

Patricia Melin
Oscar Castillo
Janusz Kacprzyk *Editors*

Design of Intelligent Systems Based on Fuzzy Logic, Neural Networks and Nature-Inspired Optimization

Studies in Computational Intelligence

Volume 601

Series editor

Janusz Kacprzyk, Polish Academy of Sciences, Warsaw, Poland
e-mail: kacprzyk@ibspan.waw.pl

About this Series

The series “Studies in Computational Intelligence” (SCI) publishes new developments and advances in the various areas of computational intelligence—quickly and with a high quality. The intent is to cover the theory, applications, and design methods of computational intelligence, as embedded in the fields of engineering, computer science, physics and life sciences, as well as the methodologies behind them. The series contains monographs, lecture notes and edited volumes in computational intelligence spanning the areas of neural networks, connectionist systems, genetic algorithms, evolutionary computation, artificial intelligence, cellular automata, self-organizing systems, soft computing, fuzzy systems, and hybrid intelligent systems. Of particular value to both the contributors and the readership are the short publication timeframe and the worldwide distribution, which enable both wide and rapid dissemination of research output.

More information about this series at <http://www.springer.com/series/7092>

Patricia Melin · Oscar Castillo
Janusz Kacprzyk
Editors

Design of Intelligent Systems Based on Fuzzy Logic, Neural Networks and Nature-Inspired Optimization



Springer

Editors

Patricia Melin

Division of Graduate Studies and Research
Tijuana Institute of Technology
Tijuana, Baja California
Mexico

Janusz Kacprzyk

Polish Academy of Sciences
Systems Research Institute
Warsaw
Poland

Oscar Castillo

Division of Graduate Studies and Research
Tijuana Institute of Technology
Tijuana, Baja California
Mexico

ISSN 1860-949X

ISSN 1860-9503 (electronic)

Studies in Computational Intelligence

ISBN 978-3-319-17746-5

ISBN 978-3-319-17747-2 (eBook)

DOI 10.1007/978-3-319-17747-2

Library of Congress Control Number: 2015939806

Springer Cham Heidelberg New York Dordrecht London

© Springer International Publishing Switzerland 2015

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, express or implied, with respect to the material contained herein or for any errors or omissions that may have been made.

Printed on acid-free paper

Springer International Publishing AG Switzerland is part of Springer Science+Business Media
(www.springer.com)

Preface

We describe, in this book, recent advances on the design of intelligent systems based on fuzzy logic, neural networks, and nature-inspired optimization and their application in areas, such as intelligent control and robotics, pattern recognition, time series prediction, and optimization of complex problems. The book is organized in eight main parts, which contain a group of papers around a similar subject. The first part consists of papers with the main theme of theoretical aspects of fuzzy logic, which basically consists of papers that propose new concepts and algorithms based on fuzzy systems. The second part contains papers with the main theme of neural networks theory, which are basically papers dealing with new concepts and algorithms in neural networks. The third part contains papers describing applications of neural networks in diverse areas, such as time series prediction and pattern recognition. The fourth part contains papers describing new nature-inspired optimization algorithms. The fifth part presents diverse applications of nature-inspired optimization algorithms. The sixth part contains papers describing new optimization algorithms. The seventh part contains papers describing applications of fuzzy logic in diverse areas, such as time series prediction and pattern recognition. Finally, the eighth part contains papers that present enhancements to meta-heuristics based on fuzzy logic techniques.

In the first part of theoretical aspects of fuzzy logic, there are five papers that describe different contributions that propose new models, concepts, and algorithms centered on fuzzy logic. The aim of using fuzzy logic is to provide uncertainty management in modeling complex problems.

In the second part of neural networks theory, there are five papers that describe different contributions that propose new models, concepts, and algorithms centered on neural networks. The aim of using neural networks is to provide learning and adaptive capabilities to intelligent systems.

In the third part of neural network applications, there are five papers that describe different contributions on the application of these kinds of neural models to solve complex real-world problems, such as time series prediction, medical diagnosis, and pattern recognition.

In the fourth part of nature-inspired optimization, there are six papers that describe different contributions that propose new models, concepts, and algorithms for optimization inspired in different paradigms of natural phenomena. The aim of using these algorithms is to provide optimization capabilities to intelligent systems or provide design methodologies for achieving optimal topological and parametric design of intelligent systems.

In the fifth part of nature-inspired optimization applications, there are seven papers that describe different contributions on the application of these kinds of algorithms to solve complex real-world optimization problems, such as time series prediction, medical diagnosis, robotics, and pattern recognition.

In the sixth part of optimization, there are seven papers that describe different contributions that propose new models, concepts, and algorithms for optimization inspired in different paradigms. The aim of using these algorithms is to provide general optimization methods and solution to some real-world problem in areas, such as scheduling, planning, and project portfolios.

In the seventh part of fuzzy logic applications, there are seven papers that describe different contributions on the application of these kinds of fuzzy logic models to solve complex real-world problems, such as time series prediction, medical diagnosis, recommending systems, education, and pattern recognition.

In the eighth part of fuzzy logic for the augmentation of nature-inspired optimization meta-heuristics, there are five papers that describe different contributions that propose new models and concepts, which can be considered as the basis for enhancing nature-inspired algorithms with fuzzy logic. The aim of using fuzzy logic is to provide dynamic adaptation capabilities to the optimization algorithms, and this is illustrated with the cases of the bat algorithm, cuckoo search, and other methods. The nature-inspired methods include variations of ant colony optimization, particle swarm optimization, the bat algorithm, as well as new nature-inspired paradigms.

In conclusion, the edited book comprises papers on diverse aspects of fuzzy logic, neural networks, and nature-inspired optimization meta-heuristics and their application in areas, such as intelligent control and robotics, pattern recognition, time series prediction, and optimization of complex problems. There are theoretical aspects as well as application papers.

January 21, 2015

Patricia Melin
Oscar Castillo
Janusz Kacprzyk

Contents

Part I Fuzzy Logic Theory

Color Image Edge Detection Method Based on Interval Type-2 Fuzzy Systems	3
Claudia I. Gonzalez, Patricia Melin, Juan R. Castro, Olivia Mendoza and Oscar Castillo	
Method for Measurement of Uncertainty Applied to the Formation of Interval Type-2 Fuzzy Sets	13
Mauricio A. Sanchez, Oscar Castillo and Juan R. Castro	
Optimization of the Interval Type-2 Fuzzy Integrators in Ensembles of ANFIS Models for Time Series Prediction:	
Case of the Mexican Stock Exchange	27
Jesus Soto and Patricia Melin	
A New Proposal for a Granular Fuzzy C-Means Algorithm.	47
Elid Rubio and Oscar Castillo	
Face Recognition with a Sobel Edge Detector and the Choquet Integral as Integration Method in a Modular Neural Networks	59
Gabriela E. Martínez, Patricia Melin, Olivia D. Mendoza and Oscar Castillo	

Part II Neural Networks Theory

Neural Network with Fuzzy Weights Using Type-1 and Type-2 Fuzzy Learning for the Dow-Jones Time Series.	73
Fernando Gaxiola, Patricia Melin and Fevrier Valdez	

Evolutionary Indirect Design of Feed-Forward Spiking Neural Networks	89
Andrés Espinal, Martín Carpio, Manuel Ornelas, Héctor Puga, Patricia Melín and Marco Sotelo-Figueroa	
Cellular Neural Network Scheme for Image Binarization in Video Sequence Analysis	103
Mario I. Chacon-Murguia and Juan A. Ramirez-Quintana	
Optimization of the LVQ Network Architecture with a Modular Approach for Arrhythmia Classification Using PSO	119
Jonathan Amezcua and Patricia Melin	
Evolution of Kernels for Support Vector Machine Classification on Large Datasets	127
Luis Carlos Padierna, Martín Carpio, Rosario Baltazar, Héctor José Puga and Héctor Joaquín Fraire	
Part III Neural Networks Applications	
Modular Neural Networks for Time Series Prediction Using Type-1 Fuzzy Logic Integration	141
Daniela Sánchez and Patricia Melin	
An Improved Particle Swarm Optimization Algorithm to Optimize Modular Neural Network Architectures	155
Alfonso Uriarte, Patricia Melin and Fevrier Valdez	
Left Ventricular Border Recognition in Echocardiographic Images Using Modular Neural Networks and Sugeno Integral Measures	163
Fausto Rodríguez-Ruelas, Patricia Melin and German Prado-Arechiga	
Optimization of Ensemble Neural Networks with Fuzzy Integration Using the Particle Swarm Algorithm for Time Series Prediction	171
Martha Pulido and Patricia Melin	
A Type-2 Fuzzy Neural Network Ensemble to Predict Chaotic Time Series	185
Victor M. Torres and Oscar Castillo	

Part IV Nature Inspired Optimization

Study of Parameter Variations in the Cuckoo Search Algorithm and the Influence in Its Behavior	199
Maribel Guerrero, Oscar Castillo and Mario García	
A New Bio-inspired Optimization Algorithm Based on the Self-defense Mechanisms of Plants	211
Camilo Caraveo, Fevrier Valdez and Oscar Castillo	
Imperialist Competitive Algorithm Applied to the Optimization of Mathematical Functions: A Parameter Variation Study	219
Emer Bernal, Oscar Castillo and José Soria	
An Improved Intelligent Water Drop Algorithm to Solve Optimization Problems	233
Diana Martinez and Fevrier Valdez	
An Improved Simulated Annealing Algorithm for the Optimization of Mathematical Functions	241
Carolina Avila and Fevrier Valdez	
Optimization of Reactive Fuzzy Controllers for Mobile Robots Based on the Chemical Reactions Algorithm	253
David de la O, Oscar Castillo, Abraham Meléndez, Patricia Melin, Leslie Astudillo and Coral Sánchez	
Part V Nature Inspired Optimization Applications	
Segmentation of Coronary Angiograms Using a Vesselness Measure and Evolutionary Thresholding	269
Ivan Cruz-Aceves and Arturo Hernández-Aguirre	
Exploring the Suitability of a Genetic Algorithm as Tool for Boosting Efficiency in Monte Carlo Estimation of Leaf Area of Eelgrass	291
Cecilia Leal-Ramirez, Héctor Echavarria-Heras and Oscar Castillo	
Obtaining Pharmacokinetic Population Models Using a Genetic Algorithm Approach	305
Oscar Montiel, J.M. Cornejo, Carlos Sepúlveda and Roberto Sepúlveda	

Parallel Evolutionary Artificial Potential Field for Path Planning—An Implementation on GPU	319
Ulises Orozco-Rosas, Oscar Montiel and Roberto Sepúlveda	
Design and Acceleration of a Quantum Genetic Algorithm Through the Matlab GPU Library	333
Oscar Montiel, Ajelet Rivera and Roberto Sepúlveda	
Implementing Pool-Based Evolutionary Algorithm in Amazon Cloud Computing Services	347
Rene Márquez Valenzuela and Mario García Valdez	
An Ant Colony Algorithm for Solving the Selection Portfolio Problem, Using a Quality-Assessment Model for Portfolios of Projects Expressed by a Priority Ranking	357
S. Samantha Bastiani, Laura Cruz-Reyes, Eduardo Fernandez, Claudia Gómez and Gilberto Rivera	
 Part VI Optimization: Theory and Applications	
A Comparison Between Memetic Algorithm and Seeded Genetic Algorithm for Multi-objective Independent Task Scheduling on Heterogeneous Machines	377
Héctor Joaquín Fraire Huacuja, Alejandro Santiago, Johnatan E. Pecero, Bernabé Dorronsoro, Pascal Bouvry, José Carlos Soto Monterrubio, Juan Javier Gonzalez Barbosa and Claudia Gómez Santillan	
Parallel Meta-heuristic Approaches to the Course Timetabling Problem	391
A. Jorge Soria-Alcaraz, Martin Carpio, Hector Puga, Jerry Swan, Patricia Melin, Hugo Terashima and A. Marco Sotelo-Figueroa	
Simplification of Decision Rules for Recommendation of Projects in a Public Project Portfolio	419
Laura Cruz-Reyes, César Medina Trejo, Fernando López Irrarragorri and Claudia G. Gómez Santillan	
A Survey of Grey Systems Applied to Multi-objective Problem	431
Fausto Balderas, Eduardo Fernandez, Claudia Gómez and Laura Cruz-Reyes	

An Efficient Representation Scheme of Candidate Solutions for the Master Bay Planning Problem	441
Paula Hernández Hernández, Laura Cruz-Reyes, Patricia Melin, Julio Mar-Ortiz, Héctor Joaquín Fraire Huacuja, Héctor José Puga Soberanes and Juan Javier González Barbosa	
Verifying the Effectiveness of an Evolutionary Approach in Solving Many-Objective Optimization Problems	455
Laura Cruz-Reyes, Eduardo Fernandez, Claudia Gomez, Patricia Sanchez, Guadalupe Castilla and Daniel Martinez	
Comparative Study on Constructive Heuristics for the Vertex Separation Problem	465
Norberto Castillo-García, Héctor Joaquín Fraire Huacuja, José Antonio Martínez Flores, Rodolfo A. Pazos Rangel, Juan Javier González Barbosa and Juan Martín Carpio Valadez	
 Part VII Fuzzy Logic Applications	
A New Approach for Intelligent Control of Nonlinear Dynamic Plants Using a Benchmark Problem	477
Leticia Cervantes and Oscar Castillo	
Fuzzy Pre-condition Rules for Activity Sequencing in Intelligent Learning Environments	489
Francisco Arce and Mario García-Valdez	
A Pre-filtering Based Context-Aware Recommender System using Fuzzy Rules	497
Xochilt Ramirez-Garcia and Mario Garcia-Valdez	
A Fitness Estimation Strategy for Web Based Interactive Evolutionary Applications Considering User Preferences and Activities Using Fuzzy Logic	507
J.C. Romero and M. García-Valdez	
Design of a Fuzzy System for Diagnosis of Hypertension	517
Juan Carlos Guzmán, Patricia Melin and German Prado-Arechiga	
Trajectory Metaheuristics for the Internet Shopping Optimization Problem	527
Mario C. López-Locés, Kavita Rege, Johnatan E. Pecero, Pascal Bouvry and Héctor J. Fraire Huacuja	

Analysis of Some Database Schemas Used to Evaluate Natural Language Interfaces to Databases.	537
Rogelio Florencia-Juárez, Juan J. González B., Rodolfo A. Pazos R., José A. Martínez F. and María L. Morales-Rodríguez	

Part VIII Fuzzy Logic and Metaheuristics

Cuckoo Search Algorithm via Lévy Flight with Dynamic Adaptation of Parameter Using Fuzzy Logic for Benchmark Mathematical Functions.	555
Maribel Guerrero, Oscar Castillo and Mario García	
Differential Evolution with Dynamic Adaptation of Parameters for the Optimization of Fuzzy Controllers.	573
Patricia Ochoa, Oscar Castillo and José Soria	
Ant Colony Optimization with Parameter Adaptation Using Fuzzy Logic for TSP Problems	593
Frumen Olivas, Fevrier Valdez and Oscar Castillo	
An Improved Harmony Search Algorithm Using Fuzzy Logic for the Optimization of Mathematical Functions	605
Cinthia Peraza, Fevrier Valdez and Oscar Castillo	
A New Algorithm Based in the Smart Behavior of the Bees for the Design of Mamdani-Style Fuzzy Controllers Using Complex Non-linear Plants.	617
Leticia Amador-Angulo and Oscar Castillo	

Part I

Fuzzy Logic Theory

Color Image Edge Detection Method Based on Interval Type-2 Fuzzy Systems

Claudia I. Gonzalez, Patricia Melin, Juan R. Castro, Olivia Mendoza and Oscar Castillo

Abstract Edge detection is one of the most commonly used operations in computer vision, image processing and pattern recognition. The efficiency of these applications depends in many cases on the quality of detected edges. A color image edge detection method based on Sobel and Interval type-2 fuzzy system IT2FSs is presented in this paper. Color images provide more information than grayscale images. Thus, more edge information is expected from a color edge detector than a grayscale edge detector. The proposed method is applied over a database of color images that include synthetic and real images. The performance of the proposed method is compared with other edge detection algorithms such as Sobel combined with type-1 fuzzy systems T1FSs and the traditional Sobel operator.

1 Introduction

Edge detection in color images is a far more difficult task than gray scale images but the color images provide more information than grayscale images; this can be vital in some computer vision applications [1]. Additionally, human perception of color images is more enriched than an achromatic picture [2]. Several color models are present such as RGB color model, YUV model, CMY color model, CMYK color model, HIS color model [3, 4].

This paper describes the application of the color image edge detection based on Sobel and interval type-2 fuzzy systems, which is performed using the RGB color model. The algorithm is tested on synthetic and real images and the results are

C.I. Gonzalez · J.R. Castro · O. Mendoza
Autonomous University of Baja California, Tijuana, Mexico

P. Melin (✉) · O. Castillo
Tijuana Institute of Technology, Tijuana, Mexico
e-mail: pmelin@tectijuana.mx

O. Castillo
e-mail: ocastillo@hafsamx.org

compared with the results of color edge image edge detection based on Type-1 fuzzy systems and the Sobel Operator.

The paper is organized as follows: In Sect. 2 the representation of a color image is defined. Section 3 presents the Color image detector using Sobel operator. Section 4 shows the color image edge detector based on type-1 and interval type-2 fuzzy systems. In Sect. 5 the simulation results are described. Finally, Sect. 6 shows the Conclusions.

2 Representation of a Color Image

Color images are of great importance in daily life and for the representation of images in the digital world. Color perception is very important but a complicated phenomenon. There are different models to represent color images, such as the RGB color model, YUV model, CMY color model, CMYK color model and HIS color model. In this paper the RGB model is applied, which is defined as follows [3].

2.1 RGB Model

The RGB color model is based on the combination of the primary colors: red (R), green (G), and blue (B). The origin of this model is in the technology of television and can be considered as the fundamental color representation in computers, digital cameras and scanners. Most programs for image processing and graphical representation are using this model for the internal representation of color.

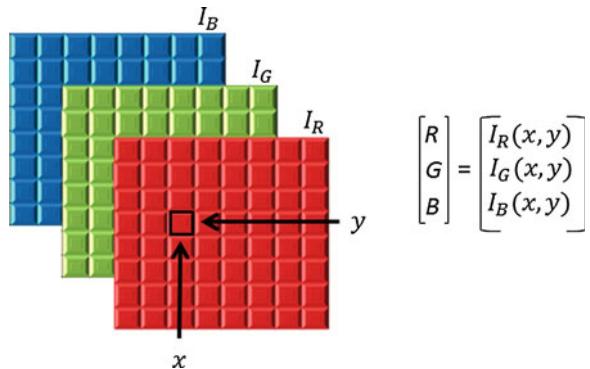
An RGB color image is an $M \times N \times 3$ array of color pixels, where each color pixel is a triplet corresponding to the red, green, and blue components of an RGB image at a specific location.

A color image can be considered as a related group of images of intensity $I_R(x, y)$, $I_G(x, y)$, $I_B(x, y)$ where the RGB value of a pixel of the color image I is obtained by accessing each of the arrays that constitute the combination of the shape [3]. The RGB color image composition is shown in Fig. 1.

3 Color Image Edge Detector Using the Sobel Operator

The edges can be considered as points in an image in which the intensity in a given direction changes dramatically. Depending on the change in intensity is the edge value to that point in the image [5].

Fig. 1 Composition of a color image RGB



The Sobel operator is one of the most used methods for edge detection. This operator uses a filter defined by 3×3 coefficients matrices which are expressed by

$$Sobelx = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \quad (1)$$

$$Sobely = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad (2)$$

The matrices *Sobelx* in (1) and *Sobely* in (2) are separately applied on the input image (*I*) to obtain two gradient components *gx* in (3) and *gy* in (4) in the horizontal and vertical orientations respectively [5, 6].

The size and direction of the gradients *gx* and *gy* are calculated by

$$gx = Sobelx * I \quad (3)$$

$$gy = Sobely * I \quad (4)$$

On a grayscale digital image, the magnitude of the edge or output edge is defined as the magnitude of the gradient *G*, which is defined by

$$G = \sqrt{g_x^2 + g_y^2} \quad (5)$$

The RGB color image can be considered as a related group of images of intensity $I_R(x, y)$, $I_G(x, y)$, $I_B(x, y)$. To apply the Sobel Operator in color image we need calculated the magnitude of the gradient (5) for each dimension of the color image I_R (Red), I_G (Green) and I_B (Blue).

The magnitude of the \mathbf{G}_R gradient for the Red dimension is defined by

$$Rg_x = Sobelx * I_R \quad (6)$$

$$Rg_y = Sobely * I_R \quad (7)$$

$$G_R = \sqrt{Rg_x^2 + Rg_y^2} \quad (8)$$

For the Green dimension, the magnitude of the \mathbf{G}_G gradient is expressed by

$$Gg_x = Sobelx * I_G \quad (9)$$

$$Gg_y = Sobely * I_G \quad (10)$$

$$G_G = \sqrt{Gg_x^2 + Gg_y^2} \quad (11)$$

An finally the magnitude of the \mathbf{G}_B gradient, for the Blue dimension is defined by

$$Bg_x = Sobelx * I_B \quad (12)$$

$$Bg_y = Sobely * I_B \quad (13)$$

$$G_B = \sqrt{Bg_x^2 + Bg_y^2} \quad (14)$$

The result of the edge detector based on the Sobel operator applied on color images is shown in Fig. 2.

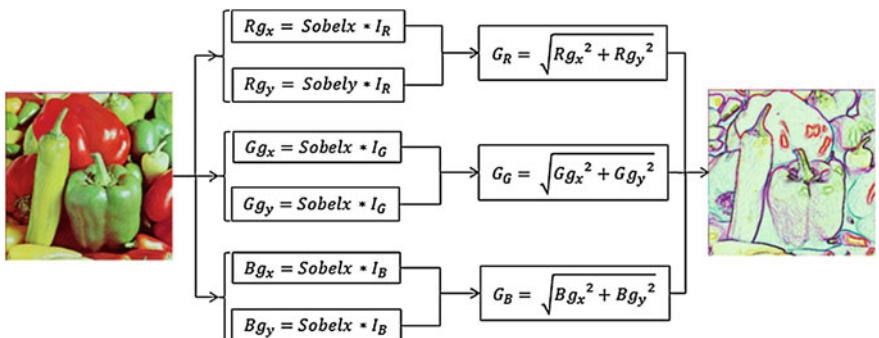


Fig. 2 Color image edge detector using the Sobel operator

4 Color Image Edge Detector Based on Fuzzy Logic

In the last years we have witnessed a rapid growth in a number of applications of fuzzy logic [7–10]. Fuzzy logic techniques have been used in many image recognition problems [11, 12], digital image processing, such as the detection edges [13–15], noise filter [16, 17], feature extraction [18], signal estimation, classification and clustering [2]. Fuzzy logic techniques represent a powerful alternative to design smart engineering systems. There are many advantages to using fuzzy inference systems [19]. Fuzzy systems are linguistic, not numerical, making it similar to the way humans think [20, 21].

In this section, an application on digital image processing using fuzzy logic is presented. The idea is to develop a color image edge detector based on Sobel operator combined with fuzzy logic; for this case study the edge detector is performed using type-1 fuzzy systems and interval type-2 fuzzy systems.

4.1 Fuzzy Inference System for Color Image Edge Detection

To apply the fuzzy approach for color image edge detection based on the Sobel combined type-1 and interval type-2 fuzzy inference system six inputs are required.

The six inputs are the gradients with respect to *x-axis* and *y-axis* for each color image dimension (Red, Green and Blue); these gradients are calculated with (6), (7), (9), (10), (12) and (13). In Fig. 3, the type-2 fuzzy inference system for edge detection in a color image is shown. The three outputs of the model presented in Fig. 3, represent the magnitude of the gradient for the color image dimension.

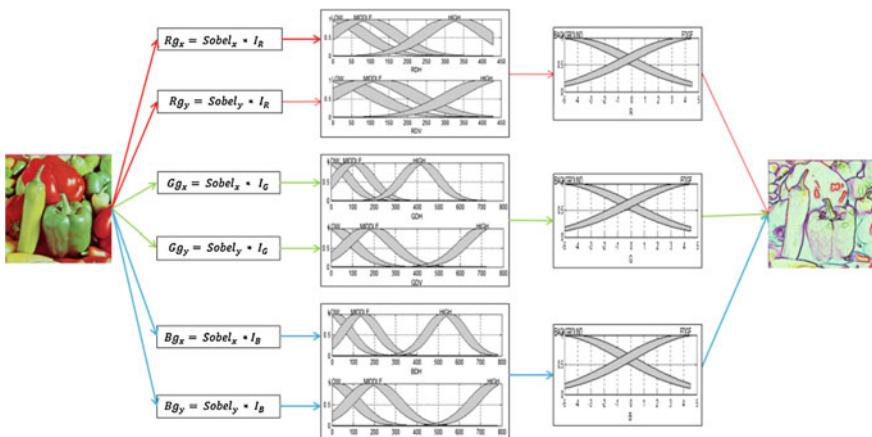


Fig. 3 Color image edge detector using Sobel combined with type-2 fuzzy systems

The two models (Sobel + T1FSs and Sobel + ITFSs) are developed under the same conditions, which are required to be able to make a comparison of both results.

4.2 Fuzzy Inference Rules

The fuzzy inference system is designed considering different groups of rules. These fuzzy rules are obtained based on the knowledge of an expert in image processing and combined with a series of tests, to achieve good results with a minimum number of rules. The rules used in the experiments are presented as follows

1. If (RDH is HIGH) or (RDV is HIGH) then (R is EDGE)
2. If (RDH is MIDDLE) or (RDV is MIDDLE) then (R is EDGE)
3. If (RDH is LOW) and (RDV is LOW) then (R is BACKGROUND)
4. If (GDH is HIGH) or (GDV is HIGH) then (R is EDGE)
5. If (GDH is MIDDLE) or (GDV is MIDDLE) then (R is EDGE)
6. If (GDH is LOW) and (GDV is LOW) then (R is BACKGROUND)
7. If (BDH is HIGH) or (BDV is HIGH) then (R is EDGE)
8. If (BDH is MIDDLE) or (BDV is MIDDLE) then (R is EDGE)
9. If (BDH is LOW) and (BDV is LOW) then (R is BACKGROUND)

5 Simulation Results

In this section we present the results of the proposed image color edge detection based on Sobel and Fuzzy Systems (T1FSs and IT2Fs). For the simulation results we used a color image database of synthetic and real images, and these images are presented in Table 1.

In the first test, the traditional Sobel operator was applied, and for this experiment we used the methodology presented in Sect. 3. The results for this edge detector are shown in Tables 2 and 3.

In another test, the proposed edge detector based on type-1 and type-2 fuzzy systems is now applied to the same color images of Table 1. These methods (Sobel + T1FSs and Sobel + IT2FSs) are designed with the structure described in Sect. 4; both methods use the same inputs, outputs and number of fuzzy rules. The edge detection obtained with these methodologies is presented in Tables 2 and 3.

The visual edge detections for the Sobel, Sobel + T1FSs, Sobel + IT2-FLSs, are shown in Tables 2 and 3. Graphically, we can clearly note that the edges detected in the Sobel operator are smaller in number than those detected by Sobel + T1FLS and Sobel + IT2-FLS. On the other hand; visually, we cannot appreciate the difference in the detected edges between Sobel + T1FLS and Sobel + IT2-FLS.

Table 1 Color images database

Synthetic Image	Real images

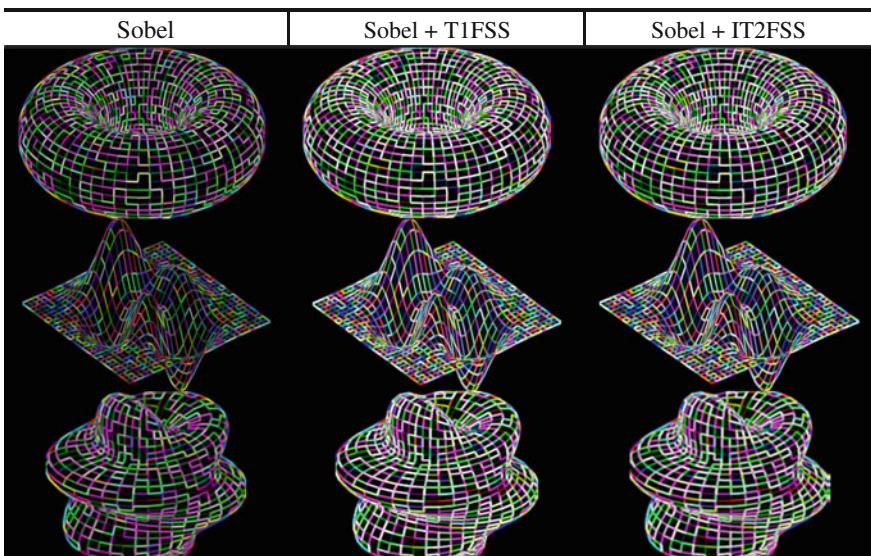
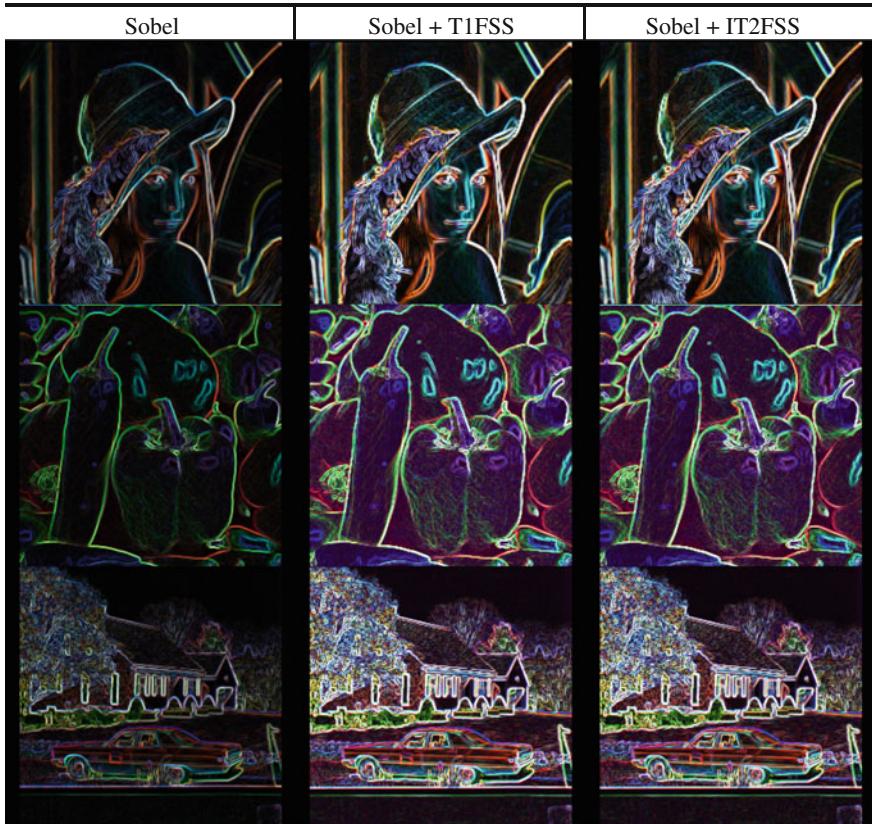
Table 2 Edge detection in color synthetic images

Table 3 Edge detection in color real images

6 Conclusions

A new method based on Type-1 and Type-2 fuzzy systems has been proposed in this paper for extracting edges from color images. On Tables 2 and 3, we can appreciate that the proposed fuzzy edge detectors are better than the traditional Sobel operator; however, we cannot appreciate clearly the difference between T1FSs and T2FSs. In future work we can apply some metric to measure the differences in the detected edges.

Acknowledgments We thank the MyDCI program of the Division of Graduate Studies and Research, UABC, and the financial support provided by our sponsor CONACYT contract grant number: 44,524.

References

1. Torre, V., Poggio, T.A.: On edge detection. *IEEE Trans. Pattern Anal. Mach. Intell.* **8**(6), 147–163 (1986)
2. Koschan, A., Abidi, M.: Detection and classification of edges in color images. *Sig. Process. Mag. IEEE* **22**(1), 64–73 (2005)
3. Gonzalez, R.C., Woods, R.E., Eddins, S.L.: Digital image processing using matlab. Prentice Hall, New Jersey (2004)
4. Kulkarni, A.D.: Computer vision and fuzzy neural systems. Prentice Hall, New Jersey (2001)
5. Sobel, I.: Camera models and perception. Ph.D. thesis, Stanford University, Stanford, CA (1970)
6. Mendoza, O., Melin, P., Licea, G.: A new method for edge detection in image processing using interval type-2 fuzzy logic. In: *IEEE International Conference on Granular Computing (GRC 2007)*, p. 151 (2007)
7. Melin, P., Mendoza, O., Castillo, O.: An improved method for edge detection based on interval type-2 fuzzy logic. *Expert Syst. Appl.* **37**(12), 8527–8535 (2010)
8. Mendel, J.: Uncertain rule-based fuzzy logic systems: introduction and new directions. Prentice Hall, New Jersey (2001)
9. Zadeh, L.A.: Fuzzy sets, vol. 8. Academic Press Inc., USA (1965)
10. Zadeh, L.A.: The concept of a linguistic variable and its application to approximate reasoning —I. *Inf. Sci.* **8**(3), 199–249 (1975)
11. Mendoza, O., Melin, P.: Quantitative evaluation of fuzzy edge detectors applied to neural networks for image recognition. *Advances in research and developments in digital systems*, pp. 324–335. In: *Stochastic Algorithms: Foundations and Applications. Lecture Notes Computer Science*, vol. 5792, pp. 169–178 (2011)
12. Mendoza, O., Melin, P., Licea, G.: A hybrid approach for image recognition combining type-2 fuzzy logic, modular neural networks and the Sugeno integral. *Inf. Sci.* **179**(13), 2078–2101 (2009)
13. Biswas, R., Sil, J.: An improved canny edge detection algorithm based on type-2 fuzzy sets. *Procedia Technol.* **4**, 820–824 (2012)
14. Bustince, H., Barrenechea, E., Pagola, M., Fernandez, J.: Interval-valued fuzzy sets constructed from matrices: application to edge detection. *Fuzzy Sets Syst.* **160**(13), 1819–1840 (2009)
15. Liang, Q., Mendel, J.: Interval type-2 fuzzy logic systems: theory and design. *IEEE Trans. Fuzzy Syst.* **8**, 535–550 (2000)
16. Schulte, S., De Witte, V., Kerre, E.E.: A fuzzy noise reduction method for color images. *IEEE Trans. Image Process.* **16**(5), 1425–1436 (2007)
17. Yuksel, M.E., Basturk, A.: Application of type-2 fuzzy logic filtering to reduce noise in color images. *Comput. Intell. Mag. IEEE* **7**(3), 25–35 (2012)
18. Teruhisa, S., Futoki, S., Hiroshi, K., Toshiaki, O.: Application of an edge detection method to satellite images for distinguishing sea surface temperature fronts near the Japanese coast. *Remote Sens. Environ.* **98**(1), 21–34 (2005)
19. Mendel, J.: Advances in type-2 fuzzy sets and systems. *Inf. Sci.* **177**(1), 84–110 (2007)
20. Karnik, N.N., Mendel, J.M., Liang, Q.: Type-2 fuzzy logic systems. *IEEE Trans. Fuzzy Syst.* **7**(6), 643–658 (1999)
21. Zadeh, L.A.: Fuzzy logic. *Computer* **1**(4), 83–93 (1988)

Method for Measurement of Uncertainty Applied to the Formation of Interval Type-2 Fuzzy Sets

Mauricio A. Sanchez, Oscar Castillo and Juan R. Castro

Abstract This paper proposes a new method for directly discovering the uncertainty from a sample of discrete data, which is then used in the formation of an Interval Type-2 Fuzzy Inference System. A Coefficient of Variation is used to measure the uncertainty on a finite sample of discrete data. Based on the maximum possible coverage area of the Footprint of Uncertainty of Gaussian membership functions, with uncertainty on the standard deviation, which then are modified according to the found index values, obtaining all antecedents in the process. Afterwards, the Cuckoo Search algorithm is used to optimize the Interval Sugeno consequents of the Fuzzy Inference System. Some sample datasets are used to measure the output interval coverage.

1 Introduction

Uncertainty, as it is currently perceived, is still something of a mistified topic. Being defined as something that is doubtful or unknown, in which by nature cannot be directly measured, therefore showing a first problem in making use of it. Although by nature, uncertainty is an unknown, it has not stopped engineers, scientists, mathematicians, etc. from using it. That is, although directly not known, an approximate of it can be modeled and used, improving the models in which it is used. By using uncertainty in a model, that model will improve its resilience, thus obtaining a better model in the end.

M.A. Sanchez · J.R. Castro
Autonomous University of Baja California, Tijuana, Mexico
e-mail: mauricio.sanchez@uabc.edu.mx

J.R. Castro
e-mail: jrcastror@uabc.edu.mx

O. Castillo (✉)
Tijuana Institute of Technology, Tijuana, Mexico
e-mail: ocastillo@hafsamx.org

Most current literature on uncertainty [1–6] is mainly based on having previous knowledge of the confidence interval around certain measurements, which translates into what is the probable uncertainty which exists within certain measurements, usually expressed with the plus-minus symbol \pm (e.g. 10.4 ± 0.02 , with in interval representation of $[10.38, 10.42]$).

As for models with uncertainty, there exists a logic which directly manages uncertainty, this being Interval Type-2 Fuzzy Logic (IT2 FL) [7], which infers Interval Type-2 Fuzzy Sets (IT2 FS) and ultimately obtains an interval or a crisp value [8]. IT2 FS manage uncertainty directly into its logic by means of confidence intervals [9], the best solution could be anywhere within such interval, and as such is an excellent tool for directly applying and inference when dealing with uncertainty. And as stated, the output interval can be used as the end result and a defuzzification process can be computed upon such interval in the case that a crisp value is required, and not an interval.

In this paper, a link is proposed between a measure of dispersion and uncertainty, which is ultimately used in the formation of IT2 FS. The platform for the model is created by a Fuzzy C-Means algorithm [10], afterwards using the Coefficient of Variation is used to calculate the Fingerprint Of Uncertainty (FOU) of each individual IT2 FS in the antecedents of the Interval Type-2 Fuzzy Inference System (IT2 FIS), and finally, a Cuckoo Search algorithm [11] is used to optimize Interval Type-2 Sugeno linear consequents [12]. The proposed method can be categorized as a hybrid algorithm because it requires multiple steps/algorithm to work in sequence for the final result to be obtained.

This paper is divided into three sections, the first is a brief introduction to the definition of Interval Type-2 Fuzzy Sets; the following section describes in detail both the premises and the proposed method; afterwards, some experimental results are shown and discussed which asses the viability of the proposed method; finally, concluding remarks are given as well as a couple of open questions as future work.

2 Interval Type-2 Fuzzy Sets

With the introduction of Fuzzy Sets in 1965 [13], it improved upon formal hard logic, where instead of only having two choices of truth values $\{0, 1\}$, any value between $[0, 1]$ was now possible. This set an unprecedented involvement in research that up to today is still very strong, first came Type-1 Fuzzy Sets [14], which can only represent vagueness or imprecision, later came Interval Type-2 Fuzzy Sets, which could now, apart from vagueness, also represent a degree of uncertainty (which is the focus of the proposed method in this paper), although recently General Type-2 Fuzzy Sets [15] are starting to gain traction in research, is still far from maturity when compared to Type-1 or Interval Type-2 Fuzzy Sets.

By nature, IT2 FS directly integrate uncertainty into its reasoning. This behavior is best applied in the case of when it is expected to deal with uncertainty in the

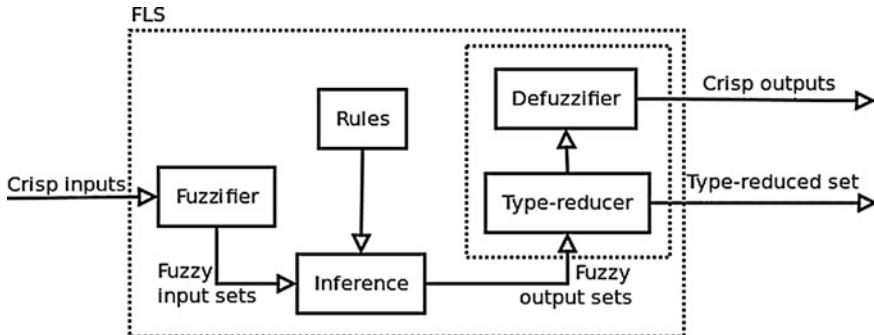


Fig. 1 Block diagram describing an IT2 FLS. With a crispst input, two outputs are possible, a confidence interval in which any possible point within such interval is a correct answer, or a crisp value, in the case a single real number is required as output

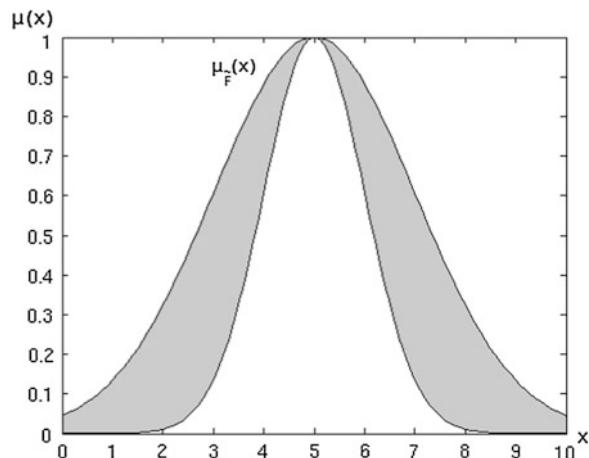
system that it is modeling, or when certain confidence intervals (uncertainty) are known a priori to designing the IT2 FIS.

The most general descriptive form of an IT2 FIS is through a block diagram, as shown in Fig. 1, which describes the basic inner functions of the complete inference. The Fuzzifier block may or may not transform the crisp input into a FS, this is chosen depending on the intended behavior of the system; the Inference block takes from the Rules block and reasons upon each input's compatibility; the Type-reducer block processes the outputs into an interval; finally, the Defuzzifier block reduces the interval from the previous block and obtains a single real number.

An IT2 FS \tilde{A} is represented by $\underline{\mu}_{\tilde{A}}(x)$ and $\overline{\mu}_{\tilde{A}}(x)$ which are the lower and upper membership functions respectively of $\mu_{\tilde{A}}(x)$, and is expressed as $\tilde{A} = \int_{w^l \in [\underline{\mu}_{F_l}(x_k), \overline{\mu}_{F_l}(x_k)]} 1/w^l$. Where, $x \in X$, k is the k th antecedent, and l the l th rule.

A sample IT2 FS is shown in Fig. 2, here a Gaussian membership function with uncertainty in the standard deviation.

Fig. 2 Sample IT2 FS membership function.
A Gaussian membership function is shown which has uncertainty through the standard deviation



The representation for rules in an IT2 FIS is formatted as shown in Eq. (1), where, $l = 1, \dots, M$ rules, $p = 1, \dots, q$ inputs, \tilde{F} is an antecedent IT2 FS, and \tilde{G} a consequent IT2 FS.

$$R^l : \text{IF } x_1 \text{ is } \tilde{F}_1^l \text{ and } \dots x_p \text{ is } \tilde{F}_p^l, \text{ THEN } y \text{ is } \tilde{G}^l \quad (1)$$

3 Proposed Method for Measuring Uncertainty

Before giving a detailed description of the proposed method, the input data must first be defined. As a starting point, a dataset is required, these data pairs, defined as Eq. (2), where φ is a set of ordered input values, and γ is a set of ordered output values, such that Γ forms a tuple of ordered sets of inputs with their respective outputs.

$$\Gamma = \langle \varphi, \gamma \rangle \quad (2)$$

Having a dataset Γ , first some pre-processing must be done in order to obtain the required inputs to the proposed method, this process is executed in order to acquire a description of the IT2 FIS, that is, to obtain the rule description ω as well as each membership function's base description, and the set of data pairs which affected the formation of each membership function $\gamma \in \Gamma$. As this is are the required inputs $\{\omega, \gamma\}$ to the proposed method, a Fuzzy C-Means (FCM) algorithm was chosen to process the raw dataset Γ into the listed required inputs $\{\omega, \gamma\}$. The FCM provides a description of rules by means of a center for each membership function for each rule. Although the FCM can define consequents for the rules in a Fuzzy System, only the antecedents are used. As the other required input is a set of data pair sets which affected the definition of each center, this can be obtained from the partition matrix that is given by the FCM; for each data pair there exists a membership value $[0, 1]$ which defines how much a certain data pair belongs to a cluster, or rule of the found FIS, to simplify building the sets of data pairs, a simple competitive rule is used: *the cluster with the highest value decides that said data pair belongs to its formation set*. With both required inputs obtained, the proposed method can now begin.

3.1 Dispersion in Data

Data dispersion in a sample of data pairs can be interpreted as a case of uncertainty. An example of varying degrees of dispersion is shown in Fig. 3, where low, medium, and high data dispersion, in relation to its center can, be perceived.

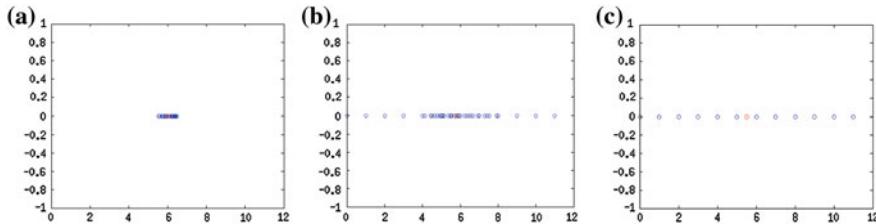


Fig. 3 Example of data dispersion. **a** Low data dispersion, **b** medium data dispersion, and **c** high data dispersion

When there is low dispersion of data samples near its representative center point, most data points are bound by only a small distance as the standard deviation is very small. This being interpreted directly into uncertainty in data, low dispersion is low uncertainty because its numerical evidence concludes that there is near zero possibility that further singular samples will fall far from the central point, unto which all previous numerical evidence is very close to. In the case of medium data dispersion, although there is a concentration of numerical evidence near its central point, there are still data points farther from its center, this leads to knowing that although future reading might obtain evidence which is far from the center, the probabilities of this occurring is low when compared to having future readings fall near the center, although not as near in the case of lower dispersion, this behavior points to having a medium amount of uncertainty. On the extreme case of high dispersion, where every sampled data point is evenly distributed throughout the range, the available numerical evidence gives way to conclude that any future sample may equally land on any section, therefore a high amount of uncertainty exists.

3.2 Relation Coefficient of Variation with Uncertainty

For the purpose of converting dispersion into uncertainty, a measure is first required which can identify a degree of dispersion in a given set, preferably a normalized value, and as such requirement, the Coefficient of Variation c_v , shown in Eq. (3), where σ is the standard deviation, and μ is the mean of the set.

$$c_v = \frac{\sigma}{\mu} \quad (3)$$

This coefficient has some limitations which can be avoided by applying some modifications. First, c_v should only be computed on non-negative values, for the case of existing negative values, the solution is to remap all values onto the positive side of the axis. Second, if μ has a value of 0 (zero), this would cause an error in computation, the solution is to add ε , which is a very small value, assuring a non-division by zero. Another note on the behavior of c_v , is that in normal distributions,

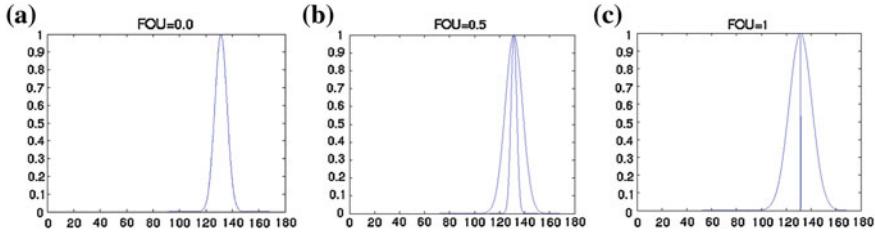


Fig. 4 Examples of varying degrees of FOU. Where **a** FOU = 0, **b** FOU = 0.5, and **c** FOU = 1

values of [0, 1] are most likely to be obtained, but non-normal distributions can obtain values above 1. Fortunately, with the FCM, all calculated sets are normal distributions, so this is a non-issue with the current implementation.

With the known limitations of c_v , an equation which modifies a set D is proposed which addresses the issue negative values, shown in Eq. (4), where if a value exists that is negative then the absolute value of the minimum is added to the set, thus remapping all values into the domain of positive values.

$$\text{IF } (\exists x \in D), \{x|x < 0\} \text{ THEN } D = D + |\min(D)| \quad (4)$$

In addition, a modification of Eq. (3) to suppress a possible division by zero, as shown in Eq. (5), where ε is a very small value.

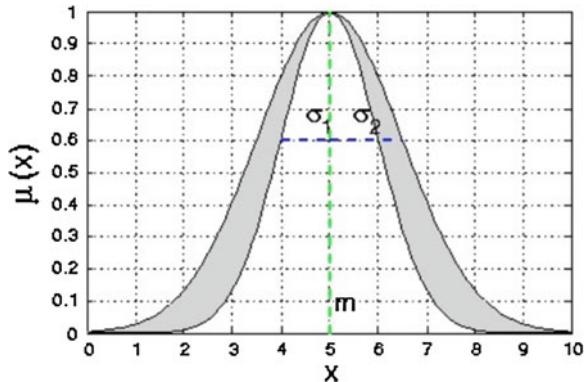
$$c_v = \frac{\sigma}{\mu + \varepsilon} \quad (5)$$

To express a relation dispersion-uncertainty, when dealing with IT2 FS, the Footprint of Uncertainty (FOU) is used. This relation is a direct proportion $FOU \propto c_v$. When there is low dispersion, there is a small FOU, when there is a medium amount of dispersion, there is a medium amount of FOU, and when there exists a high amount of dispersion, there is a high amount of FOU. This is better expressed in Fig. 4, where varying degrees of a measure of dispersion has been converted into a FOU which directly forms an IT2 FS, explained in the following sub-section.

3.3 Proposed Method

To form IT2 FS for the antecedents of a FIS, the first step is to obtain rule configuration, and data pair sets for each inputs on each rule, via a FCM algorithm. Afterwards each set of data pairs is worked on independently of each other. First, a standard deviation σ is found for the set in relation to its μ , which was found by the FCM, then the c_v is calculated. This value is now used to search for the optimal FOU area in an IT2 FS. Considering Fig. 4c, this would be the highest possible area. The initial search is done by first considering the highest possible area and the σ which was already calculated,

Fig. 5 IT2 FS represented by a Gaussian membership function with uncertainty in the standard deviation



with discrete small steps a search is performed for the FOU value which equals c_v . The smallest value is set as $\sigma_1 = \sigma_2$, shown in Fig. 4a. Each increment step λ affects σ as shown in Eq. (6), this is done iteratively while $\sigma_i \leq \|\mu, \sigma_0\|$.

$$\sigma \pm \lambda \quad (6)$$

Once the search has found the values of σ_1 and σ_2 which represent the desired FOU, the IT2 FS can be formed. Which has the form of Fig. 5, this can be formed with the values which have been calculated, by the FCM, μ , and by the proposed method, σ_1 and σ_2 . This concludes the proposed method for building the antecedents of an IT2 FIS.

3.4 IT2 Sugeno Fuzzy Consequents

The proposed method only obtains the IT2 FS for the antecedents of a FIS, the next required step is to obtain the consequents of the FIS. This is done by optimizing the IT2 Sugeno linear parameters via a Cuckoo Search algorithm. Although any other optimization algorithm can be used.

4 Experimental Results and Discussion

To test the proposed method, various datasets were used. The validation method was to verify that the interval output of the IT2 FIS had good coverage of the reference targets and at the same time not overreaching too far with the output interval.

Among the used datasets, three were used. A synthetic dataset of a 5th Order curve [16], with 1 input (x) and 1 output (y), and 94 total samples. And two real datasets; engine behavior [16], with 2 inputs (fuel rate, speed) and 2 outputs (torque, nitrous oxide emissions), and 1199 total samples; and Hahn1 [16], with 1 input (temperature) and 1 output (thermex), with 236 total samples.

4.1 Experimental Results

The obtained IT2 FIS for each dataset is shown in Figs. 6, 7 and 8, making emphasis on the FOU of the individual membership functions in the antecedents, where varying degrees of uncertainty can be seen.

As for the output coverage for each dataset, using 40 % training and 60 % training, Table 1 show the summary of the obtained coverage results.

The last set of results show graphical representations of the respective outputs for each dataset, these are shown in Figs. 9, 10 and 11. Where the blue points represent the output targets, and the lower and green lines represent the coverage of the FOU (Fig. 12).

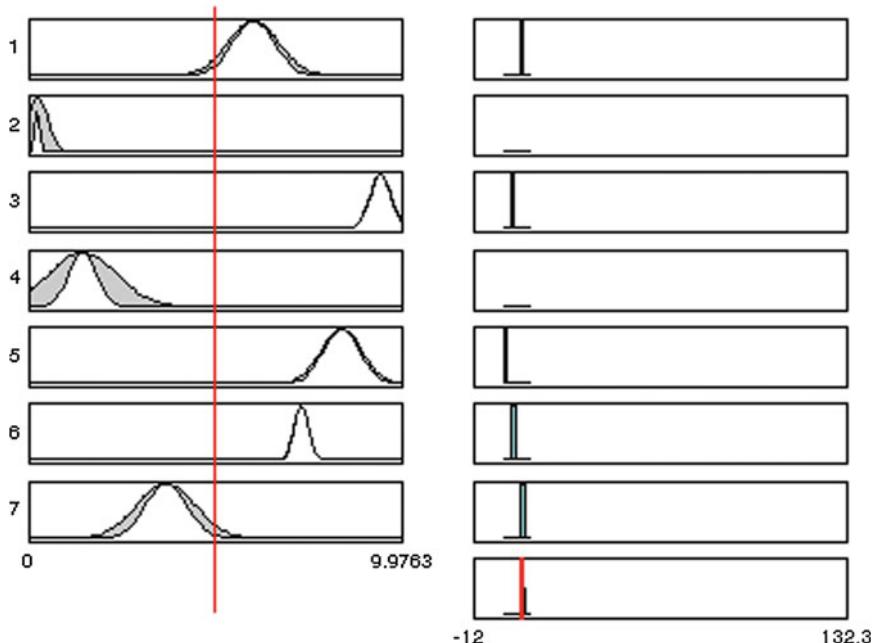


Fig. 6 IT2 FIS for solving the 5th order curve dataset

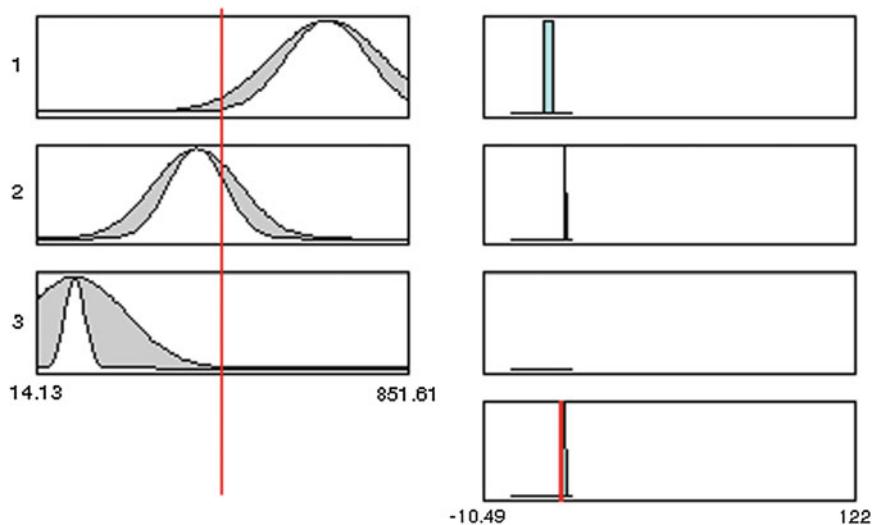


Fig. 7 IT2 FIS for solving the Hahn1 dataset

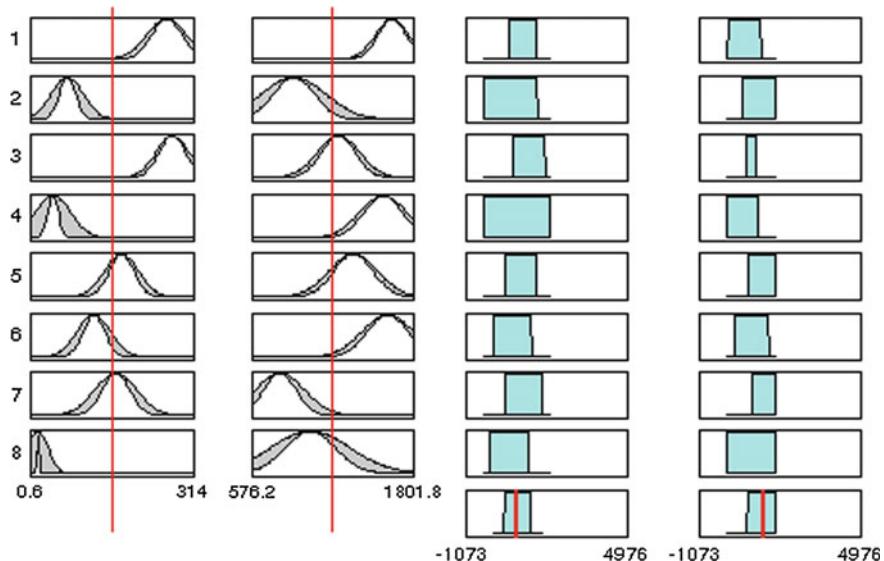


Fig. 8 IT2 FIS for solving the engine behavior dataset

Table 1 Obtained output coverage results for the chosen datasets

Dataset name	Coverage (%)
5th Order curve	100
Hahn1	100
Engine behavior	99.88/99.66

Fig. 9 Output coverage for the 5th order dataset

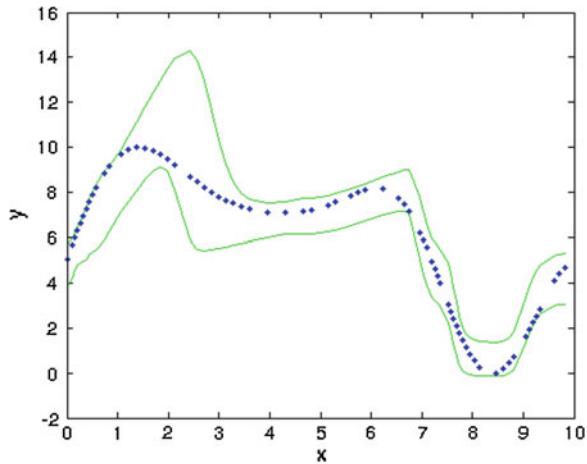
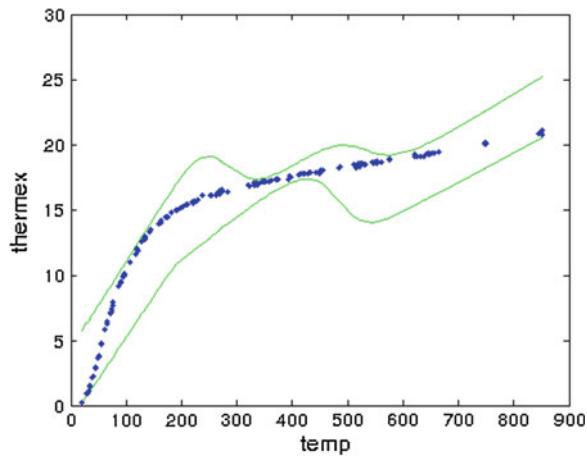


Fig. 10 Output coverage for the Hahn1 dataset



4.2 Results Discussion

With the obtained results, two facets of discussion arise, on the individual level and on the general level. On the individual level; for the 5th Order curve, a full coverage of the target is achieved although there are spikes where the curve changes slope, this is caused by the linear consequents which cannot follow abrupt changes in the

Fig. 11 Output coverage for the Hahn1 dataset. For the first output of the FIS

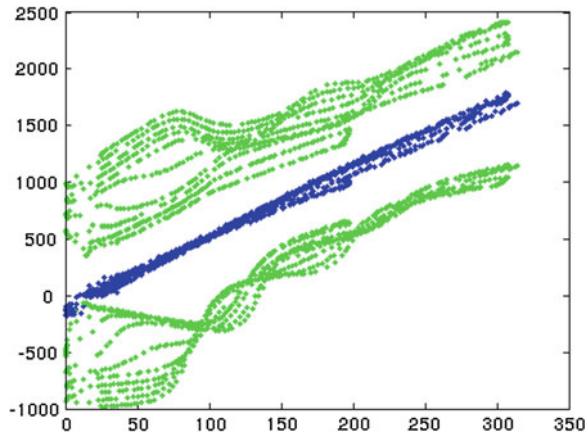
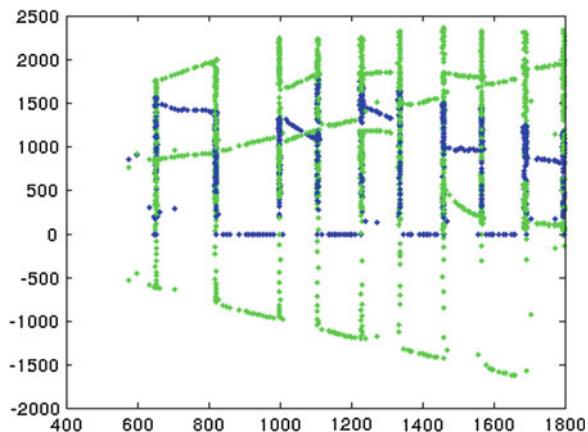


Fig. 12 Output coverage for the Hahn1 dataset. For the second output of the FIS



curve grade due to the small amount of rules used for this FIS. A solution would be to use more rules to compensate, but this would also cause an additional, and unnecessary, complexity in the system. Yet the overall behavior is acceptable as there is sufficient coverage as well as a controlled width of the output uncertainty. For the Hahn1 solution, it has the same curve behavior of the 5th Order curve, where with only three rules there is a pronounced visible behavior in the linear output of the consequents. Yet there is good coverage, of 100 %, of the target outputs via the controlled output uncertainty. Finally, for the engine, having two outputs, each ones behavior was slightly different. The first output has a more predictable behavior by better following the output targets with its coverage of 99.88 % of reference targets, whereas the second output's behavior is not as linear, such that it holds a tendency to expand as the x axis increases, although it has a coverage of 99.66 % of its reference targets. It must be noted that this specific behavior is more in line with how the Cuckoo Search algorithm optimized the

consequents, because the spreads on each individual consequent control the output interval behavior. The solution would be to adjust the Cuckoo Search for better performance or use another optimization algorithm that obtains a better solution.

On the general level of the obtained results, the formed antecedents give a good representation of uncertainty based on the dispersion of the individual sets which affected the creation of the rule configurations found by the FCM. It also depicts a behavior that IT2 FS are not always necessary, with low to no dispersion, and a T1 FS would be more than enough.

Being dependent on other algorithms can limit the general performance of the proposed method. Yet it also adds more possibilities, such as interchanging clustering algorithms to one that can obtain better rule configurations and belonging sets to be used by the proposed method. As for the optimization of the IT2 Sugeno linear consequents, there is a vast amount of optimization algorithms which could also be used for acquiring better results and thus improving the output interval behavior.

5 Conclusion and Future Work

5.1 Conclusions

With the suggested relation dispersion-uncertainty, direct uncertainty extraction is possible from existing data. This relation is found through the Coefficient of Variation, an existing equation used to measure the amount of dispersion in a set, this measure is a normalized value between 0 and 1, that although higher values than 1 are possible, this is only for non-normal distributions, which, for the purposed application, are non-existent considering that the sets are created by a clustering algorithm which only groups in normal distributions of data.

The application shown in this paper, of forming IT2 FS through the suggested equation, which relates dispersion-uncertainty, finds this relation based on the maximum possible achievable FOU, valued at 1, and relates to the maximum possible Coefficient of Variation, in a normal distribution, valued also at 1. This relation of dispersion-uncertainty-FOU is the main contribution of this paper.

With a deeper examination of the experimental results, there is much dependence on the FCM algorithm, where if such algorithm fails to provide a good model, the proposed method would fail also, since the proposed method depends on the performance of the clustering algorithm. Fortunately, if the FCM fails, other clustering algorithms could be used.

5.2 Future Work

Considering the limitation, as well as dependence, of the clustering algorithm, which other clustering or non-clustering techniques could be used to create a better pairing with the proposed method?

With the other high dependence on optimization algorithms for the consequent section of the IT2 FIS, what other optimization algorithm could be used to best pair with the proposed method?

In this paper an IT2 FS was formed, represented by a Gaussian membership function with uncertainty in the standard deviation. How would other IT2 FS membership function be adapted to use the proposed method?

How the area was directly correlated to the FOU by means of its maximum possible area was proposed. Is this the best approach?

Acknowledgments We thank the MyDCI program of the Division of Graduate Studies and Research, UABC, and Tijuana Institute of Technology the financial support provided by our sponsor CONACYT contract grant number: 314,258.

References

1. Yu, X., Mehrotra, S.: Capturing uncertainty in spatial queries over imprecise data. In: Database and Expert Systems Applications, pp. 192–201. Springer, Berlin (2003)
2. Klir, G.J.: Uncertainty and Information: Foundations of Generalized Information Theory. Wiley, New York (2005)
3. Weise, K., Woger, W.: A Bayesian theory of measurement uncertainty. *Meas. Sci. Technol.* **3** (1992)
4. Jurado, K., Ludvigson, S.C., Ng, S.: Measuring uncertainty. *Am. Econ. Rev. (AEA)* **105**(3), 1177–1216 (2013)
5. Chen, G., Ying, M., Liu, Y.: Dealing with uncertainty and fuzziness in intelligent systems. *Int. J. Intell. Syst.* **24**, 223–225 (2009)
6. Klir, G.J., Wierman, M.J.: Uncertainty-Based Information. Springer, Heidelberg (1999)
7. Mendel, J.M., John, R.I., Liu, F.: Interval type-2 fuzzy logic systems made simple. *IEEE Trans. Fuzzy Syst.* **14**, 808–821 (2006)
8. Castillo, O., Melin, P.: Recent Advances in Interval Type-2 Fuzzy Systems. Springer, Berlin (2012)
9. Mo, H., Wang, F.-Y., Zhou, M., Li, R., Xiao, Z.: Footprint of uncertainty for type-2 fuzzy sets. *Inf. Sci. (Ny)* **272**, 96–110 (2014)
10. Bezdek, J.C., Ehrlich, R., Full, W.: FCM: the fuzzy c-means clustering algorithm. *Comput. Geosci.* **10**, 191–203 (1984)
11. Yang, X.-S.: Cuckoo search via lévy flights. In: World Congress on Nature and Biologically Inspired Computing (NaBIC) 2009, pp. 210–214. IEEE (2009)
12. Ying, H.: Interval type-2 Takagi-Sugeno fuzzy systems with linear rule consequent are universal approximators. In: NAFIPS 2009—2009 Annual Meeting of the North American Fuzzy Information Processing Society, pp. 1–5. IEEE (2009)
13. Zadeh, L.A.: Fuzzy Sets. *Inf. Control* **8**, 338–353 (1965)
14. Melin, P., Castillo, O.: Fuzzy modeling fundamentals. Wiley Encycl. Comput. Sci. Eng. (2007)
15. Mendel, J.: General type-2 fuzzy logic systems made simple: a tutorial. *IEEE Trans. Fuzzy Syst.* **22**, 1 (2013)
16. The MathWorks, Inc.: Natick. MATLAB Release, Massachusetts, U.S. 2013b, (2013)

Optimization of the Interval Type-2 Fuzzy Integrators in Ensembles of ANFIS Models for Time Series Prediction: Case of the Mexican Stock Exchange

Jesus Soto and Patricia Melin

Abstract This paper describes the optimization of the fuzzy integrators in Ensembles of ANFIS model for time series prediction: case of the Mexican Stock Exchange. The Mexican stock exchange that is used corresponds to the period of 11/09/2005 to 01/15/2009 to simulate the performance of the proposed architecture. We used interval type-2 fuzzy systems to integrate the outputs (forecast) of each of the ANFIS models in the Ensemble. Genetic Algorithms (GAs) are used for the optimization of memberships function “MFs” for the 2 MFs (used linguistic labels “Small and Large”) and for the 3 MFs (used linguistic labels “Small, Middle and Large”) parameters of the fuzzy integrators. In the experiments the genetic algorithms optimized the Gaussian, Generalized Bell and Triangular membership functions for each of the fuzzy integrators; in the interval type-2 fuzzy integrator there are more parameters, thereby increasing the complexity of the training for the fuzzy integrators. Simulation results show the effectiveness of the proposed approach in comparison with other researchers.

1 Introduction

The analysis of the time series consists of a (usually mathematical) description of the movements that compose it, then build models using movements to explain the structure and predict the evolution of a variable over time [1, 2]. The fundamental procedure for the analysis of a time series as described below:

J. Soto · P. Melin (✉)
Tijuana Institute of Technology, Tijuana, Mexico
e-mail: pmelin@tectijuana.mx

J. Soto
e-mail: jesvega83@gmail.com

1. Collecting data of the time series, trying to ensure that these data are reliable.
2. Representing the time series qualitatively noting the presence of long-term trends, cyclical variations and seasonal variations.
3. Plot a graph or trend line length and obtain the appropriate trend values using method of least squares.
4. When seasonal variations are present, obtained these and adjust the data rate to these seasonal variations (i.e. data seasonally).
5. Adjust the seasonally adjusted trend.
6. Represent cyclical variations obtained in step 5.
7. Combining the results of steps 1–6 and any other useful information to make a prediction (if desired) and if possible discuss the sources of error and their magnitude.

Therefore the above ideas can assist in the important problem of prediction in the time series. Along with common sense, experience, skill and judgment of the researcher, such mathematical analysis can, however, be of value for predicting the short, medium and long term.

The Mexican Stock Exchange (MSE), the second largest exchange in Latin America and home to some of the world's leading companies, have unveiled new and innovative trading rules and practices to the international financial community that were created to streamline market access to the Exchange. Senior exchange leadership, including Luis Tellez, President and CEO of MSE Group, hosted an event, "Connect and Trade Mexico," in New York City to discuss the recent improvements and the benefits for U.S. Institutional investors, including brokers, hedge funds, high frequency traders and other professional market participants [3].

Genetic algorithms are adaptive methods, which may be used to solve search and optimization problems. They are based on the genetic process of living organisms. Over generations, the populations evolve in line with the nature of the principles of natural selection and survival of the fittest, postulated by Darwin, in imitation of this process; genetic algorithms are capable of creating solutions to real world problems. The evolution of these solutions to optimal values of the problem depends largely on the proper coding of them. The basic principles of genetic algorithms were established by Holland [4, 5] and are well described in texts Goldberg and Kalyanmoy [6], Davis [7] and Michalewicz [8]. The evolutionary modeling of fuzzy logic system can be considered as an optimization process where a part or all the fuzzy system parameters constitute a search spaces of model operational (our case), cognitive and structural.

This paper reports the results of the simulations, in which optimization of the interval type-2 fuzzy integrators in ensembles of ANFIS models for the prediction time series (case: Mexican Stock Exchange), where the results for each ANFIS are evaluated by the method of the root mean square error (RMSE). For the integration of the results of each modular in the ensemble of ANFIS we used the following integration methods: interval type-2 fuzzy systems of Mamdani kind used with two and three membership functions.

The selection of the time series for the simulations was based on the fact that these time series are widely quoted in the literature by different researchers [9, 10], which allows to compare results with other approaches such as neural networks and linear regression.

In the next section we describe the background and basic concepts of ANFIS model, Ensemble learning, Interval type-2 fuzzy systems, Genetic Algorithms and MSE time series. Section 3 presents the proposed architecture of genetic optimization of interval type-2 fuzzy integrators in ensembles of ANFIS models for the time series prediction. Section 4 presents the simulations and the results obtained with different methods of integration that are used in this work. Section 5 presents the conclusions.

2 Background and Basic Concepts

This section presents the basic concepts of ANFIS, Ensemble learning, Interval type-2 fuzzy logic, and Genetic Algorithms.

2.1 ANFIS Models

There have been proposed systems that have achieved fully the combination of fuzzy systems with neural networks, one of the most intelligent hybrid systems is the ANFIS (Adaptive Neuro Fuzzy Inference System method) as referred to by Jang [11, 12] (Fig. 1), which is a method for creating the rule base of a fuzzy system, using the algorithm of backpropagation training from the data collection process. Its architecture is functionally equivalent to a fuzzy inference system of Takagi and Sugeno [13, 14].

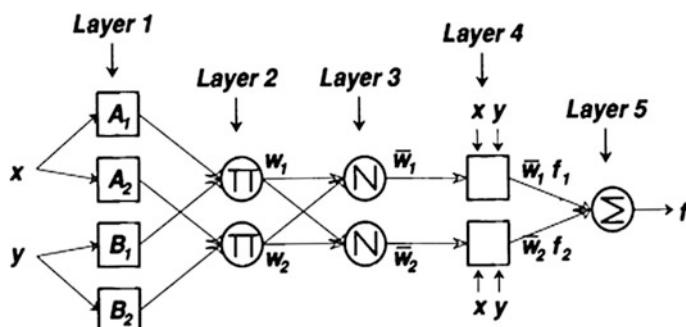


Fig. 1 ANFIS architecture

The basic learning rule of ANFIS is the gradient descent backpropagation, which calculates the error rates (defined as the derivative of the squared error for each output node) recursively from the output to the input nodes.

As a result we have a hybrid learning algorithm, which combines the gradient descent and least-squares estimation. More specifically in, the forward step of the hybrid learning algorithm, functional signals (output nodes) are processed towards layer 4 and the parameters of consequence are identified by least squares. In the backward step the premise parameters are updated by gradient descent.

2.2 Ensemble Learning

The Ensemble consists of a learning paradigm where multiple component learners are trained for a same task, and the predictions of the component learners are combined for dealing with future instances [4, 15, 16]. Since an Ensemble is often more accurate than its component learners, such a paradigm has become a hot topic in recent years and has already been successfully applied [17] to optical character recognition, face recognition, scientific image analysis, medical diagnosis.

2.3 Interval Type-2 Fuzzy Logic

Type-2 fuzzy sets are used to model uncertainty and imprecision; originally they were proposed by Zadeh [18, 19] and they are essentially “fuzzy–fuzzy” sets in which the membership degrees are type-1 fuzzy sets [20–22] (Fig. 2).

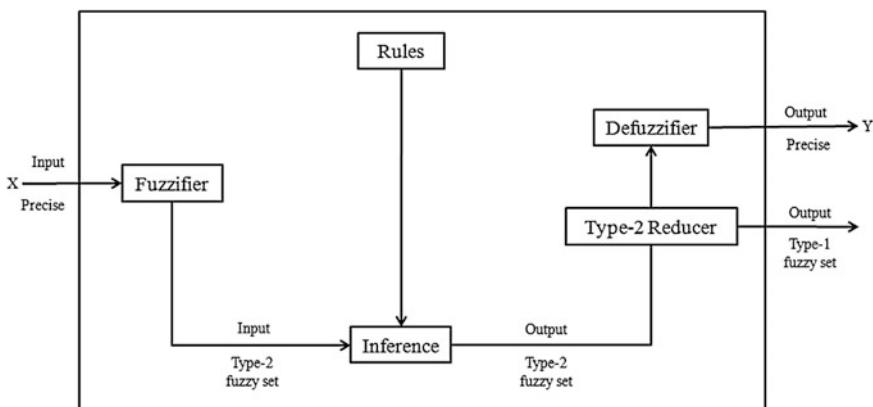
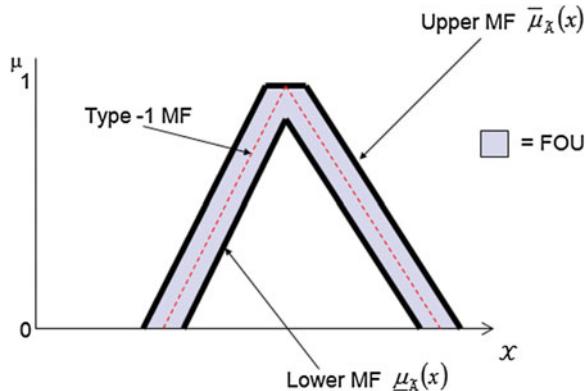


Fig. 2 Basic structure of the interval type-2 fuzzy logic system

Fig. 3 Interval type-2 membership function



The basic structure of a type-2 fuzzy system implements a nonlinear mapping of input to output space. This mapping is achieved through a set of type-2 if-then fuzzy rules, each of which describes the local behavior of the mapping.

The uncertainty is represented by a region called footprint of uncertainty (FOU). When; we have an interval type-2 membership function $\mu_{\bar{A}}(x, u) = 1, \forall u \in l_x \subseteq [0, 1]$ [20, 23–26] (Fig. 3).

The uniform shading for the FOU represents the entire interval type-2 fuzzy set and it can be described in terms of an upper membership function $\mu_{\bar{A}}(x)$ and a lower membership function $\mu_{\underline{A}}(x)$.

A fuzzy logic systems (FLS) described using at least one type-2 fuzzy set is called a type-2 FLS. Type-1 FLSs are unable to directly handle rule uncertainties, because they use type-1 fuzzy sets that are certain [27–29]. On the other hand, type-2 FLSs are very useful in circumstances where it is difficult to determine an exact certainty value, and there are measurement uncertainties.

2.4 Genetic Algorithms

Genetic Algorithms (GAs) are adaptive heuristic search algorithms based on the evolutionary ideas of natural selection and the genetic process [30–36]. The basic principles of GAs were first proposed by John Holland in 1975, inspired by the mechanism of natural selection, where stronger individuals are likely the winners in a competing environment [24]. GAs assumes that the potential solution of any problems an individual and can be represented by a set of parameters [33]. These parameters are added as the genes (individuals) of a chromosome and can be structured by string of values in binary or real form. A positive value, generally known as a fitness value, is used to reflect the degree of “goodness” of the

chromosome for the problem which would be highly related with its objective value. The pseudo code of a GAs is as follows:

1. Start with a randomly generated population of n individuals (candidate a solutions to problem).
2. Calculate the fitness of each individual in the problem.
3. Repeat the following steps until n offspring have been created:
 - a. Select a pair of parent individual from the concurrent population, the probability of selection being an increasing function of fitness. Selection is done with replacement, meaning that the same individual can be selected more than once so that the same individuals can be selected more than once to become a parent.
 - b. With the probability (crossover rate), perform crossover to the pair at a randomly chosen point to form two offspring.
 - c. Mutate the two offspring at each locus with probability (mutate rate), and place the resulting individuals in the new population.
4. Replace the current population with the new population.
5. Go to step 2.

The simple procedure just describe above is the basic one for most applications of GAs found in the literature.

2.5 Mexican Stock Exchange Time Series

MSE Group is a fully integrated Exchange Group that operates cash, listed derivatives and OTC markets for multiple asset classes, including equities, fixed income and exchange traded funds, as well as custody, clearing and settlement facilities and data products for the local and international financial community.

MSE is the second largest stock exchange in Latin America with a total market capitalization of over US\$360 billion. The Exchange is home to some of the most recognizable and profitable global corporations, including: beverage giant Modelo Group, whose brands include Corona Extra and Pacifico; America Mobil, one of the largest telecommunications companies in the world; CEMEX, the world's biggest building materials supplier; and Televisa, the largest media company in the Spanish-speaking world, among many others [3]. In addition, MexDer (the Mexican Derivatives Exchange) is also part of MSE Group and is the leading marketplace for trading benchmark Mexican derivatives products. The MSE time series we are using 800 pair data (Fig. 4) that correspond from period of 11/09/2005 to 01/15/09 [37] of the IPC (which stands for “Indice de Precios y Cotizaciones”) is the broadest indicator of the MSE overall performance.

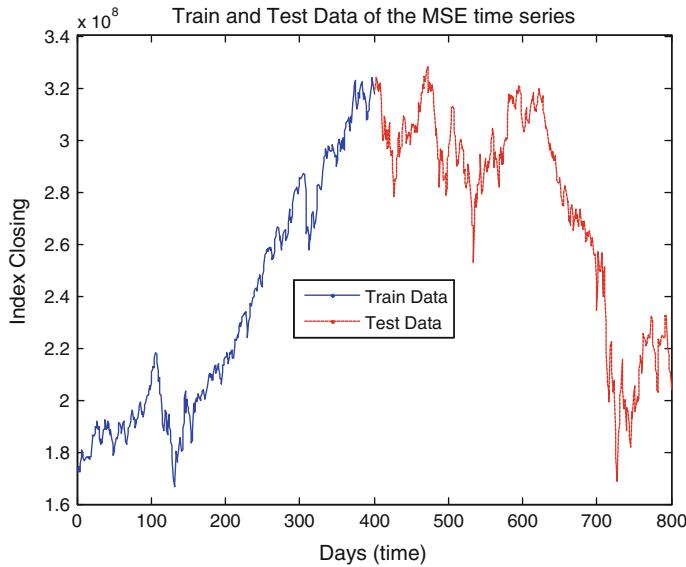


Fig. 4 Mexican stock exchange time series

3 General Architecture of the Proposed Method

The proposed method combines the ensemble of ANFIS models and the use of interval type-2 fuzzy systems as response integrators (Fig. 5).

This architecture is divided into 4 sections, where the first phase represents the data base to simulate in the Ensemble [4] of ANFIS, which in this case is the historical data of the Mexican Stock Exchange [37] time series. From the MSE time series we used 800 pairs of data points (Fig. 4), similar to [9, 10, 38].

We predict $x(t)$ from three past (delays) values of the time series, that is, $x(t - 18)$, $x(t - 12)$, and $x(t - 6)$. Therefore the format of the training and checking data is:

$$[x(t - 18), x(t - 12), x(t - 6); x(t)] \quad (1)$$

where $t = 19-818$ and $x(t)$ is the desired prediction of the time series.

In the second phase, training (the first 400 pairs of data are used to train the ANFIS) and validation (the second 400 pairs of data are used to validate the ANFIS models) is performed sequentially in each ANFIS model, where the number of ANFIS to be used can be from 1 to n depending on what the user wants to test, but in this case we are dealing with a set of 3 ANFIS in the Ensemble. Therefore each ANFIS model has three inputs variables ($x(t - 18), x(t - 12), x(t - 6)$) and one output variable ($x(t)$) is the desired prediction.

In the fourth phase we integrate the overall results of each Ensemble of ANFIS (ANFIS 1, ANFIS 2 and ANFIS 3) models, and such integration will be done the

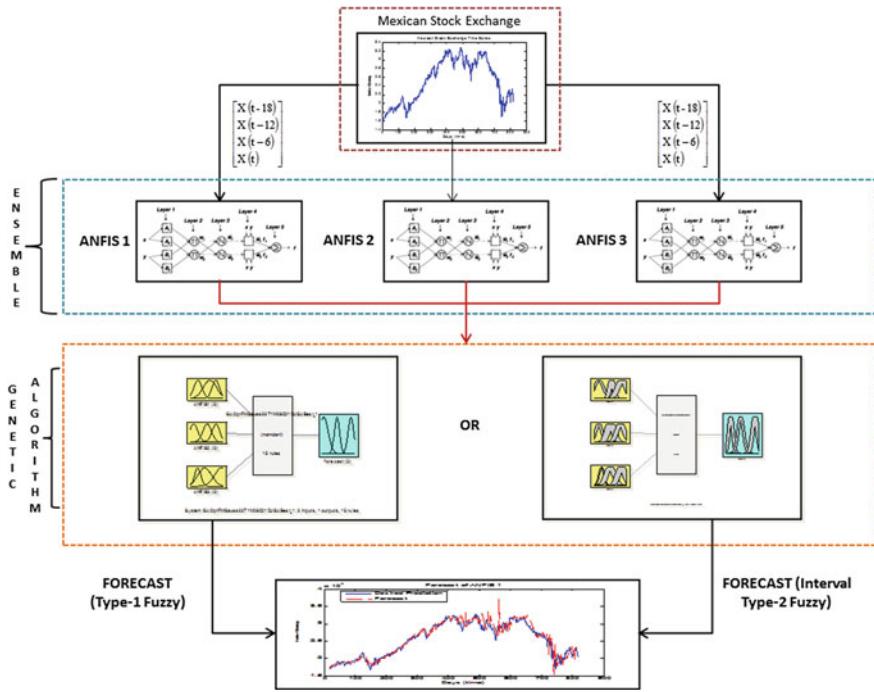


Fig. 5 The architecture of the proposed model

interval type-2 fuzzy integrators of Mamdani type, but each fuzzy integrators will optimized (GAs) of the MFs parameters. Finally the forecast output determined by the proposed architecture is obtained and it is compared with desired prediction.

4 Simulations Results

This section presents the results obtained through experiments on the architecture of genetic optimization of type-2 fuzzy integrators in ensembles of ANFIS models for the time series prediction, which show the performance that was obtained from each experiment to simulate the Mexican Stock Exchange time series.

4.1 Design of the Fuzzy Integrators

4.1.1 Interval Type-2 Fuzzy Integrators (Used Two MFs)

The design of the interval type-2 fuzzy inference system integrator is of Mamdani type and has 3 inputs (ANFIS1, ANFIS2 and ANFIS3 predictions) and 1 output

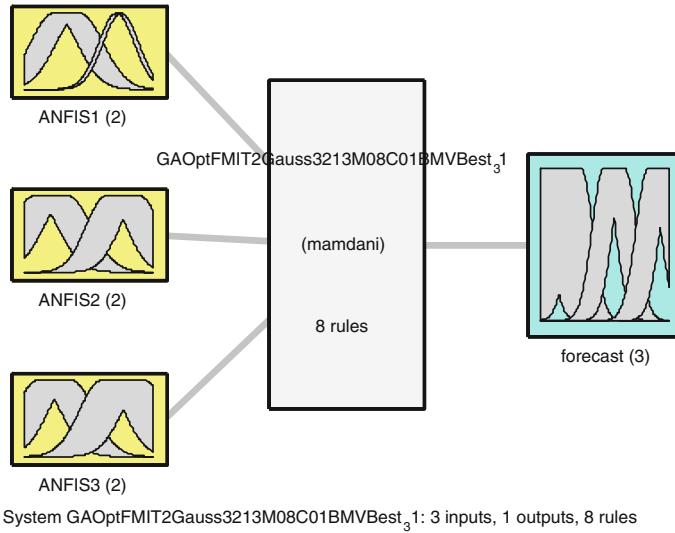


Fig. 6 Structure of the interval type-2FIS integrator (used two Gaussian MFs)

(forecast) variables, so each input will be assigned two MFs with linguistic labels “Small and Large” and the output will be assigned 3 MFs with linguistic labels “OutANFIS1, OutANFIS2 and OutANFIS3” (Fig. 6) and have 8 rules if-then.

In the interval type-2 fuzzy inference system integrator we used different MFs (Gaussian “igaussmftype2”, Generalized “igbelltype2” Bell, and Triangular “itri-type2”) to observe the behavior of each of them and determine which one provides better forecast of the time series.

4.1.2 Interval Type-2 Fuzzy Integrators (Used Three MFs)

The design (structure) of the interval type-2 fuzzy inference system integrator is of Mamdani type and has 3 inputs (ANFIS1, ANFIS2 and ANFIS3 predictions) and 1 output (forecast), so each input will be assigned three MFs with linguistic labels “Small, Middle and Large” and the output will be assigned three MFs with linguistic labels “OutANFIS1, OutANFIS2 and OutANFIS3” (Fig. 8) and have 15 rules if-then (Fig. 7).

In the interval type-2 fuzzy inference system integrator we used different MFs (Gaussian “igaussmftype2”, Generalized “igbelltype2” Bell, and Triangular “itri-type2”) to observe the behavior of each of them and determine which one provides better forecast of the time series.

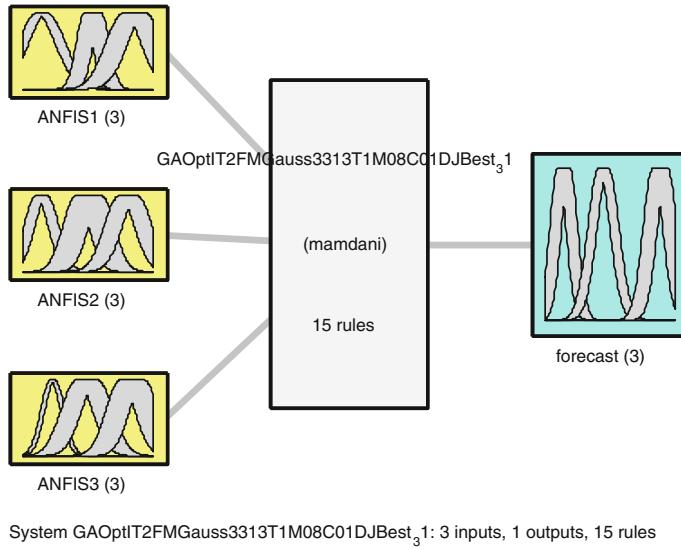


Fig. 7 Structure of the interval type-2FIS integrator (used three Gaussian MFs)

4.1.3 Design the Rules of the Fuzzy Integrators

The design rules if-then for the fuzzy inference system depends on the number of membership functions used in each input variable using the system (e.g. our fuzzy inference system uses 3 input variables which each entry contains three membership functions, therefore the total number of possible combinations for the fuzzy-rules is 27 (e.g. $3 \times 3 \times 3 = 27$)), but we used 15 fuzzy-rules for the experiments (see Fig. 8) because the performance is better and minimized the prediction error of the Dow Jones time series.

1. If (ANFIS1 is Small) and (ANFIS2 is Small) and (ANFIS3 is Small) then (forecast is OutANFIS1) (1)
2. If (ANFIS1 is Small) and (ANFIS2 is Small) and (ANFIS3 is Middle) then (forecast is OutANFIS1) (1)
3. If (ANFIS1 is Small) and (ANFIS2 is Small) and (ANFIS3 is Large) then (forecast is OutANFIS1) (1)
4. If (ANFIS1 is Middle) and (ANFIS2 is Small) and (ANFIS3 is Small) then (forecast is OutANFIS1) (1)
5. If (ANFIS1 is Large) and (ANFIS2 is Small) and (ANFIS3 is Small) then (forecast is OutANFIS1) (1)
6. If (ANFIS1 is Middle) and (ANFIS2 is Small) and (ANFIS3 is Middle) then (forecast is OutANFIS2) (1)
7. If (ANFIS1 is Middle) and (ANFIS2 is Middle) and (ANFIS3 is Middle) then (forecast is OutANFIS2) (1)
8. If (ANFIS1 is Middle) and (ANFIS2 is Middle) and (ANFIS3 is Large) then (forecast is OutANFIS2) (1)
9. If (ANFIS1 is Small) and (ANFIS2 is Middle) and (ANFIS3 is Middle) then (forecast is OutANFIS2) (1)
10. If (ANFIS1 is Large) and (ANFIS2 is Middle) and (ANFIS3 is Middle) then (forecast is OutANFIS2) (1)
11. If (ANFIS1 is Large) and (ANFIS2 is Large) and (ANFIS3 is Small) then (forecast is OutANFIS3) (1)
12. If (ANFIS1 is Large) and (ANFIS2 is Large) and (ANFIS3 is Middle) then (forecast is OutANFIS3) (1)
13. If (ANFIS1 is Large) and (ANFIS2 is Large) and (ANFIS3 is Large) then (forecast is OutANFIS3) (1)
14. If (ANFIS1 is Small) and (ANFIS2 is Large) and (ANFIS3 is Large) then (forecast is OutANFIS3) (1)
15. If (ANFIS1 is Middle) and (ANFIS2 is Large) and (ANFIS3 is Large) then (forecast is OutANFIS3) (1)

Fig. 8 Rules of interval type-2 FIS

4.2 Design the Representation of the Chromosome of Genetic Algorithms for Optimizer the MFs in the Fuzzy Integrators

The GAs are used to optimize the parameters values of the MFs in each interval type-2 fuzzy integrators. The representation of GAs is of Real-Values and the chromosome size will depend of the MFs that are used in each design of the interval type-2 fuzzy inference system integrators.

The objective function is defined to minimize the prediction error as follows:

$$f(t) = \sqrt{\frac{\sum_{t=1}^n (a_t - p_t)^2}{n}} \quad (2)$$

where a , corresponds to the real data of the time series, p corresponds to the output of each fuzzy integrator, t is de sequence time series, and n is the number of data points of time series.

In Figs. 9 and 10 the general representation of the chromosome that represents the utilized fuzzy membership functions is illustrated. In these figures, the first phase represented each input/output variables of the fuzzy systems, the second phase represents the MFs containing each input (MFs1 "Small", MFs2 "Middle" and MFs3 "Large") and output (MFs1 "OutANFIS1", MFs2 "OutANFIS2" and MFs3 "OutANFIS1") variables of the fuzzy systems, the third phase represents the MFs parameter "PL = Lower Parameter" where $PL_1 \dots PL_N$ (0.02...3.15) are the size

Fig. 9 Representation of the chromosome for the optimization of the interval type-2 fuzzy integrators (used two Gaussian MFs)

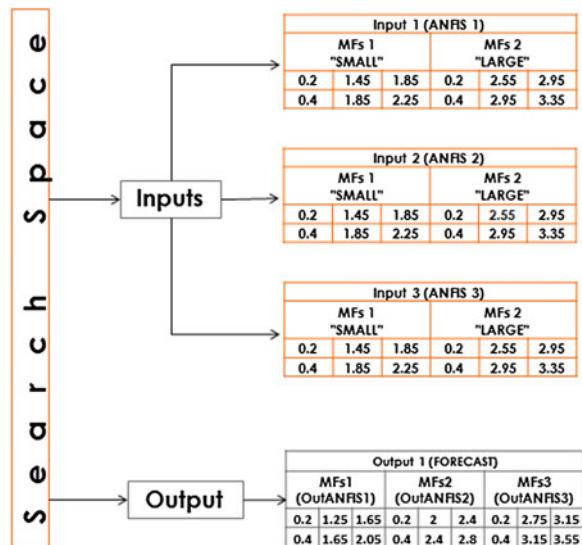
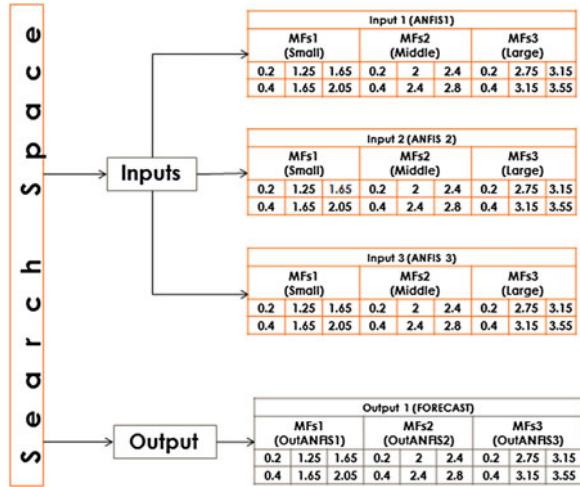


Fig. 10 Representation of the chromosome for the optimization of the interval type-2 fuzzy integrators (used three Gaussian MFs)



parameter of the MFs, the fourth phase represent the MFs parameter “PU = Upper Parameter” $PU_1 \dots PU_N$ (0.4…3.55) are the size parameter of the MFs that corresponds to each input and output. The number of parameters varies according to the kind of MFs of the interval type-2 fuzzy system (e.g. three parameter are needed to represent a Gaussian “igaussmtype2” MF’s are “sigma, mean1 and mean2”) illustrated in Figs. 9 and 10.

Therefore the number of parameters that each fuzzy inference system integrator depends of the MFs type assigned to each input and output variables.

The GAs used the following parameters for the experiments: 100 individuals or genes, 100 generations and 31 iterations (running the GAs), the selection method are the stochastic universal sampling, the percentage of crossover or recombine is 0.8 and the mutation is 0.1. Those are fundamentals parameters for test the performances of the GAs.

4.3 Result Obtained for the Genetic Optimization of Interval Type-2 Fuzzy Integration (Using Two Gaussian MFs)

In the design of the interval type-2 fuzzy integrator we have three input variables and one output variable, so each variable input will have two MFs and the variable output will have three MFs. Therefore the number of parameters that one used in the representation of the chromosome is 27, because igaussmtype2 MFs used three parameters (Variance, Mean1 and Mean2) to their representation in the interval type-2 fuzzy systems. The results obtained for the optimization of the igaussmtype2 MFs with GAs are the following: the parameters obtained with the GAs for the type-2 fuzzy MFs “igaussmtype2” (Fig. 11). The forecast data (Fig. 12) is generated

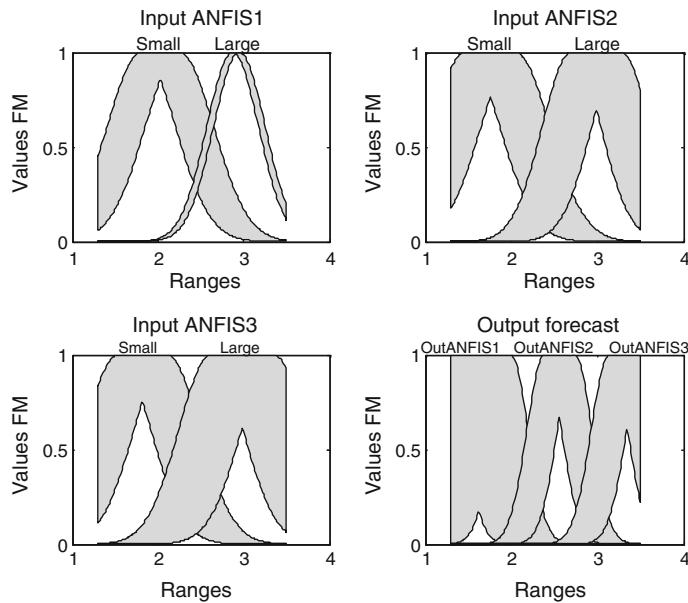


Fig. 11 Plot of the optimization of the memberships functions (input and output) parameters with the GAs

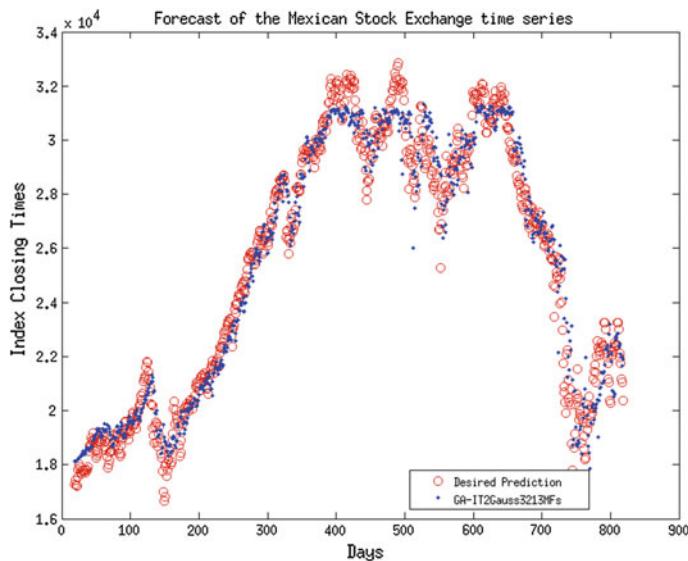


Fig. 12 Plot of the forecast generated by the genetic optimization of type-2 fuzzy integrators

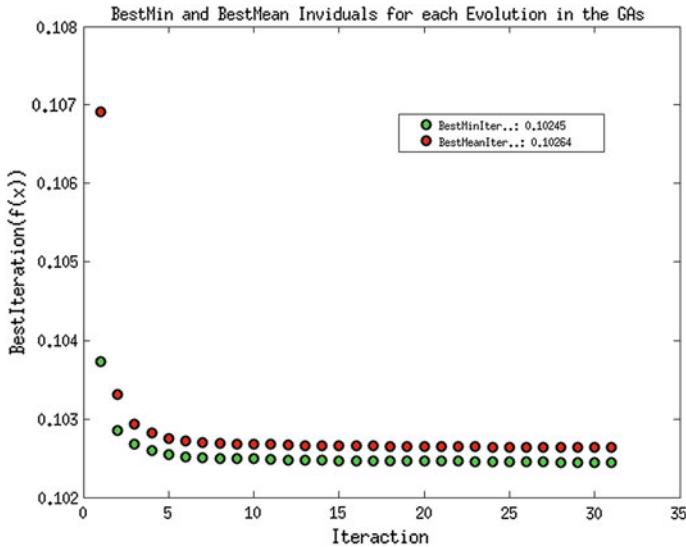


Fig. 13 Plot of the evolution error generated by GAs

by optimization of the interval type-2 fuzzy integrators. Therefore the comparison of evolution error (RMSE) between best and average (Fig. 13) obtained with the GAs for this integration are 0.10245 and 0.10264.

4.4 Result Obtained for the Genetic Optimization of Interval Type-2 Fuzzy Integration (Using Three Triangular MFs)

In the design of the interval type-2 fuzzy integrator we have three input variables and one output variable, so each inputs and output variables will have three MFs. Therefore the number of parameters of the MFs that are used in the representation of the chromosome is 72, because “igbelltype2” MFs used six’s parameters (a_1, b_1, c_1, a_2, b_2 and c_2) to their representation in the interval type-2 fuzzy systems. The results obtained for the optimization of the “itritype2” MFs with GAs are the following: the parameters obtained with the GAs for the type-2 fuzzy MFs “itritype2” (Fig. 14). The forecast data (Fig. 15) is generated by optimization of the interval type-2 fuzzy integrators. Therefore the comparison of evolution error (RMSE) between best and average (Fig. 16) obtained with the GAs for this integration are 0.1004 and 0.10071.

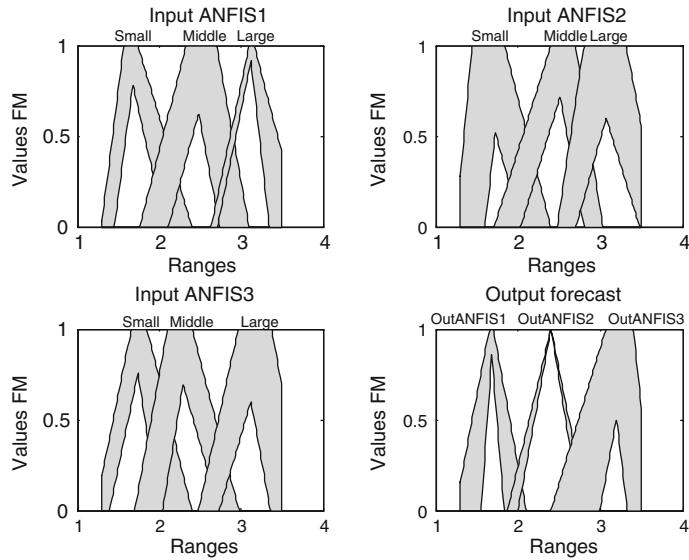


Fig. 14 Optimization of the memberships functions (input and output) parameters with the GAs

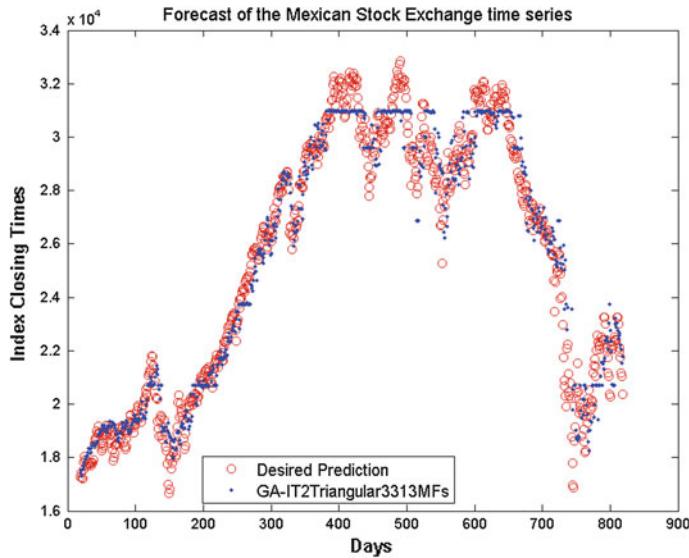


Fig. 15 Forecast generated by the genetic optimization of the type-2 fuzzy integrators

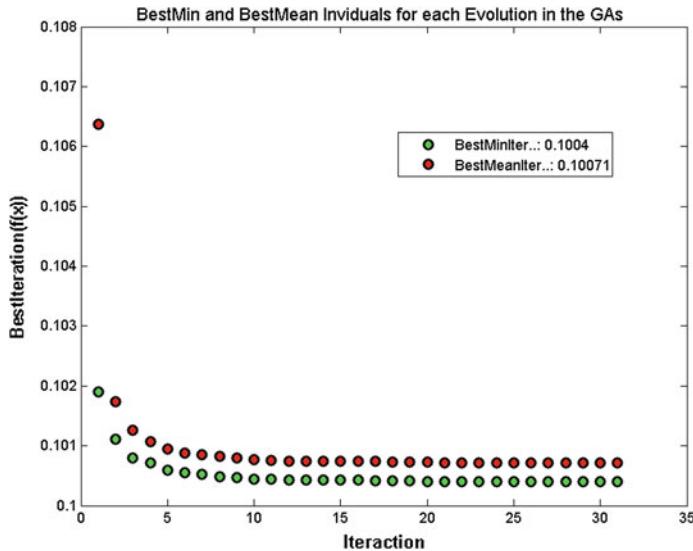


Fig. 16 Plot of the evolution error generated by GAs

4.5 Results and Comparison

Table 1 shows the results of 30 experiments that were made with the genetic optimization of interval type-2 fuzzy integrators in ensembles of ANFIS models for the time series prediction. This Table shows the comparison result (best, mean and standard deviation “STD”) of the prediction error for the optimization of the interval type-2 fuzzy integrators (used two and three MFs for each integrator). The best, averages and standard deviation prediction errors for the interval Type-2 fuzzy integrator (used two MFs) are using Generalized Bell “igbelltype2” MFs, which obtained a prediction error are 0.099952 (best), 0.100466 (mean) and 0.000596 (STD). The best, averages and standard deviation prediction errors for the interval Type-2 fuzzy integrator (used three MFs) are using Generalized Bell “igbelltype2” MFs, which obtained a prediction error are 0.097165 (best), 0.097347 (mean) and 0.000392 (STD). Therefore when we used interval type-2 fuzzy integrator (using three MFs) is better than interval type-2 fuzzy integrator (using two MFs) in because the best prediction error is 0.097165.

Table 2 shows the results of 30 experiments that were made with the genetic optimization of interval type-2 and type-1 fuzzy integrators in ensembles of ANFIS models for the time series prediction. This table shows the comparison result (best, mean and standard deviation “STD”) results of the prediction error of the optimization of the interval type-2 and type-1 fuzzy integrators (used two MFs for each integrator). The best, averages and standard deviation prediction errors for the Type-1 fuzzy integrator are using Generalized Bell MFs, which obtained a prediction error are 0.101809 (best), 0.102272 (mean) and 0.000752 (STD). The best,

Table 1 Results obtained for the optimization of the interval type-2 fuzzy integrators for the prediction error of Mexican stock exchange

Prediction error (RMSE)	Interval type-2 fuzzy integrator (used 2 MFs)			Interval type-2 fuzzy integrator (used 3 MFs)		
	igaussmtype2	igbelltype2	itrtype2	igaussmtype2	igbelltype2	itrtype2
Best	0.102447	0.099952	0.122478	0.105239	0.097165	0.100398
Average	0.102536	0.100466	0.122575	0.105494	0.097347	0.100523
STD	0.000237	0.000596	0.000282	0.000283	0.000392	0.000296
Parameter	27	54	54	36	72	72
Time (HH:MM:SS)	12:15:13	24:25:03	26:33:17	24:38:26	32:10:36	40:33:17

Table 2 Results obtained for the optimization of the interval type-2 and type 1 fuzzy integrators for the prediction error of Mexican stock exchange

Prediction error (RMSE)	Type-1 fuzzy integrator (used two MFs)			Interval type-2 fuzzy integrator (used two MFs)		
	Gaussian	GBell	Triangular	igaussmtype2	igbelltype2	itrtype2
Best	0.104759	0.101809	0.167474	0.102447	0.099952	0.122478
Average	0.106320	0.102272	0.167476	0.102536	0.100466	0.122575
STD	0.001549	0.000752	0.000013	0.000237	0.000596	0.000282
Parameter	18	27	27	27	54	54
Time (HH:MM:SS)	02:27:44	01:25:36	04:15:12	12:15:13	24:25:03	26:33:17

averages and standard deviation prediction errors for the interval Type-2 fuzzy integrator are using Generalized Bell “igbelltype2” MFs, which obtained a prediction error are 0.099952 (best), 0.100466 (mean) and 0.000596 (STD). Therefore when we used interval type-2 fuzzy integrator is better than type-1 fuzzy integrator because the best prediction error is 0.099952.

5 Conclusion

We have presented simulation results of the Mexican Stock Exchange time series (forecasting) with different hybrid intelligent approaches. The best result with the optimization of the interval type-2 fuzzy integrator (using three MFs) is better than interval type-2 fuzzy integrators (using two MFs), because the prediction error is 0.097165 (as shown Table 1), in most of the experiments that were performed with the proposed architecture of ensembles of ANFIS.

In the comparison of results between of the interval type-2 and type-1 fuzzy integrators (both using two MFs) the better prediction error is obtained when we used interval type-2 fuzzy integrator because the prediction error is 0.099952 (as shown Table 2).

We conclude that the results obtained with the optimization of the interval type-2 fuzzy integrators in ensembles of ANFIS models for the time series prediction (case Mexican Stock Exchange) is good, since we achieved 98 % of accuracy with the Mexican Stock Exchange time series.

Therefore the proposal offers efficient results in the prediction of such time series, which can help us, make decisions and avoid unexpected events in the future.

References

1. Brocklebank, J.C., Dickey, D.A.: SAS for Forecasting Series, pp. 6–140. SAS Institute Inc. Cary, NC, USA, (2003)
2. Brockwell, P.D., Richard, A.D.: Introduction to Time Series and Forecasting, pp. 1–219. Springer, New York (2002)
3. Cervantes, M., Montoya, M. Cueto, D.C.: Momentum Effect on the Mexican Stock Exchange, pp. 1–20. Social Science Electronic Publishing (2014)
4. Holland, J.H.: Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor (1975)
5. Holland, J.H.: Outline for a logical theory of adaptive systems. *J. Assoc. Comput. Mach.* **3**, 297–314 (1962)
6. Goldberg, D.E., Kalyanmoy, D.: A comparative analysis of selection schemes used in genetic algorithms. In: Gregory, J.E.R. (ed.) *Foundations of Genetic Algorithms*, pp. 69–93. Morgan Kaufmann Publishers, San Mateo, California (1991)
7. Goldberg, D.E., Korb, B., Kalyanmoy, D.: Messy genetic algorithms: motivation, analysis, and first results. *Complex Syst.* **3**, 493–530 (1989)
8. Lawrence, D.M.: *Handbook of Genetic Algorithms*. Van Nostrand Reinhold (1991)
9. Melin, P., Soto, J., Castillo, O., Soria, J.: A new approach for time series prediction using ensembles of ANFIS models. *Expert Syst. Appl.* **39**(3), 3494–3506 (2012)
10. Pulido, M., Melin, P., Castillo, O.: Particle swarm optimization of ensemble neural networks with fuzzy aggregation for time series prediction of the Mexican Stock Exchange. *Inf. Sci.* **280** (1), 188–204 (2014)
11. Jang, J.S.R.: ANFIS: Adaptive-network-based fuzzy inference systems. In: *IEEE Transaction on Systems, Man, and Cybernetics*, vol. 23, pp. 665–685 (1992)
12. Jang, J.S.R.: Rule extraction using generalized neural networks. In: *Proceedings of the 4th IFSA Wolrd Congress*, pp. 82–86 (1991)
13. Takagi, T., Sugeno, M.: Derivation of fuzzy control rules from human operation control actions. In: *Proceedings of the IFAC Symposium on Fuzzy Information, Knowledge Representation and Decision Analysis*, pp. 55–60 (1983)
14. Takagi, T., Sugeno, M.: Fuzzy identification of systems and its applications to modeling and control. In: *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 15, pp. 116–132 (1985)
15. Sharkey, A.: *Combining Artificial Neural Nets: Ensemble and Modular Multi-net Systems*. Springer, London (1999)
16. Sollich, P., Krogh, A.: Learning with ensembles: how over-fitting can be useful. In: Touretzky, D.S., Mozer, M.C., Hasselmo, M.E. (eds.) *Advances in Neural Information Processing Systems*, pp. 190–196. MIT Press, Cambridge, MA (1996)
17. Zhou, Z.H., Wu, J., Tang, W.: Ensembling neural networks: many could be better than all. *Artif. Intell.* **137**(1–2), 239–263 (2002)
18. Zadeh, L.A.: Fuzzy logic. *Computer* **1**(4), 83–93 (1988)

19. Zadeh, L.A.: Fuzzy logic = computing with words. *IEEE Trans. Fuzzy Syst.* **4**(2), 103 (1996)
20. Castro, J.R., Castillo, O., Melin, P. Rodriguez, A.: A Hybrid Learning Algorithm for Interval Type-2 Fuzzy Neural Networks: The Case of Time Series Prediction, vol. 15a, pp. 363–386. Springer, Berlin, Heidelberg (2008)
21. Jang J.S.R.: Fuzzy modeling using generalized neural networks and Kalman filter algorithm. In: Proceedings of the Ninth National Conference on Artificial Intelligence (AAAI-91), pp. 762–767 (1991)
22. Melin, P., Mendoza, O., Castillo, O.: An improved method for edge detection based on interval type-2 fuzzy logic. *Expert Syst. Appl.* **37**(12), 8527–8535 (2010)
23. Castillo, O., Melin, P.: Optimization of type-2 fuzzy systems based on bio-inspired methods: a concise review. *Appl. Soft Comput.* **12**(4), 1267–1278 (2012)
24. Castillo, O., Melin, P.: Soft Computing for Control of Non-Linear Dynamical Systems. Springer, Heidelberg (2001)
25. Castro, J.R., Castillo, O., Martínez, L.G.: Interval type-2 fuzzy logic toolbox. *Eng. Lett.* **15**(1), 89–98 (2007)
26. Castro, J.R., Castillo, O., Melin, P. Rodríguez, A.: Hybrid Learning Algorithm for Interval Type-2 Fuzzy Neural Networks, pp. 157–162. GrC (2007)
27. Mendel, J.M.: Why we need type-2 fuzzy logic systems. Article is provided courtesy of Prentice Hall, By Jerry Mendel, May 11, 2001
28. Mendel, J.M.: Uncertain Rule-Based Fuzzy Logic Systems: Introduction and New Directions, pp. 25–200. Prentice Hall, New Jersey (2000)
29. Mendel, J.M., Mouzouris, G.C.: Type-2 fuzzy logic systems. *IEEE Trans. Fuzzy Syst.* **7**, 643–658 (1999)
30. Chua, T.W., Tan, W.W.: Genetically evolved fuzzy rule-based classifiers and application to automotive classification. *Lect. Notes Comput. Sci.* **5361**, 101–110 (2008)
31. Cordon, O., Gomide, F., Herrera, F., Hoffmann, F., Magdalena, L.: Ten years of genetic fuzzy systems: current framework and new trends. *Fuzzy Sets Syst.* **141**, 5–31 (2004)
32. Cordon, O., Herrera, F., Hoffmann, F., Magdalena, L.: Genetic Fuzzy Systems: Evolutionary Tuning and Learning of Fuzzy Knowledge Bases. World Scientific, Singapore (2001)
33. Cordon, O., Herrera, F., Villar, P.: Analysis and guidelines to obtain a good uniform fuzzy partition granularity for fuzzy rule-based systems using simulated annealing. *Int. J. Approximate Reasoning* **25**, 187–215 (2000)
34. Eason, G., Noble, B., Sneddon, I.N.: On certain integrals of Lipschitz-Hankel type involving products of Bessel functions. *Phil. Trans. Roy. Soc. London* **A247**, 529–551 (1955)
35. EibenA, A.E., Smith, J.E.: Introduction to Evolutionary Computation, pp. 37–69. Springer, Berlin (2003)
36. Goldberg, D.E.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley Publishing Company, Boston (1989)
37. Mexico Bank Database: <http://www.banxico.org.mx> (2011)
38. Pulido, M., Mancilla, A., Melin, P.: An ensemble neural network architecture with fuzzy response integration for complex time series prediction. In: Evolutionary Design of Intelligent Systems in Modeling, Simulation and Control, pp. 85–110 (2009)

A New Proposal for a Granular Fuzzy C-Means Algorithm

Elid Rubio and Oscar Castillo

Abstract Fuzzy clustering algorithms are able to find the centroids and partition matrices, but are predominantly numerical, although each cluster prototype can be considered as a granule of information it continues to be a numeric value, in order to give a similar representation structure data. Granular theory and clustering algorithms can be combined to achieve this goal, resulting in granular prototypes and granular matrices of belonging and a more reflective data structure.

1 Introduction

Fuzzy Clustering algorithms [2, 5, 10–12, 14] are popular and widely used in different areas of research like pattern recognition [2], data mining [6], classification [8], image segmentation [16, 18], data analysis and modeling [3] among others, obtaining good results in these implementations. The popularity of this kind of algorithms is due to the fact that allow a datum to belong to different data clusters into a given data set, the main objective of the fuzzy clustering algorithms are find interesting patterns or group of data that share similar characteristics into a given data set.

In a general point of view the process of clustering is considered as a granular information process [4, 15], but the information granule is represented by prototypes (centers of clusters) and partition matrices (matrices of belonging) which are represented by numerical values. Due to this the data structure is not too reflective. In order to make the data structure more reflective, improvements of clustering algorithms like the FCM [2] and PCM [10, 11] in combination with Interval Type-2 Fuzzy logic techniques [9, 13] and of this combination arise the IT2FCM [7, 17]

E. Rubio · O. Castillo (✉)
Tijuana Institute of Technology, Tijuana, Mexico
e-mail: ocastillo@tectijuana.mx

E. Rubio
e-mail: elid.rubio@hotmail.com

and IT2PCM [19] algorithms, that are capable to create high order granules, due to that algorithms found upper and lower bounds of the interval to prototypes and partition matrices into a given data set.

In [1, 4, 15] a proposal of how to make Granular Clustering using the FCM algorithm is presented, and a way to create granular prototypes and granular partition matrices. Based on this work we present a new proposal of a granular fuzzy C-means algorithm.

This work is organized as follows. In Sect. 2 we show a brief overview of the Fuzzy C-Means algorithm, Sect. 3 described a new proposal of a Granular Fuzzy C-Means Algorithm presented in this paper, Sect. 4 shows the plots of the granular prototype and results with benchmark datasets, Sect. 5 contains the conclusions obtained during the elaboration of this work.

2 Fuzzy C-Means Algorithm

The FCM algorithm is a clustering unsupervised method widely used in data clustering, image segmentation, pattern recognition, etc.; this algorithm creates soft partitions where a datum can belong to different clusters with a different membership degree to each cluster of the dataset. This clustering method is an iterative algorithm, which uses the necessary condition to achieve the minimization of the objective function J_m represented by the following equation [2, 12]:

$$J_m(U, V, X) = \sum_{i=1}^c \sum_{k=1}^n \mu_i(x_k)^m \cdot d_{ik}^2 \quad (1)$$

where the variables in (1) represent the following:

- n the total number of patterns in a given data set
- c is the number of clusters, which can be found from 2 to $n - 1$
- X are data characteristics, where $X = \{x_1, x_2, \dots, x_n\} \subset R^s$
- V are the centers of the clusters, where $V = \{v_1, v_2, \dots, v_n\} \subset R^s$
- $U=\mu_{ij}$ is a fuzzy partition matrix, which contains the membership degree of each dataset x_j to each cluster v_i
- d_{ik}^2 is the Euclidean distance between each data x_k of the dataset and the centers v_i of clusters
- m is the weighting exponent

The corresponding centers of the clusters and membership degrees for each respective data to solve the optimization problem with the constraints in (1) are given by Eqs. (2) and (3), which provide an iterative procedure. The aim is to improve a sequence of fuzzy clusters until no further improvement in (1) can be performed [2, 12].

$$v_i = \frac{\sum_{k=1}^n \mu_i(x_k)^m \cdot x_k}{\sum_{k=1}^n \mu_i(x_k)^m} \quad (2)$$

$$\mu_i(x_k) = \left(\sum_{j=1}^c \left(\frac{d_{ik}}{d_{jk}} \right)^{\frac{2}{m-1}} \right)^{-1} \quad (3)$$

Equations (2) and (3) are an iterative optimization procedure. The aim is to improve a sequence of fuzzy clusters until no further improvement in $J_{m,\eta}(U, V, X)$ can be made. The Fuzzy C-Means algorithm consists of the following steps [2, 5, 12]:

1. Given a pre-selected number of clusters c and a chosen value for m , initialize the fuzzy partition matrix μ_{ij} of x_j belonging to cluster i such that:

$$\sum_{i=1}^c \mu_{ij} = 1 \quad (4)$$

2. Calculate the center of the fuzzy clusters, v_j for $j = 1, 2, \dots, c$ using Eq. (2).
3. Use Eq. (3) to update the fuzzy membership μ_{ij} .
4. If the improvement in $J_m(U, V, X)$ is less than a certain threshold (ϵ), then stop, otherwise go to step 2.

The FCM clustering is completed through a sequence of iterations, where we can start from a certain randomly initiated centroids of clusters or a certain randomly partition matrix, and iterate over the formulas (2) and (3) given above.

3 Proposal of Granular Fuzzy C-Means Algorithms

From a general point of view, fuzzy clustering is about forming information granules and revealing the structure in data. Fuzzy clusters are information granules capturing the data. Observing fuzzy clustering from a different point of view this kind of algorithms are providers of a certain granulation-degranulation mechanism.

Considering that clustering has been realized previously, we can express it in terms of cluster leading to a granular description of x to any datum x into a given dataset, the granular description mentioned above typically in fuzzy clustering algorithms is done by computing the membership degrees of x to the cluster prototypes found by the fuzzy clustering algorithm, and this process is considering a granulation phase. The degranulation is about data reconstruction on basis of the granular or internal representation (Fig. 1).

In the FCM algorithm the granulation-degranulation mechanisms can be described in the two phases mentioned below:

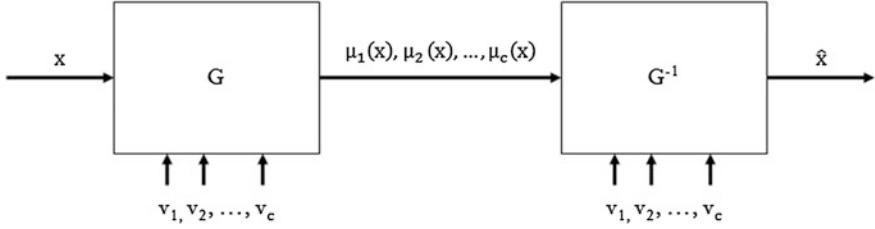


Fig. 1 Representation of the granulation-degranulation mechanisms

1. Granulation of the data x is made in terms of membership grades of the constructed information granules, the membership grades are computed by Eq. (3).
2. Degranulation provides a reconstruction of data in terms of the prototypes and membership grades computed by (2) and (3) respectively. Formally, the reconstruction of data is determined by solving the following optimization problem, with the minimization of the reconstruction error

$$\sum_{i=1}^c \mu_i(x_j) \cdot (v_i - \hat{x}_j)^2 \quad (5)$$

As a result, the equation of minimization error shown below is obtained:

$$\hat{x}_j = \frac{\sum_{i=1}^c \mu_i(x_j)^m \cdot v_i}{\sum_{i=1}^c \mu_i(x_j)^m} \quad (6)$$

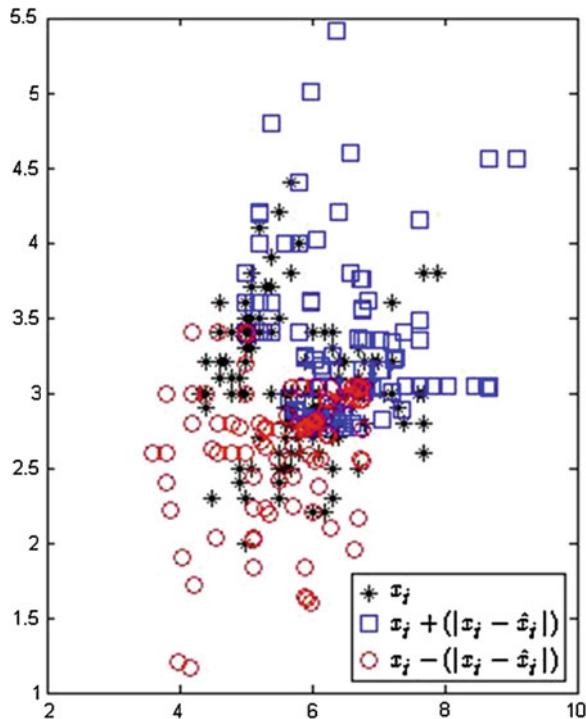
The reconstruction error becomes a function that takes into account the prototypes of the clusters and the matrices of belonging; Eq. (6) is very similar to Eq. (2) for prototype computing, observing this we can say that the reconstruction error is by finding x in Eq. (2) to compute the reconstruction error.

Based in the granulation-degranulation mechanism of the FCM algorithm, we propose a new Granular Fuzzy C-Means algorithm, which is different to that mentioned in [1, 4, 15]. This new proposal, like all algorithms that use granular computing is performed in two phases. The first phase is the granulation is a regular process of the FCM algorithm to obtain the membership degrees and prototypes of data, which are numerical values, to make this to granular membership degrees and granular prototypes. Now we proceed with the second phase, where the reconstruction error is computed with Eq. (6) and used to create two new data sets using the following equations:

$$x_j^+ = x_j + (|x_j - \hat{x}_j|) \quad (7)$$

$$x_j^- = x_j - (|x_j - \hat{x}_j|) \quad (8)$$

Fig. 2 Data distribution of the Iris flower data set and data distribution of the data set created by Eqs. (7) and (8)

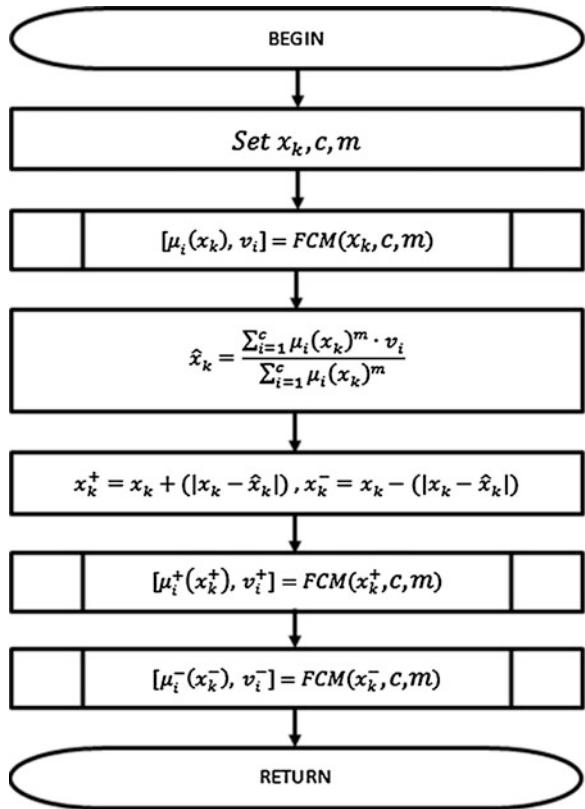


Equations (7) and (8) basically compute the upper and lower bound of the data in agreement with the reconstruction error, with the new data set created using addition and subtraction to original data, the difference between original data and reconstructed data. In Fig. 2 we can observe the distribution of the data set created with Eqs. (7) and (8).

We can obtain a granular prototype and granular membership degrees by making the grouping of the data set created with Eqs. (7) and (8) using the FCM algorithm, in Fig. 3 we can observe the process of the granular fuzzy C-means algorithm proposed and Fig. 4 show the block diagram of the FCM algorithm.

As we may observe in Fig. 3 the process of creating granular prototypes and granular membership degrees is performed by the execution of FCM algorithms over the data set created by Eqs. (7) and (8), this is due to the fact that the data created are affected by reconstruction error and make a upper and lower bound of data. In the next section we show the granular prototype found by the proposed Granular Fuzzy C-Means algorithm over some benchmark data sets.

Fig. 3 Blocks diagram of the proposed granular fuzzy c-means algorithm



4 Simulation

In order to observe, if the proposal is capable to create a granular prototype, we realized the clustering of benchmark datasets, and the benchmark dataset used to perform these tests are the following:

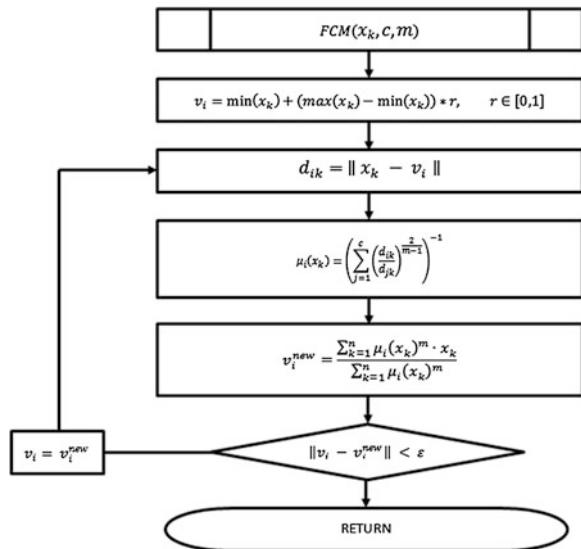
- Wine
- WDBC
- Iris Flower

Wine data set is composed by 3 classes, with 178 instances in total and 13 attributes the number of instances per class show below:

- Class 1: 59 instances
- Class 2: 71 instances
- Class 3: 48 instances

WDBC data set is composed by 2 classes, with 569 instances in total and 32 attributes, the number of instances per class show below:

Fig. 4 Blocks diagram of the original fuzzy c-means algorithm



- Class 1 (benign): 357 instances
- Class 2 (malignant): 212 instances

Iris Flower data set is composed by 3 classes, with 150 instances in total and 4 attributes the number of instances per class show below:

- Class 1 (Setosa): 50 instances
- Class 2 (Versicolour): 50 instances
- Class 3 (Virginica): 50 instances

Figures 5, 7 and 9 show the prototypes found by the FCM algorithm over the Wine, WDBC, and Iris Flower data sets respectively, and these prototypes are

Fig. 5 Prototype found by the FCM algorithm in the Wine dataset

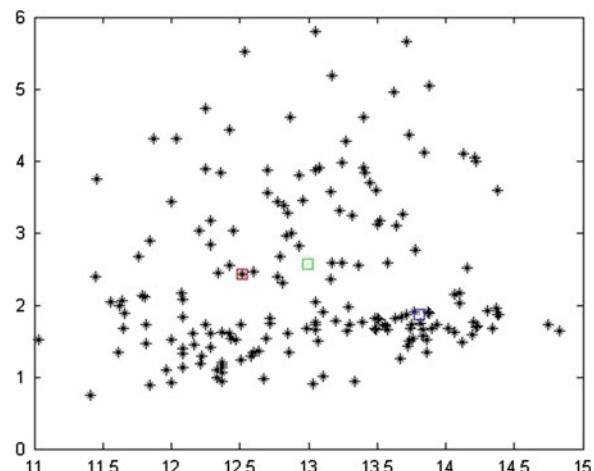


Fig. 6 Granular prototype found by the GFCM algorithm in the Wine data set

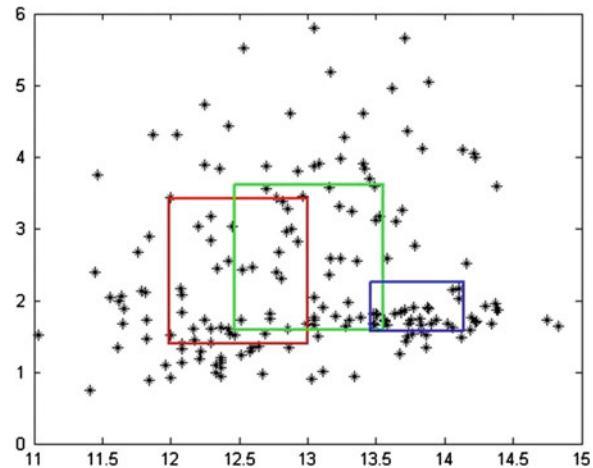
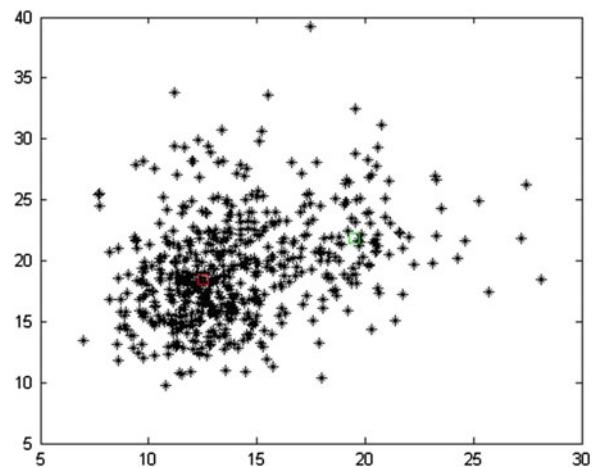


Fig. 7 Prototype found by the FCM algorithm in the WDBC dataset



represented by a numerical value and as can we see that prototype found represent the data structure but this representation is not reflective. Figures 6, 8 and 10 show the granular prototypes found by a GFCM (Granular Fuzzy C-Means) algorithm proposed over Wine, WDBC, and Iris Flower data sets respectively, in these figures we can observe that the representation of the data structure is more reflective.

5 Conclusions

Granular theory is very general and there is no a specific way of implementing this theory with clustering algorithms.

In this work a method is presented to apply the granular theory to the Fuzzy c-Means algorithm, making a granular fuzzy c-means algorithm capable of create granular prototypes and granular matrices of belonging instead of numerical prototypes and numerical matrices of belonging making more reflective the data structure.

Figures 6, 8 and 10 show the granular prototypes found by the proposed GFCM (Granular Fuzzy C-Means) algorithm applied to the Wine, WDBC, and Iris Flower data sets respectively that represent the data structure of the each dataset mentioned

Fig. 8 Granular prototype found by the GFCM algorithm in the WDBC data set

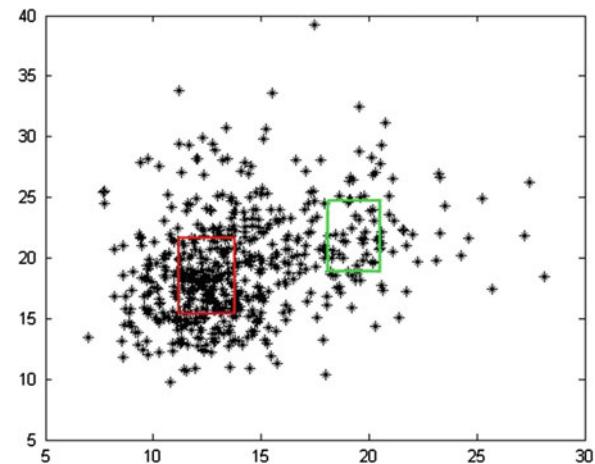


Fig. 9 Prototype found by the FCM algorithm in the Iris flower dataset

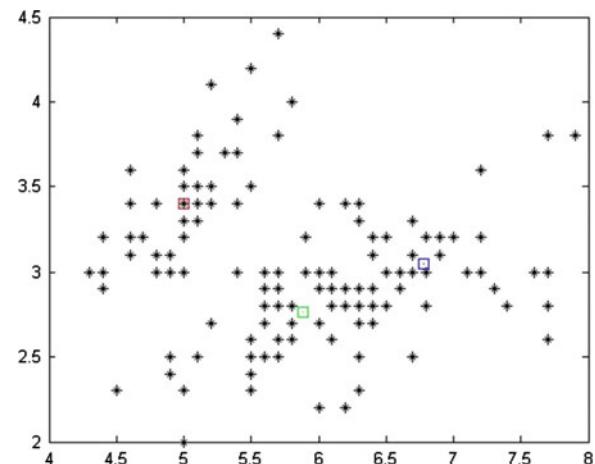
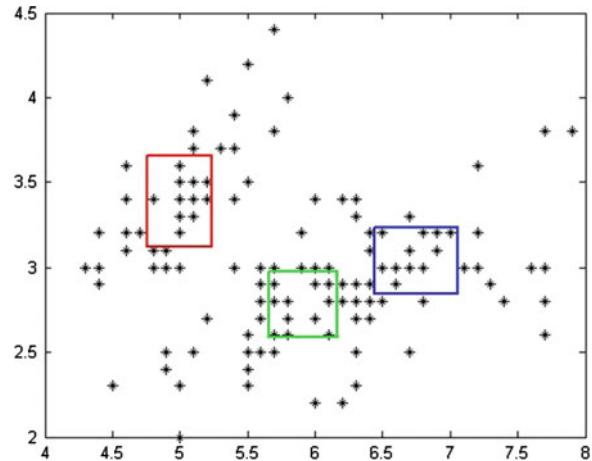


Fig. 10 Granular prototype found by the GFCM algorithm in the Iris flower data set



above. In these figures we can observe that the representation of the data structure is more reflective in comparison with the data structure found by the FCM algorithm for the same datasets.

References

1. Bargiela, A., Pedrycz, W., Hirota, K.: Granular prototyping in fuzzy clustering. *IEEE Trans. Fuzzy Syst.* **12**(5), 697–709 (2004)
2. Bezdek, J.: Pattern Recognition with Fuzzy Objective Function Algorithms. Plenum, Berlin (1981)
3. Chang, X., Li, W., Farrell, J.: A C-means clustering based fuzzy modeling method. In: The Ninth IEEE International Conference on Fuzzy Systems, 2000. FUZZ IEEE 2000, vol. 2, pp. 937–940 (2000)
4. Gacek, A.: From clustering to granular clustering: a granular representation of data in pattern recognition and system modeling. In: IFSA World Congress and NAFIPS Annual Meeting (IFSA/NAFIPS), 2013 Joint, pp. 502–506, 24–28 June 2013
5. Gustafson, D.E., Kessel, W.C.: Fuzzy clustering with a fuzzy covariance matrix. In: Proceedings of IEEE Conference on Decision and Control, San Diego, CA, pp. 761–766 (1979)
6. Hirota, K., Pedrycz, W.: Fuzzy computing for data mining. *Proc. IEEE* **87**(9), 1575–1600 (1999)
7. Hwang, C., Rhee, F.: Uncertain fuzzy clustering: interval type-2 fuzzy approach to C-means. *IEEE Trans. Fuzzy Syst.* **15**(1), 107–120 (2007)
8. Iyer, N.S., Kendel, A., Schneider, M.: Feature-based fuzzy classification for interpretation of mammograms. *Fuzzy Sets Syst.* **114**, 271–280 (2000)
9. Karnik, N., Mendel, J.: Operations on type-2 set. *Fuzzy Set Syst.* **122**, 327–348 (2001)
10. Krishnapuram, R., Keller, J.: A possibilistic approach to clustering. *IEEE Trans. Fuzzy Syst.* **1**(2), 98–110 (1993)
11. Krishnapuram, R., Keller, J.: The possibilistic c-Means algorithm: Insights and recommendations. *IEEE Trans. Fuzzy Sys.* **4**(3), 385–393 (1996)

12. Kruse, R., Döring, C., Lesot, M.J.: Fundamentals of fuzzy clustering. In: Advances in Fuzzy Clustering and its Applications. Wiley, Chichester, pp. 3–30 (2007)
13. Mendel, J.: Uncertain Rule-Based Fuzzy Logic Systems: Introduction and New Directions, pp. 213–231. Prentice-Hall, Englewood Cliffs (2001)
14. Pal, N.R., Pal, K., Keller, J.M., Bezdek, J.: A possibilistic fuzzy c-means clustering algorithm. *IEEE Trans. Fuzzy Syst.* **13**(4), 517–530 (2005)
15. Pedrycz, W., Bargiela, A.: An Optimization of allocation of information granularity in the interpretation of data structures: toward granular fuzzy clustering. *IEEE Trans. Syst. Man Cybern. B Cybern.* **42**(3), 582–590 (2012)
16. Philips, W.E., Velthuizen, R.P., Phuphanich, S., Hall, L.O., Clark, L.P., Sibiger, M.L.: Application of fuzzy c-means segmentation technique for tissue differentiation in MR images of hemorrhagic glioblastoma multiforme. *Magn. Reson. Imaging* **13**(2), 277–290 (1995)
17. Rubio, E., Castillo, O.: Interval type-2 fuzzy clustering for membership function generation. In: IEEE Workshop on Hybrid Intelligent Models and Applications (HIMA), pp. 13–18, 16–19 Apr 2013
18. Yang, M.-S., Hu, Y.-J., Lin, K.C.-R., Lin, C.C.-L.: Segmentation techniques for tissue differentiation in MRI of Ophthalmology using fuzzy clustering algorithms. *Magn. Reson. Imaging* **20**, 173–179 (2002)
19. Zarandi, M.H.F., Zarinbal, M., Türksen, I.B.: Type-II fuzzy possibilistic C-mean clustering. In: IFSA/EUSFLAT Conference, pp. 30–35 (2009)

Face Recognition with a Sobel Edge Detector and the Choquet Integral as Integration Method in a Modular Neural Networks

Gabriela E. Martínez, Patricia Melin, Olivia D. Mendoza
and Oscar Castillo

Abstract In this paper a method for response integration of Modular Neural Networks, based on Choquet Integral applied to face recognition is presented. Type-1 and Type-2 fuzzy systems for edge detections based on the Sobel, which is a pre-processing applied to the training data for better performance in the modular neural network. The Choquet integral is an aggregation operator that in this case is used as a method to integrate the outputs of the modules of the modular neural networks (MNN).

1 Introduction

A mechanism which takes as input a number n of data and combines them to result in a representative value of the information is called integration method. In the literature there exist methods which combine information from different sources. In a MNN it is common to use different methods for integrate the information, such as Type-1 and Type-2 fuzzy logic for Multimodal Biometrics [3], for Human Recognition Based on Iris, Ear and Voice Biometrics [17], for Pattern Recognition [16], the fuzzy Sugeno integral for Pattern Recognition of human face and fingerprint [10], Interval Type-2 Fuzzy Logic Sugeno Integral for Face Recognition [11], a probabilistic sum integrator in Classification using Redundant Mapping [9], a novel Bayesian learning method for information aggregation in modular neural networks [23], Self-Organizing Maps for Japanese Historical Character Recognition [4], Choquet Integral in a modular neural network [8], among others.

The Choquet integral was created by the French mathematician Gustave Choquet in 1953 [2]. The Choquet integral with respect to a fuzzy measure is a very

G.E. Martínez · O.D. Mendoza
University of Baja California, Tijuana, Mexico

P. Melin (✉) · O. Castillo
Tijuana Institute of Technology, Tijuana, Mexico
e-mail: pmelin@tectijuana.mx

popular data aggregation approach. The generalized Choquet integral with respect to a signed fuzzy measure can act as an aggregation tool, which is especially useful in many applications [6, 21, 24].

This paper is organized as follows: Sect. 2 shows the technique that is applied for the combination of the several information sources: the concepts of Fuzzy Measures and Choquet integral. Section 3 presents Edge detection based on Sobel with type-1 and type-2 fuzzy systems. Section 4 describes the architecture of the modular neural network proposed and in the Sect. 5 the simulation results are shown. And finally in Sect. 6 the Conclusions are presented.

2 Fuzzy Measures and Choquet Integral

In 1974 the concepts of “fuzzy measure and fuzzy integral” were defined by Sugeno [20] in order to define sets that do not have well-defined boundaries. A fuzzy measure is a nonnegative function monotone of values defined in “classical sets”. Currently, when referring to this topic, the term “fuzzy measures” has been replaced by the term “monotonic measures”, “non-additive measures” or “generalized measures” [14, 18, 22]. When fuzzy measures are defined on fuzzy sets, we speak of fuzzified monotonic measures [22].

2.1 Fuzzy Measures

A fuzzy measure can be defined as a fuzzy measure μ with respect to the dataset X , and must satisfy the following conditions:

1. $\mu(X) = 1; \mu(\emptyset) = 0$ Boundary conditions
2. $\text{Si } A \subset B, \text{ then } \mu(A) \leq \mu(B)$ Monotonicity

In condition two, A and B are subsets of X .

A fuzzy measure is a Sugeno measure or λ -fuzzy, if it satisfies the condition (1) of addition for some $\lambda > -1$.

$$\mu(A \cup B) = \mu(A) + \mu(B) + \lambda\mu(A)\mu(B) \quad (1)$$

where λ can be calculated with (2):

$$f(\lambda) = \left\{ \prod_{i=1}^n (1 + M_i(x_i)\lambda) \right\} - (1 + \lambda) \quad (2)$$

The value of the parameter λ is determined by the conditions of Theorem 1.

Theorem 1 Let $\mu(\{x\}) < 1$ for each $x \in X$ and let $\mu(\{x\}) > 0$ for at least two elements of X , then (2) determines a unique parameter in the following way:

If $\sum_{x \in X} \mu(\{x\}) < 1$, then λ is in the interval $(0, \infty)$.

If $\sum_{x \in X} \mu(\{x\}) = 0$, then $\lambda = 0$; That is the unique root of the equation.

If $\sum_{x \in X} \mu(\{x\}) > 1$, then λ se is in the interval $(-1, 0)$.

The fuzzy measure represents the importance or relevance of the information sources when computing the aggregation. The method to calculate Sugeno measures, carries out the calculation in a recursive way, using (3) and (4).

$$\mu(A_1) = \mu(M_i) \quad (3)$$

$$\mu(A_i) = \mu(A_{(i-1)}) + \mu(M_i) + (\lambda \mu(M_i) * \mu(A_{(i-1)})) \quad (4)$$

where A_i represents the fuzzy measure and M_i represents the fuzzy density determined by an expert, where $1 < M_i \leq \dots \leq n$, should be permuted with respect to the descending order of their respective $\mu(A_i)$.

In the literature there are 2 types of Integral, the integral of Sugeno and the Choquet Integral.

2.2 Choquet Integral

The Choquet integral can be calculated using Eq. (5) or an equivalent expression (6)

$$\text{Choquet} = \sum_{i=1}^n \{ [A_i - A_{(i-1)}] * D_i \} \quad (5)$$

With $A_0 = 0$

Or also

$$\text{Choquet} = \sum_{i=1}^n A_i * \{ [D_i - D_{(i+1)}] \} \quad (6)$$

With $D_{(n+1)} = 0$

In this case A_i represents the fuzzy measurement associated with a data D_i .

3 Edge Detection

Edge detection can be defined as a method consisting of identifying changes that exist in the light intensity, which can be used to determine certain properties or characteristics of the objects in the image. A number of edge detectors have been developed by various researchers. Amongst them, the most important operators are the Sobel Operator [19], Prewitt operator [15], Robert operator [15], Canny operator [1] and Kirsch operator [5]. The resultant images of the edge detectors preserve more details of the original images, which is a desirable feature for a pattern recognition system.

We used the Cropped Yale database to perform the training of the modular neural network, which has images of 38 people with 60 samples of each individual. To each of the images we applied a pre-processing by making use of Sobel edge detector with type-1 and type-2 fuzzy logic systems [7] in order to highlight features, some of the images can be displayed in Fig. 4a. Each image of the Cropped Yale database has a size of 168×192 .

3.1 Sobel

The Sobel operator is applied to a digital image in gray scale, is a pair of 3×3 convolution masks, one estimating the gradient in the x-direction (columns) (7) and the other estimating the gradient in the y-direction (rows) (8) [12].

$$sobel_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad (7)$$

$$sobel_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \quad (8)$$

If we have $I_{m,n}$ as a matrix of m rows and r columns, where the original image is stored, then g_x and g_y are matrices having the same dimensions as I , which at each element contains the horizontal and vertical derivative approximations and are calculated by (9) and (10) [12].

$$g_x = \sum_{i=1}^{i=3} \sum_{j=1}^{j=4} Sobel_{x,ij} * I_{r+i-2,c+j-2} \quad \begin{array}{l} \text{for } i = 1, 2, \dots, m \\ \text{for } j = 1, 2, \dots, n \end{array} \quad (9)$$

$$g_y = \sum_{i=1}^{i=3} \sum_{j=1}^{j=4} Sobel_{y,ij} * I_{r+i-2,c+j-2} \quad \begin{array}{l} \text{for } i = 1, 2, \dots, m \\ \text{for } j = 1, 2, \dots, n \end{array} \quad (10)$$

In the Sobel method the gradient magnitude g is calculated by (11).

$$g = \sqrt{g_x^2 + g_y^2} \quad (11)$$

For the type-1 and type-2 fuzzy inference systems, 3 inputs can be used as can be seen in Figs. 1 and 2, two of them are the gradients with respect to the x-axis and y-axis, calculated with (9) and (10), which we call DH and DV, respectively. The third variable M is the image after the application of a low-pass filter hMF in (12); this filter allows to detect image pixels belonging to regions of the input where the mean gray level is lower. These regions are proportionally affected more by noise, which is supposed to be uniformly distributed over the whole image [12].

$$hMF = \frac{1}{25} * \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} \quad (12)$$

Fig. 1 Variables for the edge detector with the type-1 fuzzy Sobel

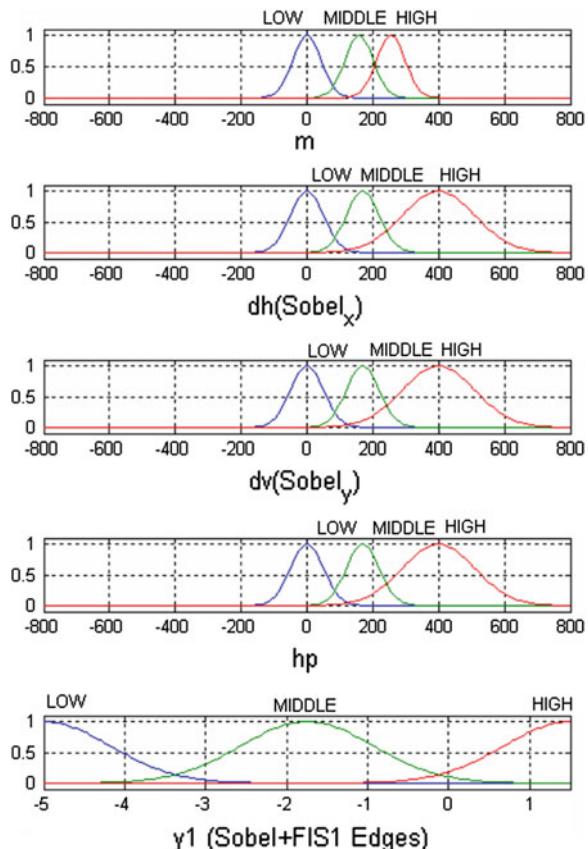
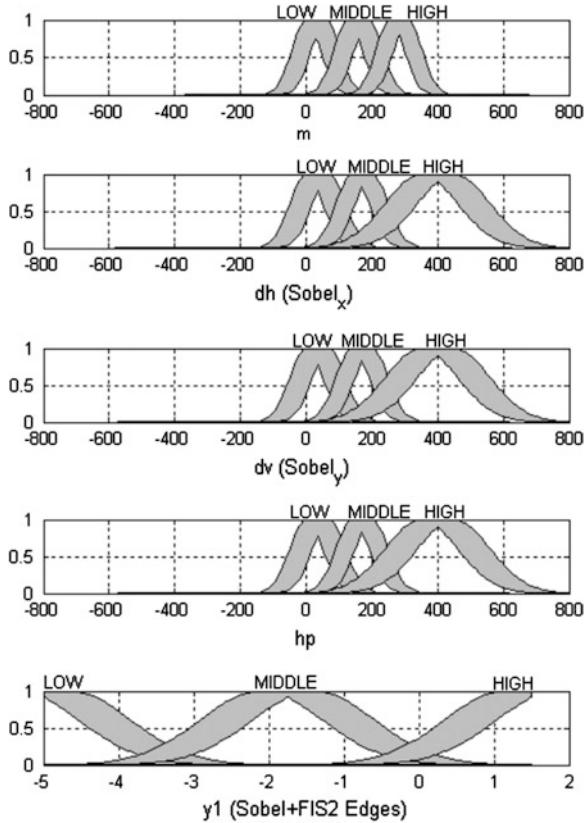


Fig. 2 Variables for the edge detector with the type-2 fuzzy Sobel



After applying the edge detector of Type-1 with Sobel, the resulting image can be viewed in Fig. 3b.

The rules used for the tests in the Type-1 and Type-2 Sobel edge detectors are [12]:

1. If (dh is LOW) and (dv is LOW) then (y1 is HIGH).
2. If (dh is MIDDLE) and (dv is MIDDLE) then (y1 is LOW).
3. If (dh is HIGH) and (dv is HIGH) then (y1 is LOW).
4. If (dh is MIDDLE) and (hp is LOW) then (y1 is LOW).
5. If (dv is MIDDLE) and (hp is LOW) then (y1 is LOW).
6. If (m is LOW) and (dv is MIDDLE) then (y1 is HIGH).
7. If (m is LOW) and (dh is MIDDLE) then (y1 is HIGH).

The images generated by the Type-2 Sobel edge detector are shown in Fig. 3c.

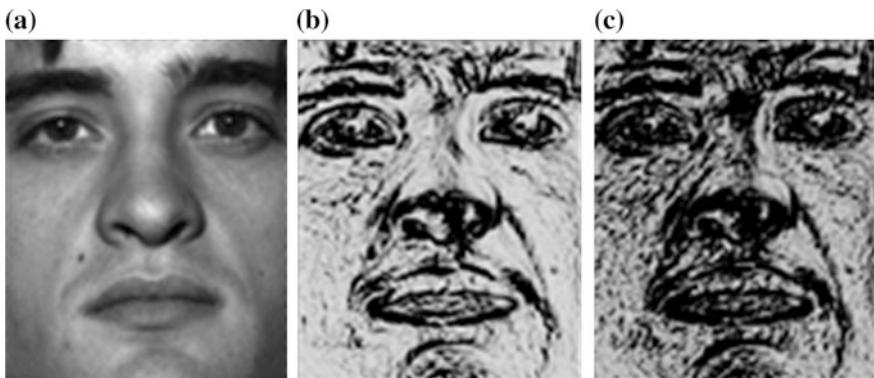


Fig. 3 **a** Original image database Cropped Yale, **b** image with type-1 Sobel edge detector, **c** image with interval type-2 Sobel edge detector

4 Modular Neural Networks

To each image of the Cropped Yale database we applied a preprocessing with the Type-1 and Type-2 Sobel edge detector, after this, each image is divided into 3 sections horizontal. Each section was used as training data in each one of the three modules of the modular neural network, as shown in Fig. 4. The integration of the simulation of each module is made with the Choquet integral.

The Training Parameters are:

Training method: gradient descendent with momentum and adaptive learning rate back-propagation (Traingdx).

Each module has two hidden layers [200 200].

Error goal: 0.0001

Epochs: 500

In Table 1 the distribution of the training data is shown.

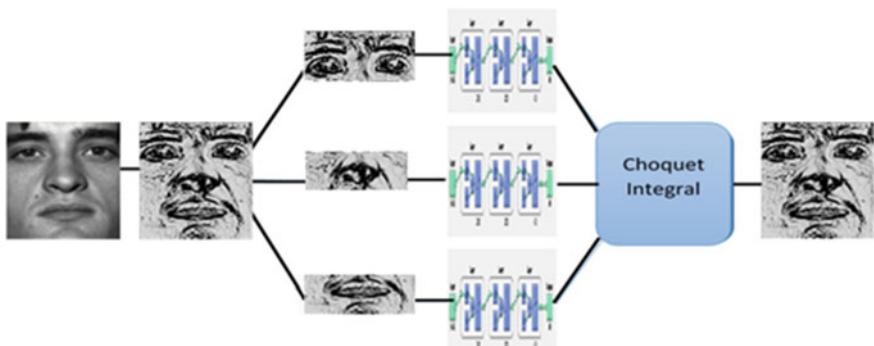


Fig. 4 Architecture proposal of the modular neural network using type-1 Sobel edge detector

Table 1 Distribution of the training data

Training	Validation	Test
70 %	15 %	15 %

4.1 The Experiment with a Modular Neural Network Recognition System and the Choquet Integral for the Modules Fusion

The experiment consist on applying each evaluated edge detector to obtain a data set of same well know benchmark data bases of images, like Cropped Yale database of faces and then train a neural network to compare the recognition rate using the k-fold cross-validation method [13].

Table 2 Fuzzy densities and λ parameter

Test	FD1	FD2	FD3	Lambda
1	0.1	0.1	0.1	-1.40165E-16
2	0.1	0.1	0.5	2.8008E-16
3	0.1	0.1	0.9	-0.540135346
4	0.1	0.5	0.1	2.76119E-16
5	0.1	0.5	0.5	-0.291796068
6	0.1	0.5	0.9	-0.910717424
7	0.1	0.9	0.1	-0.540135346
8	0.1	0.9	0.5	-0.910717424
9	0.1	0.9	0.9	-0.989141968
10	0.5	0.1	0.1	2.76119E-16
11	0.5	0.1	0.5	-0.291796068
12	0.5	0.1	0.9	-0.910717424
13	0.5	0.5	0.1	-0.291796068
14	0.5	0.5	0.5	-0.763932023
15	0.5	0.5	0.9	-0.964686549
16	0.5	0.9	0.1	-0.910717424
17	0.5	0.9	0.5	-0.964686549
18	0.5	0.9	0.9	-0.994458264
19	0.9	0.1	0.1	-0.540135346
20	0.9	0.1	0.5	-0.910717424
21	0.9	0.1	0.9	-0.989141968
22	0.9	0.5	0.1	-0.910717424
23	0.9	0.5	0.5	-0.964686549
24	0.9	0.5	0.9	-0.994458264
25	0.9	0.9	0.1	-0.989141968
26	0.9	0.9	0.5	-0.994458264
27	0.9	0.9	0.9	-0.998971986

5 Simulation Results

In the experiments we performed 27 tests in simulation of the trainings with each edge detectors making variations in the fuzzy densities with the values of 0.1, 0.5 and 0.9 and performing the calculation of the parameter λ with the bisection method, and these parameters can be find in Table 2.

In Table 3, the results obtained using preprocessing of the image, with the type-1 Sobel edge detector, are shown. The first column represents the number of tests performed for each simulation, of the column M1 to M5 the average obtained in each module for each training is displayed and finally the last three columns present

Table 3 Results of one simulation using type-1 Sobel edge detector, in test 1

Test	M1	M2	M3	M4	M5	Mean	Std.	Max.
1	100	96.0526	96.0526	98.6842	97.3684	0.9763	0.0172	1
2	100	96.0526	97.3684	98.6842	97.3684	0.9789	0.0150	1
3	100	96.0526	97.3684	98.6842	97.3684	0.9789	0.0150	1
4	100	96.0526	96.0526	98.6842	97.3684	0.9763	0.0172	1
5	100	96.0526	97.3684	98.6842	97.3684	0.9789	0.0150	1
6	100	96.0526	97.3684	98.6842	97.3684	0.9789	0.0150	1
7	100	96.0526	96.0526	98.6842	97.3684	0.9763	0.0172	1
8	100	96.0526	97.3684	98.6842	97.3684	0.9789	0.0150	1
9	100	96.0526	97.3684	98.6842	97.3684	0.9789	0.0150	1
10	100	98.6842	96.0526	98.6842	98.6842	0.9842	0.0144	1
11	100	98.6842	97.3684	98.6842	98.6842	0.9868	0.0093	1
12	100	98.6842	97.3684	98.6842	98.6842	0.9868	0.0093	1
13	100	98.6842	96.0526	98.6842	98.6842	0.9842	0.0144	1
14	100	98.6842	97.3684	98.6842	98.6842	0.9868	0.0093	1
15	100	98.6842	97.3684	98.6842	98.6842	0.9868	0.0093	1
16	100	98.6842	96.0526	98.6842	98.6842	0.9842	0.0144	1
17	100	98.6842	97.3684	98.6842	98.6842	0.9868	0.0093	1
18	100	98.6842	97.3684	98.6842	98.6842	0.9868	0.0093	1
19	100	98.68421	96.05263	98.68421	100	0.98684	0.01612	1
20	100	98.68421	97.36842	98.68421	100	0.98947	0.01101	1
21	100	98.68421	97.36842	98.68421	100	0.98947	0.01101	1
22	100	98.68421	96.05263	98.68421	100	0.98684	0.01612	1
23	100	98.68421	97.36842	98.68421	100	0.98947	0.01101	1
24	100	98.68421	97.36842	98.68421	100	0.98947	0.01101	1
25	100	98.68421	96.05263	98.68421	100	0.98684	0.01612	1
26	100	98.68421	97.36842	98.68421	100	0.98947	0.01101	1
27	100	98.68421	97.36842	98.68421	100	0.98947	0.01101	1
					Mean	0.98421	0.01315	1

Table 4 Results of training with type-1 Sobel edge detector

Test	Mean	Std.	Max.
1	0.98421	0.01315	1
2	0.98743	0.01523	1
3	0.98333	0.01606	1
Mean	0.98499		

Table 5 Results of training with type-2 Sobel edge detector

Test	Mean	Std.	Max.
1	0.98684	0.02942	1
2	0.98421	0.01915	1
3	0.98947	0.01715	1
Mean	0.98684		

the mean of the five trainings, the standard deviation and the maximum of each simulation.

In Table 4 we show the percentages of recognition of the Modular neural network using the Choquet integral as integration method with a preprocessing in the images with the Type 1 Sobel edge detector. Each result of the tests is the average of 27 simulations with different fuzzy densities, as seen in Table 3.

In Table 5 we can find the results of the simulations using the Type-2 Sobel edge detector as preprocessing method.

It can be noted that when using the Type-2 Sobel edge detector it was obtained 98.49 % of recognition whereas with type-2 Sobel edge detector a 98.68 % was obtained.

6 Conclusions

The use of the Choquet integral as an integration method of responses of a modular neural network applied to face recognition has yielded favorable results when performing the aggregation process of the preprocessed images with the detectors of Sobel edges. The resultant images of the edge detectors preserve more details of the original images, which is a desirable feature for a pattern recognition system. However it is still necessary to use a method that optimizes the values of the Sugeno measure assigned to each source of information because these were designated arbitrarily.

Acknowledgment We thank the MyDCI program of the Division of Graduate Studies and Research, UABC, Tijuana Institute of Technology, and the financial support provided by our sponsor CONACYT contract grant number: 189350.

References

1. Canny, J.: A computational approach to edge detection. *IEEE Trans. Pattern Anal. Mach. Intell.* **8**(2), 679–698 (1986)
2. Choquet, G.: Theory of capacities. *Ann. Inst. Fourier, Grenoble* **5**, 131–295 (1953). doi:[10.5802/aif.53](https://doi.org/10.5802/aif.53)
3. Hidalgo, D.: Fuzzy inference systems type 1 and type 2 as integration methods in neural networks for multimodal biometrics and Me-optimization by means of genetic algorithms. Master Thesis, Tijuana Institute of Technology (2008)
4. Horiuchi, T., Kato, S.: A study on Japanese historical character recognition using modular neural networks. In: IEEE Fourth International Conference on Innovative Computing, Information and Control, pp. 1507–1510 (2009)
5. Kirsch, R.: Computer determination of the constituent structure of biological images. *Comput. Biomed. Res.* **4**, 315–328 (1971)
6. Kwak, K.C., Pedrycz, W.: Face recognition: a study in information fusion using fuzzy integral. *Pattern Recogn. Lett.* **26**, 719–733 (2005)
7. Liu, H.C., Wu D.B., Jheng Y.D., Chen, C.C., Chien, M.F., Sheu, T.W.: Choquet integral with respect to sigma-fuzzy measure. In: International Conference on Mechatronics and Automation, Changchun, China 978-1-4244-2693-5/09. IEEE (2009)
8. Martínez, G.E., Melin, P., Olivia, M.D., Castillo, O.: Face recognition with Choquet integral in modular neural networks. In: Recent Advances on Hybrid Approaches for Designing Intelligent Systems. Springer International Publishing, Switzerland (2014). doi:[10.1007/978-3-319-05170-3_30](https://doi.org/10.1007/978-3-319-05170-3_30)
9. Meena, Y.K., Arya, K.V., Kala, R.: Classification using redundant mapping in modular neural networks. In: Second World Congress on Nature and Biologically Inspired Computing, Kitakyushu, Fukuoka, Japan, 15–17 Dec 2010
10. Melin, P., Gonzalez, C., Bravo, D., Gonzalez, F., Martínez, G.: Modular neural networks and fuzzy Sugeno integral for pattern recognition: the case of human face and fingerprint. In: Hybrid Intelligent Systems: Design and Analysis. Springer, Heidelberg (2007)
11. Melin, P., Mendoza, O., Castillo O.: Face recognition with an improved interval type-2 fuzzy logic Sugeno integral and modular neural networks. *IEEE Trans. Syst. Man Cybern. Part A Syst. Hum.* **41**(5) (2011)
12. Mendoza, O., Melin, P., Castillo, O., Castro, J.: Comparison of fuzzy edge detectors based on the image recognition rate as performance index calculated with neural networks. In: Soft Computing for Recognition Based on Biometrics. Studies in Computational Intelligence, vol. 312, pp. 389–399 (2010)
13. Mendoza, O., Melin, P.: Quantitative evaluation of fuzzy edge detectors applied to neural networks or image recognition. In: Advances in Research and Developments in Digital Systems, pp. 324–335 (2011)
14. Murofushi, T., Sugeno, M.: Fuzzy measures and fuzzy integrals. Department of Computational Intelligence and Systems Science, Tokyo Institute of Technology, Yokohama, Japan (2000)
15. Prewitt, J.M.S.: Object enhancement and extraction. In: Lipkin, B.S., Rosenfeld, A. (eds.) Picture Analysis and Psychopictorics, pp. 75–149. Academic Press, New York (1970)
16. Sánchez, D., Melin, P.: Modular neural network with fuzzy integration and its optimization using genetic algorithms for human recognition based on iris, ear and voice biometrics. In: Soft Computing for Recognition Based on Biometrics, pp. 85–102 (2010)
17. Sánchez, D., Melin, P., Castillo, O., Valdez, F.: Modular neural networks optimization with hierarchical genetic algorithms with fuzzy response integration for pattern recognition. MICAI, pp. 247–258 (2012)
18. Song, J., Li, J.: Lebesgue theorems in non-additive measure theory. *Fuzzy Sets Syst.* **149**(3), 543–548 (2005)

19. Sobel, I.: Camera models and perception. Ph.D. Thesis, Stanford University, Stanford, CA (1970)
20. Sugeno, M.: Theory of fuzzy integrals and its applications. Thesis Doctoral, Tokyo Institute of Technology, Tokyo, Japan (1974)
21. Timonin, M.: Robust optimization of the Choquet integral. *Fuzzy Sets Syst.* **213**, 27–46 (2013)
22. Wang, Z., Klir, G.: Generalized measure Theory. Springer, New York (2009)
23. Wang, P., Xua, L., Zhou, S.M., Fan, Z., Li, Y., Feng, S.: A novel Bayesian learning method for information aggregation in modular neural networks. In: *Expert Systems with Applications*, vol. 37, pp. 1071–1074. Elsevier, New York (2010)
24. Yang, W., Chen, Z.: New aggregation operators based on the Choquet integral and 2-tuple linguistic information. *Expert Syst. Appl.* **39**, 2662–2668 (2012)

Part II

Neural Networks Theory

Neural Network with Fuzzy Weights Using Type-1 and Type-2 Fuzzy Learning for the Dow-Jones Time Series

Fernando Gaxiola, Patricia Melin and Fevrier Valdez

Abstract In this paper, type-1 and type-2 fuzzy inferences systems are used to obtain the type-1 or type-2 fuzzy weights in the connections between the layers of a neural network. We use two type-1 or type-2 fuzzy systems that work in the backpropagation learning method with the type-1 or type-2 fuzzy weight adjustment. The mathematical analysis of the proposed learning method architecture and the adaptation of type-1 or type-2 fuzzy weights are presented. The proposed method is based on recent methods that handle weight adaptation and especially fuzzy weights. In this work neural networks with type-1 fuzzy weights or type-2 fuzzy weights are presented. The proposed approach is applied to the case of Dow-Jones time series prediction for evaluating its efficiency.

1 Introduction

Neural networks have been applied in several areas of research, like in the time series prediction area, which is the study case for this paper, the study case is applied for the Dow-Jones time series.

The approach presented in this paper works with type-1 and type-2 fuzzy weights in the neurons of the hidden and output layers of the neural network used for prediction of the Dow-Jones time series. These type-1 and interval type-2 fuzzy weights are updated using the backpropagation learning algorithm. We used two type-1 inference systems and two type-2 inference systems with Gaussian membership functions for fuzzy weight adjustment.

F. Gaxiola · P. Melin · F. Valdez (✉)
Tijuana Institute of Technology, Tijuana, Mexico
e-mail: fevrier@tectijuana.edu.mx

F. Gaxiola
e-mail: fergaor_29@hotmail.com

P. Melin
e-mail: pmelin@tectijuana.mx

The proposed approach is applied to time series prediction for the Dow-Jones time series. The objective is obtaining the minimal prediction error for the data of the time series.

We used a supervised neural network, because this type of network is the most commonly used in the area of time series prediction.

The weights of a neural network are an important part in the training phase, because these affect the performance of the learning process of the neural network.

This paper is focused in the managing of weights, because on the practice of neural networks, when performing the training of neural networks for the same problem is initialized with different weights or the adjustment are in a different way each time it is executed, but at the final is possible to reach a similar result.

The next section presents a background about modifications of the backpropagation algorithm and different management strategies of weights in neural networks, and basic concepts of neural networks. Section 3 explains the proposed method and the problem description. Section 4 describes the neural network with type-1 fuzzy weights proposed in this paper. Section 5 describes the neural network with type-2 fuzzy weights proposed in this paper. Section 6 presents the simulation results for the proposed method. Finally, in Sect. 7 some conclusions are presented.

2 Background and Basic Concepts

In this section a brief review of basic concepts is presented.

2.1 Neural Network

An artificial neural network (ANN) is a distributed computing scheme based on the structure of the nervous system of humans. The architecture of a neural network is formed by connecting multiple elementary processors, this being an adaptive system that has an algorithm to adjust their weights (free parameters) to achieve the performance requirements of the problem based on representative samples [8, 26].

The most important property of artificial neural networks is their ability to learn from a training set of patterns, i.e. they are able to find a model that fits the data [9, 34].

The artificial neuron consists of several parts (see Fig. 1). On one side are the inputs, weights, the summation, and finally the transfer function. The input values are multiplied by the weights and added: $\sum x_i w_{ij}$. This function is completed with the addition of a threshold amount i . This threshold has the same effect as an input with value -1 . It serves so that the sum can be shifted left or right of the origin. After addition, we have the f function applied to the sum, resulting in the final value of the output, also called y_i [28], obtaining the following equation:

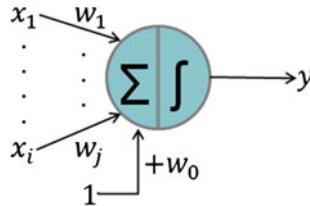


Fig. 1 Schematics of an modular artificial neural network

$$y_i = f \left(\sum_{i=1}^n x_i w_{ij} \right) \quad (1)$$

where f may be a nonlinear function with binary output ± 1 , a linear function $f(z) = z$, or as sigmoid logistic function:

$$f(z) = \frac{1}{1 + e^{-z}}. \quad (2)$$

2.2 Overview of Related Works

The backpropagation algorithm and its variations are the most useful basic training methods in the area of neural networks. However, these algorithms are usually too slow for practical applications.

When applying the basic backpropagation algorithm to practical problems, the training time can be very high. In the literature we can find that several methods have been proposed to accelerate the convergence of the algorithm [2, 18, 28, 37].

There exists many works about adjustment or managing of weights but only the most important and relevant for this research will be mentioned here [4, 10, 31, 36]:

Momentum method—Rumelhart, Hinton and Williams suggested adding in the increased weights expression a momentum term β , to filter the oscillations that can be formed a higher learning rate that lead to great change in the weights [19, 32].

Adaptive learning rate—focuses on improving the performance of the algorithm by allowing the learning rate changes during the training process (increase or decrease) [19].

Conjugate Gradient algorithm—A search of weight adjustment along conjugate directions. Versions of conjugate gradient algorithm differ in the way in which a constant βk is calculated.

- Fletcher-Reeves update [12].
- Polak-Ribiere updated [12].
- Powell-Beale Restart [3, 33].
- Scaled Conjugate Gradient [29].

Kamarthi and Pittner [24], focused in obtaining a weight prediction of the network at a future epoch using extrapolation.

Ishibuchi et al. [21], proposed a fuzzy network, where the weights are given as trapezoidal fuzzy numbers, denoted as four trapezoidal fuzzy numbers for the four parameters of trapezoidal membership functions.

Ishibuchi et al. [22], proposed a fuzzy neural network architecture with symmetrical fuzzy triangular numbers for the fuzzy weights and biases, denoted by the lower, middle and upper limit of the fuzzy triangular numbers.

Feuring [11], based on the work by Ishibuchi, where triangular fuzzy weights are used, developed a learning algorithm in which the backpropagation algorithm is used to compute the new lower and upper limits of weights. The modal value of the new fuzzy weight is calculated as the average of the new computed limits.

Castro et al. [6], use interval type-2 fuzzy neurons for the antecedents and interval of type-1 fuzzy neurons for the consequents of the rules. This approach handles the weights as numerical values to determine the input of the fuzzy zneurons, as the scalar product of the weights for the input vector.

Gaxiola et al. [13–17], proposed a neural network with type-2 fuzzy weights using triangular membership functions.

In addition, recent works on type-2 fuzzy logic have been developed in time series prediction, like that of Castro et al. [6], and other researchers [1, 7].

3 Proposed Method and Problem Description

The objective of this work is to use type-1 and interval type-2 fuzzy sets to generalize the backpropagation algorithm to allow the neural network to handle data with uncertainty. The Dow-Jones time series is utilized for testing the proposed approach.

The updating of the weights will be done differently to the traditional updating of the weights performed with the backpropagation algorithm (Fig. 2).

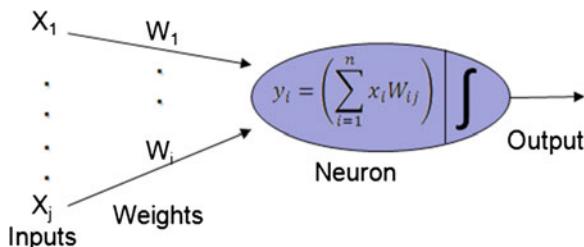
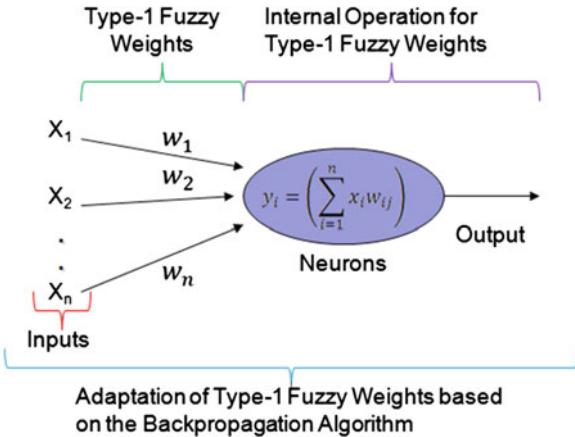


Fig. 2 Scheme of current management of numerical weights (type-0) for the inputs of each neuron

Fig. 3 Schematics of each neuron with the proposed management of weights using type-1 fuzzy sets

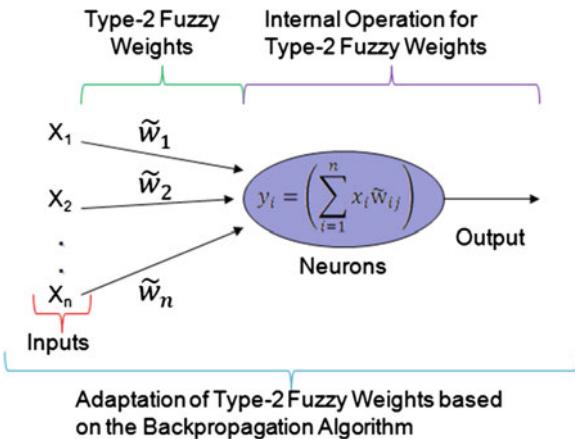


The proposed method performs the updating of the weights working with interval type-2 fuzzy weights. This method uses two type-1 and two type-2 inference systems with Gaussian membership functions for obtaining the type-1 and interval type-2 fuzzy weights using in the neural network, and obtaining the outputs taking into account the possible change in the way we work internally in the neuron, and the adaptation of the weights given in this way (Figs. 3 and 4) [30].

We developed a method for adjusting weights to achieve the desired result, searching for the optimal way to work with type-1 fuzzy weights or type-2 fuzzy weights [23].

We used the sigmoid activation function for the hidden neurons and the linear activation function for the output neurons, and we utilized this activation functions because these functions have obtained good results in similar approaches.

Fig. 4 Schematics of each neuron with the proposed management of weights using interval type-2 fuzzy sets



4 Neural Network Architecture with Type-1 Fuzzy Weights

The proposed neural network architecture with type-1 fuzzy weights (see Fig. 5) is described as follows:

Layer 0 Inputs.

$$x = [x_1, x_2, \dots, x_n] \quad (3)$$

Layer 1 type-1 fuzzy weights for the hidden layer.

$$w_{ij} \quad (4)$$

Layer 2 Equations of the calculations in the hidden neurons using type-1 fuzzy weights.

$$Net = \sum_{i=1}^n x_i w_i \quad (5)$$

Layer 3 Equations of the calculations in the outputs neurons using type-1 fuzzy weights.

$$Out = \sum_{i=1}^n y_i w_i \quad (6)$$

Layer 4 Obtain the output of the neural network.

We considered a neural network architecture with 1 neuron in the output layer and 30 neurons in the hidden layer.

This neural network uses two type-1 fuzzy inference systems, one in the connections between the input neurons and the hidden neurons, and the other in the connections between the hidden neurons and the output neuron. In the hidden layer and output layer of the network we are updating the weights using the two type-1

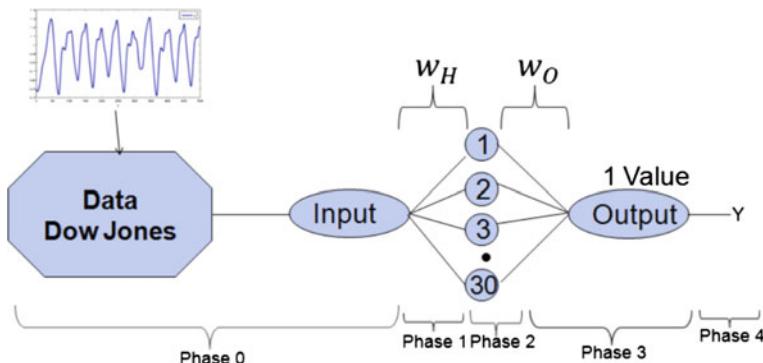


Fig. 5 Proposed neural network architecture with type-1 fuzzy weights

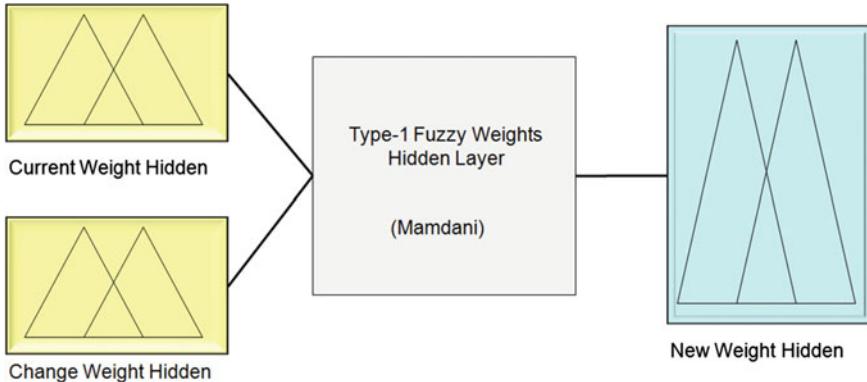


Fig. 6 Structure of the two type-1 fuzzy inference systems that are used to obtain the type-1 fuzzy weights in the hidden and output layer

fuzzy inference system that obtains the new weights in each epoch of the network on base at the backpropagation algorithm.

The two type-2 fuzzy inference systems have the same structure and consist of two inputs (the current weight in the actual epoch and the change of the weight for the next epoch) and one output (the new weight for the next epoch) (see Fig. 6).

We used two Gaussian membership functions with their corresponding range for delimiting the inputs and outputs of the two type-1 fuzzy inference systems (see Figs. 7 and 8).

We obtain the two type-1 fuzzy inference systems empirically.

The two type-1 fuzzy inference systems used the same six rules, the four combinations of the two Gaussian membership function and two rules added for null change of the weight (see Fig. 9).

5 Neural Network Architecture with Type-2 Fuzzy Weights

The proposed neural network architecture with interval type-2 fuzzy weights (see Fig. 10) is described as follows:

Layer 0 Inputs.

$$x = [x_1, x_2, \dots, x_n] \quad (7)$$

Layer 1 Interval type-2 fuzzy weights for the connection between the input and the hidden layer of the neural network.

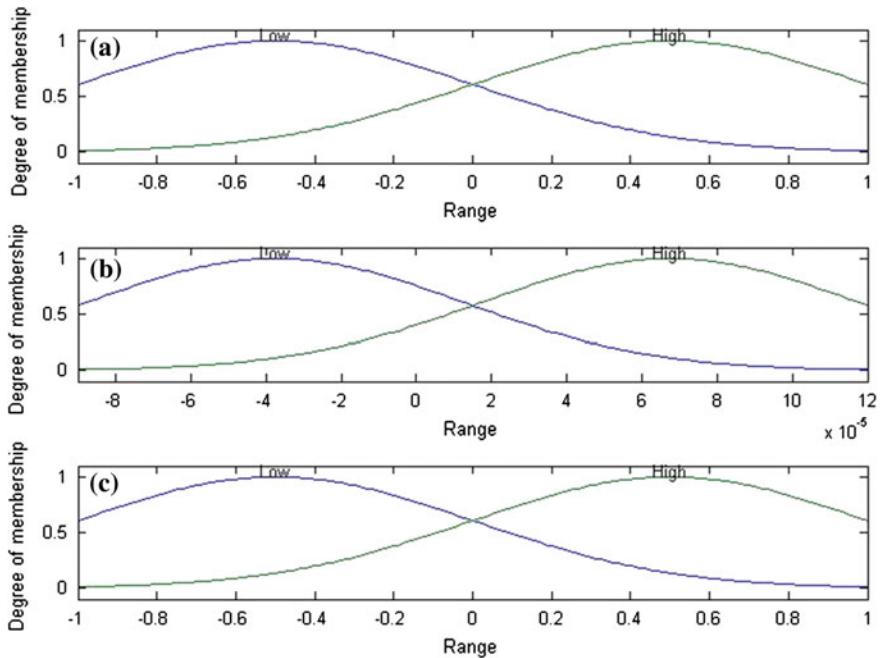


Fig. 7 Inputs **(a** current weight and **b** change of weight) and output **(c** new weight) of the type-1 fuzzy inference systems that are used to obtain the type-1 fuzzy weights in the hidden layer

$$\tilde{w}_{ij} = [\bar{w}_{ij}, \underline{w}_{ij}] \quad (8)$$

where \tilde{w}_{ij} are the weights of the consequents of each rule of the type-2 fuzzy system with inputs (current type-2 fuzzy weight, change of weight) and output (new fuzzy weight).

Layer 2 Equations of the calculations in the hidden neurons using interval type-2 fuzzy weights.

$$Net = \sum_{i=1}^n x_i \tilde{w}_{ij} \quad (9)$$

Layer 3 Equations of the calculations in the output neurons using interval type-2 fuzzy weights.

$$Out = \sum_{i=1}^n y_i \tilde{w}_{ij} \quad (10)$$

Layer 4 Obtain a single output of the neural network.

We applied the same neural network architecture used in the type-1 fuzzy weights for the type-2 fuzzy weights (see Fig. 6).

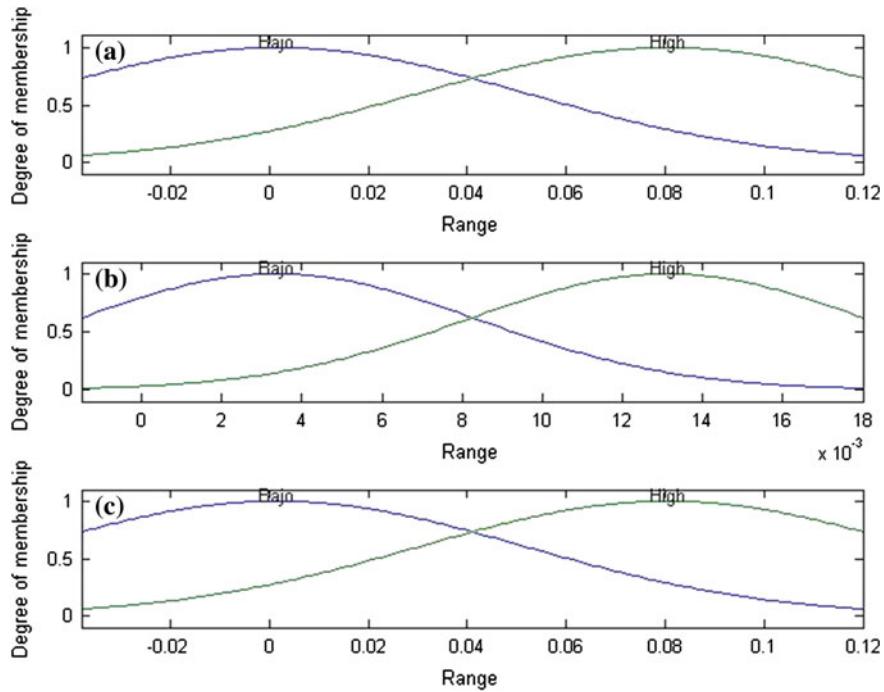


Fig. 8 Inputs (a current weight and b change of weight) and output (c new weight) of the type-1 fuzzy inference systems that are used to obtain the type-1 fuzzy weights in the output layer

- | |
|--|
| 1. (Current_Weight is lower) and (Change_Weight is lower) then (New_Weight is lower) |
| 2. (Current_Weight is lower) and (Change_Weight is upper) then (New_Weight is lower) |
| 3. (Current_Weight is upper) and (Change_Weight is lower) then (New_Weight is upper) |
| 4. (Current_Weight is upper) and (Change_Weight is upper) then (New_Weight is upper) |
| 5. (Current_Weight is lower) then (New_Weight is lower) |
| 6. (Current_Weight is upper) then (New_Weight is upper) |

Fig. 9 Rules of the type-1 fuzzy inference system used in the hidden and output layer for the neural network with type-1 fuzzy weights

We used two type-2 fuzzy inference systems to obtain the type-2 fuzzy weights and work in the same way like with the type-1 fuzzy weights.

The structure and rules (see Fig. 9) of the two type-2 fuzzy inference systems are the same of the type-1 fuzzy inference systems, the difference is in the memberships functions, Gaussian membership functions for type-2 [5, 20, 27, 35].

We used two Gaussian membership functions with their corresponding range for delimiting the inputs and outputs of the two type-2 fuzzy inference systems (see Figs. 11 and 12).

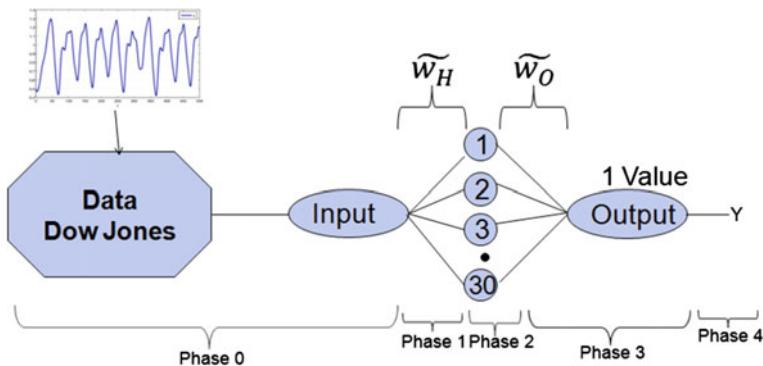


Fig. 10 Proposed neural network architecture with interval type-2 fuzzy weights

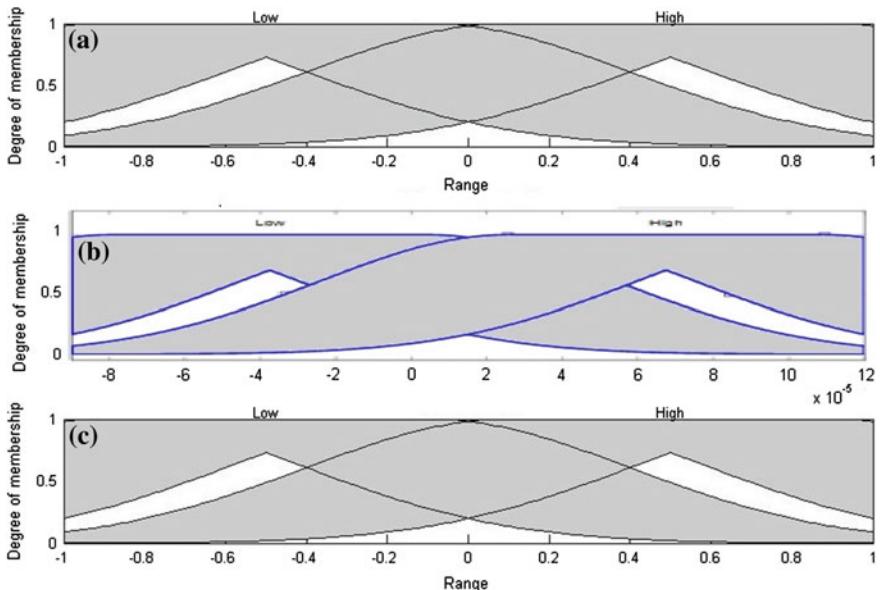


Fig. 11 Inputs (a current weight and b change of weight) and output (c new weight) of the type-2 fuzzy inference systems that are used to obtain the type-2 fuzzy weights in the hidden layer

We obtain the type-2 fuzzy inference systems incrementing and decrementing 20 percent the values of the centers of the Gaussian membership functions and the same standard deviation of the type-1 Gaussians membership functions, we use this method to obtain the footprint of uncertainty (FOU) for the type-2 fuzzy inference systems used in the neural network with type-2 fuzzy weights.

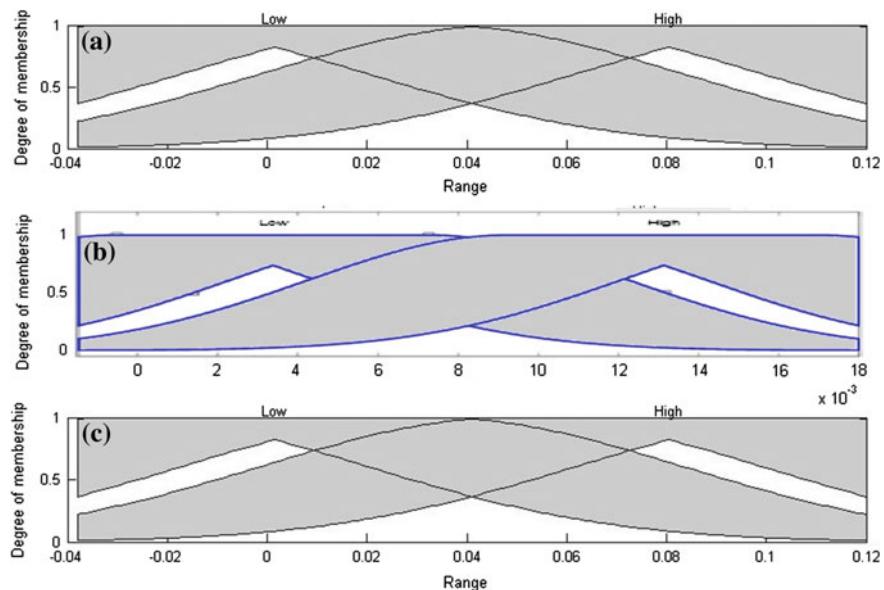


Fig. 12 Inputs (a current weight and b change of weight) and output (c new weight) of the type-2 fuzzy inference system that are used to obtain the type-2 fuzzy weights in the output layer

6 Simulation Results

We performed experiments in time-series prediction, specifically for the Dow-Jones time series.

We presented the obtained results of the experiments performed with the neural network with type-1 fuzzy weights (NNT1FW) and the neural network with type-1 fuzzy weights (NNT1FW), these results are achieved without optimizing of the neural network and the type-1 fuzzy systems, which means that all parameters of the neural network and the range and values of the membership functions of the type-1 fuzzy systems are established empirically. The average error was obtained of 30 experiments.

In Table 1, we present the prediction error obtained with the results achieved as output of NNT1FW. The best prediction error is of 0.0097 and the average prediction error is of 0.0201.

In Fig. 13 we show prediction data with type-1 fuzzy weights against the test data of the Dow-Jones time series

In Table 2, we present the prediction error obtained with the results achieved as output of NNT2FW. The best prediction error is of 0.0080 and the average prediction error is of 0.0104.

In Fig. 14 we show the prediction data with type-2 fuzzy weights against the test data of the Dow-Jones time series.

Table 1 Prediction error for the neural network with type-1 fuzzy weights for Dow-Jones time series

No.	Epochs	Network error	Prediction error
E1	100	1×10^{-8}	0.0121
E2	100	1×10^{-8}	0.0195
E3	100	1×10^{-8}	0.0198
E4	100	1×10^{-8}	0.0097
E5	100	1×10^{-8}	0.0238
E6	100	1×10^{-8}	0.0213
E7	100	1×10^{-8}	0.0231
E8	100	1×10^{-8}	0.0171
E9	100	1×10^{-8}	0.0216
E10	100	1×10^{-8}	0.0147
<i>Average prediction error</i>			0.0201

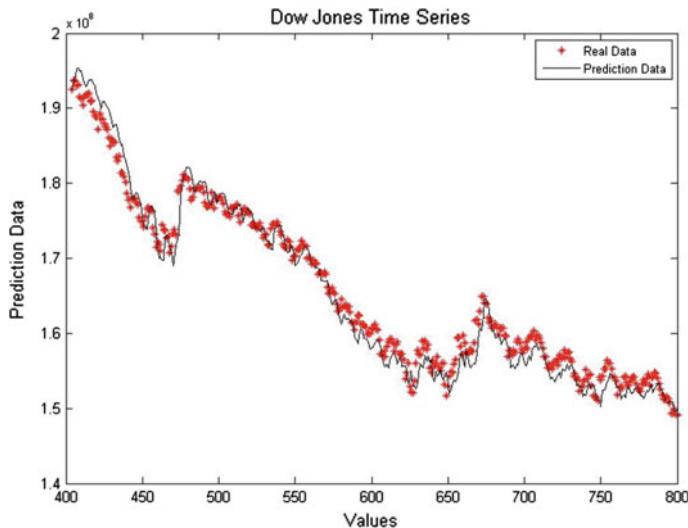


Fig. 13 Plot of the prediction data with NNT1FW against the test data of the Dow-Jones time series

7 Conclusions

In this paper, we proposed a new learning method that updates weights (type-1 or type-2 fuzzy weights) in each connection between the neurons of the layers of neural network using a type-1 or type-2 fuzzy inference system with Gaussians membership functions applied to the Dow-Jones time series.

Additionally, the neurons work internally with the type-1 or type-2 fuzzy weights and therefore, obtaining results at the output of each neuron of the neural network. The modifications performed in the neural network, that allows working

Table 2 Prediction error for the neural network with interval type-2 fuzzy weights for the Dow-Jones time series

No.	Epochs	Network error	Prediction error
E1	100	1×10^{-8}	0.0105
E2	100	1×10^{-8}	0.0082
E3	100	1×10^{-8}	0.0125
E4	100	1×10^{-8}	0.0114
E5	100	1×10^{-8}	0.0108
E6	100	1×10^{-8}	0.0095
E7	100	1×10^{-8}	0.0080
E8	100	1×10^{-8}	0.0101
E9	100	1×10^{-8}	0.0093
E10	100	1×10^{-8}	0.0117
<i>Average prediction error</i>			0.0104

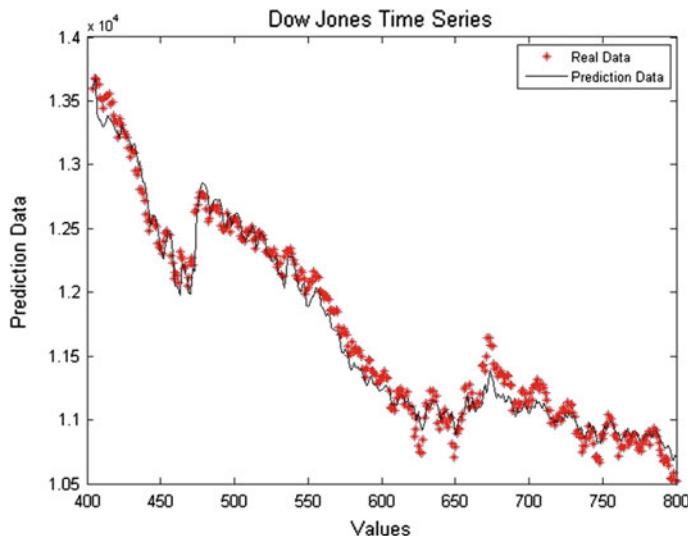


Fig. 14 Plot of the prediction data with NNT2FW against the test data of the Dow-Jones time series

with type-1 or type-2 fuzzy weights, provide the neural network with greater robustness and less susceptibility at the noise in the data of the time series.

The prediction error of 0.0080 of the neural network with type-2 fuzzy weights for the Dow-Jones time series is better than the prediction error of 0.0097 of the neural network with type-1 fuzzy weights (as shown in Tables 1 and 2).

This result is good considering that the used parameters for the neural networks at the moment are determined in an empirical way.

References

1. Abiyev, R.: A Type-2 Fuzzy Wavelet Neural Network for Time Series Prediction. Lecture Notes in Computer Science, vol. 6098, pp. 518–527 (2010)
2. Barbounis, T.G., Theocharis, J.B.: Locally recurrent neural networks for wind speed prediction using spatial correlation. *Inf. Sci.* **177**(24), 5775–5797 (2007)
3. Beale, E.M.L.: A derivation of conjugate gradients. In: Lootsma, F.A. (ed.) Numerical Methods for Nonlinear Optimization, pp. 39–43. Academic Press, London (1972)
4. Casasent, D., Natarajan, S.: A classifier neural net with complex-valued weights and square-law nonlinearities. *Neural Netw.* **8**(6), 989–998 (1995)
5. Castillo, O., Melin, P.: A review on the design and optimization of interval type-2 fuzzy controllers. *Appl. Soft Comput.* **12**(4), 1267–1278 (2012)
6. Castro, J., Castillo, O., Melin, P., Rodríguez-Díaz, A.: A hybrid learning algorithm for a class of interval type-2 fuzzy neural networks. *Inf. Sci.* **179**(13), 2175–2193 (2009)
7. Castro, J., Castillo, O., Melin, P., Mendoza, O., Rodríguez-Díaz, A.: An interval type-2 fuzzy neural network for chaotic time series prediction with cross-validation and Akaike test". In: Soft Computing for Intelligent Control and Mobile Robotics, pp. 269–285 (2011)
8. Cazorla, M., Escolano, F.: Two Bayesian methods for junction detection. *IEEE Trans. Image Proc.* **12**(3), 317–327 (2003)
9. De Wilde, O.: The magnitude of the diagonal elements in neural networks. *Neural Netw.* **10**(3), 499–504 (1997)
10. Draghici, S.: On the capabilities of neural networks using limited precision weights. *Neural Netw.* **15**(3), 395–414 (2002)
11. Feuring, T.: Learning in fuzzy neural networks. In: IEEE international conference on neural networks, vol. 2, pp. 1061–1066 (1996)
12. Fletcher, R., Reeves, C.M.: Function minimization by conjugate gradients. *Comput. J.* **7**, 149–154 (1964)
13. Gaxiola, F., Melin, P., Valdez, F., Castillo, O.: Neural network with type-2 fuzzy weights adjustment for pattern recognition of the human Iris biometrics. In: Proceedings of the 11th Mexican international conference on advances in computational intelligence, MICAI '12, vol. 2, pp. 259–270 (2012)
14. Gaxiola, F., Melin, P., Valdez, F., Castillo, O.: Optimization of type-2 fuzzy weight for neural network using genetic algorithm and particle swarm optimization. In: 5th World Congress on Nature and Biologically Inspired Computing, pp. 22–28 (2013)
15. Gaxiola, F., Melin, P., Valdez, F.: Backpropagation method with type-2 fuzzy weight adjustment for neural network learning. In: Fuzzy Information Processing Society (NAFIPS). Annual meeting of the North American, pp. 1–6 (2012)
16. Gaxiola, F., Melin, P., Valdez, F.: Genetic optimization of type-2 fuzzy weight adjustment for backpropagation ensemble neural network. In: Recent Advances on Hybrid Intelligent Systems, pp. 159–171 (2013)
17. Gaxiola, F., Melin, P., Valdez, F.: Neural network with lower and upper type-2 fuzzy weights using the backpropagation learning method. In: IFSA World Congress and NAFIPS annual meeting (IFSA/NAFIPS), pp. 637–642 (2013)
18. Gedeon, T.: Additive neural networks and periodic patterns. *Neural Netw.* **12**(4–5), 617–626 (1999)
19. Hagan, M.T., Demuth, H.B., Beale, M.H.: Neural Network Design. PWS Publishing, Boston, p. 736 (1996)
20. Hagras, H.: Type-2 fuzzy logic controllers: a way forward for fuzzy systems in real world environments. In: IEEE World Congress on Computational Intelligence, pp. 181–200 (2008)
21. Ishibuchi, H., Morioka, K., Tanaka, H.: A fuzzy neural network with trapezoid fuzzy weights, fuzzy systems. In: IEEE World Congress on Computational Intelligence, vol. 1, pp. 228–233 (1994)

22. Ishibuchi, H., Tanaka, H., Okada, H.: Fuzzy neural networks with fuzzy weights and fuzzy biases. In: IEEE international conference on neural networks, vol. 3, pp. 1650–165 (1993)
23. Jang, J.S.R., Sun, C.T., Mizutani, E.: Neuro-fuzzy and soft computing: a computational approach to learning and machine intelligence, p. 614. Prentice-Hall, Inc., New Jersey (1997)
24. Kamarthi, S., Pittner, S.: Accelerating neural network training using weight extrapolations. *Neural Netw.* **12**(9), 1285–1299 (1999)
25. Karnik, N., Mendel, J.: Applications of type-2 fuzzy logic systems to forecasting of time-series. *Inf. Sci.* **120**(1–4), 89–111 (1999)
26. Martinez, G., Melin, P., Bravo, D., Gonzalez, F., Gonzalez, M.: Modular neural networks and fuzzy sugeno integral for face and fingerprint recognition. *Adv. Soft Comput.* **34**, 603–618 (2006)
27. Melin, P.: Modular Neural Networks and Type-2 Fuzzy Systems for Pattern Recognition, pp. 1–204. Springer, Berlin (2012)
28. Meltser, M., Shoham, M., Manevitz, L.: Approximating functions by neural networks: a constructive solution in the uniform norm. *Neural Netw.* **9**(6), 965–978 (1996)
29. Moller, M.F.: A scaled conjugate gradient algorithm for fast supervised learning. *Neural Netw.* **6**, 525–533 (1993)
30. Monirul Islam, M.D., Murase, K.: A new algorithm to design compact two-hidden-layer artificial neural networks. *Neural Netw.* **14**(9), 1265–1278 (2001)
31. Neville, R.S., Eldridge, S.: Transformations of Sigma-Pi nets: obtaining reflected functions by reflecting weight matrices. *Neural Netw.* **15**(3), 375–393 (2002)
32. Phansalkar, V.V., Sastry, P.S.: Analysis of the back-propagation algorithm with momentum. *IEEE Trans. Neural Netw.* **5**(3), 505–506 (1994)
33. Powell, M.J.D.: Restart procedures for the conjugate gradient method. *Math. Program.* **12**, 241–254 (1977)
34. Salazar, P.A., Melin, P., Castillo, O.: A new biometric recognition technique based on hand geometry and voice using neural networks and fuzzy logic. In: Soft Computing for Hybrid Intelligent Systems, pp. 171–186 (2008)
35. Sepúlveda, R., Castillo, O., Melin, P., Montiel, O.: An efficient computational method to implement type-2 fuzzy logic in control applications. In: Analysis and Design of Intelligent Systems using Soft Computing Techniques, pp. 45–52 (2007)
36. Yam, J., Chow, T.: A weight initialization method for improving training speed in feedforward neural network. *Neurocomputing* **30**(1–4), 219–232 (2000)
37. Yeung, D., Chan, P., Ng, W.: Radial basis function network learning using localized generalization error bound. *Inf. Sci.* **179**(19), 3199–3217 (2009)

Evolutionary Indirect Design of Feed-Forward Spiking Neural Networks

**Andrés Espinal, Martín Carpio, Manuel Ornelas, Héctor Puga,
Patricia Melín and Marco Sotelo-Figueroa**

Abstract The present paper proposes the automatic design of Feed-Forward Spiking Neural Networks by representing several inherent aspects of the neural architecture in a proposed Context-Free Grammar; which is evolved through an Evolutionary Strategy. In the indirect design, the power of the design and the capabilities of the designed neural network are strongly related with the complexity of the grammars. The neural networks designed with the proposed grammar are tested with two well-known benchmark datasets of pattern recognition. Finally, neural networks derived from the proposed grammar are compared with other generated by similar grammars which were designed for the same purposed, the neural network design.

1 Introduction

Artificial Neural Networks (ANNs) have been successfully applied to solve problems in several fields such as pattern recognition. Several topologies have been proposed for ANNs, which are defined around three aspects: computing nodes (neuron models), communications links (synapse connections) and message types (coding schemes) [14]; which all together play an important role in the impact on ANNs' performance [6].

The Spiking Neural Networks (SNNs), which are considered as the third generation of ANNs [18], are formed by spiking neurons, which deal with infor-

A. Espinal · M. Carpio · M. Ornelas · H. Puga · M. Sotelo-Figueroa
Tecnológico Nacional de México - Instituto Tecnológico de León,
Av. Tecnológico S/N, León, GTO, Mexico
e-mail: andres.espinal@itleon.edu.mx

P. Melín (✉)
Tecnológico Nacional de México - Instituto Tecnológico de Tijuana,
Calz. del Tecnológico S/N, Tijuana, B.C., Mexico
e-mail: pmelin@tectijuana.mx

mation encoded in timing of events (spikes). SNNs are computationally stronger than sigmoid neural networks [17].

Lately, methauristic algorithms have been used for dealing with the problem of designing ANNs for solving some given problems [6, 23]. Metaheuristics can design ANNs by either direct or indirect representation. This paper focuses in the indirect representation, which represents aspects of an ANN, that is codified in some way, and the algorithm tries to design this ANN for a given problem [4].

In this work is explored the capabilities of indirect design of SNNs by means of Backus-Naur Form (BNF) grammar. The current paper proposes the modification of a grammar for designing SNNs reported in the state of the art by integrating several aspects of these neural networks. The SNNs are designed for solving pattern recognition problems and the results are compared with those obtained in [7]. The paper is organized as follows: Sect. 2 gives all the fundamentals required for this work, Sect. 3 explains the grammar designed for this work, in Sect. 4 some experiments and their results are explained and showed, and finally, Sect. 5 gives the conclusions and proposes future work.

2 Background

This paper is based on the framework for the automatic generation of SNNs to solve pattern recognition problems proposed in [7]. In this section is explained the framework and each of its components.

The framework in [7] requires as inputs: training and testing sets from the pattern recognition problem, a BNF grammar as indirect representation of the architectures of SNN and the output firing times for each class in the dataset (see Fig. 1).

The BNF grammar presented in [7] adds m Gaussian functions, that are used to encode the real valued vectors into spike trains (see Grammar 1.1), for the BNF grammar designed to generate Fully-Connected Feed-Forward Artificial Neural Networks with sigmoid functions (first approach in [4]).

The design process is carried out by the Grammatical Evolution (GE) based on Evolutionary Strategy (ES), which basically by means of the BNF grammar and the mapping process creates architectures of SNNs candidates. The GE evaluates the quality of its candidates by using a fitness function and evolves them trying to improve the candidates by minimizing their fitness values.

The GE generates words that are transformed to candidates SNNs. For calculating the quality of each SNN, three stages are required. The first stage consists in training $2K + 1$ (with $K \geq 1$) times the candidate architecture using the input training set, the desired firing output times and the supervised-learning based on ES proposed in [1]. The second stage consists in assigning to the candidate architecture, the median training process from the training set generated in the first stage. Finally, the third stage consists in obtaining the performance of the candidate architecture of SNN (trained) over unseen patterns using the input testing set. The fitness function used by the GE is showed in Eq. 1.

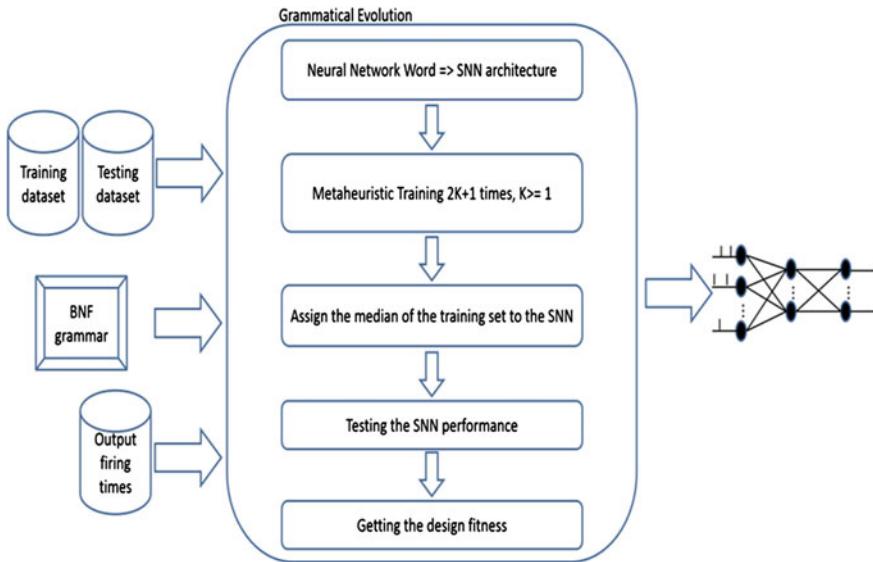


Fig. 1 Generic framework diagram for designing SNNs architectures

```

<network> |= <receptiveFields> - <layers>
<receptiveFields> |= <digit>
<layers> |= <layer> | <layer>, <layers>
<layer> |= <digit>
<digit> |= 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

```

Grammar 1.1 BNF grammar for generating fully-connected feed-forward SNNs

$$f = \frac{\#GRFs}{9} + \sum_{i=1}^{\#nhl_i} i * \frac{\#nhl_i}{9} + ((1 - perfTest) * 100) \quad (1)$$

where:

- $\#GRFs$ is the number of Gaussian Receptive Fields (GRFs) selected by the GE.
- $\#nhl_i$ is the number of neurons in the i -th hidden layer set by the GE.
- $perfTest$ is the performance of the trained SNN with the candidate architecture over the input testing set.

In the following section are described the components required for implement the framework and design SNN.

2.1 Spiking Neural Networks

The Spiking Neural Networks (SNNs) are formed by the interconnection of spiking neurons, which handle the information by timing of events (spikes). When a SNN is applied to solve pattern recognition problems, original patterns can not be fed into the SNN; they need to be transformed as spikes in an interval of time by using an encoding scheme. Several encoding schemes have been proposed, such as the Gaussian Receptive Fields (GRFs); this encoding scheme has been extensively used in [1, 2, 22]. Basically, this encoding scheme requires for encoding a variable, m neurons (with Gaussian functions) used for covering the whole range of the variable, γ as a coefficient for setting the width of Gaussian functions and the encoding simulation time t .

After transforming original vectors into spikes in an interval of time, they can be fed into the SNN. The architecture of the SNN can vary depending on the kind of the problem to solve. This work focuses on SNNs with architecture known as Fully-Connected Feed-Forward as used in [1, 2, 12]. There are several spiking neuron models that can be implemented in a SNN, this work use the Spike Response Model which is detailed in next section.

2.1.1 Gaussian Receptive Fields

The traditional form of patterns as multidimensional raw data, which consist of real values can't be used to feed a SNNs in a simulation process. The patterns need to be transformed into temporal patterns (a set of events in time or spikes as known as spike trains) before being processed by the SNN.

In [3], was proposed an encoding scheme to generate firing times from real values; the Gaussian Receptive Fields (GRFs). This scheme enables the representation of continuously valued input variables by a population of neurons with graded and overlapping sensitivity profiles, such as Gaussian activation functions (the receptive fields). To encode values into a temporal pattern, it is sufficient to associate highly stimulated neurons with early firing times and less stimulated neurons with later (or no) firing times. Each input variable is encoded independently, it means that each input dimension is encoded by an array of one-dimensional receptive fields; the GRFs constitute a biologically plausible and well studied method for representing real-valued parameters [2].

In [13] is given a detailed definition about the construction and use of the GRFs. Each input datum is fed to all the conversion neurons covering the whole data range. For a range $[n_{min}, n_{max}]$ of a variable n , m neurons were used with GRF. For the i -th neuron coding for variable n , the center of the Gaussian function is set to C_i according to Eq. 2.

$$C_i = n_{min} + \frac{2i - 3}{2} * \frac{n_{max} - n_{min}}{m - 2}, \quad m > 2 \quad (2)$$

The width w of each Gaussian function is set according to Eq. 3. Where $1 \leq \gamma \leq 2$ is a coefficient of multiplication which is inversely proportional to the width of the Gaussian functions.

$$w = \frac{n_{max} - n_{min}}{\gamma(m - 2)} \quad (3)$$

Using the centers of each neuron/Gaussian function and their width, the magnitude of firing $f(x)$ for each sample points of the input x is calculated using Eq. 4.

$$f_i(x) = e^{-\frac{(x - C_i)^2}{2w^2}} \quad (4)$$

The magnitude values are then converted to time delay values by associating the highest magnitude value to a value close to zero milliseconds and neurons with lower firing magnitude are associated with a time delay value close to the limit maximum of time of simulation. Conversion from magnitude to time delay values is done using Eq. 5. Where t is the total time of simulation.

$$T_i = (1 - f_i) * t, \quad i = 1, 2, \dots, m \quad (5)$$

Time delay values greater than some value $t_{threshold} < t$ are coded no to fire as they are consider being insufficiently excited. Therefore, neurons with time delay value between zero and $t_{threshold}$ milliseconds carry all the encoded information of the input data.

2.1.2 Spike Response Model

The Spike Response Model (SRM) [9, 10] is an approximation of the dynamics of the integrate-and-fire neuron. For this work, the spiking neurons use the time-to-first-spike as coding scheme for sending/receiving messages. Due to the coding scheme being used, a reduced version of the SRM is implemented, which has been used in [1, 2, 22].

The reduced SRM is defined according [1] as follows. Let us consider that a neuron j has a set Γ_j of immediate predecessors called presynaptic neurons and receives a set of spikes with firing times t_i , $i \in \Gamma_j$. Neurons fire when their state variable $x(t)$, called membrane potential, reaches a certain threshold θ . The internal state of a neuron is determined by Eq. (6), where w_{ji} is the synaptic weight to modulate $y_i(t)$, which is the unweighted postsynaptic potential of a single spike coming from neuron i and impinging on neuron j .

$$x_j(t) = \sum_{i \in \Gamma_i} w_{ji} y_i(t) \quad (6)$$

The unweighted contribution $y_i(t)$ is given by Eq. (7), this uses a function $\varepsilon(t)$; which describes the form of the postsynaptic potential and its input parameter are formed by the next three values: t is the current time, t_i is the firing time of the presynaptic neuron i and d_{ji} is the associated synaptic delay.

$$y_i(t) = \varepsilon(t - t_i - d_{ji}) \quad (7)$$

The form of the postsynaptic potential $\varepsilon(t)$ is given by Eq. (8), the function has a τ parameter, that is the membrane potential time constant defining the decay time of the postsynaptic potential.

$$\varepsilon(t) = \begin{cases} \frac{t}{\tau} e^{1-\frac{t}{\tau}} & \text{if } t > 0 \\ 0 & \text{else} \end{cases} \quad (8)$$

Each neuron fires once at most, the firing time t_j of neuron j is determined as the first time the state variable crosses the threshold from below.

2.2 Evolutionary Strategy

The Evolutionary Strategies (ES) [20], deal natively with problems in real domain. In [1] was designed a Self-Adaptive ES originally for training SNNs, but it could be used to solve other optimization problems. In this ES each population member consists of n -dimensional vectors. The population at any given generation g is denoted as $P(g)$. Each individual is taken as a pair of real-valued vectors, (x_i, η_i) , where x_i 's are objective variables which depend of the optimization problem, and η_i 's are standard deviations for mutations. Each individual generates a single offspring (x'_i, η'_i) , where each variable $x'_i(j)$ of the offspring can be randomly defined by either Eq. (9) (local search) or Eq. (10) (global search) and the standard deviation for mutation of the offspring is defined by Eq. (11).

$$x'_i(j) = x_i(j) + \eta_i(j)N_j(0, 1) \quad (9)$$

$$x'_i(j) = x_i(j) + \eta_i(j)\delta_j \quad (10)$$

$$\eta'_i(j) = \eta_i(j)\exp(\tau' N(0, 1) + \tau N_j(0, 1)) \quad (11)$$

where:

- $N(0, 1)$ denotes a normally distributed one dimensional random number with $\mu = 0$ and $\sigma = 1$.

- $N_j(0, 1)$ indicates that the random number is generated anew for each value of j .
- δ_j is a Cauchy random variable, and it is generated anew for each value of j (Scale = 1).
- Factor $\tau = \frac{1}{\sqrt{2\sqrt{n}}}$
- Factor $\tau' = \frac{1}{\sqrt{2n}}$

The Self-adaptive ES is presented in the Algorithm 1.

Algorithm 1 Self-adaptive ES

- 1: Generate the initial population of μ individuals.
 - 2: Evaluate the fitness score for each individual $(x_i, \eta_i), i = 1, \dots, \mu$ of the population based on the fitness function.
 - 3: **while** the maximum iteration is not reached **do**
 - 4: Each parent (x_i, η_i) generates a single offspring (x'_i, η'_i)
 - 5: Calculate the fitness of each offspring $(x'_i, \eta'_i), i = 1, \dots, \mu$.
 - 6: Generate a new population $P(g)$ using tournament selection and elitism to keep track of the best individual at each generation
 - 7: **end while**
-

2.3 Grammatical Evolution

Grammatical Evolution (GE) [21] is a grammar-base form of Genetic Programming (GP) [16]. GE joins the principles from molecular biology, which are used by the GP, and the power of formal grammars. Unlike GP, the GE adopts a population of lineal genotypic integer strings, or binary strings, which are transformed into functional phenotypic through a genotype-to-phenotype mapping process, this process is also known as Indirect Representation [8]. This transformation is governed through a Backus Naur Form grammar (BNF). Genotype strings are evolved with no knowledge of their phenotypic equivalent, only use the fitness measure.

Eventhough the GE uses the Genetic Algorithm (GA) [5, 11, 21] as search strategy it is possible to use another search strategy like the Particle Swarm Optimization, called Grammatical Swarm (GS) [19]. In the GE each individual is mapped into a program using the BNF.

2.3.1 Mapping Process

When approaching a problem using GE, initially a BNF grammar must be defined. This grammar specifies the syntax of desired phenotypic programs to be produced by GE. The development of a BNF grammar also affords the researcher the ability to incorporate domain biases or domain specific functions.

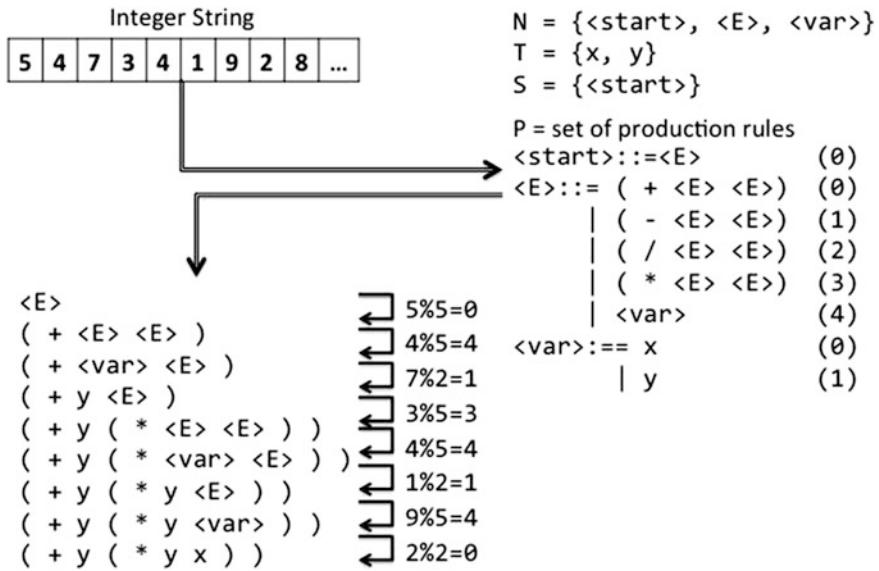


Fig. 2 An example of a transformation from genotype to phenotype using a BNF grammar. It begins with the start symbol, if the production rule from this symbol is only one rule, then the production rule gets instead of the start symbol, and the process begins to choose the productions rules base on the current genotype. It is taking each genotype and the non-terminal symbol from the left to realize the next production using Eq. (12) until all the genotypes are mapped or there aren't more non-terminals in the phenotype

A BNF grammar is made up of the tuple N, T, P, S ; where N is the set of all non-terminal symbols, T is the set of terminals, P is the set of production rules that map $N \rightarrow T$, and S is the initial start symbol where $S \in N$. Where there are a number of production rules that can be applied to a non-terminal, a “|” (or) symbol separates the options.

Using the grammar as the GE input, Eq. (12) is used to choose the next production based on the non-terminal symbol.

$$Rule = c\%r \quad (12)$$

where c is the codon value and r is the number of production rules available for the current non-terminal.

An example of the mapping process employed by GE is shown in Fig. 2.

3 Methodology

This paper proposes an extension to the BNF grammar presented in [7]. The grammar proposed herein keeps the characteristics of the BNF grammar in [7] and adds more aspects to define the whole SNN. The BNF grammar defines the

```

⟨network⟩ ⊨ ⟨decayFactor⟩ – ((receptiveFields)), · · · , ((receptiveFields)) – ⟨layers⟩
⟨decayFactor⟩ ⊨ ⟨digit⟩
⟨receptiveFields⟩ ⊨ ⟨gamma⟩, ⟨m⟩
⟨gamma⟩ ⊨ 1.0 | 1.1 | 1.2 | 1.3 | 1.4 | 1.5 | 1.6 | 1.7 | 1.8 | 1.9 | 2.0
⟨m⟩ ⊨ ⟨digit⟩
⟨layers⟩ ⊨ ⟨layer⟩ | ⟨layer⟩, ⟨layers⟩
⟨layer⟩ ⊨ ⟨digit⟩
⟨digit⟩ ⊨ 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

```

Grammar 1.2 BNF grammar for SNN design based on the grammar proposed in [7]

characteristics of the neuron and the encoding scheme for each feature, this generates more SNNs than the previous grammar by increasing the characteristics for defining the SNN, in other words this grammar allows to expand the search space when is used by the Grammatical Evolution (see Grammar 1.2).

The aspects integrated in the proposed grammar are:

- Decay time of the postsynaptic potential τ for all neurons in the SNN (Eq. 8).
- Coefficient of multiplication γ (Eq. 3) and number of Gaussian functions m (Eqs. 2 and 3) of the encoding scheme for each feature in the original input pattern.
- The quantity of hidden layers and the neurons per hidden layer in the SNN.

4 Experiments and Results

The capabilities of the proposed grammar were tested following the same experimentation used in [7]. The experimentation was carried out with two well-known benchmark datasets of pattern recognition problems from the UCI Machine Learning Repository. Next is the description of each dataset:

- Iris plant: The iris plant dataset contains 3 classes (Iris Setosa, Iris Versicolour and Iris Virginica). There are 50 instance patterns for each class. Each pattern is described by 4 attributes.
- Wine: The wine dataset contains 3 classes. There are 59, 41 and 48 instance patterns for classes 1, 2 and 3 respectively. Each pattern is described by 13 attributes.

The K-folds cross validation [15] was used to create test instances for each dataset and it was used a value of $K = 10$, giving a set of 10 test instances. The same configuration was used through all the test instances for the Grammatical Evolution and the Supervised-learning based on ES process. The configuration of the experiments is presented next.

Table 1 Classification performance of the designed SNNs (using grammar in [7]) over training and testing patterns through 10 tests for Iris plant dataset

# test	Architecture	Training	Testing
1	4(5)-3-1	0.8740	1.0000
2	4(4)-7-1	0.9481	1.0000
3	4(5)-2-1	0.8592	1.0000
4	4(3)-7-1	0.9111	1.0000
5	4(9)-6-1	0.9185	1.0000
6	4(6)-6-1	0.8666	1.0000
7	4(7)-6-1	0.9407	1.0000
8	4(5)-3-1	0.9777	1.0000
9	4(5)-5-1	0.9333	1.0000
10	4(7)-4-1	0.9333	1.0000
Average accuracy		0.91625	1.0000

GE based on ES. The parameters size of population = 30, function calls = 600 and boundaries $\in [0, 255]$.

Supervised-learning based on ES. The parameters size of population = 30, function calls = 15,000, weights $\in [-1000, 1000]$ and delays $\in [0.1, 16]$.

GRFs. m could be from 1 to 9 and it is defined by the framework. For the grammar in [7] $\gamma = 1.5$ and for the grammar proposed here this is defined by the framework. The parameter t = 4 ms.

SRM. The parameters $\theta = 1.0$ mV. For the grammar in [7] $\tau = 9.0$ ms and for the grammar proposed here this is defined by the framework. The simulation start time = 5 ms and simulation end time = 19 ms.

Output firing times for the classes 1, 2 and 3 were 6.0, 10.0, 14 ms respectively.

The results reported in [7] for Iris Plant and Wine datasets are showed in Tables 1 and 3. The results obtained with the proposed grammar herein are showed in Tables 2 and 4. In all tables the architecture column describes the structure of a

Table 2 Classification performance of the designed SNNs (using grammar proposed herein) over training and testing patterns through 10 tests for Iris plant dataset

# test	Architecture	Training	Testing
1	$\tau = 6$, Net: [(1.7, 3), (1.4, 3), (1.0, 7), (1.3, 9)]-5-1	0.9851	1.0000
2	$\tau = 9$, Net: [(1.5, 5), (1.0, 3), (1.2, 5), (2.0, 5)]-6-1	0.9259	1.0000
3	$\tau = 5$, Net: [(1.8, 3), (1.9, 6), (1.4, 5), (1.3, 4)]-2-1	0.9407	0.8000
4	$\tau = 2$, Net: [(1.0, 3), (1.0, 8), (2.0, 3), (1.1, 8)]-7-1	0.9111	1.0000
5	$\tau = 6$, Net: [(1.2, 6), (1.4, 3), (1.2, 8), (1.4, 8)]-3-1	0.9407	0.9333
6	$\tau = 8$, Net: [(1.1, 4), (1.1, 4), (1.7, 5), (1.9, 8)]-4-1	0.9555	0.9333
7	$\tau = 2$, Net: [(1.6, 9), (2.0, 8), (1.5, 6), (1.9, 3)]-2-1	0.8148	0.8000
8	$\tau = 6$, Net: [(1.2, 5), (1.7, 3), (1.9, 7), (1.0, 3)]-7-1	0.9111	1.0000
9	$\tau = 8$, Net: [(1.7, 5), (1.2, 3), (1.4, 9), (1.9, 4)]-4-1	0.9407	1.0000
10	$\tau = 3$, Net: [(1.3, 7), (1.7, 3), (1.8, 3), (1.6, 4)]-6-1	0.9555	1.0000
Average accuracy		0.9281	0.9466

Table 3 Classification performance of the designed SNNs (using grammar in [7]) over training and testing patterns through 10 tests for wine dataset

# test	Architecture	Training	Testing
1	13(7)-3-1	0.6037	0.6842
2	13(4)-8-1	0.8187	0.9444
3	13(8)-6-1	0.8000	0.8333
4	13(7)-7-1	0.7687	0.9444
5	13(7)-9-1	0.7750	0.8333
6	13(5)-7-1	0.7937	0.9444
7	13(8)-3-1	0.8062	0.8888
8	13(4)-8-1	0.7625	0.8333
9	13(7)-8-1	0.8757	0.8823
10	13(4)-6-1	0.9012	1.0000
	Average accuracy	0.79054	0.87884

SNN. In Tables 1 and 3 the sign “-” separates the neurons into each layer, the first (input) layer is described by the number of problem’s features and the m Gaussian functions of the GRFs inside parenthesis. In Tables 2 and 4, first the neuron parameter τ is set, next it is described the neural structure; the sign “-” separates the neurons into each layer, the first (input) layer is described by the GRFs for each problem’s feature where inside the parenthesis is showed de GRFs’s parameters: γ and the m Gaussian functions.

Table 4 Classification performance of the designed SNNs (using grammar proposed herein) over training and testing patterns through 10 tests for wine dataset

# test	Architecture	Training	Testing
1	$\tau = 7$, Net: [(1.9, 3), (1.0, 4), (1.5, 8), (1.6, 3), (1.6, 3), (1.2, 8), (1.3, 6), (1.1, 7), (1.6, 10), (2.0, 8), (1.5, 6), (1.8, 9), (1.4, 11)]-5-1	0.4528	0.4736
2	$\tau = 3$, Net: [(1.6, 8), (1.3, 8), (1.1, 11), (2.0, 8), (1.7, 7), (1.2, 10), (1.2, 10), (1.9, 11), (1.0, 8), (1.4, 8), (1.2, 4), (2.0, 11), (1.6, 8)]-9-1	0.4687	0.3888
3	$\tau = 3$, Net: [(1.2, 10), (1.9, 8), (1.6, 11), (1.7, 10), (1.3, 3), (1.8, 11), (1.1, 10), (1.3, 4), (1.6, 5), (1.1, 4), (1.5, 3), (1.6, 8), (1.2, 7)]-4-7-1	0.3875	0.3888
4	$\tau = 4$, Net: [(1.0, 10), (2.0, 6), (2.0, 11), (1.5, 6), (1.8, 10), (1.2, 9), (1.7, 5), (1.1, 3), (1.0, 6), (1.5, 9), (1.2, 9), (1.5, 11), (1.3, 8)]-9-1	0.6562	0.6111
5	$\tau = 3$, Net: [(1.0, 4), (1.0, 6), (2.0, 6), (1.1, 8), (1.2, 8), (1.2, 6), (1.5, 7), (1.5, 5), (1.1, 8), (1.7, 10), (1.3, 7), (1.4, 5), (1.1, 10)]-5-1	0.7312	0.7222
6	$\tau = 9$, Net: [(1.7, 11), (1.4, 7), (1.1, 8), (1.3, 8), (1.7, 6), (1.2, 9), (1.4, 8), (1.0, 9), (1.9, 7), (1.9, 6), (1.6, 10), (1.7, 10), (1.0, 8)]-9-1-1	0.5437	0.4444
7	$\tau = 4$, Net: [(1.0, 6), (1.8, 3), (2.0, 7), (2.0, 9), (1.1, 10), (1.0, 11), (1.1, 8), (1.8, 10), (1.1, 10), (1.2, 7), (1.2, 11), (1.8, 5), (1.6, 7)]-6-1	0.8375	0.8888
8	$\tau = 5$, Net: [(2.0, 11), (1.1, 5), (2.0, 6), (1.9, 10), (1.1, 5), (1.2, 11), (1.1, 3), (1.5, 10), (1.2, 8), (2.0, 4), (2.0, 11), (1.1, 7), (1.4, 8)]-7-1	0.6894	0.8235
9	$\tau = 5$, Net: [(1.9, 4), (1.2, 5), (1.9, 4), (1.6, 11), (1.8, 11), (1.2, 3), (1.4, 9), (2.0, 5), (1.9, 8), (1.1, 9), (1.8, 5), (1.8, 10), (1.1, 8)]-4-1	0.6543	0.6250
10	$\tau = 9$, Net: [(1.7, 7), (1.8, 11), (1.3, 8), (1.8, 3), (1.1, 7), (1.0, 7), (1.7, 6), (1.9, 10), (1.4, 8), (1.4, 6), (1.8, 9), (1.9, 6), (1.0, 5)]-7-1	0.6187	0.8888
	Average accuracy	0.6040	0.6255

5 Conclusions

In this paper is presented an extension of the grammar proposed in [7] for the automatic generation of SNN to solve pattern recognition problems.

The framework using the grammar proposed in this work, obtained similar results to those reported in [7] for the Iris plant dataset, however for the wine dataset, only some results are similar to those obtained by the grammar in [7].

The proposed grammar herein can describe more SNNs and does it in a more detailed way, it is evident that this description increases the search space for the Grammatical Evolution.

Authors propose to improve the results using the grammar proposed herein, by testing other metaheuristics and looking for a way to avoid the training process which is costly.

Acknowledgments The authors thank to Consejo Nacional de Ciencia y Tecnología (CONACyT) and Instituto Tecnológico de México - *Instituto Tecnológico de León* for the support to this research.

References

1. Belatreche, A.: Biologically Inspired Neural Networks: Models, Learning, and Applications. VDM Verlag, Saarbrücken, Germany (2010)
2. Bohte, S.M., Kok, J.N., LaPoutre, H.: Error-backpropagation in temporally encoded networks of spiking neurons. *Neurocomputing* **48**, 17–37 (2002)
3. Bohte, S.M., Poutre, H.La, Kok, J.N.: Unsupervised clustering with spiking neurons by sparse temporal coding and multilayer RBF networks. *IEEE Trans. Neural Netw.* **13**, 426–435 (2002)
4. De Mingo Lopez, L.F., Gomez Blas, N., Arteta, A.: The optimal combination: grammatical swarm, particle swarm optimization and neural networks. *J. Comput. Sci.* **3**(1–2), 46–55 (2012)
5. Dempsey, I., O'Neill, M., Brabazon, A.: Foundations in grammatical. In: Foundations in Grammatical Evolution for Dynamic Environments, vol. 194. Springer, New York (2009)
6. Ding, S., Li, H., Su, C., Yu, J., Jin, F.: Evolutionary artificial neural networks: a review. *Artif. Intell. Rev.* **39**(3), 251–260 (2013)
7. Espinal, A., Carpio, M., Ornelas, M., Puga, H., Melin, P., Sotelo-Figueroa, M.: Developing architectures of spiking neural networks by using grammatical evolution based on evolutionary strategy. In Martinez-Trinidad, J., Carrasco-Ochoa, J., Olvera-Lopez, J., Salas-Rodriguez, J., Suen, C. (eds.) *Pattern Recognition. Lecture Notes in Computer Science*, vol. 8495, pp. 71–80. Springer, Berlin (2014)
8. Fang, H.-l., Ross, P., Corne, D.: A promising genetic algorithm approach to jobshop scheduling, rescheduling, and open-shop scheduling problems. In Proceedings of the 5th international conference on genetic algorithms, pp. 375–382. Morgan Kaufmann, Massachusetts (1993)
9. Gerstner, W.: Time structure of the activity in neural network models. *Phys. Rev. E* **51**(1), 738–758 (1995)
10. Gerstner, W., Kistler, W.: Spiking neuron models: single neurons, populations, plasticity. Cambridge University Press, Cambridge (2002)

11. Holland, J.: Adaptation in natural and artificial systems. University of Michigan Press, Michigan (1975)
12. Hong, S., Ning, L., Xiaoping, L. Qian, W.: A cooperative method for supervised learning in spiking neural networks. In: IEEE on CSCWD, pp. 22–26 (2010)
13. Johnson, C., Roychowdhury, S., Venayagamoorthy, G.K.: A reversibility analysis of encoding methods for spiking neural networks. In: IJCNN, pp 1802–1809 (2011)
14. Judd, J.S.: Neural network design and the complexity of learning. In: Neural Network Modeling and Connectionism Series. Massachusetts Institute Technology, Massachusetts (1990)
15. Kohavi, R.: A study of cross-validation and bootstrap for accuracy estimation and model selection. In: IJCAI, pp. 1137–1145 (1995)
16. Koza, J.R., Poli, R.: Genetic programming. In Burke, E.K., Kendall, G. (eds.) Search methodologies: introductory tutorials in optimization and decision support techniques, pp. 127–164. Kluwer, Boston (2005)
17. Maass, W.: Noisy spiking neurons with temporal coding have more computational power than sigmoidal neurons, pp. 211–217. MIT Press, Massachusetts (1996)
18. Maass, W.: Networks of spiking neurons: the third generation of neural network models. Neural Netw. **10**(9), 1659–1671 (1997)
19. O'Neill, M., Brabazon, A.: Grammatical differential evolution. In: International conference on artificial intelligence (ICAI '06), CSEA Press, Las Vegas (2006)
20. Rechenberg, I.: Evolutions Strategie: optimierung technischer systeme nach prinzipien der biologischen evolution. Frommann-Holzboog (1973)
21. Ryan, C., Collins, J., O'Neill, M.: Grammatical evolution: evolving programs for an arbitrary language. In: Proceedings of the first European workshop on genetic programming. Lecture notes in computer science 1391, pp. 83–95. Springer, Berlin (1998)
22. Shen, H., Liu, N., Li, X., Wang, Q.: A cooperative method for supervised learning in spiking neural networks. In: IEEE on CSCWD, pp. 22–26 (2010)
23. Yao, X.: Evolving artificial neural networks. Proc. IEEE **87**(9), 1423–1447 (1999)

Cellular Neural Network Scheme for Image Binarization in Video Sequence Analysis

Mario I. Chacon-Murguia and Juan A. Ramirez-Quintana

Abstract This chapter presents a novel cellular neural network architecture for image binarization in video sequence. The cellular network is part of a neuro-inspired system used to detect dynamic objects in video sequences. Among its novelty is that besides binarization it is able to reduce also noise, and its parameters are self-adapted. Qualitative findings are used to show the advantage of using the cellular network.

1 Introduction

Neuro-inspired computation is an emergent area that provides new alternatives in real world applications. Intelligent vision systems, IVS, as well as other human activity monitoring, HAM, applications represent important real world applications where neuro-inspired systems can generate important contributions. Detection of dynamic objects in video sequences is a paramount task in IVS, and HAM. Methods for dynamic object detection involving artificial neural models, ANN, have been reported in the literature; PCNN [1], CNN [2–4], PNN [5] and SOM [6–10]. The SOM model has also been used to represent motion perception mechanisms based on neuro-inspired principles [10, 11].

Based on the previous mentioned literature, it can be stated that there exist two forms to achieve dynamic object motion analysis using ANN and neuro-inspired models; motion perception systems like in [1, 4, 10, 11], and background modeling systems [6–9].

M.I. Chacon-Murguia (✉) · J.A. Ramirez-Quintana
Visual Perception Applications on Robotic Lab, Chihuahua Institute of Technology,
Chihuahua, Mexico
e-mail: mchacon@itchihuahua.edu.mx

J.A. Ramirez-Quintana
e-mail: jaramirez@itchihuahua.edu.mx

In this work a cellular neural network, CNN, to improve dynamic object detection with self-adapting parameters is reported. The CNN is part of a neuro-inspired model termed SOM-CNN, designed for dynamic object detection in video sequences, and which also includes a SOM like network to estimate the video background.

The organization of the work is as follows. Section 2 describes the SOM-CNN model including the SOM like, and CNN networks. The results are presented in Sect. 3, and the conclusions are discussed in Sect. 4.

2 SOM-CNN Video Object Detection Scheme

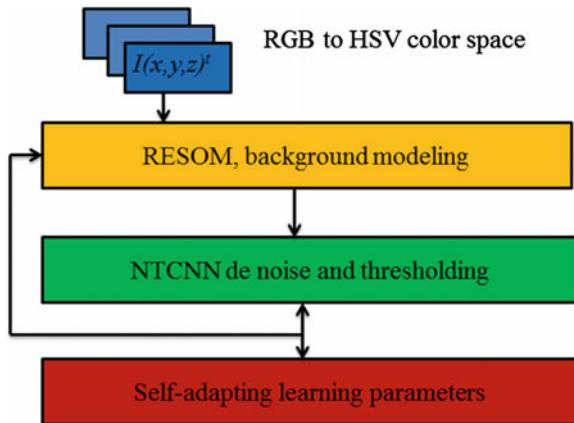
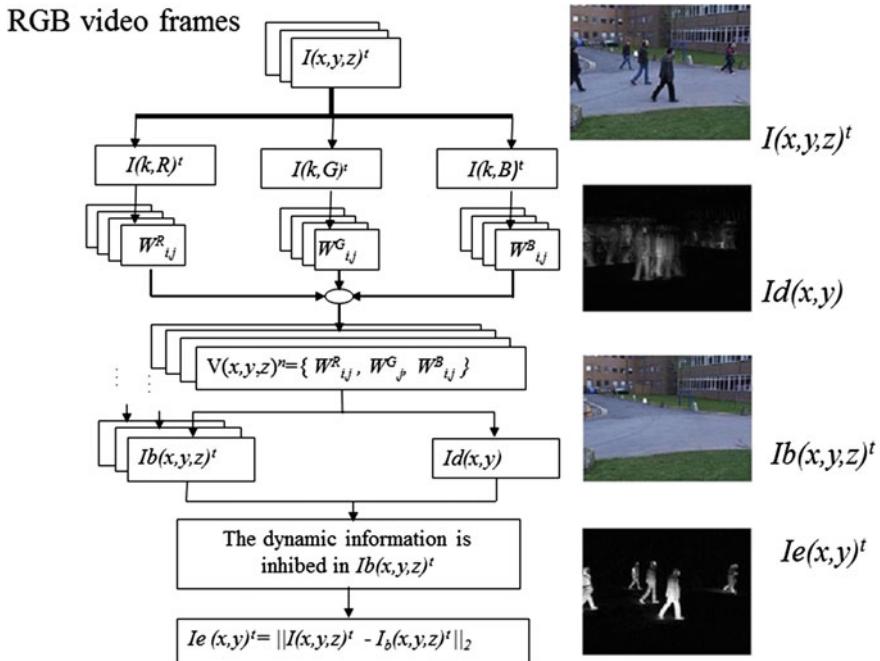
The self-organizing map cellular network, SOM-CNN, is a bio-inspired model designed for video analysis. The model is composed of three main modules; background modeling, noise and thresholding, and self-adapting parameters. The cellular neural network scheme for image binarization, neighbor thresholding cellular network NTCNN in video sequence analysis is embedded in the noise and thresholding module. In as much as the cellular network works on the results of the background modeling, the cellular neural network scheme is described previous to the explanation of the NTCNN.

2.1 SOM-CNN General Scheme

The SOM-CNN is a method to detect dynamic objects in video sequences based on the well-known background subtraction method. In this method a background model is estimated from the video sequence and then subtracted from the current frames to detect the dynamic objects. SOM-CNN achieves a background modeling task based on the SOM retinotopic network, RESOM, and a binarization to locate dynamic objects in video sequences by means of the NTCNN. Unlike other SOM based methods, RESOM is inspired in learning mechanism of the cortical layers of simple cells of V1. The NTCNN is a neural network architecture based on CNN and inspired from some parts of the model LAMINART. The general scheme of the SOM-CNN is shown in Fig. 1. In the next sections the RESOM and NTCNN networks are described.

2.2 RESOM Background Modeling

The RESOM is responsible of generating a background model estimation based on the analysis of the RGB color information. A detailed scheme is illustrated in Fig. 2. The input to this network is a color frame defined as $I(x, y, z)^t$, where $z = \{R, G, B\}$. This frame is decomposed in each color channel and fed into three different

**Fig. 1** SOM-CNN model**Fig. 2** RESOM model

RESOMs. Each RESOM is composed of M of neurons, in this work 4. The neurons are represented by the weight retinotopic maps $w(k, i, j)^z$. Each channel is vectorized to form the vectors $I(k, R)^t$, $I(k, G)^t$, and $I(k, B)^t$ to be the inputs to the networks $w(k, i, j)^R$, $w(k, i, j)^G$, and $w(k, i, j)^B$ respectively. Once the RESOM processes the

frame for N_s iterations, the resulting weights are ordered in an array of four images $V_w(x, y, z)^{m,t}$, where:

$$m = j + (i - 1)J \quad (1)$$

In this case, J is the maximum dimension of j , $m = 1, \dots, 4$. $V_w(x, y, z)^{m,t}$ is used to define $I_b(x, y, z)^t$ and $I_d(x, y)^t$. $I_b(x, y, z)^t$ corresponds to the expected value of $V_w(x, y, z)^{m,t}$. $I_d(x, y)^t$ is the sum of the image $V_w(x, y, z)^{m,t}$ differences. $I_d(x, y)^t$ is used to locate and inhibit information in $I_b(x, y, z)^t$ related to dynamic objects. Finally, dynamic object detection is achieved by background subtraction.

The RESOM network has a winner takes all competition given by the distance between the weight vectors $W(m)^z$ and the input frame $I(x, y, z)^t$ transformed into $I(k, z)^t$, computed by

$$\eta(m)^{z,n} = \sum_k |I(k, z)^t - \omega(k, m)^{z,n}| \quad (2)$$

K is the number of pixels in each color channel of $I(k, z)^t$. $\eta(m)^{z,n}$ is computed in each iteration n , to obtain the winner neurons.

For each network z , the neurons with the lowest response $\eta(m)^{z,n}$ are considered the winners in their respective neighborhood of neurons. Figure 3 shows a more specific illustration of the RESOM architecture.

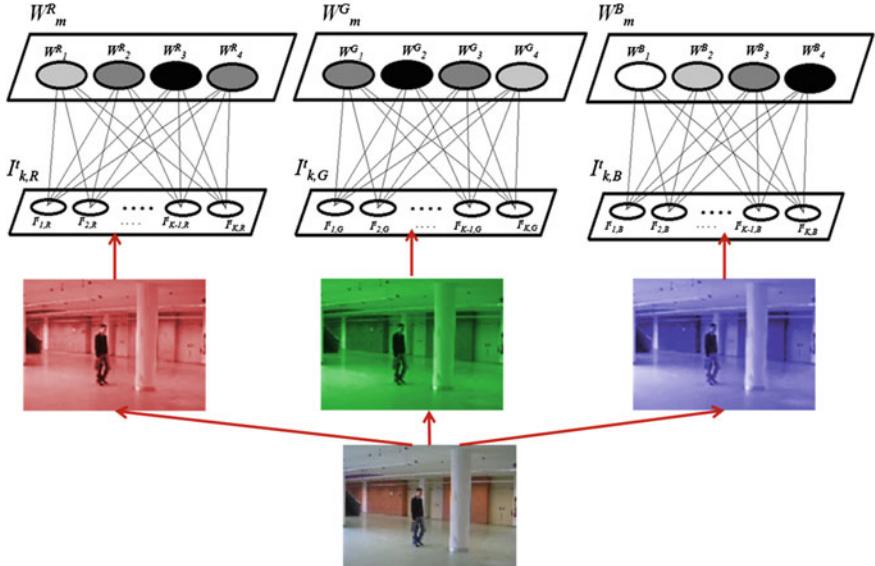


Fig. 3 RESOM architecture for $M = 4$

The neuron weights are updated in each iteration with a Hebbian learning scheme following the next rule

$$\begin{aligned}\omega(k, m)^{z,n+1} &= \omega(k, m)^{z,n} + \Delta\omega(k, m)^{z,n} \\ \Delta\omega(k, m)^{z,n} &= \alpha[I(k, z) - \omega(k, m)^{z,n}] \beta(m)^z\end{aligned}\quad (3)$$

where n is the number of iterations the RESOM processes the frame $I(k, z)^t$, α is the learning rate, $\beta(m)^z$ is a neighborhood function. α and $\beta(m)^z$ are based on the Hebbian parameters of the SOM retinotopic map [12]. In this work they are defined as

$$\alpha = \max \left[\exp \left(-\frac{6t}{T_f} \right), \alpha_s \right] \quad (4)$$

where T_f defines the exponential decay, in this work $T_f = 15$, α_s is a parameter that maintains the learning of the RESOM different from zero during the video sequence when $t \rightarrow \infty$.

The neighborhood function characterizes the retinotopic behavior in the weight vectors $W(m)^z$, and it is given by:

$$\beta(m)^z = \exp \left(-\frac{(m_w^z - m)^2}{\sigma_\beta^2} \right) \quad (5)$$

where m_w^z is the winner neuron index in the network z , and σ_β is the neighborhood radius computed by

$$\sigma_\beta = \max \left[13.5 \exp \left(-\frac{5t}{T_f} \right), \sigma_s \right] \quad (6)$$

σ_s maintains the neighborhood radius different from zero during the video sequence when $t \rightarrow \infty$.

Figure 4 shows the RESOM background estimation of each color channel for $M = 4$, where the weights of the neurons were ordered to form 4 images in the RGB color space with the next relation

$$V_w(k, z)^{m,t} = \omega(k, m)^{z,n} \quad (7)$$

Then, $V_w(k, z)^{m,t}$ is transformed to $V_w(x, y, z)^{m,t}$ and after the first frame, the weights learn the scene and the expected value of $V_w(x, y, z)^{m,t}$ is computed by:

$$I_b(x, y, z)^t = \frac{1}{M} \sum_{m=1}^M V_w(x, y, z)^{m,t}, \quad m = 1, \dots, M \quad (8)$$

that is considered the initial background model.

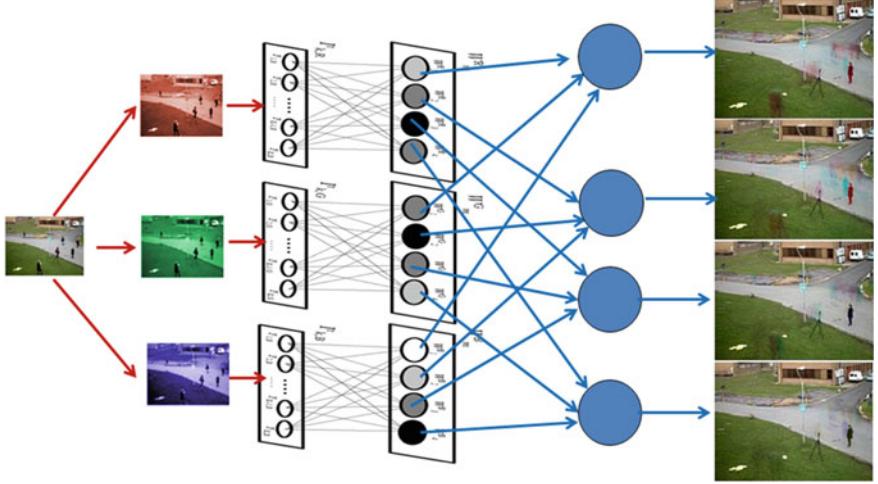


Fig. 4 RESOM partial background estimation for $M = 4$

After this frame, the learning parameters of the RESOM are adapted in order to learn the background properly in all neurons, and the moving objects only partially learned. At this point, $I_b(x, y, z)^t$ is an initial background model with incomplete information of dynamic objects as shown in Fig. 5.

A correct background model can be obtained from the RESOM if we remove the different component information. This information can be removed or inhibited from $I_b(x, y, z)^t$ by detecting the differences among the RESOM neurons.

The first step is to define the number of differences to consider $M_1 = \lceil (M/2)^2 \rceil$. Then, a set of images that contain the difference in the images of $V_w(x, y, z)^{m,t}$, is computed by

$$IV(x, y, z, m_k) = |V_w(x, y, z)^{m_1} - V_w(x, y, z)^{m_2}| \quad m_k = 1, 2, \dots, M_1 \quad (9)$$

where $m_1 = m_k - A_v M/2$, $m_2 = M/2 + A_v + 1$. Initially $A_v = 0$, and then it is updated as follows:

$$A_v = \begin{cases} A_v + 1 & \text{Remainder}(M_1/m_k) = 1 \\ A_v & \text{Otherwise} \end{cases} \quad (10)$$

The total difference is obtained with

$$I_d(x, y)^t = \sum_{m_k} \left(\frac{1}{3} \sum_z IV(x, y, z, m_k) \right) \quad (11)$$

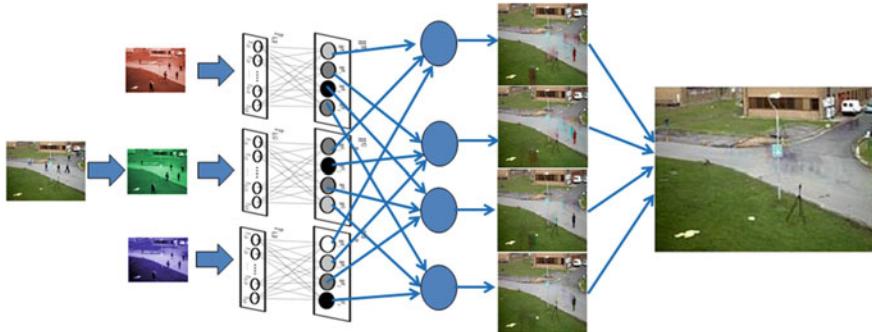


Fig. 5 Initial background with incomplete information of dynamic objects

The result is an image that contains the differences between the images in $V_w(x, y, z)^{m,t}$. $I_d(x, y)^t$ represents a moving pattern of the dynamic objects, which is used to find the pixels contaminated by dynamic objects in $I_b(x, y, z)^t$ through the next updating rule:

$$I_b(x, y, z)^{t+1} = \begin{cases} I_b(x, y, z)^{t+1} & I_d(x, y)^{t+1} < Th_1 \\ I_b(x, y, z)^t & \text{Otherwise} \end{cases} \quad (12)$$

here $Th_1 = 0.02$. In fact, Th_1 could be equal to zero, but, $I_d(x, y)^t$ could retain noise and dynamic object information from past positions. $I_b(x, y, z)^t$ is updated only if $I_d(x, y)^t < Th_1$ which means that each pixel of $I_b(x, y, z)^t$ preserves the background information if each pixel of $I_d(x, y)^t$ has information of dynamic objects. Therefore, an analysis of the gray levels of $I_d(x, y)^t$ was developed to establish a threshold and remove the noise and information of dynamic objects in $I_b(x, y, z)^t$. The background model is used to perform dynamic segmentation by background subtraction [13] based on the Euclidean distance between $I(x, y, z)^t$ and $I_b(x, y, z)^t$ given by

$$I_e(x, y)^t = \|I(x, y, z)^t - I_b(x, y, z)^t\|_2 \quad (13)$$

Figure 6 shows two examples where the background is improved by using $I_d(x, y)^t$, and the background subtraction result $I_e(x, y)^t$.

By analysis of other $I_e(x, y)^t$ binarization results, the following issues are detected.

- (i) Videos with dynamic backgrounds yield noise in $I_e(x, y)^t$ composed of a set of irregular regions during several frames. If the dynamic background spread that noise through a large part of the scene, the segmentation could cause serious false positives errors (Fig. 7).
- (ii) Illumination changes cause a noise evenly spread in $I_e(x, y)^t$ binarized. This noise is caused because background modeling takes several frames to adapt to the new illumination conditions, causing false positive errors during this adaptation period, Fig. 8.

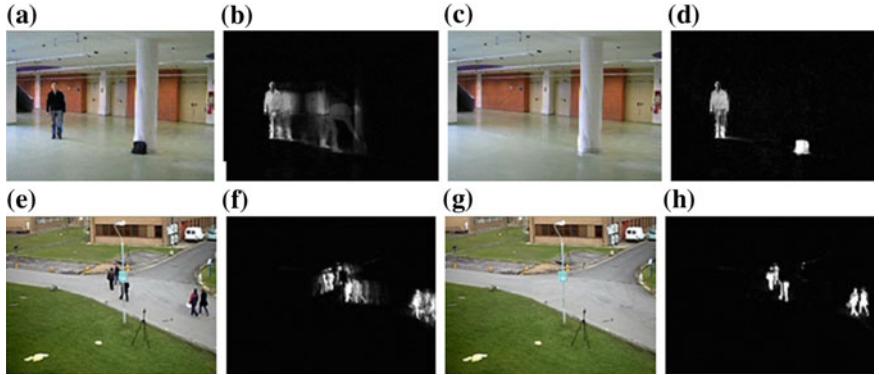


Fig. 6 Video ‘MSA’. **a** $I(x, y, z)^t$, frame 200. **b** $I_d(x, y)^t$. **c** $I_b(x, y, z)^t$. **d** $I_e(x, y)^t$. Video ‘P2009_L1_1’ after bootstrapping condition. **e** $I(x, y, z)^t$ frame 24. **f** $I_d(x, y)^t$. **g** $I_b(x, y, z)^t$. **h** $I_e(x, y)^t$

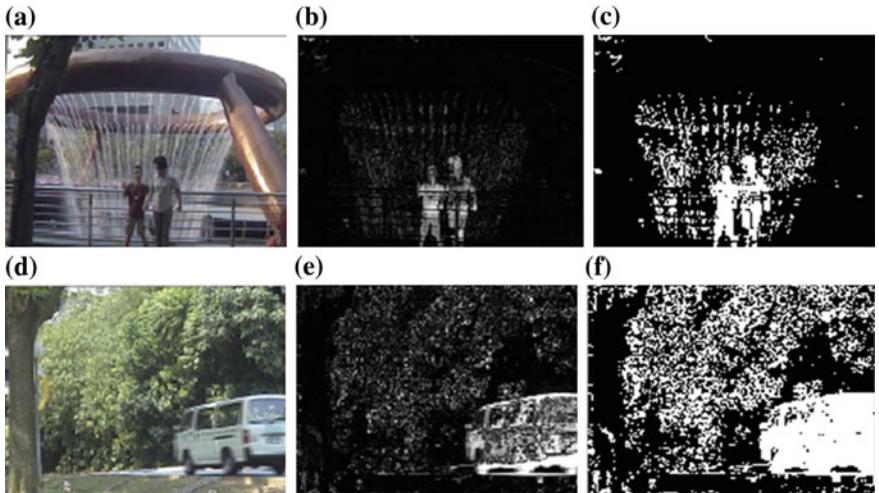


Fig. 7 Video ‘FT’. **a** Frame 179. **b** $I_e(x, y)^t$. **c** $I_e(x, y)^t$ binarized. Video ‘CAM’. **d** Frame 205. **e** $I_e(x, y)^t$. **f** $I_e(x, y)^t$ binarized

- (iii) Regions in $I_e(x, y)^t$ that correspond to dynamic objects are solid and the size is defined by the size of the dynamic object, Fig. 9.
- (iv) The value of the threshold Th_I has a great influence on the binarization of $I_e(x, y)^t$ called $I_{eb}(x, y)^t$, Fig. 10.

Based on these observations, we know that it is necessary to decrement the amount of noise on dynamic and illumination conditions, as well as to automatically compute Th_I . This issue is faced through a cellular network architecture, named neighbor threshold cellular network, that is described in the next section.



Fig. 8 Illumination effect in $I_e(x, y)^t$ due to illumination changes



Fig. 9 $I_e(x, y)^t$ result for dynamic objects

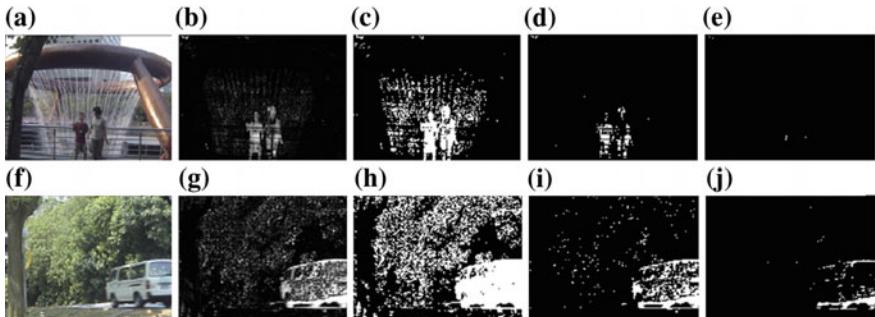


Fig. 10 Binarization process video ‘FT’. **a** Frame 179. **b** $I_e(x, y)^t$. **c** $I_{eb}(x, y)^t$, $Th_1 = 0.1$. **d** $I_{eb}(x, y)^t$, $Th_1 = 0.5$. **e** $I_{eb}(x, y)^t$, $Th_1 = 0.85$. Binarization process video ‘CAM’. **f** Frame 205. **g** $I_e(x, y)^t$. **h** $I_{eb}(x, y)^t$, $Th_1 = 0.1$. **i** $I_{eb}(x, y)^t$, $Th_1 = 0.5$. **j** $I_{eb}(x, y)^t$, $Th_1 = 0.85$

2.3 Neighbor Threshold Cellular Neural Network

The discrete time cellular network is defined by its difference equation [14]

$$\begin{aligned} \chi(x, y)^{n_c} &= \sum_{(x_N, y_N) \in N_v} A(x_N, y_N) y_c(x + x_N, y + y_N)^{n_c} \\ &+ \sum_{(x_N, y_N) \in N_v} B(x_N, y_N) I(x + x_N, y + y_N)^{n_c} + Z_c(x, y) \end{aligned} \quad (14)$$

where n_c is the iteration index, N_c the neighborhood of the element $p_e(x, y) \in I(x, y)$, (x_N, y_N) the index of the neighborhood N_c , $\chi(x, y)$ is the state of each cell, A and B are the weight kernels of the network, $Z_c(x, y)$ is the cell threshold, and $y_c(x, y)$ is the cell output defined by

$$y_c(x, y)^{n_c+1} = \frac{1}{2}(|\chi(x, y)^{n_c} + 1| - |\chi(x, y)^{n_c} - 1|) = f[\chi(x, y)] \quad (15)$$

From this model several configuration to perform image processing operations are defined. A binarization, thresholding, mode is described in the next section.

2.3.1 Discrete Thresholding CNN

The discrete thresholding CNN, TCNN, is defined by the kernels and threshold

$$A = 2 \quad B = 0 \quad Z_c(i, j) = -z_o \quad (16)$$

and initial conditions $\chi(x, y)^0 = 1 - 2I_e(x, y)^t$. z_o is a constant in the interval $[-1, 1]$. Thus, the difference equation is simplified to

$$\chi(n_c + 1) = f[\chi(n_c)] - z_o \quad (17)$$

and the output $y_c(x, y)^\infty$ corresponds to

$$\begin{aligned} y_c(x, y) &= -1, \text{ if } I_{ec}(x, y) > z_o \\ y_c(x, y) &= 1, \text{ if } I_{ec}(x, y) \leq z_o \end{aligned} \quad (18)$$

Although this model can perform image binarization, it lacks of the fact that the binarization decision only takes into account the information of one pixel. This issue is not well recommended in image processing tasks, specially, when noise is present. Therefore, the alternative is to design a more robust thresholding model by considering neighbor pixels. This model is presented in the following section.

2.3.2 Discrete Neighbor Thresholding CNN

The discrete neighbor thresholding CNN, NTCNN, defines the threshold of a $p_e(x, y) \in I_e(x, y)^t$ based on its gray level and the gray levels of their neighbors. The synaptic kernels are

$$A = \begin{bmatrix} a_2^{n_c} & a_2^{n_c} & a_2^{n_c} & a_2^{n_c} & a_2^{n_c} \\ a_2^{n_c} & a_1^{n_c} & a_1^{n_c} & a_1^{n_c} & a_2^{n_c} \\ a_2^{n_c} & a_1^{n_c} & 2 & a_1^{n_c} & a_2^{n_c} \\ a_2^{n_c} & a_1^{n_c} & a_1^{n_c} & a_1^{n_c} & a_2^{n_c} \\ a_2^{n_c} & a_2^{n_c} & a_2^{n_c} & a_2^{n_c} & a_2^{n_c} \end{bmatrix}, a_1 < 1, a_2 < 1 \quad B = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (19)$$

$$Z = -z_o$$

This new configuration yields the new difference equation of the NTCNN model

$$\begin{aligned} \chi(x, y)^{n_c+1} = & 2f[\chi(x, y)^{n_c}] + a_1^{n_c} \sum_{(x_N, y_N) \in N_8} f[\chi(x_N, y_N)^{n_c}] \\ & + a_2^{n_c} \sum_{(x_N, y_N) \in N_{(25-8)}} f[\chi(x_N, y_N)^{n_c}] - z_o(x, y) \end{aligned} \quad (20)$$

where N_8 correspond to the 8 neighbor pixels $p_e(x, y) \in I_{ec}(x, y)$, meanwhile $N_{(25-8)}$ defines the next surrounding of N_8 .

NTCNN performs noise reduction in the first iterations and then binarization in the next iterations. The initial conditions are the same as for TCNN.

At each iteration n_c , NTCNN assigns to each pixel a threshold, $Th(x, y)^{n_c}$, that depends on $z_o(x, y)$, a decaying neighbor product weighted by a_1 and a_2 , since $a_1 < 1$, $a_2 < 1$ and $n_c \rightarrow \infty$, then $a_1^{n_c} \rightarrow 0$ & $a_2^{n_c} \rightarrow 0$. Thus, the threshold $Th(x, y)$ is determined by

$$\begin{aligned} Th(x, y)^{n_c} = & a_1^{n_c} \sum_{(x_N, y_N) \in N_8} f[\chi(x_N, y_N)^{n_c}] + a_2^{n_c} \sum_{(x_N, y_N) \in N_{(25-8)}} f[\chi(x_N, y_N)^{n_c}] \\ & - z_o(x, y) \end{aligned} \quad (21)$$

The output $y_c(x, y)^\infty$ is

$$\begin{aligned} y_c(x, y) = & -1, \text{ if } I_{ec}(x, y) > zc + Nac. \\ y_c(x, y) = & 1, \text{ if } I_{ec}(x, y) \leq zc + Nac. \end{aligned} \quad (22)$$

where

$$Nac = a_1^{n_c} \sum_{(x_N, y_N) \in N_8} f[\chi(x_N, y_N)^{n_c}] + a_2^{n_c} \sum_{(x_N, y_N) \in N_{(25-8)}} f[\chi(x_N, y_N)^{n_c}] \quad (23)$$

Nac modifies the threshold for each pixel based on neighbor pixel information. After eight iterations the NTCNN yields the binary image, $Is(x, y)^t$ corresponding to the dynamic objects

$$Is(x, y)^t = \left[y_c(x, y)^8 - 1 \right] / 2 \quad (24)$$

Considering Eq. (20),

If $z_o = 1$, all $p_e(x, y) \in I_{ec}(x, y)^t \neq 0$ generate $Is(x, y)^t = 1$ Foreground

If $z_o = -1$, all $p_e(x, y) \in I_{ec}(x, y)^t \neq 1$ generate $Is(x, y)^t = 0$ Background

Therefore, in order to select the value of z_o , it is considered that it is better to have false positives with respect dynamic objects, therefore the option $z_o = 1$ is selected in this work. This value avoids false negatives and camouflage.

The parameters a_1 , a_2 , are used to reduce noise in $I_s(x, y)^t$. These parameters allow selecting a threshold for each pixel based on its neighborhood. Figures 11 and 12 illustrate two cases where the advantage of using parameters a_1 , a_2 is compared against the segmentation using only Th_1 .

At this point the proposed segmentation method has a NTCNN with constant $NC = 8$, $z_o = 1$ and parameters a_1 and a_2 . The last stage proposes the automatic mechanism to update the parameters a_1 and a_2 .

SOM-CNN assumes normal conditions in the scenario. No noise generated by dynamic background, camera jitter, camouflage, etc. In this circumstance it is recommended that a_1 and $a_2 \approx 0$. Thus, the initial conditions are, $a_1 = 0.05$, $a_2 = 0.01$, to make all $p_e(x, y) \in I_e(x, y)^t \neq 0 \rightarrow 1$.

After the first frame the parameters are adapted based on the scenario conditions related to illumination. This conditions are determined based on V (Value—luminance) changes on consecutive frames. The V component was selected because it is related to the frame luminance. The illumination change is detected by the expected value of inter-frame difference

$$\begin{aligned} Iv(k)^t &= |I(k, V)^t - I(k, V)^{t-1}| \\ \mu_v &= \frac{1}{K} \sum_k Iv(k)^t \end{aligned} \quad (25)$$

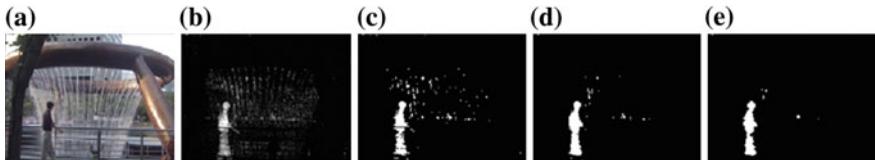


Fig. 11 Binarization process. **a** Frame 415, video ‘FT’. **b** $Ie(x, y)^t$. **c** Binarization with $Th_1 = 0.35$. **d** $I_s(x, y)^t$ obtained with NTCNN, $a_1 = 0.11$, $a_2 = 0.11$. **e** $I_s(x, y)^t$ obtained with NTCNN, $a_1 = 0.35$, $a_2 = 0.05$

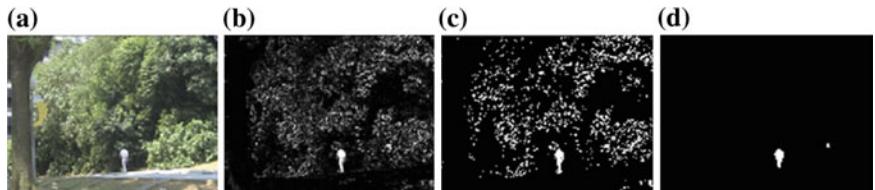


Fig. 12 Binarization process. **a** Frame $t = 415$, video ‘CAM’. **b** $Ie(x, y)^t$. **c** Binarization with $Th_2 = 0.35$. **d** $I_s(x, y)^t$ obtained with NTCNN, $a_1 = 0.55$, $a_2 = 0.5$

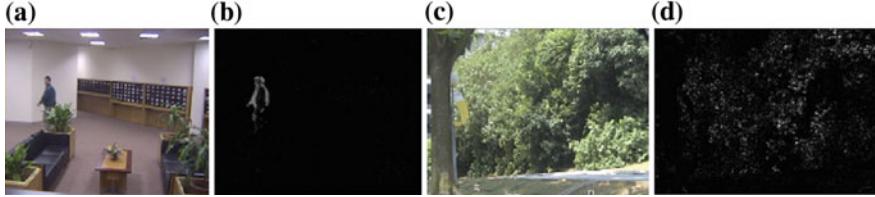


Fig. 13 $Iv(x, y)^t$ results. **a** Video ‘LB’, $t = 380$. **b** $Iv(x, y)^t$ of **a**, $\mu_v = 0.00991$. **c** Video ‘CAM’, $t = 580$. **d** $Iv(x, y)^t$ of **(c)**, $\mu_v = 0.0235$

μ_v measures the luminance change between frames, because it is related to amount of values different from zero in $Iv(k)^t$ and its gray level values. Figure 13 shows that if a dynamic object is present in the scenario, $Iv(x, y)^t$ yields a pattern similar to the object edges, Fig. 13b. If the scenario involves noise, several non-uniform regions are generated, Fig. 13d. If the dynamic object are too small or the noise is not very significant, $\mu_v \approx 0$. If strong changes are produced, like dynamic object size changes or there exist noise, then μ_v will be in the interval $(0,0.1]$, like in Fig. 13d, where $\mu_v = 0.0235$.

In general, we have that $\mu_v > 0.1$ occurs only in cases of camera jittering, and dynamic background. In these cases parameter adjustment is required to reduce segmentation noise. Parameter adjustment is based on μ_v , through the sigmoidal function:

$$s_d(s_1, s_2, \mu_d) = \frac{s_1}{1 + \exp(-40(10\mu_d - s_2))} \quad (26)$$

The sigmoidal function is tuned to increase the parameter values when $\mu_d \approx 0.1$. s_1 is maximum value of $s_d(s_1, s_2, \mu_d)$, s_2 is the inflection point of the sigmoidal, and μ_d is given by:

$$\mu_d = \frac{1}{\Lambda} \sum_{\tau=t-\Lambda}^t \mu_v^\tau \quad (27)$$

where Λ is a time window in frames. Λ allows estimating the scenario changes during the last Λ frames. In SOM-CNN, if $t > 300$ $\Lambda = 300$, if not $\Lambda = t$. Any value may be selected for Λ , although it is recommended to select a value that avoids an increment in μ_d when drastic changes occurs in μ_v which not correspond to the normal dynamic of the scenario.

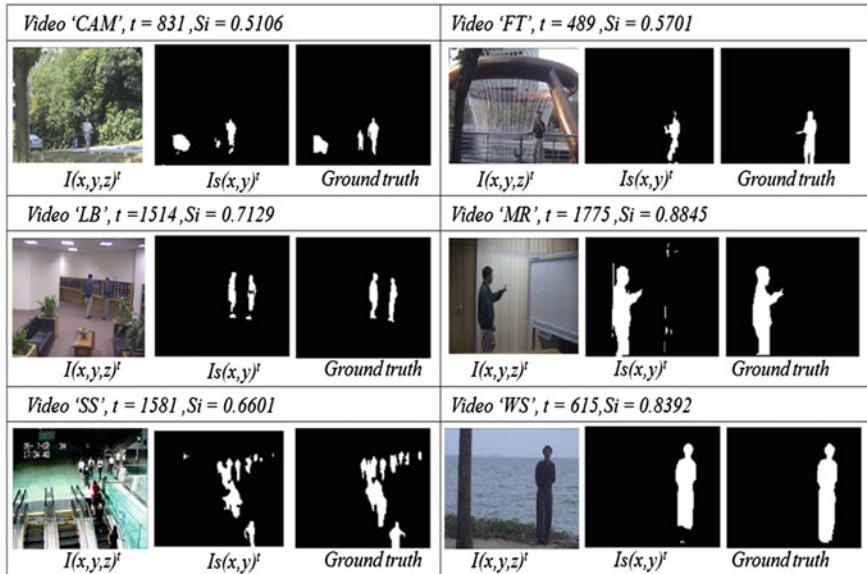
Thus, the automatic update of the parameters a_1 , a_2 is computed by

$$\begin{aligned} a_1 &= a_{1o} + s_d(0.1, s_2, \mu_d) \\ a_2 &= a_{2o} + s_d(0.1, s_2, \mu_d) \end{aligned} \quad (28)$$

The initial values, a_{1o} , a_{2o} , are established according to scene conditions as stated in Table 1.

Table 1 Initial conditions for a_1 , a_2

Scenario	a_{1o}	a_{2o}	s_2
Normal conditions	0.05	0.01	0.40
Normal conditions with reflections or shadows	0.10	0.10	0.40
Dynamic background	0.08	0.02	0.25
Low contrast	0.03	0.00	0.40

**Fig. 14** Results of the SOM-CNN model showing the advantage of using the NTCNN

3 Results

The SOM-CNN model was tested in several video data bases in order to evaluate its performance. Figure 14 illustrates the results obtained in six videos. The ground truth, expected segmentation, is included for the purpose of comparison. The videos included illumination changes, as well as dynamic background. In each case, it can be observed that the NTCNN module certainly reduced the amount of false positives.

4 Conclusions

This work described a discrete cellular network, NTCNN, approach for noise reduction and binarization in dynamic object detection. The neural scheme represents an improvement to the normal threshold CNN because it involves neighbor

self-adapting parameters. The NTCNN is derived from bio-inspired systems which represents a good alternative for video analysis according to the results of RESOM and SOM-CNN. The results of this work indicated that bio-inspired systems can deal with real scenario conditions related to: dynamic background. The NTCNN turned to be a good option to improve object detection results of the RESOM, and it was shown that an automatic mechanism for automatic parameter update was possible for the NTCNN.

Acknowledgments The authors thank the Fondo Mixto de Fomento a la Investigación Científica y Tecnológica CONACYT-Gobierno del Estado de Chihuahua under grant CHIH-2012-C03-193760 and Tecnológico Nacional de México under grants CHI-MCIET-2013-230 and CHI-IET-2012-105.

References

1. Yu, B., Zhang, L.: Pulse-coupled neural networks for contour and motion matching. *IEEE Trans. Neural Netw.* **15**(5), 1186–1201 (2004)
2. Faro, A., Giordano, D., Spampinato, C.: Evaluation of the traffic parameters in a metropolitan area by fusing visual perceptions and CNN processing of webcam images. *IEEE Trans. Neural Netw.* **19**(6), 1108–1129 (2008)
3. Costantini, G., Casali, D., Carota, M., Perfetti, R.: A CNN-based algorithm for moving object detection in stereovision applications. In: Proceedings of International Conference on Circuit Theory and Design European, pp. 500–503, Aug 2007
4. Wei-Song, L., An-Te, L., Chun-Hsiung, F.: Computational autonomous visual perception using cellular neural networks. In: Proceedings of IEEE International Conference on Computational Intelligence for Measurement Systems and Applications, pp. 198–202, July 2005
5. Dubravko, C., Marques, O., Socek, D.: Neural network approach to background modeling for video object segmentation. *IEEE Trans. Neural Netw.* **18**(6), 1614–1627 (2007)
6. Maddalena, L., Petrosino, A.: A self-organization approach to background subtraction for visual surveillance applications. *IEEE Trans. Image Process.* **17**(7), 1168–1177 (2008)
7. Chacon-Murguia, M.I., Gonzalez-Duarte, S., Vega-Pineda, J.: Simplified SOM-neural model for video segmentation of moving objects. In: Proceedings of International Joint Conference on Neural Networks, pp. 14–19, June 2009
8. Chacon-Murguia, M.I., Gonzalez-Duarte, S.: An adaptive neural-fuzzy approach for object detection in dynamic backgrounds for surveillance systems. *IEEE Trans. Ind. Electron.* **59**(8), 3286–3298 (2012)
9. Ghasemi, A., Safabakhsh, R.: Unsupervised foreground-background segmentation using growing self organizing map in noisy backgrounds. In: Proceedings of 3rd International Conference on Computer Research and Development, pp. 334–338, Mar 2011
10. Baier, V.: Motion perception with recurrent self-organizing maps based models. In: Proceedings of International Joint Conference on Neural Networks, pp. 1182–1186, July 2005
11. Avila-Mora, I.M., Castellanos-Sánchez, C.: Bio-inspired clustering of moving objects. In: Proceedings of Congress on Intelligent Systems, pp. 58–62, May 2009
12. Miikkulainen, R., Bednar, J.A., Choe, Y., Sirosh, J.: Computational Maps in the Visual Cortex. Springer Sciences Media Inc., Ed., New York (2005)
13. Brutzer, S., Höferlin, B., Heidemann, G.: Evaluation of background subtraction techniques for video surveillance. In: Proceedings of Conference on Computer Vision and Pattern recognition, pp. 1937–1944, June 2011
14. Chua, L., Roska, T.: Cellular Neural Networks and Visual Computing Foundation and Applications. Cambridge University Press, New York (2004)

Optimization of the LVQ Network Architecture with a Modular Approach for Arrhythmia Classification Using PSO

Jonathan Amezcu and Patricia Melin

Abstract In this paper, the optimization of LVQ neural networks with modular approach is presented for classification of arrhythmias, using particle swarm optimization. This work focuses only in the optimization of the number of modules and the number of cluster centers. Other parameters, such as the learning rate or number of epochs are static values and are not optimized. Here, the MIT-BIH arrhythmia database with 15 classes was used. Results show that using 5 modules architecture could be a good approach for classification of arrhythmias.

1 Introduction

In this paper the optimization of a modular LVQ neural network architecture [1, 4] with Particle Swarm Optimization (PSO) for arrhythmia classification [22, 24] is presented. The optimization focuses on the number of modules and the number of cluster centers. Other parameters such as the epochs or learning rate are static values, obtained from a previous research on optimization of parameters [2].

LVQ is an adaptive learning method used to solve classification problems; although it uses supervised training, LVQ applies unsupervised data clustering techniques, to pre-process the dataset and obtain the centers of the clusters [13].

Particle Swarm Optimization (PSO) is used for the architecture optimization, which is a stochastic optimization technique based on the social behavior of animals. In this work PSO is applied to optimize the number of modules in the

J. Amezcu · P. Melin (✉)
Tijuana Institute of Technology, Tijuana, Mexico
e-mail: pmelin@tectijuana.edu.mx

J. Amezcu
e-mail: jonathan.aguiluz@yahoo.com

architecture of the LVQ network [7, 8], which is variable in a range of [2, 5]. The distribution of classes in each module depends of the number of modules in the architecture.

The rest of this paper is organized as follows, in Sect. 2 some basic concepts of PSO are described, Sect. 3 presents the problem statement of this work, Sect. 4 shows the proposed model, in Sect. 5 presents the simulation results and finally in Sect. 6 the conclusions.

2 PSO Method

Particle Swarm Optimization (PSO) [2, 6, 9, 14] is a technique based on social behaviors observed in animals. This method has gained popularity as a robust technique for solving optimization problems. In PSO, individual particles of a swarm represent possible solutions to the problem.

The position of each particle is adjusted according to its velocity and the difference between its current position and the best position found by its neighbors, and the best position found so far. The position of a particle i is updated as follows:

$$v_{ij}(t+1) = v_{ij}(t) + c_1 r_{1j}(t)[y_{ij}(t) - x_{ij}(t)] + c_2 r_{2j}(t)[\hat{y}_j(t) - x_{ij}(t)] \quad (1)$$

where $v_{ij}(t)$ is the velocity of a particle i in dimension j at time step t , $x_{ij}(t)$ is the position of a particle i in dimension j at time step t ; c_1 and c_2 are constants used to scale the contribution of the cognitive and social components and $r_{1j}(t)$ and $r_{2j}(t)$ are random values in the range of [0, 1] that introduce stochastic element to the algorithm.

The personal best position y_i is the best position the particle has visited, the personal best position at a time step $t + 1$ is calculated as follows:

$$y_i(t+1) = \begin{cases} y_i(t), & f(x_i(t+1)) \geq y_i(t) \\ x_i(t+1), & f(x_i(t+1)) < y_i(t). \end{cases} \quad (2)$$

The global best position $\hat{y}(t)$ at a time step t is defined as:

$$\hat{y}(t) \in \{y_0(t), \dots, y_{n_s}(t)\} | f(\hat{y}(t)) = \min(\{y_0(t), \dots, y_{n_s}(t)\}) \quad (3)$$

where n_s is the total number of particles in the swarm. The definition in Eq. (3) states that \hat{y} is the best position discovered by any of the particles so far.

The algorithm for the PSO [28, 29] model is as follows, where $S.x_i$ is used to denote the position of particle i in the swarm S .

```

repeat
    for each particle  $i = 1, \dots, S.n_s$ do
        //Set the personal best position
        if  $(S.x_i) < f(S.y_i)$  then
             $S.y_i = S.x_i;$ 
        end
        //Set the global best position
        if  $S.y_i < S.\hat{y}_i$  then
             $S.\hat{y}_i = S.y_i;$ 
        end
    end
    for each particle  $i = 1, \dots, S.n_s$ do
        update velocity;
        update position;
    end
until stopping condition is true;

```

3 Problem Statement

As mentioned above, PSO is a bio-inspired optimization method that has proved to be successful in the implementation of various problems, such as the optimization of membership functions of fuzzy systems, parameters optimization of neural networks, and fuzzy systems as well, and, in this case, the optimization of a modular neural network architecture for classification. In this work, such optimization consists into find the architecture for the minimum error accuracy in the classification of arrhythmias [3, 5, 21, 23].

On the other hand, classification tasks consists into assigning objects to only one of many predefined categories, a widespread problem that covers many different areas of application, such as spam detection in emails, classification of galaxies, among others [24]. In this research, modular LVQ network [12, 17, 18] was used as classification technique, Fig. 1 shows the architecture of a LVQ network [15].

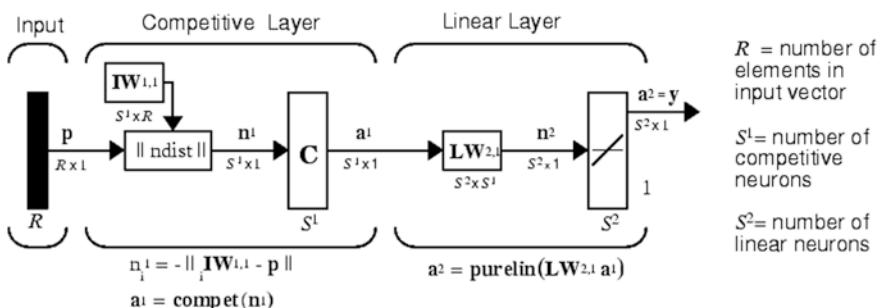


Fig. 1 Architecture of LVQ network

3.1 Arrhythmia Dataset

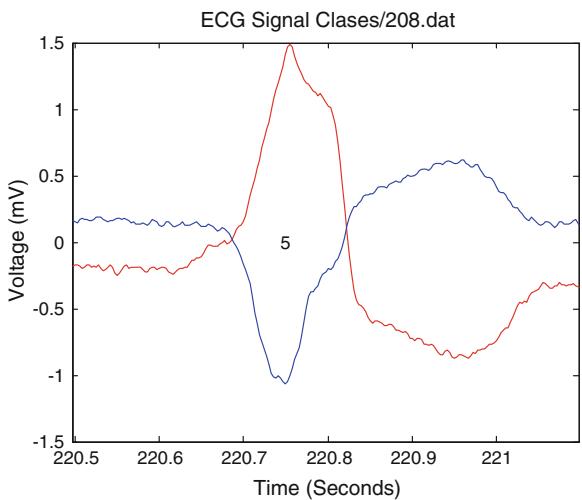
Arrhythmias are expressed as changes in the normal sequences of electrical heart impulses, these impulses may happen too fast, too slowly or erratically, and can be measured with a Holter device in ECG signals [10]. Figure 2 shows an example of one of these ECG signals [11].

The MIT-BIH [19] arrhythmia dataset was used for this research. This database consists of 15 different types of arrhythmias. This database contains 48 half-hour excerpts of ECG recordings, obtained from 47 subjects studied by BIH Arrhythmia Laboratory. The recordings were digitized at 360 samples per second per channel with 11-bit resolution over a 10 mV range. The recordings were preprocessed as follows:

- Analog outputs were filtered using a passband from 0.1 to 100 Hz. The passband-filtered signals were digitized at 360 Hz per signal.
- Taking the R point [27] as reference, in the obtained signals was located the start and end point for each wave.
- The 38 higher voltages and 38 lower voltages of each vector were taken, resulting in vectors of 76 voltages.

In the next section we discuss the proposed optimization model for modular LVQ network architecture.

Fig. 2 Example of an ECG signal



4 Proposed Model

In this work, the development of a PSO [16, 20] model for the optimization of a modular LVQ network architecture for classification is presented. In this section, we explain in depth the architecture of the PSO model and its parameters as well, and how are used for the optimization of a modular LVQ network architecture.

Within the architecture of modular LVQ network there is a parameter that defines how many cluster centers the network will be training with. Therefore we have a 2-dimension problem, the optimal number of modules in the modular architecture, and the number of cluster centers for each of the modules.

Table 1 shows the parameters, range, minimal and maximal values that PSO model works with. The values for the number of cluster centers were taken from [2], where the optimization of parameters for LVQ [25, 26] networks is presented, and it was with this range of cluster centers that the best classification accuracy was achieved.

About the number of modules in the architecture, in the work of [18] many architectures were developed to work with this same arrhythmia database, one of the best architectures with high classification accuracy was composed by 5 modules, so for this reason was it decided to develop this research working with a range of [2, 5] for the number of modules. Table 2 shows the rest of parameters for the PSO method.

So far we have that PSO model works with 2-dimension problem, the optimization of the number of modules in a modular architecture, and the number of cluster centers. In Sect. 4.1 we discuss how the database was partitioned, since it was quite difficult because of the dynamical number of modules and the information was to be evenly distributed in each module.

Table 1 Parameters of the architectures included in PSO model

Parameter	Minimal value	Maximal value
Number of modules	2	5
Number of clusters centers	15	30

Table 2 PSO parameters

Parameter	Value
Population size	15
Maximum number of iterations	15
C_1	2
C_2	$4 - C_1$
Inertia weight	Linear decreasing

4.1 Partitioning the Database

As mentioned earlier, MIT-BIH arrhythmia database consists of 15 classes. In this work was considered to have the classes the most evenly distributed in each module. The classes are labeled from 1 to 15, so to solve this particular problem, a method was developed; this method takes the total number of classes and divided it by the number of modules, depending on the architecture to evaluate. For example, if the PSO method is evaluating a five-module architecture, then this method distributes 3 classes in each module.

When PSO method is evaluating a two-module architecture, then classes are distributed 8 classes in the first module, and 7 classes in the second one. The classes are consecutively distributed, which means that in a two-module architecture, first module contains classes 1–8, and the second one, classes 9–15, and so on for each of the architectures to evaluate by the PSO method.

The rest of the LVQ network architecture, such as the learning rate (LR), and the maximum number of epochs, were left static, in 0.0994 and 200 respectively, taking into account that these parameters were already optimized in [2] achieving a good classification percentage.

5 Simulation Results

A set of 15 experiments were conducted using de PSO model described above, Table 3 shows the obtained results, where **Time** is expressed in HH:MM format. Experiments were performed in a Windows 7 PC x64, Quad-Core i7 processor and 16 GB of RAM.

Table 3 Simulation results

	Modules	Clusters	Epochs	Error	Time
1	5	17	97	1.66×10^{-8}	06:31
2	5	25	30	1.33×10^{-7}	07:22
3	5	29	76	1.25×10^{-8}	06:45
4	5	30	55	1.66×10^{-8}	06:29
5	5	17	51	1.38×10^{-7}	07:33
6	5	23	28	1.66×10^{-8}	07:18
7	5	17	51	1.34×10^{-7}	06:30
8	5	20	50	1.34×10^{-7}	06:57
9	5	25	61	1.44×10^{-8}	06:47
10	5	30	104	1.22×10^{-7}	07:28
11	5	24	38	1.66×10^{-8}	06:55
12	5	27	94	1.72×10^{-8}	06:32
13	5	19	37	1.33×10^{-7}	06:41
14	5	21	55	1.66×10^{-8}	07:12
15	5	24	62	1.66×10^{-8}	06:45

Notice that for all experiments, the best LVQ network architecture consisted of 5 modules, which makes sense since the smaller number of data in each module is easier to train the LVQ network. The errors reached in each experiment are very similar, so basing on the obtained error the best experiment was the number 3.

6 Conclusions

In this paper, a PSO model to find an optimal modular LVQ [30] network architecture for classification of arrhythmias [1, 22] was presented. The results show that a five modules LVQ network architecture can be an optimal architecture for the classification of arrhythmias.

The main reason for these results is that, like many other methods, if a LVQ network module receives several records for training, the classification accuracy for that module tends to be lower, in this case this is for the similarity grade between certain records; there are many training records that are very similar each other but they belong to different classes in the network module, therefore, with less training records in each module this grade of similarity is low, and the classification percentage tends to be higher.

Regarding with the PSO model, it has proved to be a good approach for the optimization of the LVQ network model presented, for the classification of arrhythmias.

References

1. Amezcua, J., Melin, P.: A modular LVQ neural network with fuzzy response integration for arrhythmia classification. In: IEEE 2014 Conference on Norbert Wiener in the 21st Century, Boston, June 2014
2. Amezcua, J., Melin, P.: Optimization of modular neural networks with the LVQ algorithm for classification of arrhythmias using particle swarm optimization. In: Recent Advances on Hybrid Approaches for Designing Intelligent Systems. Studies in Computational Intelligence Book, vol. 547, pp. 307–314. Springer, Berlin (2014)
3. Anuradha, B., Veera-Reddy, V.C.: Cardiac arrhythmia classification using fuzzy classifiers. *J. Theor. Appl. Inform. Technol.* 353–359 (2005)
4. Biswal, B., Biswal, M., Hasan, S., Dash, P.K.: Nonstationary power signal time series data classification using LVQ classifier. *Appl. Soft Comput.* **18**, 158–166 (2014)
5. Castillo, O., Melin, P., Ramirez, E., Soria, J.: Hybrid intelligent system for cardiac arrhythmia classification with fuzzy K-nearest neighbors and neural networks combined with a fuzzy system. *J. Expert Syst. Appl.* **39**(3), 2947–2955 (2012)
6. Cavuslu, M., Karakuzu, C., Karakaya, F.: Neural identification of dynamic systems on FPGA with improved PSO learning. *Appl. Soft Comput.* **12**, 2707–2718 (2012)
7. Frasconi, P., Gori, M., Soda, G.: Links between LVQ and backpropagation original research article. *Pattern Recogn. Lett.* **18**(4), 303–310 (1997)
8. Grbovic, M., Vučetić, S.: Regression learning vector quantization. In: Ninth IEEE International Conference on Data Mining. Miami, Dec 2009

9. Hashemi, A.B., Meybodi, M.R.: A note on the learning automata based algorithms for adaptive parameter selection in PSO. *Appl. Soft Comput.* **11**, 689–705 (2011)
10. Hu, Y.H., Palreddy, S., Tompkins, W.A.: Patient adaptable ECG beat classifier using a mixture of experts approach. *IEEE Trans. Biomed. Eng.* 891–900 (1997)
11. Hu, Y.H., Tompkins, W., Urrusti, J.L., Afonso, V.X.: Applications of ANN for ECG signal detection and classification. *J. Electrocardiol.* **28**, 66–73 (1994)
12. Kim, J., Sik-Shin, H., Shin, K., Lee, M.: Robust algorithm for arrhythmia classification in ECG using extreme learning machine. *Biomed. Eng. (Online)*, Oct 2009
13. Kohonen, T.: Improved versions of learning vector quantization. In: *International Joint Conference on Neural Networks*, vol. 1, pp. 545–550, San Diego (1990)
14. Krishna, N.L., Kadetotad Deepak, V., Manikantan, K., Ramachandran, S.: Face recognition using transform domain feature extraction and PSO-based feature selection. *Appl. Soft Comput.* **22**, 141–161 (2014)
15. Learning Vector Quantization Networks. Site: http://www.mathworks.com/help/nnet/ug/bss4b_1-15.html. Last Access: 24 June 2014
16. Lee, C., Leu, Y., Yang, W.: Constructing gene regulatory networks from microarray data using GA/PSO with DTW. *Appl. Soft Comput.* **12**, 1115–1124 (2012)
17. Martín-Valdivia, M.T., Ureña-López, L.A., García-Vega, M.: The learning vector quantization algorithm applied to automatic text classification tasks. *Neural Networks* **20**(6), 748–756 (2007)
18. Melin, P., Amezcua, J., Valdez, F., Castillo, O.: A new neural network model based on the LVQ algorithm for multi-class classification of arrhythmias. *Inf. Sci.* **279**, 483–497 (2014)
19. MIT-BIH Arrhythmia Database.: PhysioBank, Physiologic Signal Archives for Biomedical Research. Site: <http://www.physionet.org/physiobank/database/mitdb/>. Last access: 24 June 2014
20. Narayan, R., Chatterjee, D., Kumar-Goswami, S.: An application of PSO technique for harmonic elimination in a PWM inverter. *Appl. Soft Comput.* **9**, 1315–1320 (2009)
21. Nasiri, J.A., Naghibzadeh, M., Yazdi, H.S., Naghibzadeh, B.: ECG arrhythmia classification with support vector machines and genetic algorithm. In: *Third UKSim European Symposium on Computer Modeling and Simulation* (2009)
22. Nouaouria, N., Boucadoum, M.: Improved global-best particle swarm optimization algorithm with mixed-attribute data classification capability. *Appl. Soft Comput.* **21**, 554–567 (2014)
23. Owis, M.I., Abou-Zied, A.H., Youssef, A.M., Kadah, Y.M.: Study of features based on non-linear dynamical modeling in ECG arrhythmia detection and classification. *IEEE Trans. Biomed. Eng.* **49**(7) (2002)
24. Pang-Ning, T., Steinbach, M., Kumar, V.: *Introduction to Data Mining*, pp. 145–148. Pearson, Addison Wesley, Boston (2006)
25. Pedreira, C.: Learning vector quantization with training data selection. *IEEE Trans. Pattern Anal. Mach. Intell.* **28**, 157–162 (2006)
26. Torrecilla, J.S., Rojo, E., Oliet, M., Domínguez, J.C., Rodríguez, F.: Self-organizing maps and learning vector quantization networks as tools to identify vegetable oils and detect adulterations of extra virgin olive oil. *Comput. Aided Chem. Eng.* **28**, 313–318 (2010)
27. Tsipouras, M.G., Fotiadis, D.I., Sideris, D.: An arrhythmia classification system based on the RR-interval signal. *Artif. Intell. Med.* 237–250 (2005)
28. Vellasques, E., Sabourin, R., Granger, E.: Fast intelligent watermarking of heterogeneous image streams through mixture modeling of PSO populations. *Appl. Soft Comput.* **13**, 3130–3148 (2013)
29. Vieira, S., Mendoca, L., Farinha, G., Souza, J.: Modified binary PSO for feature selection using SVM applied to mortality prediction of septic patients. *Appl. Soft Comput.* **13**, 3494–3504 (2013)
30. Xinye, Z., Jubai, A., ZhiFeng, Y.: A method of LVQ network to detect vehicle based on morphology. In: *WRI Global Congress on Intelligent Systems*, Xiamen, China, May 2009

Evolution of Kernels for Support Vector Machine Classification on Large Datasets

Luis Carlos Padierna, Martín Carpio, Rosario Baltazar,
Héctor José Puga and Héctor Joaquín Fraire

Abstract Kernel selection is a main factor in the designing of support vector machines. Evolutionary techniques have been applied to select the fittest kernel for specific classification problems. However, technical issues emerge when attempting to apply this methodology to deal with large datasets. On the other hand, a new method for improving the training time of support vector machines was recently developed. In this chapter, the new method is integrated in a kernel evolution scheme. Ten benchmark datasets are tested. Results indicate that the new method speeds up the evolution process when datasets are greater than 1000 instances.

1 Introduction

Support Vector Machines (SVMs) have been widely used for pattern classification. Key design aspects for achieving high performance rates in SVM classification involve: selection of the quadratic programming solver, kernel parameter tuning and kernel selection [1–3].

L.C. Padierna · M. Carpio · R. Baltazar · H.J. Puga (✉)

Tecnológico Nacional de México-Instituto Tecnológico de León, Leon, Mexico
e-mail: pugahector@yahoo.com

L.C. Padierna
e-mail: luiscarlos.padierna@itleon.edu.mx

M. Carpio
e-mail: juanmartin.carpio@itleon.edu.mx

R. Baltazar
e-mail: r.baltazar@ieee.org

H.J. Fraire
Tecnológico Nacional de México-Instituto Tecnológico de Cd. Madero, Cd. Madero, Mexico
e-mail: automatas2002@yahoo.com.mx

Kernel selection is useful when training an SVM with non-linearly separable data. The reason is that a kernel is generally a non-linear function that maps the original input space into a high-dimensional dot-product feature space in order to enhance linear separability [4]. Until now, there not exists a systematic way to choose the fittest kernel for a given dataset [5].

Furthermore, one single kernel could be not enough to reach good generalization rates. A current trend consists in combining two or more kernels with the intention to increase the generalization capability of an SVM [6]. This strategy is called Multiple Kernel SVM [2].

Recent approaches had considered evolutionary strategies to automatically choose a multiple kernel that best fit to a specific dataset [1, 7]. Approximate methods such as evolutionary strategies are needed since the problem of finding the best multiple kernel for a specific dataset is NP-complete [8]. However, these strategies involve prohibitive computational costs as the size of the dataset increases.

It is known that training a standard SVM has a complexity between $O(n^2)$ and $O(n^3)$ where n is the number of input vectors [3, 9]. Furthermore, the size of the associated Gram matrix is $n \times n$ (see Sect. 2.2), therefore, the cost in time and space that should be paid in every evaluation of the fitness function when evolving kernels is high.

In this chapter a new method is integrated into an evolutionary scheme in order to prove the hypothesis that, by applying it to a dataset, is possible to reduce the computational burden inherent to the evolution process and, in consequence, to accelerate the evolution of kernels for large datasets.

Section 2 provides fundamental definitions and a brief description of the techniques applied in this work. The evolutionary scheme integrating the new method, the materials and configurations used to carry out experiments are established in Sect. 3. Finally, a discussion about main results, future research guidelines and conclusions are provided in Sect. 4.

2 Theoretical Basis

In this section three methods are briefly described, namely, C-SVM, Genetic Programming (GP) and Decision Tree Support Vector Machines (DTSVM). In addition, some fundamental concepts are given in logical order.

2.1 Support Vector Machines

SVMs are non-probabilistic binary classifiers that can be used to construct a hyperplane to separate data into one of two classes. Its formulation is as follows [4, 10]:

Given a training dataset $D = \{x_i, y_i\}_{i=1}^m$ where $x_i \in X, X \subset R^n, y_i \in \{+1, -1\}$, SVM classifies with an optimal separating hyperplane, which is given by:

$$h(x) = w^T x + b \quad (1)$$

When working with data non-linearly separable, this hyperplane is obtained by solving the following quadratic programming problem:

$$\begin{aligned} & \min \left(\frac{1}{2} w^T w + C \sum_{i=1}^m \xi_i \right) \\ & \text{s.t. } y_i(w^T x_i + b) \geq 1 - \xi_i, \xi_i > 0 \quad \text{for } i = 1, \dots, m \end{aligned} \quad (2)$$

where ξ_i are slack variables to tolerate miss classifications and $C > 0$ is a regularization parameter. Introducing the nonnegative Lagrange multipliers α and β and following the Karush-Kuhn-Tucker conditions:

$$\nabla_w L = w - \sum_{i=1}^m \alpha_i y_i x_i = 0 \quad (3)$$

$$\nabla_b L = - \sum_{i=1}^m \alpha_i y_i = 0 \quad (4)$$

$$\nabla_{\xi_n} L = C - \alpha_i - \beta_i = 0 \quad (5)$$

the problem (2) can be proved to be equivalent to the following dual problem

$$\begin{aligned} \text{Max } L(\alpha) &= \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m y_i y_j \alpha_i \alpha_j k(x_i, x_j) \\ \text{s.t. } C &\geq \alpha_i \geq 0 \quad \forall i = 1, \dots, m \\ \text{and } \sum_{i=1}^m \alpha_i y_i &= 0 \end{aligned} \quad (6)$$

This expression is considered the standard version of an SVM and is called C-SVM.

2.2 Kernel, Multiple Kernel and Gram Matrix

Kernel A function $K(x, x')$ defined on $R^n \times R^n$ is called a kernel on $R^n \times R^n$ or kernel briefly if there exists a map ϕ from the space R^n to the Hilbert space

Table 1 Some common kernel functions

Kernel	$K(x, x') =$	Kernel	$K(x, x') =$
Lineal	$x^T x'$	Powered	$- x - x' ^\beta \quad 0 < \beta \leq 1$
Polynomial	$(\sigma \times x^T x' + r)^d$	Log	$-\log(1 + x - x')^\beta \quad 0 < \beta \leq 1$
RBF	$e^{-\frac{ x-x' ^2}{\sigma^2}}$	Generalized Gaussian	$e^{-(x-x')^T A(x-x')}$ where A is a symmetric PD matrix
Sigmoid	$\tanh(\sigma \times x^T x' + r)$	Hybrid	$e^{-\frac{ x-x' ^2}{\sigma^2}} \times (\tau + x^T x')^d$

$\phi: R^n \rightarrow \mathcal{H}$ such that $K(x, x') = (\phi(x) \cdot \phi(x'))$ where (\cdot) denotes the inner product of space \mathcal{H} [11]. Some common kernel functions are shown in Table 1 [12].

Multiple Kernel Is denoted as $K_\eta(x_i, x_j) = f_\eta \left(\left\{ K_m(x_i^m, x_j^m) \right\}_{m=1}^P \right)$ where the combination function $f_\eta: R^P \rightarrow R$, can be a linear or a nonlinear function and η parameterizes the combination function. The more common implementation is:

$$K_\eta(x_i, x_j) = f_\eta \left(\left\{ K_m(x_i^m, x_j^m) \right\}_{m=1}^P | \eta \right) \quad (7)$$

where the parameters are used to combine a set of predefined kernels (i.e., the kernel functions and corresponding kernel parameters are known before training) [2]. To implement a multiple kernel in an SVM, the requirement is that it fulfills the Mercer conditions [13]:

$$\begin{aligned} K(x, x') &= \sum_i^{\infty} a_i \varphi_i(x) \varphi_i(x'), \quad a_i > 0 \\ &\int_a^b \int_a^b K(x, x') g(x) g(x') dx dx' > 0 \end{aligned} \quad (8)$$

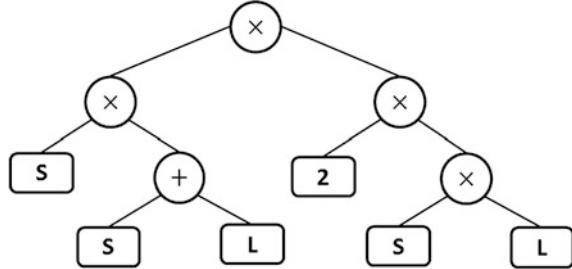
where the variables x and x' are defined in the closed square $a \leq x \leq b$, $a \leq x' \leq b$ and $g(x)$ and $g(x')$ are any function continuous in (a, b) .

Gram Matrix For a function $K(x, x'): R^n \times R^n \rightarrow R$ and l points $x_1, \dots, x_l \in R^n$, the $l \times l$ matrix G , of which the i -th row and the j -th column element is $G_{ij} = K(x_i, x_j)$, is called the Gram matrix of the function $K(x, x')$ with respect to x_1, \dots, x_l [11].

2.3 Genetic Programming

This paradigm can search the space of possible computer programs for an individual computer program that is highly fit in solving the problem at hand. Genetic

Fig. 1 Multiple kernel represented as a tree



Programming (GP) has been shown to be capable of inducing programs for fields as optimal control, planning, symbolic regression, automatic programming, and pattern classification among others. The induction is a result of the combination of an efficient learning procedure and expressive symbolic representations [14].

The program structure is closely related to a fitness function that guides certain evolution process. In the case of kernel evolution, tree data structures are commonly used for encoding programs as chromosomes. GP has been applied to find optimized kernel functions for SVM classification [1, 7]. In Fig. 1 a multiple kernel encoded as a tree is exemplified and correspond to the expression:

$$K_{\eta}(x_i, x_j) = (K_S(x_i, x_j) + K_L(x_i, x_j))(2 \times K_L(x_i, x_j) \times K_S^2(x_i, x_j)) \quad (9)$$

where $S = K_S(x_i, x_j)$ and $L = K_L(x_i, x_j)$ are the Sigmoid and Linear Kernel, respectively.

For kernel construction the next three steps are suggested: find out basic kernels, find out the operations keeping kernels and construct kernels from basic kernels applying operations [11]. In this work, the basic kernels are four: Linear, RBF, Sigmoid and Polynomial. Two keeping kernels operations are considered, addition (+) and multiplication (×). GP is used to construct kernels from basic kernels applying operations.

2.4 Decision Tree Support Vector Machine

Decision Tree Support Vector Machine (DTSVM) is a method for data reduction and was recently proposed in [3]. DTSVM aims to build a subset (X_R) of original set (X) such that X_R be much smaller than X , i.e., $X_R \subset X$: $|X_R| \ll |X|$. To obtain such a subset, the C4.5 algorithm is used to derive a Decision Tree (DT) which partitions the input space into regions with low entropy. Then, adjacent regions with opposite class label are detected. Finally, a Fisher Linear Discriminant (FLD) is applied to choose a proportion of elements nearest to the decision boundary whose could be support vectors. Figure 2 shows an illustration of the method.

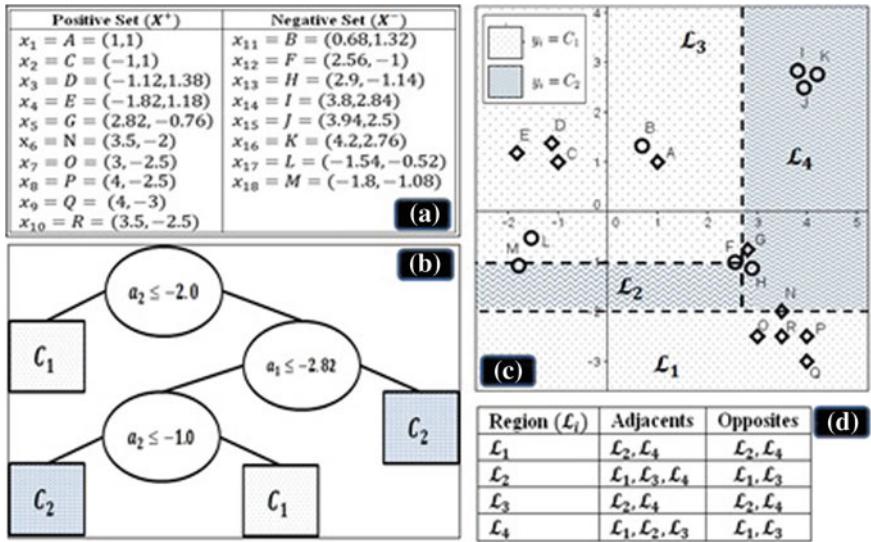


Fig. 2 Numerical example of DTSVM method. **a** Original dataset. **b** Induced decision tree from dataset. **c** Partition derived from DT. **d** Opposite regions to consider in FLD

In Fig. 2, a two class problem in R^2 space is presented. The original dataset (X) involves 18 instances, ten positives and eight negatives. C4.5 algorithm is applied to X and a DT is obtained as a result. Each leave (\mathcal{L}_i) of the induced DT represents one of the regions in which R^2 space has been divided. Each region is labeled with the majority class in order to detect regions with opposite label. Finally a FLD is applied to each pair of opposite regions to select a subset X_R whose elements are supposed to be support vectors. For deeper insight refer to the original work [3].

3 Methodology and Experiments

In this section, a kernel evolution process is described. Materials and experiments settings are specified.

3.1 General Overview for Kernel Evolution

Figure 3 illustrates the phases into which the kernel evolution process is split. First, a random population is created and genetically modified. Kernels obtained with genetic operators are used for training an SVM classifier. The classifier is training for a specific dataset by means of the libsvm method [15]. Accuracy is considered

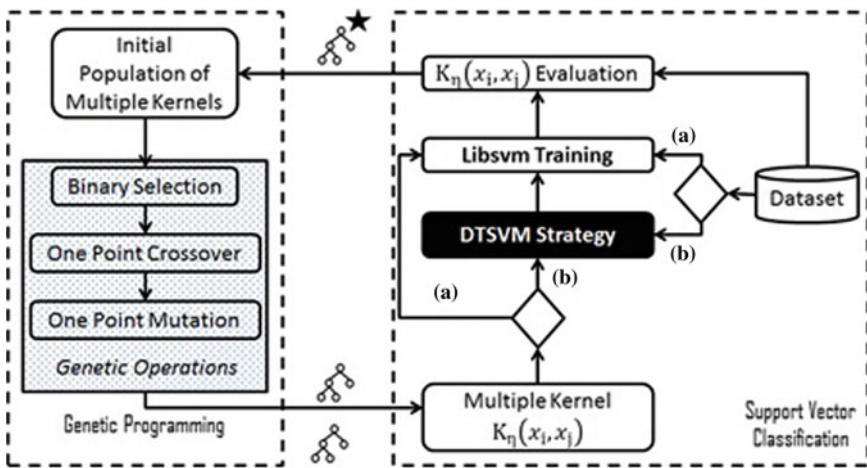


Fig. 3 Multiple kernel evolution. **a** Without data reduction. **b** Using data reduction

as a quality measure. The kernel with the best quality is passed to the population for the next generation replacing the worst individual.

In order to keep the evolution process as simple as possible, the accuracy rate for each candidate solution was calculated over the original data.

3.2 Datasets

Ten datasets with different number of instances and attributes were used in experiments. Table 2 provides relevant information about each dataset. The dataset sources are: the UCI machine learning [16], libsvm and delve [15] repositories.

Table 2 Datasets for experiments

Dataset	Instances	Attributes	Number of positives	Number of negatives	Source
Haberman	306	3	225	81	UCI
Ionosphere	351	34	124	217	UCI
Breast	683	10	444	239	UCI
Pima	768	8	500	268	UCI
Fourclass	862	2	307	555	libsvm
Splice	1000	60	517	483	delve
a4a	4781	123	1188	3593	UCI
a5a	6414	123	1284	5130	UCI
a6a	11,220	123	2692	8528	UCI
cod-rna	59,535	8	19,845	39,690	libsvm

Table 3 Configuration of algorithms for kernel evolution

Parameters for genetic programming	Value	Parameters for SVM and DTSVM	Value
Replications	50	C	2
Population size	5	d (degree)	2
Tree depth	3	σ (scale)	1
Generations	2	r (offset)	0
ϵ (Stop criterion)	$\epsilon < 10^{-3}$	DT algorithm	C4.5
Mutation rate	0.2	Impurity measure.	$-\sum_{i=0}^{C_L-1} p(i t) \log_2 p(i t)$
Crossover rate	0.8	Entropy (t)	
Operator set	{ $\times, +$ }	DT pruning	Post pruning
Terminal set	{Linear, sigmoid, polynomial, RBF}	DT stop splitting criterion	10 % of dataset in one leaf
Initialization method	Grow	Proportion of support vectors candidates	10 % of closest element to the FLD threshold
Mutation method	One point		
Selection method	Binary selection		

Table 3 lists the settings for the evolution process. Replications are the times a whole evolution process was run for a specific dataset. Population size indicates the number of multiple kernels randomly generated. Tree depth specifies the maximum number of levels a tree can reach.

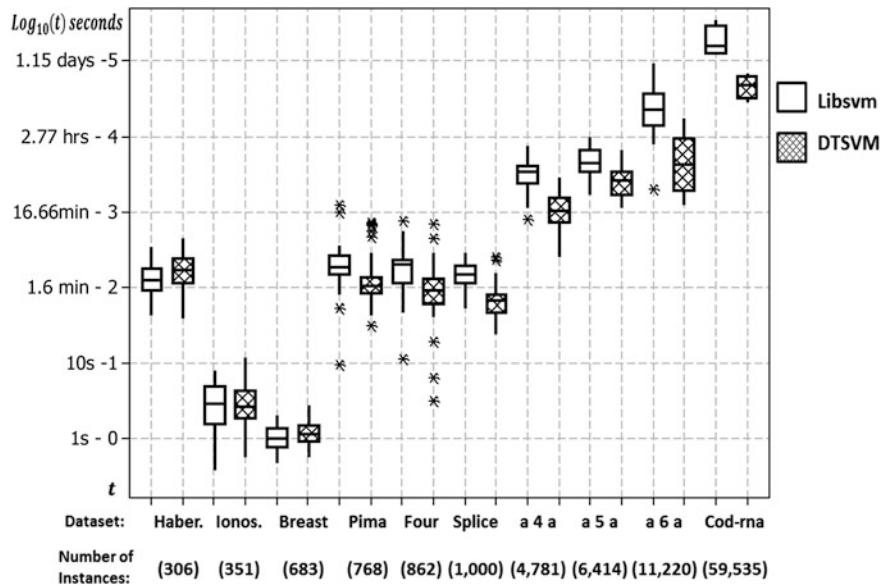
Generations establishes the times the whole kernel population is genetically modified and evaluated. ϵ stands for the tolerance error in accuracy rate. C represents the SVM error penalty. d, σ and r are kernel parameters. C_L is the number of classes. $p(i|t) = \frac{|x_i=t|}{|X|}$ is the probability of example i to be in class t .

Parameters were chosen to construct a context in which the evolution process could be fully measured using a small amount of time. It is important to point out that any other parameter configuration could have been used without loss of expression since the main goal is to observe if the evolution process implementing the DTSVM method is finished faster than without implementing it under the same scenario.

All experiments were run in a computer with the next features: i7 core 3.2 GHz processor, 8 GB RAM, 100 GB SSD. Java 8 SE was the programming language and Windows 7 ultimate the operating system. The random number generator for tests was the one implemented by Random java class. For each replication the current time was considered as initial seed.

4 Results and Discussion

In Fig. 4 it is shown the time required for evolving kernels with and without using DTSVM. The vertical axis measures the time in $\log_{10}(t)$ scale, where t is in seconds. The horizontal axis measures the number of instances a dataset contains.

**Fig. 4** Box plot of time for evolving kernels with genetic programming**Table 4** Numerical results for kernel evolution

Dataset	Instances	Method	Max.	Avg.	Med	Min	Std. dvt.
Haberman	306	libsvm	5.75	2.31	2.03	0.70	1.10
		DT-SVM	7.38	3.08	2.83	0.66	1.63
Ionosphere	351	libsvm	0.13	0.05	0.05	0.01	0.03
		DT-SVM	0.19	0.05	0.04	0.01	0.04
Breast	683	libsvm	0.03	0.02	0.02	0.01	0.01
		DT-SVM	0.05	0.02	0.02	0.01	0.01
Pima	768	libsvm	20.48	3.83	3.11	0.16	3.33
		DT-SVM	12.10	2.61	1.74	0.52	2.79
Fourclass	862	libsvm	12.83	3.28	3.28	0.19	2.05
		DT-SVM	11.37	2.07	1.53	0.05	1.92
Splice	1000	libsvm	4.85	2.47	2.46	0.89	0.90
		DT-SVM	4.10	1.25	1.10	0.40	0.74
a4a	4781	libsvm	124.19	55.63	56.81	13.08	22.62
		DT-SVM	47.25	19.63	17.51	4.23	9.71
a5a	6414	libsvm	163.15	82.42	73.97	27.85	32.46
		DT-SVM	111.58	46.87	44.01	19.03	22.53
a6a	11,220	libsvm	1570.51	486.55	386.66	33.67	355.86
		DT-SVM	292.68	100.87	71.96	20.41	82.93
cod-rna	59,535	libsvm	5857.27	3321.75	2680.18	2069.38	1725.03
		DT-SVM	1138.75	808.94	814.31	468.38	274.86

Times are presented in minutes

In each pair, the first box and the second box represents the behavior of training an SVM with or without using DTSVM respectively.

It can be observed from the chart that the bigger the dataset is, the more benefit from the new method is obtained. With small datasets (less than 1000 instances) the method has random behavior and the time required for evolving kernels is more likely to be dependent on the dataset complexity or any other variable rather than the dataset size.

Table 4 summarizes the statistics of training time (in minutes). For each dataset the first row represents the results of training only with libsvm. The second row shows results of applying data reduction DTSVM and then training with libsvm. Large datasets are notoriously benefited from data reduction.

5 Conclusion and Future Directions

In this chapter the DTSVM method was tested for kernel evolution on large datasets. Results indicate that evolutionary strategies could take advantage from the data reduction method when datasets are greater than 1000 instances. For smaller datasets, variations in time make explicit the need to carry out further research in order to determine if a correlation exists between the dataset size and the time required for evolving kernels with DTSVM.

Based on the fact that kernel evolution has been recently analyzed only with small datasets, and considering that methods for training SVMs faster are emerging as a current trend, authors believe that this is, likely, the first work that focus the kernel evolution for large datasets.

Results motivate to explore future directions. Authors suggest: to analyze the effect of more methods for fast training of SVM in kernel evolution; to extend the diversity of datasets to increase the confidence training time results; to study how the new method impacts in the algorithmic complexity for training an SVM.

Acknowledgments Luis Carlos Padierna García wishes to acknowledge the financial support of the Consejo Nacional de Ciencia y Tecnología (CONACYT grant 375524). The authors also thank the support of the Tecnológico Nacional de México – Instituto Tecnológico de León.

References

1. Diosan, L., Rogozan, A., Pecuchet, J.-P.: Learning SVM with complex multiple kernels evolved by genetic programming. *Int. J. Artif. Intell. Tools* **19**(5), 647–677 (2010)
2. Gönen, M., Alpaydin, E.: Multiple kernel learning algorithms. *J. Mach. Learn. Res.* **12**, 2211–2268 (2011)
3. Asdrúbal, C., Xiaoou, L., Wen, Y.: Support vector machine classification for large datasets using decision tree and fisher linear discriminant. *Future Gener. Comput. Syst.* **36**, 57–65 (2014)

4. Shigeo, A.: Support vector machines for pattern classification. Springer, New York (2010)
5. Essam, A.D., Hamza, T.: New empirical nonparametric kernels for support vector machines classification. *Appl. Soft Comput.* **13**, 1759–1765 (2013)
6. Castro, E., Gómez-Verdejo, V., Martínez-Ramón, M., Kiehl, K. A., Kalhoun, V.D.: A multiple kernel learning approach to perform classification of groups from complex-valued fMRI data analysis-application to schizophrenia. *NeuroImage* **87**, 1–17 (2014)
7. Koch, P., Bischl, B., Flasch, O., Bartz-Beielstein, T., Weihs, C., Konen, W.: Tuning and evolution of support vector machines. *Evol. Intell.* 1–30 (2011)
8. Padierna, L.C., Carpio, J.M., Baltazar, M.D.R., Puga, H.J., Fraire, H.J.: Muliple kernel support vector machine is np-complete. In: Gelbukh, A., Félix, C., Galicia-Haro, S. (eds.) *Nature Inspired Computation and Machine Learning*, Springer International Publishing, Switzerland (2014)
9. Tsang, I., Kwok, J., Cheung, P.-M.: Core vector machines-fast SVM training on very large data sets. *J. Mach. Learn. Res.* **363**–392 (2005)
10. Vapnik, V.: *Statistical Learning Theory*. Wiley, New York (1998)
11. Deng, N., Tian, Y., Zhang, C.: *Support Vector Machines*. CRC Press, Boca Raton (2013)
12. Bollchandani, D., Sahula, V.: Exploring efficient kernel functions for support vector machines based feasibility models for analog circuits. *Int. J. Des. Anal. Tools Circ. Syst.* 1–8 (2011)
13. Mercer, J.: Functions of positive and negative type, and their connection with the theory of integral equations. *Philoso. Trans. Roy. Soc. London A Math. Phys. Eng. Sci.* **209**, 415–446 (1909)
14. Koza, J.R.: *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge (1992)
15. Chih-Chung, C., Chih-Jen, L.: Libsvm-a library for support vector machines. *ACM Trans. Intell. Syst. Technol.* **2**(3), 1–27 (2011)
16. Asuncion, A., Newman, D.J.: UCI machine learning repository. University of California, Irvine, CA (2007). <http://www.ics.uci.edu/~mlearn/MLRepository.html>

Part III

Neural Networks Applications

Modular Neural Networks for Time Series Prediction Using Type-1 Fuzzy Logic Integration

Daniela Sánchez and Patricia Melin

Abstract In this paper, a new method to perform the times series prediction using modular neural networks with type-1 fuzzy logic integration is proposed. The proposed method consists in the division of a dataset into modules and each module learns a specific period of the time series. Each module has a prediction, but the final prediction is obtained using type-1 fuzzy logic. To prove the effectiveness of the proposed method, 800 points of the Mackey-Glass time series are used; 500 points are used for the training phase and 300 for the testing phase. In this work the number of modules is fixed established but the number of points in each module for the training phase is randomly established. Different trainings are performed using different modular neural architectures (number of hidden layers and neurons) and a type-1 fuzzy integrator is used to perform a final comparison.

1 Introduction

A time series is a set of observations x_t , each one being recorded at a specific time t [3]. Current time series forecasting methods generally fall into two groups: methods based on statistical concepts and computational intelligence techniques. Computational intelligence techniques for time series forecasting generally fall into two major categories: first, the methods based on neural networks (NNs); and second, the methods based on evolutionary computation. The second category is divided into methods based on genetic algorithm (GA), evolutionary programming

D. Sánchez · P. Melin (✉)
Tijuana Institute of Technology, Tijuana, Mexico
e-mail: pmelin@tectijuana.mx

D. Sánchez
e-mail: danielasanchez.itt@hotmail.com

(EP), and genetic programming (GP). All these methods are inspired by the study of biological processes [22].

Hybrid intelligent systems are computational systems that integrate different intelligent techniques. These systems are now being used to support complex problem solving and decision making in a wide variety of tasks. This kind of systems allows the representation and manipulation of different types and forms of data and knowledge which may come from various sources [28]. There are various works where this kind of systems have been proposed and they combine different techniques and good results have been obtained, such as in [4, 12, 15, 20], where techniques as modular neural networks (MNNs), fuzzy logic (FL) or GA has been integrated among them. In this work, two intelligence techniques are combined; modular neural networks and type-1 fuzzy logic. These techniques are used because in other works have demonstrated to be a good combination of techniques, such as in [13, 14, 19].

This paper is organized as follows: Sect. 2 contains the basic concepts used in this research work, Sect. 3 contains the proposed method, Sect. 4 presents experimental results and in Sect. 5, the conclusions of this work are presented.

2 Basic Concepts

In this section, a brief overview of the basic concepts used in this research work is presented.

2.1 Times Series

A time series is a sequential set of data points, measured typically over successive times. A time series in general is supposed to be affected by four main components, which can be separated from the observed data. These components are: Trend, Cyclical, Seasonal and Irregular components. Time series observations are frequently encountered in many domains such as business, economics, industry, engineering and science, etc. [21, 26, 27]. Depending on the nature of analysis and practical need, there can be various different kinds of time series [1].

2.2 Modular Neural Networks

A Modular Neural Network (MNN) is a Neural Network (NN) that consists of several modules, each module carrying out one subtask of the global task [3], it

means that the network can be decomposed into two or more modules (sub-systems) that operate on distinct inputs without communicating with each other [2, 7] (each neural network works independently in its own domain). The final decision is based on the results of the individual networks, called agents or experts [11]. Modular artificial neural networks are especially efficient for certain classes of regression and classification problems, as compared to the conventional monolithic artificial neural networks. The modular neural networks are comprised of modules which can be categorized on the basis of both distinct structure and functionality which are integrated together via an integrating unit. With functional categorization, each module is a neural network which carries out a distinct identifiable subtask [2].

2.3 Type-1 Fuzzy Logic

Zadeh introduced the term fuzzy logic in his seminal work “Fuzzy sets” which described the mathematics of fuzzy set theory (1965). FL techniques have been used in image-understanding applications such as detection of edges, feature extraction, classification, and clustering. Fuzzy logic poses the ability to mimic the human mind to effectively employ modes of reasoning that are approximate rather than exact [5]. Fuzzy logic is a useful tool for modeling complex systems and deriving useful fuzzy relations or rules [16]. However, it is often difficult for human experts to define the fuzzy sets and fuzzy rules used by these systems [23]. The basic structure of a fuzzy inference system consists of three conceptual components: a rule base, which contains a selection of fuzzy rules, a database (or dictionary) which defines the membership functions used in the rules, and a reasoning mechanism that performs the inference procedure [6, 24, 25].

3 General Architecture of the Proposed Method

The proposed method combines modular neural networks (MNNs) and type-1 fuzzy logic as integration techniques to perform the time series prediction. As it known, any kind of neural network (artificial, ensemble or modular neural network) has two main phases; training and testing phase. In the artificial or ensemble neural networks, all the data points used for the training phase are learning for the module or sub modules (depending on the king of neural network) [8, 17, 18]. In the proposed method the idea is to divide the data points used for the training phase into sub module and so on, each sub module will be an expert of a part of the time series. The number of data points for each module will be randomly established. An

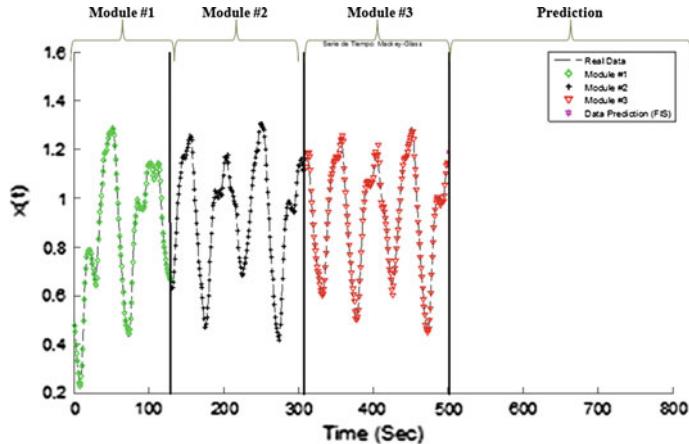


Fig. 1 Example of the proposed method

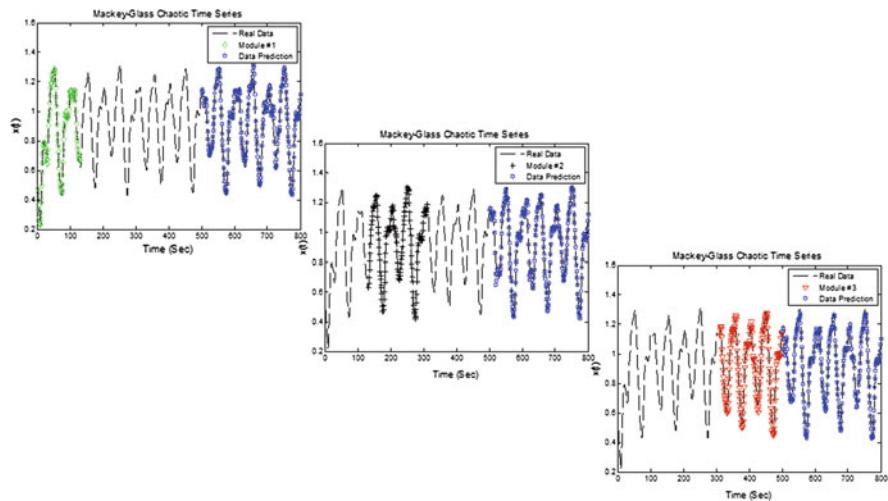


Fig. 2 Example of the proposed method showing the three modules

example of this is shown in Fig. 1, where 3 sub modules are used and each module learns a part of the data points used for the training phase.

The question would be, how to obtain a final prediction? Firstly, each module has a prediction as Fig. 2 shows, where each module has an individual prediction.

To obtain a final prediction these predictions are integrated. In the proposed method type-1 fuzzy logic is used. In this case, 3 modules are used in all the

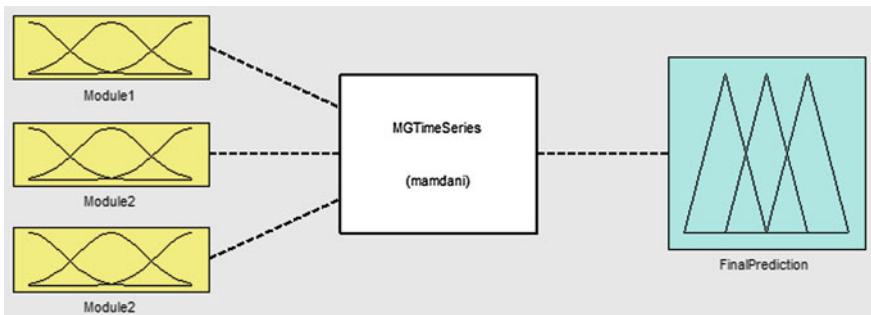


Fig. 3 Example of the fuzzy integrator

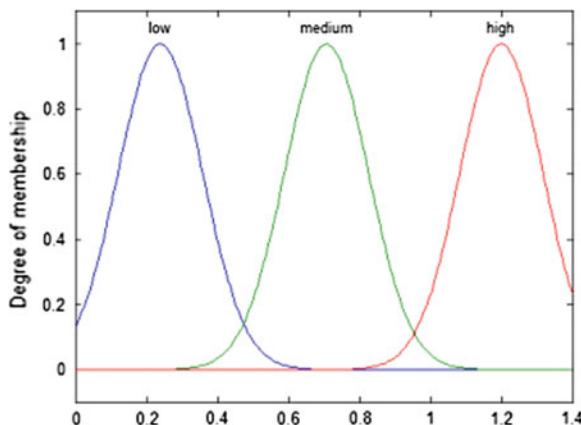


Fig. 4 Example of variable of the fuzzy integrator

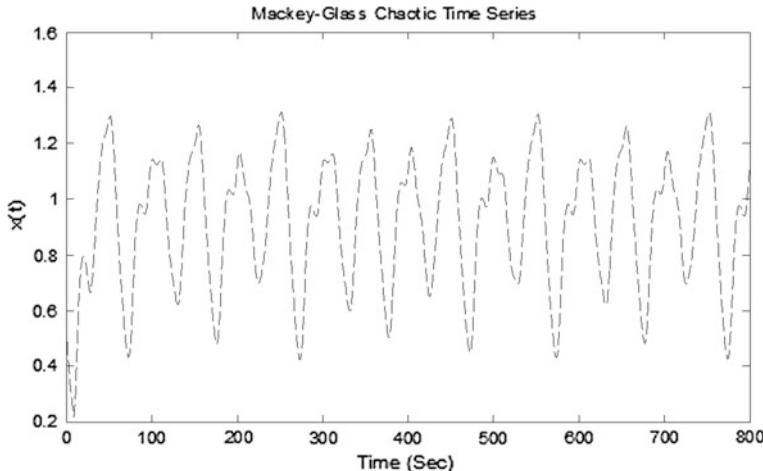
trainings for this reason the fuzzy integrator has 3 inputs, each input represents a module. The fuzzy integrator has an output to obtain a final prediction. An example of the fuzzy integrator is shown in Fig. 3.

The fuzzy integrator is Mamdani type, and each variable (inputs and outputs) has 3 Gaussian membership functions. In Fig. 4, an example of these variables is shown. The fuzzy rules used in the fuzzy inference system are shown in Fig. 5.

3.1 Time Series

To prove the effectiveness of the proposed method, 800 data points of the Mackey-Glass time series are used that correspond to a period from 09/11/05 to 15/01/09,

1. If (input1 is low) and (input2 is low) and (input3 is low) then (output is low) (1)
2. If (input1 is low) and (input2 is low) and (input3 is medium) then (output is low) (1)
3. If (input1 is low) and (input2 is low) and (input3 is high) then (output is low) (1)
4. If (input1 is low) and (input2 is medium) and (input3 is low) then (output is low) (1)
5. If (input1 is low) and (input2 is medium) and (input3 is medium) then (output is medium) (1)
6. If (input1 is low) and (input2 is medium) and (input3 is high) then (output is high) (1)
7. If (input1 is low) and (input2 is high) and (input3 is low) then (output is low) (1)
8. If (input1 is low) and (input2 is high) and (input3 is medium) then (output is medium) (1)
9. If (input1 is low) and (input2 is high) and (input3 is high) then (output is high) (1)
10. If (input1 is medium) and (input2 is low) and (input3 is low) then (output is low) (1)
11. If (input1 is medium) and (input2 is low) and (input3 is medium) then (output is medium) (1)
12. If (input1 is medium) and (input2 is low) and (input3 is high) then (output is high) (1)
13. If (input1 is medium) and (input2 is medium) and (input3 is low) then (output is medium) (1)
14. If (input1 is medium) and (input2 is medium) and (input3 is medium) then (output is medium) (1)
15. If (input1 is medium) and (input2 is medium) and (input3 is high) then (output is medium) (1)
16. If (input1 is medium) and (input2 is high) and (input3 is low) then (output is low) (1)
17. If (input1 is medium) and (input2 is high) and (input3 is medium) then (output is medium) (1)
18. If (input1 is medium) and (input2 is high) and (input3 is high) then (output is high) (1)
19. If (input1 is high) and (input2 is low) and (input3 is low) then (output is low) (1)
20. If (input1 is high) and (input2 is low) and (input3 is medium) then (output is medium) (1)
21. If (input1 is high) and (input2 is low) and (input3 is high) then (output is high) (1)
22. If (input1 is high) and (input2 is medium) and (input3 is low) then (output is low) (1)
23. If (input1 is high) and (input2 is medium) and (input3 is medium) then (output is medium) (1)
24. If (input1 is high) and (input2 is medium) and (input3 is high) then (output is high) (1)
25. If (input1 is high) and (input2 is high) and (input3 is low) then (output is high) (1)
26. If (input1 is high) and (input2 is high) and (input3 is medium) then (output is high) (1)
27. If (input1 is high) and (input2 is high) and (input3 is high) then (output is high) (1)

Fig. 5 Fuzzy rules**Fig. 6** Mackey-Glass time series

500 data points are used for the training phase and 300 for the testing phase. The Mackey-Glass time series [9, 10] is shown in Fig. 6. The division of the data points for training and testing phase is represented in Fig. 7.

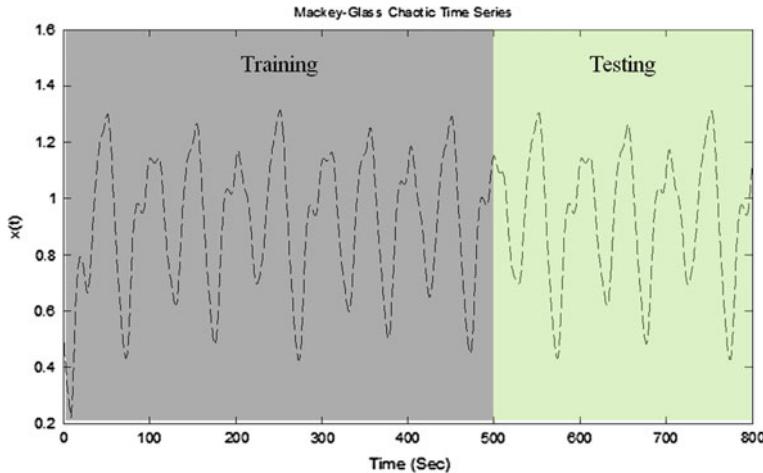


Fig. 7 Data points for training and testing phase

4 Experimental Results

The Different trainings performed using different modular neural network architectures (number of hidden layers and neurons) using the type-1 fuzzy integrator is shown in this section. In this work 3 tests were performed, in each case 30 trainings were performed where the number of neurons is randomly established.

4.1 Using One Hidden Layer in Each Module (Test #1)

In this test, the modules have one hidden layer in all the modules. In Table 1, the 5 trainings with the lowest prediction error are shown. The training #26 is the best training in this test. The individual prediction of each module of the best training is presented in Fig. 8.

The final prediction can be observed in Fig. 9. In this training 0.058734 of prediction error was obtained.

4.2 Using Two Hidden Layer in Each Module

In this test, the modules have two hidden layer in all the modules. In Table 2, the 5 trainings with the lowest prediction error are shown. The training #29 is the best

Table 1 Best results using one hidden layer in each module

Training	Data points per module	Neurons	Error
6	190	32	0.062634
	240	35	
	70	33	
10	140	27	0.062603
	130	28	
	230	15	
17	80	21	0.061375
	220	24	
	200	32	
24	180	30	0.060838
	230	22	
	90	35	
26	260	30	0.058734
	30	31	
	210	25	

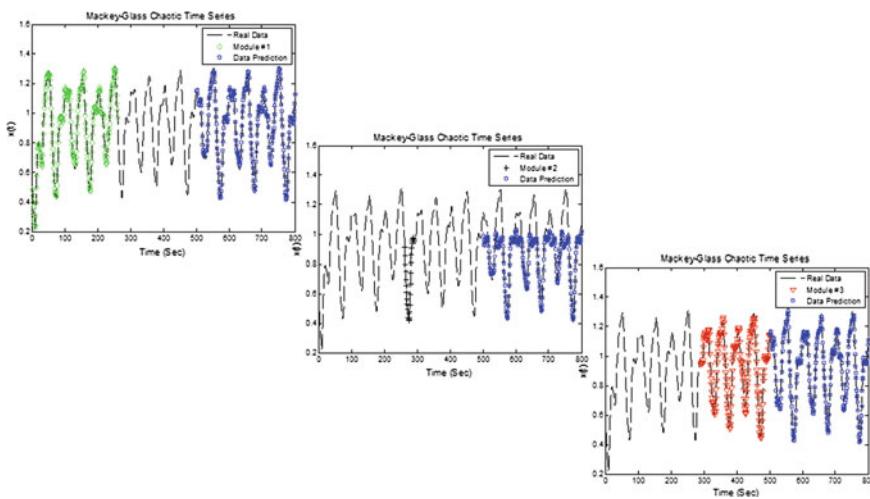


Fig. 8 Individual prediction of each module (best training, test #1)

training in this test. The individual prediction of each module of the best training is presented in Fig. 10.

The final prediction can be observed in Fig. 11. In this training 0.061378 of prediction error was obtained.

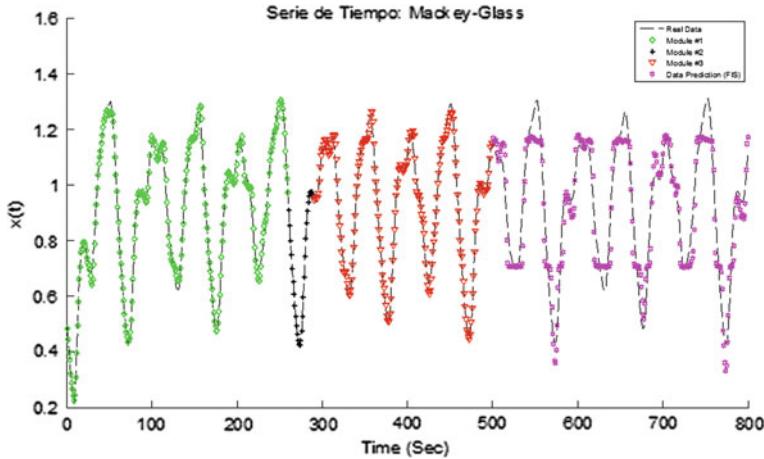


Fig. 9 Best prediction for the test #1

Table 2 Results using two hidden layer in each module

Training	Data points per module	Neurons	Error
15	220	25, 32	0.063906
	240	15, 20	
	40	19, 21	
23	340	25, 33	0.061837
	70	35, 36	
	90	19, 16	
27	70	24, 39	0.064014
	240	18, 17	
	190	24, 24	
29	260	21, 32	0.061378
	220	14, 20	
	20	32, 27	
30	250	12, 29	0.064351
	140	33, 37	
	110	38, 34	

4.3 Using One and Two Hidden Layer in Each Module

In this test, the modules have one and two hidden layer, the number in this case was randomly established. In Table 3, the 5 trainings with the lowest prediction error are shown. The training #16 is the best training in this test. The individual prediction of each module of the best training is presented in Fig. 12.

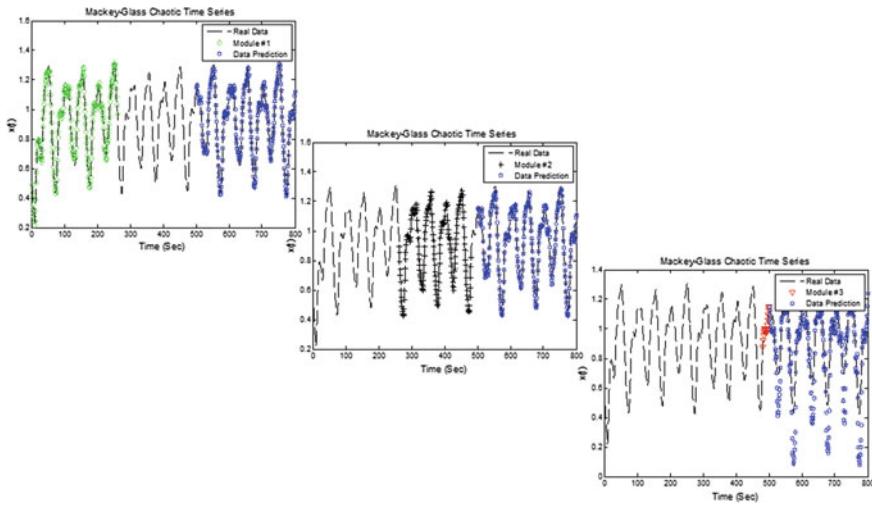


Fig. 10 Individual prediction of each module (best training, test #2)

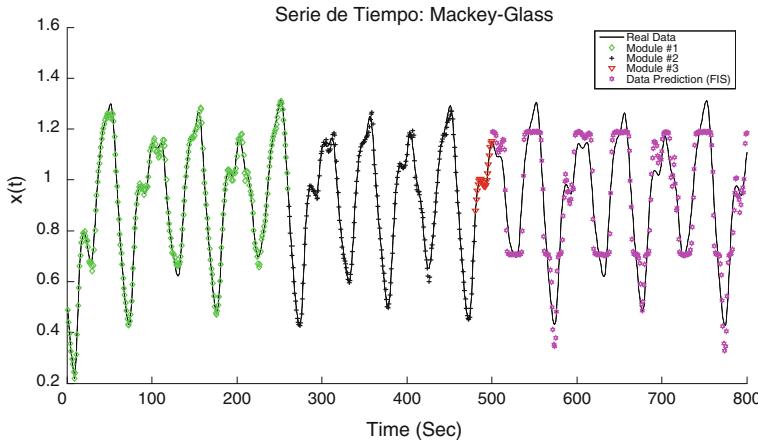


Fig. 11 Best prediction for the test #2

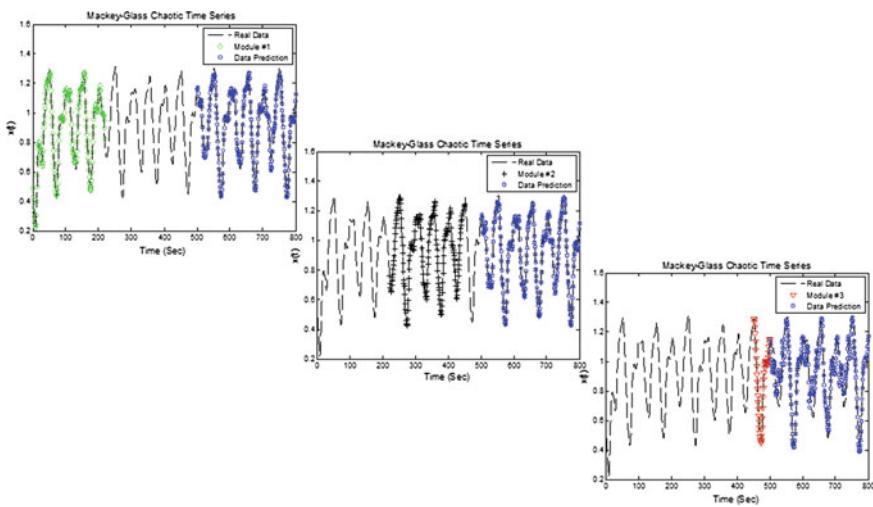
The final prediction can be observed in Fig. 13. In this training 0.058308 of prediction error was obtained.

4.4 Comparison of Results

In Table 4, a comparison of the results obtained is presented, where it can be observed that the lowest prediction error was obtained in the test #3. This training

Table 3 Results using one and two hidden layer in each module

Training	Data points per module	Neurons	Error
2	180	29, 24	0.063888
	270	33	
	50	33, 24	
15	80	22	0.062376
	170	24	
	250	23, 15	
16	220	28	0.058308
	230	27	
	50	24, 23	
19	170	23	0.063545
	220	29	
	110	19, 26	
21	130	11	0.064204
	180	33	
	190	13, 14	

**Fig. 12** Individual prediction of each module (best training, test #3)

uses one hidden layer in the first and the second module and two hidden layers in the third module and has a prediction error of 0.058308. But, the best average of the 30 trainings was obtained in the test #1.

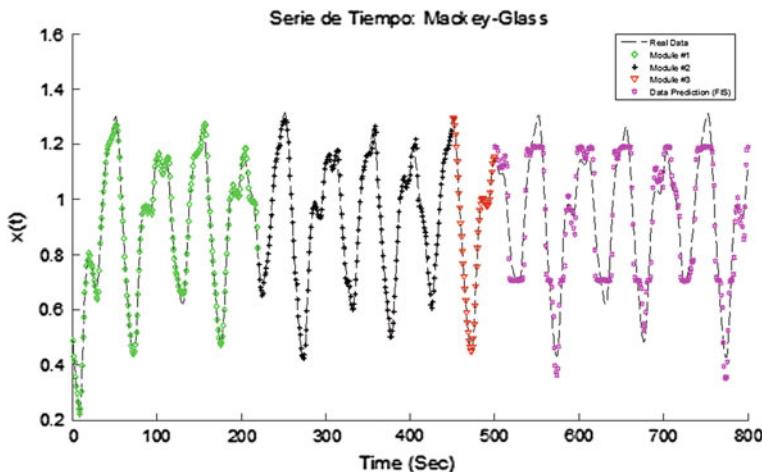


Fig. 13 Best prediction for the test #3

Table 4 Comparison of results

Test	Training	Data points per module	Neurons	Error	Average
1	26	260	30	0.058734	0.065408
		30	31		
		210	25		
2	29	260	21, 32	0.061378	0.069158
		220	14, 20		
		20	32, 27		
3	16	220	28	0.058308	0.067453
		230	27		
		50	24, 23		

5 Conclusions

In this work a new method combining modular neural networks and type-1 fuzzy logic used for time series prediction is proposed, where the main idea is to divide the data points for the training phase into sub modules, and each sub module learns a part of the time series and has a prediction. The final prediction is obtained using a type-1 fuzzy integrator Mamdani Type with 3 Gaussian membership functions in each variable (inputs and outputs) with 27 fuzzy rules. 3 tests were performed, where the number of hidden layers and neurons were varied, each test has 30 non-optimized trainings. The best prediction was obtained in the test #3, but the best average was obtained with the test #1. As the difference among the different tests is small, obtaining optimal architectures is better. For this reason as future work the

optimization of some parameters of the neural networks is proposed as the number of data points used for the training phase, the number of modules, hidden layers and their neurons. Also the optimization of some parameters of the fuzzy integrator is proposed as the type of membership function, the number of membership functions with their parameters and the fuzzy rules.

References

1. Adhikari, R., Agrawal, R.K.: An introductory study on time series modeling and forecasting. LAP Lambert Academic Publishing, Germany (2013)
2. Azamm, F.: Biologically inspired modular neural networks. PhD thesis, Virginia Polytechnic Institute and State University, Blacksburg, Virginia (2000)
3. Brockwell, P.J., Davis, R.A.: Introduction to Time Series and Forecasting, 2nd edn. Springer, Berlin (2002)
4. Hidalgo, D., Castillo, O., Melin, P.: Type-1 and type-2 fuzzy inference systems as integration methods in modular neural networks for multimodal biometry and its optimization with genetic algorithms. *Soft Comput. Hybrid Intell. Syst.* pp. 89–114 (2008)
5. Hofmann, H.D.: Application of intelligent measurements with metrical image processing for quality control. In: Resented at the 5th International Conference, PEDAC' 92, Alexandria, Egypt, Dec 2005
6. Jang, J., Sun, C., Mizutani, E.: Neuro-Fuzzy and Soft Computing. Prentice Hall, New Jersey (1997)
7. Khan, A., Bandopadhyaya, T., Sharma, S.: Classification of stocks using self organizing map. *Int. J. Soft Comput. Appl.* **4**, 19–24 (2009)
8. Khashai, M., Bijari, M.: An artificial neural network (p, d, q) model for time series forecasting. *Expert Syst. Appl. Int. J.* **37**(1), 479–489 (2010)
9. Mackey, M.C.: Adventures in Poland: having fun and doing research with Andrzej Lasota. *Mat. Stosow.* pp. 5–32 (2007)
10. Mackey, M.C., Glass, L.: Oscillation and chaos in physiological control systems. *Science* **197**, 287–289 (1997)
11. Melin, P., Castillo, O.: Hybrid Intelligent Systems for Pattern Recognition Using Soft Computing: An Evolutionary Approach for Neural Networks and Fuzzy Systems, 1st edn, pp. 119–122. Springer, Berlin (2005)
12. Melin, P., Kacprzyk J., Pedrycz, W: Bio-inspired Hybrid Intelligent Systems for Image Analysis and Pattern Recognition. Springer, Berlin (2009)
13. Melin, P., Sánchez, D., Castillo, O.: Genetic optimization of modular neural networks with fuzzy response integration for human recognition. *Inf. Sci.* **197**, 1–19 (2012)
14. Muñoz, R., Castillo, O., Melin, P.: Face, fingerprint and voice recognition with modular neural networks and fuzzy integration. In: Bio-inspired Hybrid Intelligent Systems for Image Analysis and Pattern Recognition 2009, pp. 69–79. Springer, Berlin (2009)
15. Nawa, N., Takeshi, F., Hashiyama, T., Uchikawa, Y.: A study on the discovery of relevant fuzzy rules using pseudobacterial genetic algorithm. *IEEE Trans. Indu. Electron.* **46**(6), 1080–1089 (1999)
16. Okamura, M., Kikuchi, H., Yager, R., Nakanishi, S.: Character diagnosis of fuzzy systems by genetic algorithm and fuzzy inference. In: Proceedings of the Vietnam-Japan Bilateral Symposium on Fuzzy Systems and Applications, Halong Bay, Vietnam, pp. 468–473 (1998)
17. Pulido, M., Melin, P., Castillo, O.: Optimization of Ensemble Neural Networks With Type-2 Fuzzy Response Integration for Predicting the Mackey-Glass Time Series. NaBIC 2013, Fargo, USA, pp. 16–21 (2013)

18. Pulido, M., Melin, P., Castillo, O.: Optimization of Type-2 Fuzzy Integration in Ensemble Neural Networks for Predicting the US Dolar/MX Pesos Time Series. IFSANAFIPS 2013, Edmonton, Canada, pp. 1508–1512 (2013)
19. Sánchez, D., Melin, P.: Modular neural network with fuzzy integration and its optimization using genetic algorithms for human recognition based on iris, ear and voice biometrics. In: Soft Computing for Recognition Based on Biometrics Studies in Computational Intelligence, 1st edn, pp. 85–102. Springer, Berlin (2010)
20. Santos, J.M., Alexandre, L.A., Marques de Sá J.: Modular Neural Network Task Decomposition Via Entropic Clustering. ISDA 2006, Jinan, China, pp. 62–67 (2006)
21. Tong, H.: Threshold Models in Non-Linear Time Series Analysis. Springer, New York (1983)
22. Wagner, N., Michalewicz, Z., Schellenberg, S., Chiriac, C., Mohais, A.: Intelligent techniques for forecasting multiple time series in real-world systems. *Int. J. Intell. Comput. Cybern.* **4**(3), 284–310 (2011)
23. Wang, W., Bridges, S.: Genetic Algorithm Optimization of Membership Functions for Mining Fuzzy Association Rules. Department of Computer Science Mississippi State University (2000)
24. Zadeh, L.A.: Fuzzy Sets. *J. Inf. Control* **8**, 338–353 (1965)
25. Zadeh, L.A.: Some reflections on soft computing, granular computing and their roles in the conception, design and utilization of information/intelligent systems. *Soft. Comput.* **2**, 23–25 (1998)
26. Zhang, G.P.: A neural network ensemble method with jittered training data for time series forecasting. *Inf. Sci.* **177**, 5329–5346 (2007)
27. Zhang, G.P.: Time series forecasting using a hybrid ARIMA and neural network model. *Neurocomputing* **50**, 159–175 (2003)
28. Zhang, Z., Zhang, C.: An Agent-Based Hybrid Intelligent System for Financial Investment Planning. PRICAI 2002, Tokyo, Japan, pp. 355–364 (2002)

An Improved Particle Swarm Optimization Algorithm to Optimize Modular Neural Network Architectures

Alfonso Uriarte, Patricia Melin and Fevrier Valdez

Abstract According to the literature of Particle Swarm Optimization (PSO), there are problems of local minimum and premature convergence with this algorithm. A new algorithm is presented called the Improved Particle Swarm Optimization using the gradient descent method (BP algorithm) as operator of particle swarm incorporated into the Algorithm, as a function to test the improvement. The Gradient Descent Method (BP Algorithm) helps not only to increase the global optimization ability, but also avoid the premature convergence problem. The Improved PSO Algorithm IPSO is applied to Neural Network to optimize the architecture. The results show that there is an improvement with respect to using the conventional PSO Algorithm.

1 Introduction

The Particle Swarm Optimization (PSO) algorithm is inspired by the movements of bird flocks and the mutual collaboration among themselves in seeking food within the shortest period of time [1]. PSO is one of the most popular optimization algorithms due to its extremely simple procedure, easy implementation and very fast rate of convergence. Apart from all the advantages, the algorithm has its drawbacks too. The PSO algorithm faces problems with premature convergence as it is easily trapped into local optima. It is known that it is almost impossible for the PSO algorithm to escape from the local optima once it has been trapped, causing the

A. Uriarte · P. Melin (✉) · F. Valdez
Tijuana Institute of Technology, Tijuana, Mexico
e-mail: pmelin@tectijuana.edu.mx

A. Uriarte
e-mail: alfonso.uriarteg@gmail.com

F. Valdez
e-mail: fevrier@tectijuana.edu.mx

algorithm to fail in achieving the near optimum result. Many methods have been proposed throughout the years to counter the drawback of the PSO algorithm [2, 3].

Although some improvement measures have been proposed, such as increasing population scale, the dynamic adjustment coefficient of inertia, rights to a certain extent the improve the optimization algorithm performance, but the algorithm itself has some of the nature problem that is not solved.

In order to apply the particle swarm algorithm in training neural network avoiding slow convergence speed and so on, this paper combines the particle swarm algorithm with the gradient descending method, the article proposes an improved particle swarm optimization algorithm, this method has the purpose of combining the advantages of particle swarm algorithm global parallel search and the gradient descending method local certainty search, improve of convergence speed and avoid the local minima.

2 Modular Neural Networks

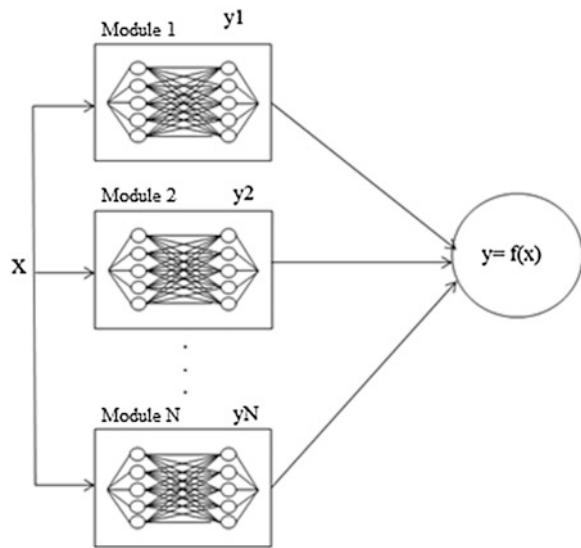
The modular neural network consists of a network of interconnected simple neural networks (also called modules) each of which solves a sub problem of the complete problem. The greatest benefit of modular neural networks is the inherent separation of the overall problem into sub problems and concerns.

A modular neural network is a set of simple networks and an integrating unit that determines how the outputs of the networks are applied to the final output. The simple network is also called an expert network. The knowledge networks and integrating unit work together to learn how to divide a task into subtasks that are functionally independent. The integration unit measures the competence of each expert network and assigns each task to learn different networks. This modular architecture avoids learning a global control policy set that cannot be good for the entire control each task under different operating points. Figure 1 shows the architecture of a Modular Neural Network.

3 Gradient Descent Method

The gradient descent method is an algorithm to find a local minimum of a function. The way it works is we start with an initial guess of the solution and we take the gradient of the function at that point. We move the solution in the negative direction of the gradient and we repeat the process. The algorithm will eventually converge where the gradient is zero (which corresponds to a local minimum). Its brother, the gradient ascent, finds the local maximum nearer the current solution by stepping it towards the positive direction of the gradient. They are both first-order algorithms because they take only the first derivative of the function.

Fig. 1 Modular neural network architecture



The BP algorithm is a kind of simple deterministic local search method, it uses the local adjustments aspects with strong performance, along the direction of gradient descent can quickly find local optimal solution, but the BP algorithm is very sensitive to the choice of the initial position, and does not ensure that the optimal solution is the global optimal.

4 Particle Swarm Optimization

PSO was formulated originally by Eberhart and Kennedy in 1995. The thought process behind the algorithm was inspired by the social behavior of animals, such as bird flocking or fish schooling. PSO is similar to the continuous GA in that it begins with a random population matrix. Unlike the GA, PSO has no evolution operators such as crossover and mutation. The rows in the matrix are called particles (similar to the GA chromosome). They contain the variable values and are not binary encoded. Each particle moves about the cost surface with a velocity. The particles update their velocities and positions based on the local and global best solutions:

The equation to update the velocity is:

$$V_{ij}(t+1) = V_{ij}(t) + c_1 r_{1j}(t) [y_{ij}(t) - x_{ij}(t)] + c_2 r_{2j}(t) [\hat{y}_{ij}(t) - x_{ij}(t)] \quad (1)$$

The equation to update the position is:

$$x_i(t+1) = x_i(t) + v_i(t+1) \quad (2)$$

where

x_i	particle i position
v_i	particle i velocity
y_i	best position of particle i
\hat{y}	best global position
c_1	cognitive coefficient
c_2	social coefficient
r_{1j} and r_{2j}	social coefficient

The PSO algorithm updates the velocity vector for each particle then adds that velocity to the particle position or values. Velocity updates are influenced by both the best global solution associated with the lowest cost ever found by a particle and the best local solution associated with the lowest cost in the present population. If the best local solution has a cost less than the cost of the current global solution, then the best local solution replaces the best global solution. The particle velocity is reminiscent of local minimizers that use derivative information, because velocity is the derivative of position. The constant c_1 is called the cognitive parameter. The constant c_2 is called the social parameter. The advantages of PSO are that it is easy to implement and there are few parameters to adjust.

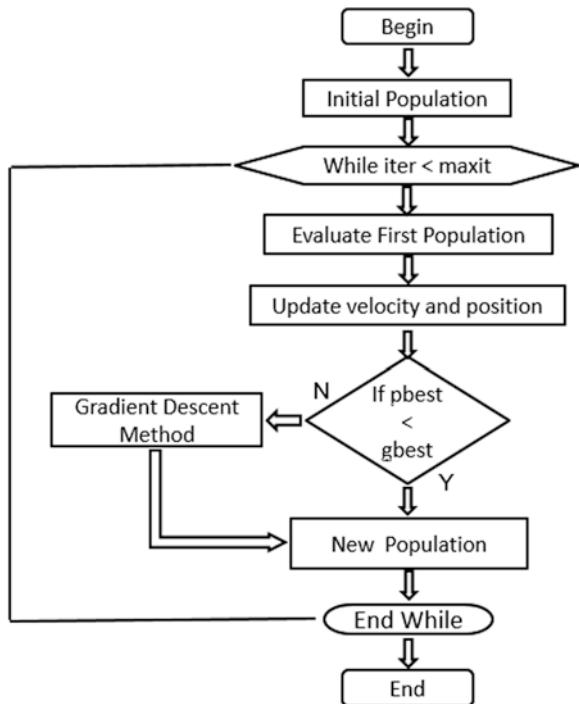
5 Improved Particle Swarm Optimization

The process is as follows: all particles first are improved by PSO in the group of every generation algorithm, according to the type (1) and (2) update each particle speed and position, and calculate for each particle the fitness function of value.

According to the particle fitness value choose one or several of the adaptable particles. These particles called elite individuals. The elite individual is not directly into the next generation of algorithm in the group, but rather through BP operator to improve performance. That is the BP operator in the area around to develop individual elite more excellent performance of particle position, and they lead the species particles rapid evolution in the next generation.

From the overall perspective, the improved particle swarm optimization algorithm is in the PSO algorithm of the next generation groups merge with BP Algorithm (Gradient Descent Method). Figure 2 shows a flow diagram of the Improved Particle Swarm Optimization.

Fig. 2 Improved particle swarm optimization diagram



IPSO Pseudocode

```

-Begin
  -Start population
  -While (no stop condition is met) do
    -Evaluate first population
    -Update velocity and position of particle
    -Evaluate new population
    -If the best local particle is better than best global
      particle
      - Move to the new population
    -Else
      -Evaluate with Gradient Descent Method
  -End while
-End
  
```

6 Experimental Results

In this case 6-dimensional particles are generated, the first dimension represents the layers and the other 5 dimensions are the neurons per layer. Figure 3 shows the architecture of particles of the PSO algorithm with 3 layers. Figure 4 shows the

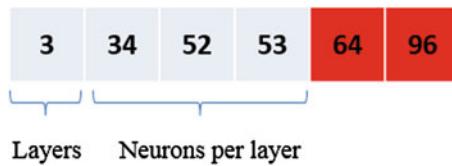


Fig. 3 Architecture of a particle of the PSO algorithm with 3 layers

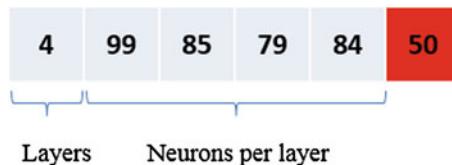


Fig. 4 Architecture of a particle of the PSO algorithm with 4 layers

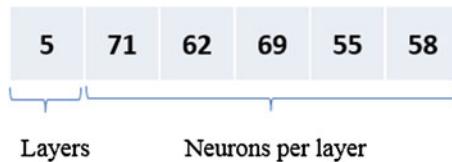


Fig. 5 Architecture of a particle of the PSO algorithm with 5 layers

architecture of particles of the PSO algorithm with 4 layers. Figure 5 shows the architecture of particles of the PSO algorithm with 5 layers.

The IPSO algorithm was executed with an initial population of 30 individuals, with which the neural network is evaluated.

A modular neural network with the database is trained and recognition tests are made.

Table 1 Parameters of training with IPSO

Parameter	Value
Training method	Trainscg
Epochs	300
Goal error	0.02
Transfer function hidden layer	Sigmoidal tangent
Transfer function output layer	Sigmoidal tangent

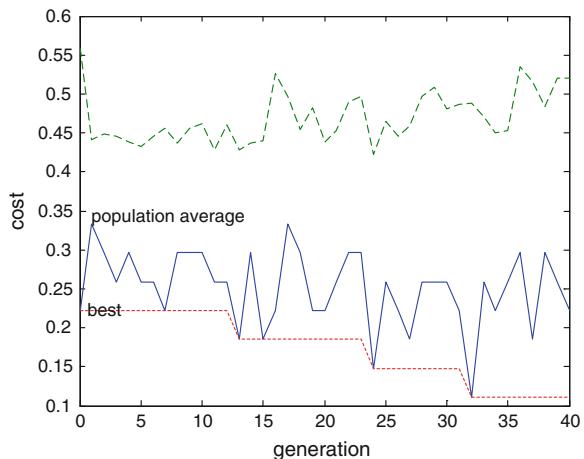
Table 2 Experimental results with IPSO

# Training	Goal error	Training method	Epochs	Error	% Recognition
1	0.02	Trainscg	300	0.28041944	71.96
2	0.02	Trainscg	300	0.23638695	76.36
3	0.02	Trainscg	300	0.28692699	71.31
4	0.02	Trainscg	300	0.2501262	74.99
5	0.02	Trainscg	300	0.20442754	79.56
6	0.02	Trainscg	300	0.27878932	72.12
7	0.02	Trainscg	300	0.26231238	73.77
8	0.02	Trainscg	300	0.16208492	83.79
9	0.02	Trainscg	300	0.16963158	83.04
10	0.02	Trainscg	300	0.22724654	77.28
11	0.02	Trainscg	300	0.24959946	75.04
12	0.02	Trainscg	300	0.29901556	70.10
13	0.02	Trainscg	300	0.2380238	76.20
14	0.02	Trainscg	300	0.2654024	73.46
15	0.02	Trainscg	300	0.14333041	87.67
16	0.02	Trainscg	300	0.17492481	82.51
17	0.02	Trainscg	300	0.21244602	78.76
18	0.02	Trainscg	300	0.16749188	83.25
19	0.02	Trainscg	300	0.22795085	77.20
20	0.02	Trainscg	300	0.20725308	79.27
21	0.02	Trainscg	300	0.29869609	70.13
22	0.02	Trainscg	300	0.15157061	84.84
23	0.02	Trainscg	300	0.20517433	79.48
24	0.02	Trainscg	300	0.24783431	75.22
25	0.02	Trainscg	300	0.14848664	85.15
26	0.02	Trainscg	300	0.21558876	78.44
27	0.02	Trainscg	300	0.28585093	71.41
28	0.02	Trainscg	300	0.14974138	85.03
29	0.02	Trainscg	300	0.16655319	83.34
30	0.02	Trainscg	300	0.14920687	85.08

Table 1 shows the parameters of training with IPSO. Table 2 shows the experimental results with IPSO.

Figure 6 shows the convergence graph of the IPSO.

Fig. 6 Convergence graph of the IPSO



7 Conclusion

In order to reduce the search process for the particle swarm algorithm and the existing early-convergence problem, a kind of “variation” into the idea of the particle swarm algorithm is proposed, that is algorithm not constraints particle boundary make them within the search range, but have the same number of random particles replace those particles of flying away from the search area.

The gradient descent method (BP algorithm) as a particle swarm operator is embedded in particle swarm algorithm and helps to solve problems of local minimum and premature convergence with faster convergence velocity approaching the global optimal solution for the best design of modular neural network.

References

1. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of IEEE International Conference on Neural Networks, IV, pp. 1942–1948. IEEE Service Center, Piscataway, NJ (1995)
2. Palupi, R.D., Siti, M.S.: Particle swarm optimization: technique, system and challenges. Int. J. Appl. Inf. Syst. **1**, 19–27 (2011)
3. Bai, Q.H.: Analysis of particle swarm optimization algorithm. Comput. Inf. Sci. **3**, 180–184 (2010)

Left Ventricular Border Recognition in Echocardiographic Images Using Modular Neural Networks and Sugeno Integral Measures

Fausto Rodríguez-Ruelas, Patricia Melin
and German Prado-Arechiga

Abstract The Echocardiography and the 2D ultrasound images are widely used to assess patients with heart diseases. The Observer (cardiologist) qualitatively deduces the heart morphology and left and right ventricular functions. In this paper we use the modular neural networks and Sugeno Measures to find patterns in echocardiogram images to recognize left ventricular borders of the heart and derive quantitative parameters. We studied 39 echocardiographic images that are used as an input to modular neural networks to find patterns and recognize the left ventricular border and also to a monolithic neural network to compare the results. We used the percentage of error recognition to evaluate the two neural networks, where modular neural networks offered better results with a 98 % of recognition versus 80 % recognition of monolithic Neural Network. Modular neural networks proved that they are an effective technique to recognize the left ventricular border of the heart.

1 Introduction

There have been many works with modular neural networks for pattern recognition, solving problems such as identification of face, fingerprints, voice, among many others [1, 4, 6, 8].

In this paper we use the modular neural network to find patterns in echocardiogram images to recognize the left ventricular border of the heart.

The 2D echocardiography is widely used technique to evaluate patients with heart diseases in medical hospitals [3, 7]. Ultrasound images allow a visual structure observation and motion observation of the heart. Observer qualitatively deduces the cardiac morphologic state and quantitatively measures the ventricular functions.

F. Rodríguez-Ruelas · P. Melin (✉)
Tijuana Institute of Technology, Tijuana, BC, Mexico
e-mail: epmelin@hafsamx.org

G. Prado-Arechiga
Cardio-Diagnóstico, Tijuana, BC, Mexico

Since cardiologists based on their experience provide a diagnosis of the heart using ultrasound imaging, it is recognized that new Physicians do not give an accurate diagnosis [7]. Quantification of LV parameters is important to provide accurate diagnosis [2, 3]. To obtain these parameters is necessary to recognize and draw the borders of the ventricle as show in Fig. 1. In Fig. 1a we show a typical gray scale 2D Left Ventricular (LV) ultrasound image and in Fig. 1b a 2D gray scale LV ultrasound image with traced border.

Left ventricle border recognition in echocardiographic images is limited by noise, gain-dependence and endocardial dropout [5, 7]. In the literature, the traditional methods to find edges such as Sobel, Prewitt, Canny, Roberts, do not provide favorable results in this particular problem. Instrumentation is recently available and can automatically identify and track the endocardial border of the left ventricle. The automatically tracker borders are then subject to calculation of volume using some methodology and the ejection fraction can be calculated from the maximal and minimal volumes [10, 11]. While manual manipulation of the contour is commonly needed to insure an accurate left ventricular cavity boundary, as the border detection algorithms. Neural networks on the other hand have the advantages of being able to work, learn and produce results in noisy environments [1, 4, 6, 8, 9]. We use this well documented soft computing technique to solve the issue of left ventricle border recognition in ultrasound images that could help us in the future to evaluate the mobility of each segment in the LV more precisely.

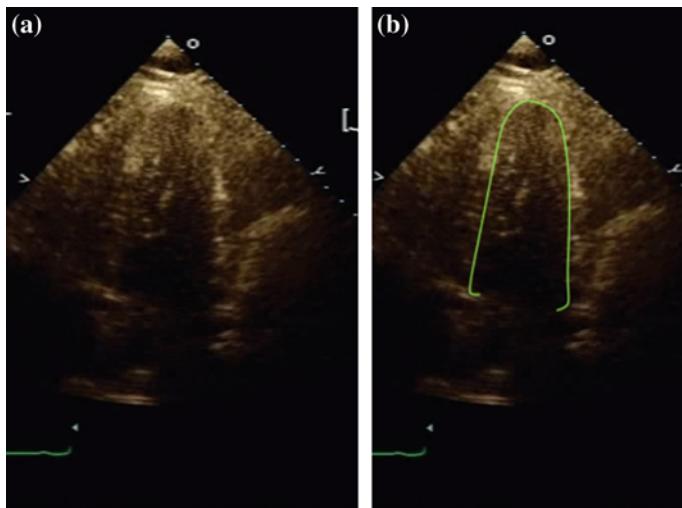


Fig. 1 **a** A typical gray scale 2D LV ultrasound image. **b** A 2D gray scale LV ultrasound image with traced border

1.1 Modular Neural Network

The main principle is simple, divide a task into less complex subtasks. Then, every subtask is assigned to an expert (Module), which generates a result. If a task can be separated into various subtasks, each subtask can be trained apart and then be integrated into an overall architecture for a solution [6, 8, 9].

1.2 Advantages of Modular Neural Network

- Learning abilities are better than monolithic networks abilities.
- Modularity may involve reducing the size of the parameters of neural networks that improves computation and generalization capabilities process.
- Helps to determine the activity that is taking place in every part of the system, identifying the function that performs each neural network in the whole system.
- If there are changes in the environment, editing is easier in modular system than a monolithic system, because modularity allows changes in a part of the system instead of change the whole entire system [8].

2 Methods

We worked with thirty-nine ultrasound images provided by Cardio-Diagnostic Center of Tijuana, Mexico. In these images we applied a pre-processing that consisted on segmenting the image to extract the region of interest, obtaining an image size of 51 pixels high by 46 pixels wide.

We worked with two neural networks, a modular and a monolithic to compare the results. We provided to the monolithic network, the complete image of 51×46 pixels as input and we provided to modular neural network the 51×46 image divided in three parts, where every part was given to an each expert module. Both networks used a number from 1 to 39 as a desired output. This number represents an indexed binary image with border traced. So the neural networks output is a number that represents an indexed image in a database. This database contains a set of traced border images. All networks are trained with two random samples of noise. Every pixel of the noisy samples is calculated according to Eq. (1). Where the new pixel (NP_i) is the sum of the intensity of P_i plus a random variable taken from a normal distribution, with a media equal to 0 and standard deviation of 0.1 and 0.2 that represents a level of noise for every sample.

$$NP_i = P_i + u \quad (1)$$

where:

NP_i is the new valor of pixel i

P_i is the value of pixel i

u is random probability sample from a normal distribution with media equal to 0 and standard equal to 0.2.

2.1 Modular Neural Network Architecture

Three modules (experts) for the modular neural network where taken in place, where each expert is responsible for recognizing a part of the image. Each module contains three monolithic networks. The result of each monolithic network is send to an integrator that used Sugeno Measures to provide a final result [6, 8]. Figure 2 illustrates the Modular Neural Network architecture used for recognition of LV borders.

The decision is made according to the Eq. (2).

$$g(M_i) = h(A) + h(B) + \lambda h(A)h(B) \quad (2)$$

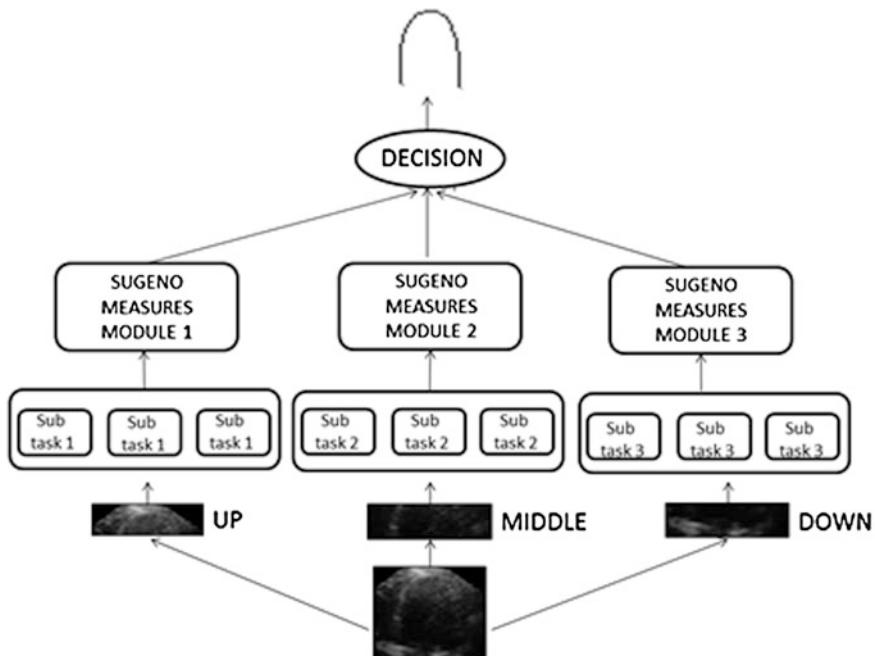


Fig. 2 Modular neural network architecture

where:

g matrix that stores the results of each module

M_i module i

$h(A)$ best result of the module i

$h(B)$ second best result of the module i

λ is equal to 1.

3 Simulation Results

We describe in this section the simulation results that were obtained with the proposed approach.

3.1 Results and Experiments of the Monolithic Network

We performed 50 experiments for the monolithic network. Each experiment consisted of training the neural network and later performed a recognition rate test with 39 images. The monolithic network scored an average of 80 % of recognition.

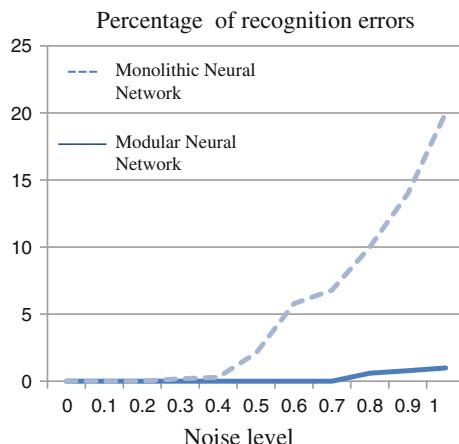
3.2 Experiments and Results of Modular Neural Network

We performed 50 experiments for the modular neural network. Each experiment consisted on training the network with the 39 images. Each module was trained separately, and then we used the Sugeno Measures to integrate the results and make a final decision. After, a test is performed with the 39 images to obtain a percentage of recognition result. The 50 experiments produce an average of 98.1 % recognition.

3.3 Neural Networks Experiments with Noise

We performed 50 experiments with random noise levels of 0.1–1, with an increment of 0.1 for both neural networks and percentage recognition of error test was performed. Results are shown in Fig. 3.

Fig. 3 Test of noise level for neural networks



4 Conclusions

This study demonstrated the effectiveness of modular neural networks to find left ventricular border patterns. We used only segmentation as pre-processing, with routine adjustment settings in the input image from the capture.

Despite the difficulties like low contrast, speckle noise, and signal dropouts the modular neural network scores well on gray scale routine images. The modular neural network had a recognition rate of 98.1 %. Despite this success, the percentage declined with increasing levels of noise.

References

1. Alvarado, M., Melin, P., Lopez, M., Mancilla, A., Castillo, O.: A hybrid approach with the wavelet transform, modular neural networks and fuzzy integrals for face and fingerprint recognition. *Curr. Develop. Theory Appl. Wavelets* **1**, 235–250 (2007)
2. Cannesson, M., Tanabe, M., Suffoletto, M., McNamara, D., Madan, S.: Real-time 3D echocardiographic quantification of left atrial volume. *JACC Cardiovasc. Imag.* **5**, 769–777 (2012)
3. Hammoude, A.: Endocardial border identification in two-dimensional echocardiographic images: review of methods. *Comput. Med. Imag. Graph.* **2**, 181–193 (1998)
4. Hidalgo, D., Castillo, O., Melin, P.: Optimization with genetic algorithms of modular neural networks using interval type-2 fuzzy logic for response integration: The case of multimodal biometry. In: IEEE International Joint Conference on Neural Networks, pp. 738–745 (2008)
5. Kirckpatrick, J., Lang, R., Savitri, E., James, B., Fedson, S., Anderson, A., Bednarz, J., Spencer, K.: Automated border detection on contrast enhanced echocardiographic images. *Int. J. Cardiol.* **18**(103), 164–167 (2005)
6. Martinez, G., Melin, P., Castillo, O.: Optimization of modular neural networks using hierarchical genetic algorithms applied to speech recognition. In: IEEE International Joint Conference on Neural Networks, vol. 3, pp. 1400–1405 (2005)

7. Maxime, C., Masaki, T., Matthew, S., Dennis, M.: A novel two-dimensional echocardiographic image analysis system using artificial intelligence-learned pattern recognition for rapid automated ejection fraction, vol. 49, pp. 217–226 (2007)
8. Melin, P., Castillo, O.: Hybrid Intelligent Systems for Pattern Recognition Using Soft Computing: An Evolutionary Approach for Neural Networks and Fuzzy Systems. Springer, Berlin (2005)
9. Melin, P., Gonzalez, C., Castillo, O.: Face recognition using modular neural networks and fuzzy Sugeno integral for response integration. In: IEEE International Joint Conference on Neural Networks. vol. 1, pp. 349–354 (2005)
10. Rasalingam, R., Makan, M., Perez, J.: The Washington Manual of Echocardiography, 2nd edn. Lippincott Williams and Wilkins, Philadelphia (2013)
11. Thavendiranathan, P., Liu, S., Verhaert, D., Calleja, A., Nitinunnu, A., Van, T., Michelis, N., Simonetti, O., Rajagopal, S., Ryan, T., Vannan, M.: Feasibility, accuracy, and reproducibility of real-time full-volume 3D transthoracic echocardiography to measure LV volumes and systolic function. *JACC Cardiovasc. Imag.* **5**, 239–251 (2012)

Optimization of Ensemble Neural Networks with Fuzzy Integration Using the Particle Swarm Algorithm for Time Series Prediction

Martha Pulido and Patricia Melin

Abstract This paper describes an optimization method based on the particle swarm algorithm for designing ensemble neural networks with fuzzy response aggregation to forecast complex time series. The time series that was considered in this paper, to compare the hybrid approach with traditional methods, is the Mackey Glass benchmark time series. Simulation results are presented for the optimization of the structure of the ensemble neural network with type-1 and type-2 fuzzy response integration and its optimization with genetic algorithms. The Simulation results show that the ensemble approach produces good prediction of the Mackey Glass time series.

1 Introduction

Time Series are defined as a set of measurements of some phenomenon or experiment recorded sequentially in time. The first step in analyzing a time series is to plot it, and this allows: to identify the trends, seasonal components and irregular variations. A classic model for a time series can be expressed as a sum or product of three components: trend, seasonality and the random error term [18].

Time series predictions are very important because based on them we can analyze past events to know the possible behavior of future events and thus we can take preventive or corrective decisions to help avoid unwanted circumstances.

The contribution of this paper is the proposed approach hybrid for ensemble neural network optimization using particle swarm optimization. In the literature there has been recent work of time series with different approaches [2, 3, 4, 9, 10, 12, 15, 16, 20, 24, 26, 27].

M. Pulido · P. Melin (✉)
Tijuana Institute of Technology, Tijuana, Mexico
e-mail: epmelin@hafsamx.org

M. Pulido
e-mail: marthapulido_84@hotmail.com

The rest of the paper is organized as follows: Sect. 2 describes the concepts of optimization, Sect. 3 describes the concepts of particle swarm optimization, Sect. 4 describes the concepts of Fuzzy Systems as Methods of integration, Sect. 5 describes the problem and the proposed method of solution, Sect. 6 describes the simulation results of the proposed method, and Sect. 7 shows the conclusions.

2 Architecture of the Neural Network Ensemble

Artificial neural networks are inspired by the biological nervous system architecture, which consists of a large number of relatively simple neurons that work in parallel to facilitate rapid decision making [17].

A neural network is a system of parallel processors interconnected as a directed graph. Schematically each processing element (neurons) in the network is represented as a node. Its most important advantage is to solve problems that are too complex for conventional technologies, problems that have no solution algorithm and its solution algorithm is very difficult finding. Artificial neural networks are formed by a large number of neurons, and these are not called artificial neurons but output nodes or units. Since these ensemble models behave remarkably well, recently it has become a very hot topic in both the neural networks and machine learning communities [23], and has already been successfully applied to diverse areas such as face recognition [8, 19], optical character recognition [5, 16, 25], scientific image analysis [22], medical diagnosis [5, 23], seismic signals classification [21], etc.

3 Particle Swarm Optimization

The Particle Swarm Optimization algorithm maintains a swarm of particles, where each particle represents a potential solution. In analogy with evolutionary computation paradigms, a swarm is a population, while a particle is similar to an individual. In simple terms, the particles are “flown” through a multidimensional search space where the position of each particle is adjusted according to its own experience and that of their neighbors. Let $x_i(t)$ denote the position of particle i in the search space at time step t unless otherwise selected, t denotes discrete time steps. The position of the particle is changed by adding a velocity, $v_i(t)$ to the current position i.e.

$$x_i(t + 1) = x_i(t) + v_i(t + 1) \text{ m} \quad (1)$$

with $x_i(0) \sim U(X_{min}, X_{max})$.

It is the velocity vector the one that drives of the optimization process, and reflects both the experimental knowledge of the particles and the information exchanged in the vicinity of particles. The experimental knowledge of a particle

which is generally known as the cognitive component, which is proportional to the distance of the particle from its own best position (hereinafter, the personal best position particles) that are from the first step. Socially exchanged information is known as the social component of the velocity equation [6, 7, 11].

For the gbest PSO, the particle velocity is calculated as:

$$v_{ij}(t+1) = v_{ij}(t) + c_1 r_1 [y_{ij}(t) - x_{ij}(t)] + c_2 r_2(t) [\hat{y}_j(t) - x_{ij}(t)] \quad (2)$$

where $v_{ij}(t)$ is the velocity of the particle i in dimension j at time step t , c_1 and c_2 are positive acceleration constants used to scale the contribution of cognitive and social skills, respectively, $y r_{1j}(t), y r_{2j}(t) \sim U(0, 1)$ are random values in the range [0,1].

The best personal position in the next time step $t + 1$ is calculated as:

$$y_i(t+1) = \begin{cases} y_i(t) & \text{if } f(x_i(x_i(t+1))) \geq f(y_i(t)) \\ x_i(t+1) & \text{if } f(x_i(x_i(t+1))) > f(y_i(t)) \end{cases} \quad (3)$$

where $f: \mathbb{R}^{nx} \rightarrow \mathbb{R}$ is the fitness function, as with EAs, measuring fitness with the function will help find the optimal solution, for example the objective function quantifies the performance, or the quality of a particle (or solution).

The overall best position, $\hat{y}(t)$ at time step t , is defined as:

$$\hat{y}(t) \in \{y_o(t), \dots, y_{ns}(t)\} f(y(t)) = \min\{f(y_o(t)), \dots, f(y_{ns}(t))\} \quad (4)$$

where n_s is the total number of particles in the swarm. Importantly, the above equation defining and establishing \hat{y} the best position is uncovered by either of the particles so far as this is usually calculated from the best personal position [5, 6, 8].

The overall best position may be selected from the actual swarm particles, in which case:

$$\hat{y}(t) = \min\{f(x_o(t)), \dots, f(x_{ns}(t))\} \quad (5)$$

4 Fuzzy Systems as Methods of Integration

Fuzzy logic was proposed for the first time in the mid-sixties at the University of California, Berkeley by Lofti A. Zadeh, who proposed what it's called the principle of incompatibility: "As the complexity of system increases, our ability to be precise instructions and build on their behavior decreases to the threshold beyond which the accuracy and meaning are mutually exclusive characteristics." Then introduced the concept of a fuzzy set, under which lies the idea that the elements on which to build human thinking are not numbers but linguistic labels. Fuzzy logic can represent the

common knowledge as a form of language that is mostly qualitative and not necessarily a quantity in a mathematical language [10].

Type-1 Fuzzy system theory was first introduced by Zadeh [14] in 1965, and has been applied in many areas such as control, data mining, time series prediction, etc.

The basic structure of a fuzzy inference system consists of three conceptual components: a rule base, which contains a selection of fuzzy rules, a database (or dictionary) which defines the membership functions used in the rules, and reasoning mechanism, which performs the inference procedure (usually fuzzy reasoning) [11].

Type-2 Fuzzy systems were proposed to overcome the limitations of a type-1 FLS, the concept of type-1 fuzzy sets was extended into type-2 fuzzy sets by Zadeh in 1975. These were designed to mathematically represent the vagueness and uncertainty of linguistic problems; thereby obtaining formal tools to work with intrinsic imprecision in different type of problems; it is considered a generalization of the classic set theory. Type-2 fuzzy sets are used for modeling uncertainty and imprecision in a better way [1].

5 Problem Statement and Proposed Method

The objective of this work was to develop a model that is based on integrating the responses of an ensemble neural network using type-1 and type-2 fuzzy systems and their optimization. Figure 1 represents the general architecture of the proposed method, where historical data, analyzing data, creation of the ensemble neural network and integrate responses of the ensemble neural network with type-2 fuzzy system integration and finally obtaining the outputs are shown. The information can be historical data, these can be images, time series, etc., in this case we show the application to time series prediction of the Dow Jones where we obtain good results with this series.

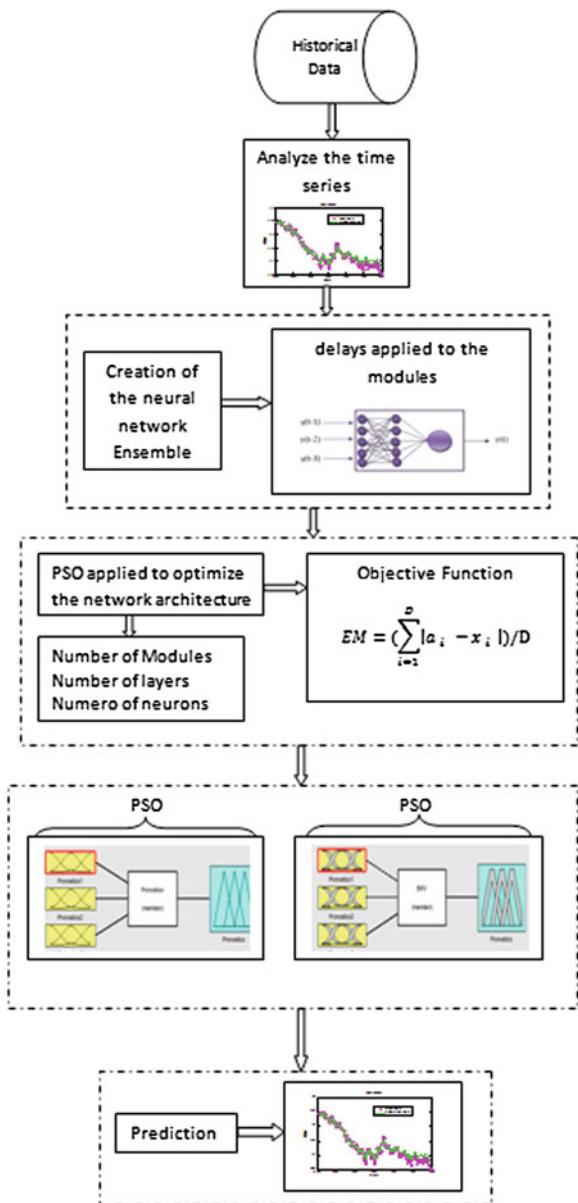
Figure 2 represents the Particle Structure to optimize the ensemble neural network, where the parameters that are optimized are the number de modules, number of layers, and number of neurons of the ensemble neural network. PSO determines the number of modules, number of layers and number of neurons per layer, that the neural network ensemble should have, that meets the objective to achieve a better Prediction error.

For the optimization of the structure of the ensemble neural network a particle swarm optimization algorithm was also used. The structure of the swarm particle represents the structure of the ensemble.

The objective function is defined to minimize the prediction error as follows:

$$EM = \left(\sum_{i=1}^D |a_i - x_i| \right) / D \quad (6)$$

Fig. 1 General architecture of the proposed method



where a_i corresponds to the predicted data depending on the output of the network modules, X represents real data, D the Number of Data points and EM is the total average prediction error.

The corresponding particle structure is shown in Fig. 2.

The parameters for the particle swarm optimization algorithm are: 100 Particles, 100 iterations, Cognitive Component (C1) = 2, Social Component (C2) = 2,

Number Of Module	Number Of Layers	Neurons 1	...	Neurons n
------------------------	------------------------	-----------	-----	--------------

Fig. 2 Particle structure to optimize the ensemble neural network

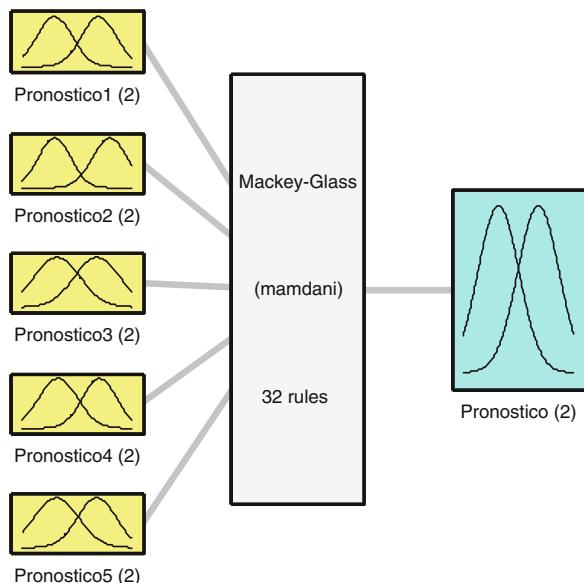
Constriction coefficient of linear increase (C) = (0–0.9) and Inertia weight with linear decrease (W) = (0.9–0). We consider a number of 1–5 modules, number of layers of 1–3, neurons number of 1–30.

Figure 3 shows a type-2 fuzzy system consisting of 5 inputs depending on the number of modules of the neural network ensemble and one output. Each input and output linguistic variable of the fuzzy system uses 2 Gaussian membership functions. The performance of the type-2 fuzzy integrators is analyzed under different levels of uncertainty to find out the best design of the membership functions and consist of 32 rules. For the type-2 fuzzy integrator using 2 membership functions, which are called low prediction and high prediction for each of the inputs and output of the fuzzy system. The membership functions are of Gaussian type, and we consider 3 sizes for the footprint uncertainty 0.3, 0.4 and 0.5 to obtain a better prediction of the time series.

In Fig. 4 we show the possible rules of a type-2 fuzzy system.

The objective function for the fuzzy system optimization defined to minimize the prediction error is given by:

Fig. 3 Type-2 fuzzy system for the Mackey Glass time series



System Mackey-Glass: 5 inputs, 1 outputs, 32 rules

1. If (Prediction1 is Pred1Low) and (Prediction2 is Pron2Low) and (Prediction3 is Pred3Low) and (Prediction4 is Pred4Low) and (Prediction5 is Pred5Low) then (Prediction is Low)
2. If (Prediction1 is Pred1High) and (Prediction2 is Pron2High) and (Prediction3 is Pred3High) and (Prediction4 is Pred4Low) and (Prediction5 is Pred5High) then (Prediction is High)
3. If (Prediction1 is Pred1Low) and (Prediction2 is Pron2Low) and (Prediction3 is Pred3High) and (Prediction4 is Pred4Low) and (Prediction5 is Pred5High) then (Prediction is Low)
4. If (Prediction1 is Pred1High) and (Prediction2 is Pron2High) and (Prediction3 is Pred3High) and (Prediction4 is Pred4Low) and (Prediction5 is Pred5Low) then (Prediction is High)
5. If (Prediction1 is Pred1Low) and (Prediction2 is Pron2Low) and (Prediction3 is Pred3Low) and (Prediction4 is Pred4High) and (Prediction5 is Pred5High) then (Prediction is Low)
6. If (Prediction1 is Pred1High) and (Prediction2 is Pron2High) and (Prediction3 is Pred3High) and (Prediction4 is Pred4Low) and (Prediction5 is Pred5Low) then (Prediction is High)
7. If (Prediction1 is Pred1Low) and (Prediction2 is Pron2Low) and (Prediction3 is Pred3High) and (Prediction4 is Pred4High) and (Prediction5 is Pred5High) then (Prediction is High)
8. If (Prediction1 is Pred1High) and (Prediction2 is Pron2High) and (Prediction3 is Pred3Low) and (Prediction4 is Pred4Low) and (Prediction5 is Pred5Low) then (Prediction is Low)
9. If (Prediction1 is Pred1Low) and (Prediction2 is Pron2High) and (Prediction3 is Pred3High) and (Prediction4 is Pred4High) and (Prediction5 is Pred5Low) then (Prediction is High)
10. If (Prediction1 is Pred1High) and (Prediction2 is Pron2Low) and (Prediction3 is Pred3Low) and (Prediction4 is Pred4Low) and (Prediction5 is Pred5Low) then (Prediction is Low)
11. If (Prediction1 is Pred1Low) and (Prediction2 is Pron2High) and (Prediction3 is Pred3Low) and (Prediction4 is Pred4High) and (Prediction5 is Pred5Low) then (Prediction is Low)
12. If (Prediction1 is Pred1High) and (Prediction2 is Pron2Low) and (Prediction3 is Pred3High) and (Prediction4 is Pred4Low) and (Prediction5 is Pred5High) then (Prediction is High)
13. If (Prediction1 is Pred1Low) and (Prediction2 is Pron2High) and (Prediction3 is Pred3Low) and (Prediction4 is Pred4Low) and (Prediction5 is Pred5Low) then (Prediction is Low)
14. If (Prediction1 is Pred1High) and (Prediction2 is Pron2Low) and (Prediction3 is Pred3High) and (Prediction4 is Pred4Low) and (Prediction5 is Pred5High) then (Prediction is High)
15. If (Prediction1 is Pred1Low) and (Prediction2 is Pron2High) and (Prediction3 is Pred3Low) and (Prediction4 is Pred4Low) and (Prediction5 is Pred5Low) then (Prediction is Low)
16. If (Prediction1 is Pred1High) and (Prediction2 is Pron2Low) and (Prediction3 is Pred3High) and (Prediction4 is Pred4High) and (Prediction5 is Pred5High) then (Prediction is High)
17. If (Prediction1 is Pred1Low) and (Prediction2 is Pron2Low) and (Prediction3 is Pred3High) and (Prediction4 is Pred4Low) and (Prediction5 is Pred5Low) then (Prediction is Low)
18. If (Prediction1 is Pred1High) and (Prediction2 is Pron2High) and (Prediction3 is Pred3Low) and (Prediction4 is Pred4Low) and (Prediction5 is Pred5High) then (Prediction is High)
19. If (Prediction1 is Pred1Low) and (Prediction2 is Pron2Low) and (Prediction3 is Pred3High) and (Prediction4 is Pred4High) and (Prediction5 is Pred5Low) then (Prediction is Low)
20. If (Prediction1 is Pred1High) and (Prediction2 is Pron2High) and (Prediction3 is Pred3Low) and (Prediction4 is Pred4Low) and (Prediction5 is Pred5High) then (Prediction is High)
21. If (Prediction1 is Pred1Low) and (Prediction2 is Pron2High) and (Prediction3 is Pred3High) and (Prediction4 is Pred4Low) and (Prediction5 is Pred5Low) then (Prediction is Low)
22. If (Prediction1 is Pred1High) and (Prediction2 is Pron2Low) and (Prediction3 is Pred3Low) and (Prediction4 is Pred4High) and (Prediction5 is Pred5Low) then (Prediction is High)
23. If (Prediction1 is Pred1Low) and (Prediction2 is Pron2High) and (Prediction3 is Pred3Low) and (Prediction4 is Pred4Low) and (Prediction5 is Pred5Low) then (Prediction is Low)
24. If (Prediction1 is Pred1High) and (Prediction2 is Pron2High) and (Prediction3 is Pred3Low) and (Prediction4 is Pred4Low) and (Prediction5 is Pred5High) then (Prediction is High)
25. If (Prediction1 is Pred1Low) and (Prediction2 is Pron2High) and (Prediction3 is Pred3High) and (Prediction4 is Pred4Low) and (Prediction5 is Pred5High) then (Prediction is High)
26. If (Prediction1 is Pred1Low) and (Prediction2 is Pron2High) and (Prediction3 is Pred3Low) and (Prediction4 is Pred4High) and (Prediction5 is Pred5Low) then (Prediction is Low)
27. If (Prediction1 is Pred1Low) and (Prediction2 is Pron2High) and (Prediction3 is Pred3High) and (Prediction4 is Pred4High) and (Prediction5 is Pred5Low) then (Prediction is Low)
28. If (Prediction1 is Pred1Low) and (Prediction2 is Pron2High) and (Prediction3 is Pred3Low) and (Prediction4 is Pred4High) and (Prediction5 is Pred5Low) then (Prediction is Low)
29. If (Prediction1 is Pred1High) and (Prediction2 is Pron2High) and (Prediction3 is Pred3High) and (Prediction4 is Pred4High) and (Prediction5 is Pred5Low) then (Prediction is High)
30. If (Prediction1 is Pred1High) and (Prediction2 is Pron2Low) and (Prediction3 is Pred3Low) and (Prediction4 is Pred4Low) and (Prediction5 is Pred5High) then (Prediction is Low)
31. If (Prediction1 is Pred1Low) and (Prediction2 is Pron2High) and (Prediction3 is Pred3Low) and (Prediction4 is Pred4High) and (Prediction5 is Pred5High) then (Prediction is High)
302 If (Prediction1 is Pred1High) and (Prediction2 is Pron2Low) and (Prediction3 is Pred3High) and (Prediction4 is Pred4Low) and (Prediction5 is Pred5Low) then (Prediction is Low)

Fig. 4 Rules of the type-2 fuzzy inference system for the Mackey Glass time series

$$f_1 = \left(\sum_{i=1}^D |\mathbf{a}_i - \mathbf{X}_i| \right) / D \quad (7)$$

$$f = ((f_1 + f_2 + \dots + f_n) / n) + ((R/32)/100))$$

where \mathbf{a} , corresponds to the predicted data depending on the outputs of the network modules, X represents real data, D the Number of Data, f is the total prediction error, and R the number rules generated by the genetic algorithm. In this case we are considering that the maximum number of rules is 32 and this why is this value is used in (10) as weight to limit the value f .

The parameters for particle swarm optimization, C_1 and C_2 have a value of 2, the maximum speed is 1. Figure 5 represents the Particle Structure to optimize the Type-1 Fuzzy System, and is composed of 180 parameters which allow us to

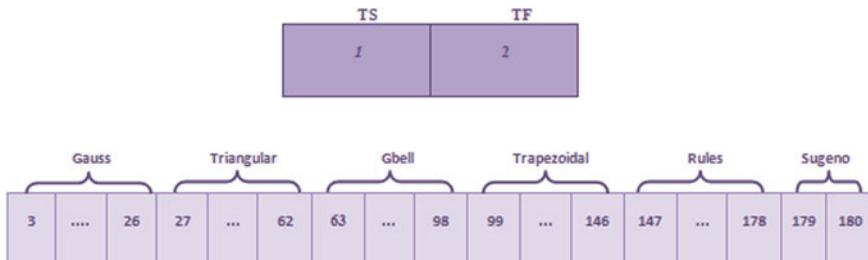


Fig. 5 Particle structure to optimize the type-1 fuzzy integration

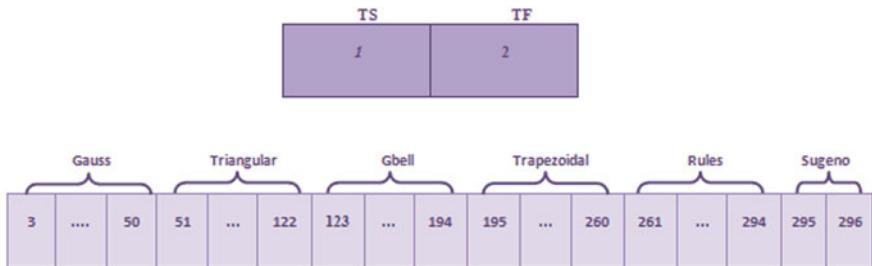


Fig. 6 Particle structure to optimize the type-2 fuzzy integration

optimize the structure of the fuzzy integrator. The parameter 1 determines the type of system (Mamdani or Sugeno), parameter 2 and determines the type of membership functions (Gaussian, generalized bell, triangular and trapezoidal). Parameters 3–26 (real numbers) allow us to manage the parameters of the membership functions for the inputs and output (depending on the type of system and type of functions we use only part of the chromosome). Parameters 147–178 allow us to reduce the number of fuzzy rules, activating or deactivating them.

Figure 6 represents the Particle Structure to optimize the Type-2 Fuzzy System and it is composed of 296 parameters, which allow us to optimize the structure of the fuzzy integrator. The parameter 1 determines the type of system (Mamdani or Sugeno), parameter 2 and determines the type of membership functions (Gaussian, generalized bell, triangular and trapezoidal). Parameters 3–260 (real numbers) allow us to manage the parameters of the membership functions for the inputs and output (depending on the type of system and type of functions we use only part of the chromosome). Parameters 261–294 allow us to reduce the number of fuzzy rules, activating or deactivating them.

Data of the Mackey-Glass time series was generated using Eq. (6). We are using 800 points of the time series. We use 70 % of the data for the ensemble neural network trainings and 30 % to test the network.

The Mackey-Glass Equation is defined as follows:

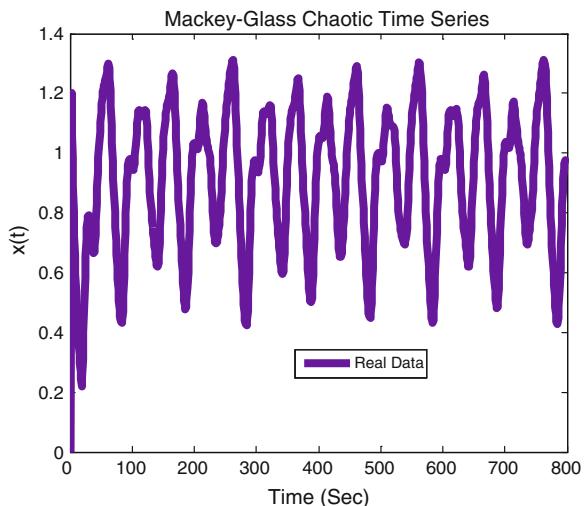
$$\dot{x}(t) = \frac{0.2x(t - \tau)}{1 + x^{10}(t - \tau)} - 0.1x(t) \quad (8)$$

where it is assumed $x(0) = 1.2$, $\tau = 17$, $\tau = 34$, and $68x(t) = 0$ for $t < 0$.

Figure 7 shows a plot of the time series for these parameter values.

This time series is chaotic, and there is no clearly defined period. The series does not converge or diverge, and the trajectory is extremely sensitive to the initial conditions. The time series is measured in number of points, and we apply the fourth order Runge-Kutta method to find the numerical solution of the equation [13, 14].

Fig. 7 Mackey Glass time series



6 Simulation Results

In this section we present the simulation results obtained with the integration of the ensemble neural network with type-2 fuzzy integration and its optimization with the particle swarm optimization for the Mackey-Glass time series.

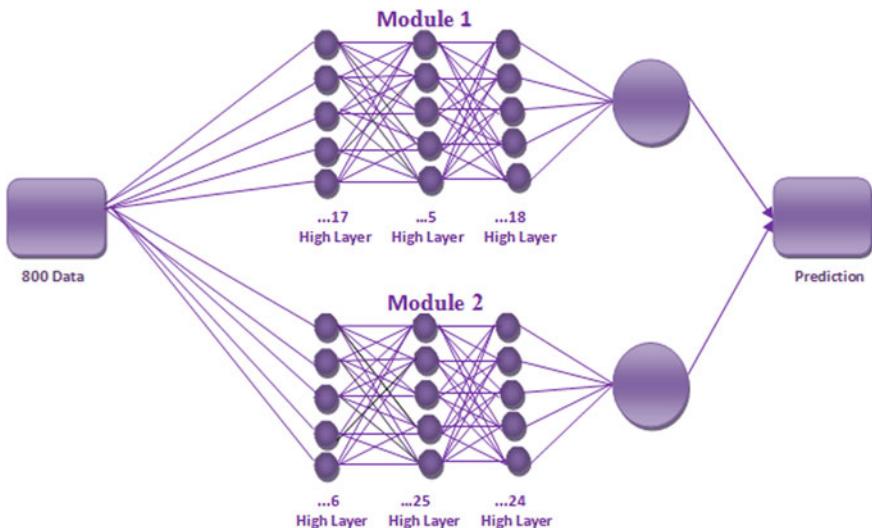


Fig. 8 Ensemble neural network architecture PSO of MG for $\tau = 17$

Table 1 Particle swarm results for the ensemble neural network for Mackey-Glass with $\tau = 17$

No.	Iterations	Particles	Number of modules	Number of modules	Number of neurons	Duration	Prediction error
1	100	100	4	2	20, 14	02:23:18	0.0076048
					13, 16		
					17, 8		
					6, 26		
2	100	100	2	2	12, 16	01:45:45	0.0063313
					12, 26		
3	100	100	2	3	17, 5, 18	01:28:42	0.0018838
					6, 25, 24		
4	100	100	2	2	14, 10	01:23:28	0.0061291
					14, 21		
5	100	100	3	2	9, 5	02:17:06	0.0053679
					23, 20		
					22, 13		
6	100	100	3	3	23, 14, 16	02:20:04	0.0061983
					19, 10, 23		
					22, 12, 11		
7	100	100	2	3	11, 19, 26	01:33:18	0.0052105
					8, 16, 20		
8	100	100	2	2	2, 24	01:26:21	0.0065942
					24, 14		
9	100	100	2	2	14, 13	01:40:25	0.0030335
					5, 24		
10	100	100	4	2	25, 15	01:35:13	0.0069241
					23, 20		
					25, 4		
					11, 21		

When using a particle swarm algorithm to optimize the structure of the ensemble neural network and considering 5 modules at the most, then the best achieved architecture for the time series is shown in Fig. 8 and Table 1 in row number 5.

In this architecture we used three layers in each module. In module 1, in the first layer we used 17 neurons, 5 neurons in second layer and 18 third layer. In module 2 we used 17 neurons, 5 neurons in second layer and 18 third layer. In module 2, 6 neurons in the first layer, 25 neurons in the second and 24 third layer, the Levenberg-Marquardt (LM) training method was used; we also applied 3 delays to the network.

Table 1 shows the particle swarm optimization where the best prediction error is of 0.0018838.

Table 2 Results of type-1 fuzzy integration for $\tau = 17$

Experiments	Prediction error with fuzzy integration type-1
Experiment 1	0.8695
Experiment 2	1.0538
Experiment 3	0.9538
Experiment 4	0.9224
Experiment 5	1.2676
Experiment 6	1.4773
Experiment 7	0.2454
Experiment 8	2.1092
Experiment 9	2.1103
Experiment 10	0.5250

Fuzzy integration is performed initially by implementing a type-1 fuzzy system in which the best result was in the experiment of row number 5 of Table 2 with an error of: 0.2454.

Fuzzy integration is performed by implementing a type-1 fuzzy system in which the results were as follows: for the best evolution with a degree of uncertainty of 0.3 a forecast error of 0.1158 was obtained, and with a degree of uncertainty of 0.4 a forecast error of 0.1386 and with a degree of uncertainty of 0.5 a forecast error of 0.1496 was obtained, as shown in Table 3.

Table 4 shows the particle swarm results of the optimization of type-1 fuzzy system, where 30 iterations were performed for each of the best iterations of the neural network. Where the best prediction error is of 0.00106, which is shown in row number four.

Table 3 Results of type-2 fuzzy integration for $\tau = 17$

Experiment	Prediction error 0.3 uncertainty	Prediction error 0.4 uncertainty	Prediction error 0.5 uncertainty
Experiment 1	0.3393	0.2175	0.2240
Experiment 2	0.1780	0.1980	0.2559
Experiment 3	0.1632	0.1829	0.2145
Experiment 4	0.2042	0.2042	0.1998
Experiment 5	0.6667	0.6655	0.6687
Experiment 6	0.6666	0.6650	0.6683
Experiment 7	0.2049	0.2102	0.1985
Experiment 8	0.3296	0.3085	0.3163
Experiment 9	0.1158	0.1386	0.1497
Experiment 10	0.5228	0.5233	0.5226

Table 4 Particle swarm results for the type-1 fuzzy system for the MG with $\tau = 17$ series

No.	Iterations	Particles	TS	TF	Number of rules	Duration	Error PSO	Error prediction
1	100	100	M	Gau	3	00:41:40	0.009375	0.008375
2	100	100	M	Gbell	4	00:40:35	0.00625	0.00525
3	100	100	M	Gbell	4	00:48:22	0.00125	0.00115
4	100	100	S	Gau	3	00:39:16	0.00116	0.00106
5	100	100	M	Gbell	8	00:40:20	0.0021875	0.0011875
6	100	100	M	Gbell	6	00:41:19	0.0025	0.0015
7	100	100	M	Gau	4	00:52:04	0.002651	0.001651
8	100	100	M	Gau	4	00:52:04	0.002584	0.001584
9	100	100	M	Gbell	4	00:42:17	0.00136	0.00126
10	100	100	S	Gau	8	00:53:24	0.002815	0.001815

Table 5 Particle swarm results for the type-2 fuzzy system for the MG with $\tau = 17$ series

No.	Iterations	Particles	TS	TF	Number of rules	Duration	Error PSO	Error prediction
1	100	100	M	Gau	3	01:43:46	0.008817	0.007817
2	100	100	M	Gbell	4	01:46:05	0.007982	0.006982
3	100	100	M	Gbell	4	01:48:22	0.005287	0.004287
4	100	100	S	Gau	3	01:39:16	0.006113	0.005113
5	100	100	M	Gbell	8	01:45:28	0.006872	0.005872
6	100	100	S	Gbell	7	01:47:52	0.00352	0.00252
7	100	100	M	Gau	4	01:49:28	0.0038	0.002981
8	100	100	M	Gbell	3	01:56:04	0.004792	0.003792
9	100	100	M	Gau	4	01:47:17	0.002540	0.001540
10	100	100	S	Gau	9	01:57:29	0.0027835	0.0017835

Table 5 shows the particle swarm results of the optimization of type-2 fuzzy system, where 30 iterations were performed for each of the best iterations of the neural network. Where the best prediction error is of 0.001540, which is shown in row number nine.

7 Conclusions

In this paper we considered the PSO algorithm to optimize the architecture of ensemble neural network and optimization for the structure of type-1 and type-2 fuzzy system to predict the time series of the Mackey-Glass, where good results were obtained with Particle Swarm Optimization PSO is an effective and efficient

metaheuristic to find the solution of problems. In conclusion applying this technique neural network architecture helps reduce the ensemble neural networks.

Acknowledgments We would like to express our gratitude to the CONACYT, Tijuana Institute of Technology for the facilities and resources granted for the development of this research.

References

1. Castillo, O., Melin, P.: Type-2 Fuzzy Logic: Theory and Applications Neural Networks, pp. 30–43. Springer, Berlin (2008)
2. Castillo, O., Melin, P.: Hybrid intelligent systems for time series prediction using neural networks, fuzzy logic, and fractal theory. *IEEE Trans. Neural Netw.* **13**(6), 1395–1408 (2002)
3. Castillo, O., Melin, P.: Simulation and forecasting complex economic time series using neural networks and fuzzy logic. In: Proceedings of the International Neural Networks Conference, vol. 3, pp. 1805–1810 (2001)
4. Castillo, O., Melin, P.: Simulation and forecasting complex financial time series using neural networks and fuzzy logic. In: Proceedings the IEEE the International Conference on Systems, Man and Cybernetics vol. 4, pp. 2664–2669 (2001)
5. Drucker, H., Schapire, R., Simard, P.: Improving performance in neural networks using a boosting algorithm. In: Hanson, S.J., Cowan, J.D., Giles, C.L. (eds.) *Advances in Neural Information Processing Systems*, vol. 5, pp. 42–49. Morgan Kaufmann, San Mateo (1993)
6. Eberhart, R.C., Kennedy, J.: A new optimizer particle swarm theory. In: Proceedings of the Sixth Symposium on Micromachine an Human Science, pp. 39–43 (1995)
7. Eberhart, R.C.: *Fundamentals of Computational Swarm Intelligence*. Wiley, London, pp. 93–129 (2005)
8. Huarng, K.: Effective lengths of intervals to improve forecasting in fuzzy time series. *Fuzzy Sets Syst.* **123**(3), 387–394 (2001)
9. Jang, J.S.R., Sun, C.T., Mizutani, E.: *Neuro-Fuzzy and Soft Computing*. Prentice Hall, New Jersey (1996)
10. Karnik, N., Mendel, J.M.: Applications of type-2 fuzzy logic systems to forecasting of time-series. *Inf. Sci.* **120**(1–4), 89–111 (1999)
11. Kennedy, J., Eberhart, R.C.: Particle swarm optimization. In: *Proceedings Intelligent Symposium*, pp. 80–87 (2003)
12. Krogh, A., Vedelsby, J.: Neural network ensembles, cross validation, and active learning. In: Tesauro, G., Touretzky, D., Leen, T. (eds.) *Advances in Neural Information Processing Systems* vol. 7, pp. 231–238. MIT Press, Cambridge (1995)
13. Mackey, M.C.: Adventures in Poland: having fun and doing research with Andrzej Lasota. *Mat. Stosow.* pp 5–32 (2007)
14. Mackey, M.C., Glass, L.: Oscillation and chaos in physiological control systems. *Science* **197**, 287–289 (1997)
15. Maguire, L.P., Roche, B., McGinnity, T.M., McDaid, L.J.: Predicting a chaotic time series using a fuzzy neural network. *Inf. Sci.* **12**(1–4) pp. 125–136 (1998)
16. Melin, P., Castillo, O., Gonzalez, S., Cota, J., Trujillo, W., Osuna, P.: Design of Modular Neural Networks with Fuzzy Integration Applied to Time Series Prediction, vol. 41, pp. 265–273. Springer, Berlin (2007)
17. Multaba, I.M., Hussain, M.A.: *Application of Neural Networks and Other Learning Technologies in Process Engineering*. Imperial College Press, London (2001)
18. Plummer, E.A.: *Time Series Forecasting With Feed-Forward Neural Networks: Guidelines and limitations*. University of Wyoming, Wyoming (2000)

19. Pulido, M., Mancilla, A., Melin, P.: An ensemble neural network architecture with fuzzy response integration for complex time series prediction. In: Evolutionary Design of Intelligent Systems in Modeling, Simulation and Control, vol. 257, pp. 85–110. Springer, Berlin (2009)
20. Sharkey, A.J.: Combining artificial neural nets: ensemble and modular multi-net systems. Springer, London (1999)
21. Shimshoni, Y., Intrator, N.: Classification of seismic signal by integrating ensemble of neural networks. *IEEE Trans. Signal Process.* **46**(5), 1194–1201 (1998)
22. Sollich, P., Krogh, A.: Learning with ensembles: how over-fitting can be useful. In: Touretzky, D.S., Mozer, M.C., Hasselmo, M.E. (eds.) *Advances in Neural Information Processing Systems*, vol. 8, pp. 190–196. MIT Press, Cambridge (1996)
23. Xue, J., Xu, Z., Watada, J.: Building an integrated hybrid model for short-term and mid-term load forecasting with genetic optimization. *Int. J. Innov. Comput. Inf. Control* **8**(10), 7381–7391 (2012)
24. Yadav, R.N., Kalra, P.K., John, J.: Time series prediction with single multiplicative neuron model. *Soft Computing for Time Series Prediction. Appl. Soft Comput.* **7**(4), 1157–1163 (2007)
25. Yao, X., Liu, Y.: Making use of population information in evolutionary artificial neural networks. *IEEE Trans. Syst. Man Cybern. Part B Cybern.* **28**(3), 417–425 (1998)
26. Zhao, L., Yang, Y.: PSO-based single multiplicative neuron model for time series prediction. *Expert Syst. Appl.* **36**(2), 2805–2812 (2009)
27. Zhou, Z.-H., Jiang, Y., Yang, Y.-B., Chen, S.-F.: Lung cancer cell identification based on artificial neural network ensembles. *Artif. Intell. Med.* **24**(1), 25–36 (2002)

A Type-2 Fuzzy Neural Network Ensemble to Predict Chaotic Time Series

Victor M. Torres and Oscar Castillo

Abstract This paper presents a type-2 adaptive fuzzy neural network ensemble to predict chaotic time series in combination with the well known M8 algorithm. The chaotic time series is depicted by the register of seismic events and their seismic coordinates in a catalog. ANFIS model are used as components of the Ensemble to train and evaluate seven chaotic time series that are used by the M8 algorithm to make a prediction.

1 Introduction

A prediction is significant only to the extent it is successful beyond chance [1].

As a science, earthquake prediction is still not very mature [2]. Statistical methods have been proposed to determine the probability that something can happen, especially in the seismic field of investigation, but they are not well suited to deal with uncertain systems [3].

Prediction systems have been proposed based upon statistical methods and neural systems [1], most of them using a single adaptive neural system or an adaptive fuzzy neural system [4, 5]. The Ensemble paradigm introduces the use of several fuzzy neural networks functioning together in order to do a better prediction.

In this paper, a prediction system is described. The statistical method used is the well known M8 algorithm. This method has proven to be useful in determining

V.M. Torres

Carretera Ensenada-Tijuana No. 3918, Zona Playitas, 22860 Ensenada, B.C., Mexico

O. Castillo (✉)

Tijuana Institute of Technology, P.O. Box 4207 Chula Vista, USA

e-mail: ocastillo@hafsamx.org

seismic events of 6.0+ magnitudes, but not accurate enough. This statistical method is combined with an Ensemble of Fuzzy Neural Networks (ANFIS) to overcome the uncertainty of a real system.

2 Definitions

2.1 Neural Network

Neural networks are a form of computational multi-threaded model involving simple processing elements with a high degree of interconnection and adaptive interaction among them [4] (see Fig. 1).

A neural network is a system that approximates the operation of a human brain.

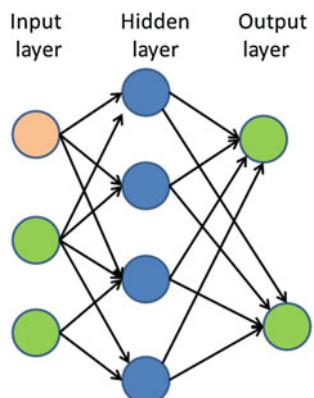
2.2 Adaptive Neural Network

An adaptive neural network is a system that processes information and makes adjustments to the network when necessary based on their own evaluation of how to most efficiently carry out their functions.

There are two main ways an adaptive neural network “learns”: supervised learning and unsupervised learning. Supervised learning requires a human counterpart who instructs the network on how to interpret and interact with various inputs in order to ensure that there are no errors in the methods that the adaptive neural network uses to process information.

Non-supervised learning relies on the central processing unit interacting with its environment and making its own decisions on how it should operate based on its original programming.

Fig. 1 Neural network diagram



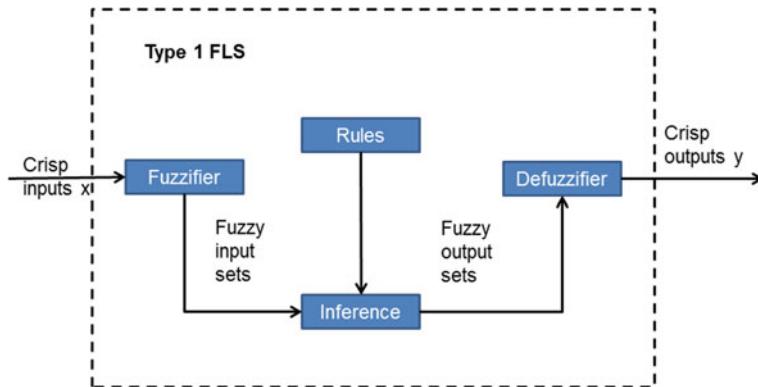


Fig. 2 Type 1 fuzzy logic system diagram

2.3 Fuzzy Logic

Fuzzy sets were first proposed by Zadeh [6] in 1965 as a way to deal with imprecise or uncertain data. Fuzzy set theory is effective in the linguistic representation of information and incorporates human knowledge dealing with uncertainty and imprecision.

A type-1 fuzzy logic system (FLS) is a system that uses fuzzy set theory to map inputs to outputs. It's contains a fuzzifier block that becomes in fuzzy sets any crisp input data, an inference block that acts in the same way a person do who has do deal with imprecise data, a set of rules or expert knowledge depicted in linguistic values and an defuzzifier block that reduces the output fuzzy set in a crisp output. (see Fig. 2).

In 1975, Zadeh [6] introduces inference systems diffuse type-2 (IT2FLS) as an extension to type 1 fuzzy inference systems. Mendel [7] develops the theory for these sets which have been shown as more effective than types 1.

A type-2 FLS is very similar to a type 1 FIS but a type-2 FLS incorporates a type-reduction block to reduce an interval type-2 fuzzy set into an interval-valued type-1 fuzzy set. A fuzzy set type-1 reaches the defuzzifier block to produce a crisp output (see Fig. 3).

2.4 Fuzzy Neural Network

Union of neural networks and fuzzy logic produces a fuzzy neural network. This network contains a set of nodes arranged on layers that perform the same task independently. Each node is fixed or adaptive [8] (see Fig. 4) and each layer performs different tasks, as described below:

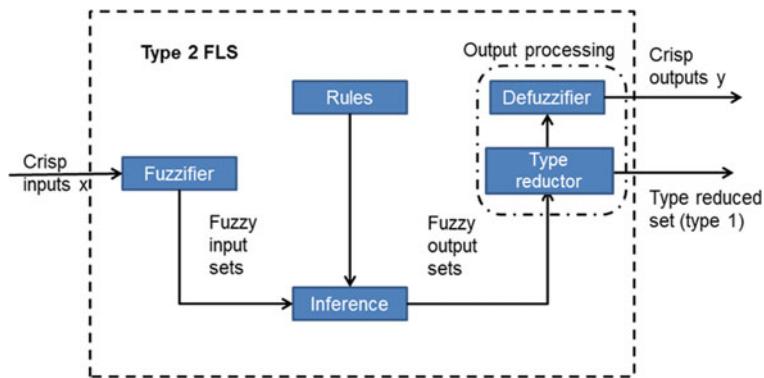


Fig. 3 Type 2 fuzzy logic system diagram

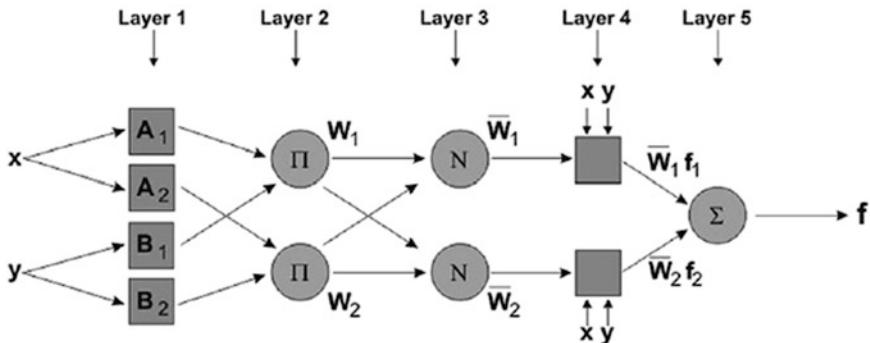


Fig. 4 Fuzzy neural network. *Source* Monika [8]

- Layer 1—Fuzzification layer. Each node in this layer is an adaptive node. The nodes in this layer are called I . The parameters in this layer are called assumptions.
- Layer 2—Inference layer. Each node in this layer is static. The nodes in this layer are called P , whose output is the product of all input signals. The output of each node represents the application of a rule.
- Layer 3—Implication layer. Each node in this layer is static and they are called by N . The i -th node calculates the radius of the i -input rules. The output of this layer is called normalization.
- Layer 4—Aggregation layer. Each i -th node in this layer is an Adaptive node. The parameters in this layer are referred to as the consequences.
- Layer 5—The single node in this layer is a static node, called R , which calculates the sum of all signals output.

2.5 Adaptive Neural Network Fuzzy Inference System (ANFIS)

ANFIS is an implementation of an adaptive neural network of a Takagi-Sugeno fuzzy inference system which integrates Least Square Estimation and Back-propagation algorithms for a rapid learning speed. ANFIS are widely used to explain past data and predict future behavior [9].

2.6 Ensemble

Zhou [10] proposes that the use of several systems of neural networks functioning together do better predictions than separately. This arrangement of fuzzy neural systems is called Ensemble. An ensemble is a learning paradigm that includes multiple components learners that are trained to deal with the same task [11], in other words, several neural networks are attached and are trained to solve a problem [12, 13].

Neural ensemble models can be formed in many different manners. One general approach to creating ensemble members is to hold the training data constant and to vary different factors/parameters of neural networks. For example, ensemble components can be networks trained with different initial random weights, networks trained with different algorithms, networks with different architectures that may include varying number of input and/or hidden nodes, or even networks with different types.

The main idea of the Ensemble is that each fuzzy neural network that integrates has a different way of training and be stimulated, so it brings a different point of view of the same task [3].

3 Data Used in This Study

3.1 Seismic Coordinates

In seismology, a tremor is “a sudden and transient shaking of the Earth’s crust which releases energy in the form of strong elastic waves” [13]. Each registered tremor has at least the same basic data, which are known as seismic coordinates. These data are:

- Magnitude
- Time of the event
- Latitude and longitude, and
- Depth.

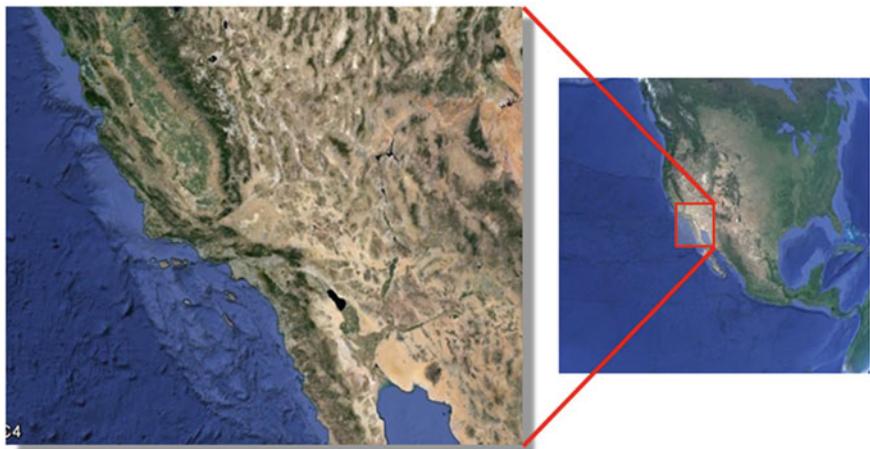


Fig. 5 Region of study

After a seismic event, it is common that events of lesser magnitude following the main arise. These events are known as aftershocks, and decreases with the time.

3.2 Seismic Catalog

Data used in this approach were acquired from the catalog of Southern California Earthquake Data Center. This catalog contains information about seismic events spanning from 1933 to present day.

Seismic coordinates listed above are present in the catalog. Data accumulated over decades allows the use predictive algorithms to estimate future events, such as M8 [14], M8S, CN among others.

An initial set of data spanning from 1993/01/01 thru 2014/09/30 is used in this study. Geographically, this catalog includes data from 30.0° to 39.0° latitude and -111.0° to -124.0° longitude (see Fig. 5). Data contained in this set were pre-processed to eliminate possible duplication and aftershocks [15].

4 Preprocessing Data and M8 Algorithm

4.1 Duplication and Aftershocks

The elimination of duplication of events in the catalog was performed by simple comparison of seismic coordinates. For the removal of aftershocks, Gutenberg-Richter and Omori law [16] were applied.

Gutenberg-Richter law [16] allows setting the scale of the tremors, determining the number of aftershocks that follow a strong land movement in a given region and over a period of time. This relationship is calculated as:

$$N = 10^{(a-b)M} \quad (1)$$

where:

N is the number of events with $M \geq M_0$

M is magnitude

a and b are constants

Typically, b is equal to 1 in seismic active regions and is equal to the rate of earthquakes in the region.

In the other hand, Omori law [17] expresses the decay of aftershocks in a time interval. This law is expressed as:

$$n(t) = \frac{k}{(c + t)} \quad (2)$$

where:

$n(t)$ is the probability that a seismic event shows up in time t

c is the days after the main shock

t is the time of the event

Utsu [5] in 1961 proposed an amendment to this rule so that the probability of the aftershock decreases exponentially within time. As shown in the next equation.

$$n(t) = \frac{k}{(c + t)^p} \quad (3)$$

Both of these laws were used to determine independent and dependent seismic events and to eliminate aftershocks from the catalog.

4.2 M8 Algorithm

The M8 algorithm [14, 18] was designed by retrospective analysis of the preceding seismicity of the great movements of the world ($M \geq 8$). This prediction scheme is implemented as follows:

- The territory to be considered is explored by overlapping circles of diameter equal to $D(M_0)$, where D is diameter of the event of M_0 magnitude.

- In each circle, the sequence of earthquakes is considered eliminating their replicas. The sequence is standardized by the lower magnitude $M_{\min}(N)$, with N as a standard value.
- Time-dependent functions characterize the sequence that is calculated in a window of time (t, t) and a range of magnitudes $M_0 > M_i \geq M_{\min}(N)$. These functions include:
 - $N(t)$, number of main shocks.
 - $L(t)$, standard deviation de $N(t)$ from the long-term trend.

$$L(t) = N(t) - [N_{\text{cum}}(t-s)] \cdot \frac{t-t_0}{t-t_0-s} \quad (4)$$

- $N_{\text{cum}}(t)$ is the total number of main shocks with $M \geq M_{\min}$ from the beginning of the sequence t_0 to t .
- $Z(t)$ is the concentration of major shocks estimated as the average of the diameter of the source radio l to the average distance r between them.
- $B(t) = \max\{b_i\}$, the maximum number of aftershocks, seen in a time window.
- Each N , L y Z function is calculated for $N = 10$ and $N = 20$. As a result seven functions, including the B -function, describes the earthquake sequence.
- High values in functions are identified by a condition that exceed Q percentiles ($\geq Q\%$).
- An alarm is declared in the following five years, when at least six functions, including B , present values that exceed Q . The prediction becomes more stable for two moments, t and $t + 05$ (in years).

5 Architecture of the Ensemble

The architecture of the ensemble is divided into five sections (Fig. 6). The first one is represented by the data base, which consists in the preprocessed catalog.

The second section consists of the ANFIS that are trained and used to validate data; three of them are enough to carry out this study. In the third section, the trained ANFIS are used to generate an initial prediction. The outcomes of the ANFIS are combined in the fourth section and the fifth section a final prediction is made.

Seven time series recorded at six months intervals are set up in the Ensemble. Two series characterize the current level of seismicity within a given region of study. Two others characterize the seismic level based on a ratio of events with magnitude $M > M_0$ over the total number of events. Two more are defined as departure values for a six year period. The seventh series represents the maximum number of aftershocks associated with a main series event.

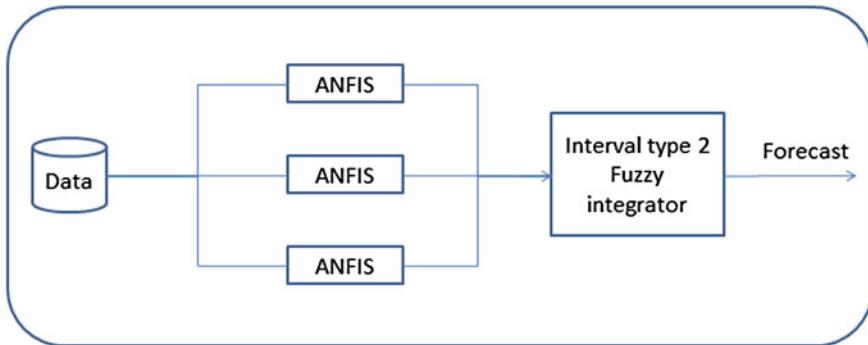


Fig. 6 Seismic hazard predictor ensemble

6 Assumptions

Due the nature of the data, the ensemble is dealing with a chaotic time series. The Database contains the seismic coordinates of all the registered events through the years.

The rates of $M > M_0$ target earthquakes are not known exactly and their empirical estimates are unreliable. Earthquakes with lower magnitude than 5.6 are being used to generate a short and medium term prediction. A variant of the M8 algorithm has to be used to predict earthquakes of magnitude 4.0 an above.

ANFIS deals with the same circles of interest in the study region. A minimum change in the architecture of the Ensemble will allow to ANFIS deal with different and overlapping circles of interest in the same research region. In that case, the resulting prediction would be only over the overlapping region of the circles of interest. Moreover, this minimum change modifies the general approach for creating ensemble members becoming the Ensemble in a modular system.

7 Conclusions

To be useful, an earthquake prediction must be precise enough to warrant the cost of increased precautions; including disruption of ordinary activities and commerce, and timely enough that preparation can be made [19].

The M8 algorithm requires a notoriously extensive research region to evaluate the possibility of a seismic event. For events of magnitude 6.0, 6.5, 7.0 and 7.5 circles of interest (CI) have a diameter of 277,078 km (172.20 miles), 384.69 km (239.08 miles), 562.11 km (349.35 miles) and 854.63 km (531.15 miles) respectively.

The above dimensions described about circles of interest shed some criticize on forecasts made by this algorithm as some researchers believe that it is likely that the

successes be more by chance than the goodness of the method. Therefore, a modification of diameter of the circles of interest or how a sequence of earthquake are associate to each circle is necessary in order to predict tremors with magnitude 6.0 and lower.

An adaptation of the M8 algorithm has to be made for various reasons: for the period from 1993/1/1 up to date, the catalog has little information about tremors with 6.0+ in magnitude. In just 20 years there have been seven (7) earthquakes in the region with these characteristics. The maximum magnitude of a tremor has been of 7.2 on April 4th 2010, and only two have been magnitude 7.0+ in 20 years.

The circles of interest should consider sequences of earthquakes that allow predicting tremors magnitudes 4.0 and above that are more numerous in the catalog and are also of interest in the field of seismic prediction.

The amendments to the M8 algorithm modify the behavior of ANFIS that must respond to new requirements to establish more reliable predictions.

Also, using the Ensemble as a base system, a modular approach can be tested bringing a different perspective over the same problem.

References

1. Zamani, A., Sorbi, M.R., Safavi, A.A.: Application of neural network and ANFIS model for earthquake occurrence in Iran. *Earth Sci. Inform* **6**, 71–85 (2013). Springer, Berlin
2. Bolt, B.A.: *Earthquakes and Geological Discovery*. Scientific American Library, New York (1993)
3. Soto, J., Melin, P., Castillo, O.: Time series prediction using ensembles of neuro-fuzzy models with interval type-2 and type-1 fuzzy integrators. In: Conference: IJCNN 2013, Dallas Texas (2013)
4. Tsunekawa, H.: A Fuzzy Neural Network Prediction Model of the Principal Motions of Earthquakes Based on Preliminary Tremors. IEEE, New Jersey (1998). ISBN: 0-7803-4503-7
5. Utsu, T.: A statistical study of the occurrence of aftershocks. *Geophys. Mag.* **30**, 521–605 (1961)
6. Zadeh, L.A.: Fuzzy sets. *Inf. Control* **8**(3), 338–353 (1965)
7. Karnik, N.N., Mendel, J.M.: Introduction to type-2 fuzzy logic systems. In: Proceedings in 1998 IEEE FUZZY Conference, pp. 915–920. Anchorage, May 1998
8. Monika, A.K.: Comparison of mamdani fuzzy model and neuro fuzzy model for load sensor. *Int. J. Eng. Innovative Technol. (IJEIT)* **2**(9) (2013)
9. Chen D.W., Zhang, J.-P.: Time series prediction based on ensemble ANFIS. In: Proceedings of the Fourth International Conference on Machine Learning and Cybernetics, Guangzhou, 18–21 August 2005
10. Zhou, Z., Wu, J., Tang, W.: Ensembling neural networks: many could be better than all. *Artif. Intell.* **137**(1–2), 239–263 (2002)
11. Dietterich, T.G.: Machine learning research: four current directions. *Artif. Intell.* **18**(4), 97–136 (1998)
12. Yang, J., Yu, P.S.: Mining asynchronous periodic patterns in time series data. *IEEE Trans. Knowl. Data Eng.* **15**(3) (2003)
13. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning representations by back-propagating errors. *Nature* **323**, 533–536 (1986)

14. Keilis-Borok, V.I., Kossobokov, V.G.: Premonitory activation of seismic flow: algorithm M8. *Phys. Earth Planet. Int.* **61**, 73–83 (1990)
15. Keilis-Borok, V.I., Knopoff, L., Rotvain, I.M.: *Nature* **283**, 259–263 (1980)
16. Gutenberg, B., Richter, C.F.: *Seismicity of the Earth and Associated Phenomena*, 2nd ed. Princeton University Press, Princeton (1954)
17. Omori, F.: On the aftershocks of earthquakes. *J. Coll. Sci. Imperial Univ. Tokyo* **7**, 111–200 (1894)
18. Keilis-Borok, V.I.: The Algorithm M8. Russian Academic of Sciences. <http://www.mitp.ru/en/m8pred.html> (2009)
19. Thomas, A.M.: Economic impacts of earthquake prediction. In: Proceedings of the Seminar on Earthquake Prediction Case Histories, pp. 179–185. UNDP/PRO, Geneva, 12–15 October 1982 (1983)

Part IV

Nature Inspired Optimization

Study of Parameter Variations in the Cuckoo Search Algorithm and the Influence in Its Behavior

Maribel Guerrero, Oscar Castillo and Mario García

Abstract We describe in this paper the Cuckoo Search Algorithm via Lévy flights and present the study of parameters behavior in the algorithm. CS is a stochastic algorithm, inspired by the natural behavior of a family of birds called Cuckoos. The parameters considered in the study are: Pa (probability to be discovered by the host bird), α (is the step size), β (variable included in Lévy distribution). The experiments were performed with the Rosenbrock function. We will observe the behavior of Pa , α and β parameters according to the variation of their rank, the number of iterations and the effect they have on the optimal value.

1 Introduction

This paper is concerned with the study of the parameters behavior in the Cuckoo Search Algorithm. The experiments were performed with the Rosenbrock function because it is a complex function for this algorithm, which is a fact based on the previously performed experiments.

Some species, such as the Ani and Guira cuckoos lay their eggs in communal nests, though they may remove others' eggs to increase the hatching probability of their own eggs. Quite a number of species engage the obligate brood parasitism by laying their eggs in the nests of other host birds (often other species) [15].

There are many stochastic algorithms used in the field of optimization, and some of these are inspired by nature, and this is the case of the cuckoo bird, which has served as inspiration for the proposal of a new stochastic algorithm.

The Cuckoo Search Algorithm is a meta-heuristic optimization method proposed by Xin-She Yang and Suash Deb in 2009. CS is based on the brood parasitism of some cuckoo species. In addition, this algorithm is enhanced by the so-called Lévy flights.

M. Guerrero · O. Castillo (✉) · M. García
Tijuana Institute of Technology, Tijuana, BC, Mexico
e-mail: ocastillo@tectijuana.mx

Despite being a relatively young algorithm, CS has multiple applications and researchers continue to propose new variants that help to improve the algorithm. Some variants are: Cuckoo Search via Lévy flights [14] an efficient Cuckoo Search Algorithm for numerical function optimization [10], and Multimodal function optimization [5]. However, a complete understanding of the method is lacking and this is why the study presented here was made.

The parameters in this study are: Pa (probability to be discovered by the host bird), α (is the step size), β (variable included in Lévy distribution). The study graphically shows the behavior of each of them to gradually increase the number of iterations, the study was conducted with the Rosenbrock function due to the results in a previous paper “*Cuckoo Search via Lévy Flights and a Comparison with Genetic Algorithms*” [4], and we want to know the behavior of the parameters in order to implement a fuzzy system to dynamically adjust the parameters for the purpose of helping the CS algorithm performance.

The rest of the paper is organized as follows. Section 2 describes the Cuckoo Search Algorithm via Lévy flights, rules of the algorithm, pseudo code are presented and mathematical Equations, in Sect. 3 the Rosenbrock benchmark mathematical function used to evaluate the performance of optimization algorithms is explained, Sect. 4 presents the study of parameters behavior in the cuckoo search algorithm, and Sect. 5 shows the conclusions of this paper.

2 Cuckoo Search Algorithm

The algorithm is inspired by the reproduction strategy of cuckoos. At the most basic level, cuckoos lay their eggs in the nests of other host birds, which may be of different species. The host bird may discover that the eggs are not its own and either destroy the egg or abandon the nest all together.

This has resulted in the evolution of cuckoo eggs, which mimic the eggs of local host birds [2].

2.1 Lévy Flight

In general, the animals search for food in a random or quasi-random manner. The foraging path of any animal is effectively a random walk, as the next move is based on the current location or state and the transition probability to the next location. The probability for choosing depends implicitly on a probability, which can be modelled mathematically. The flight behavior of many animals and insects has demonstrated the typical characteristics of Lévy flights [1]. Hence, such behavior has been applied for optimization and optimal search, and preliminary results show its promising capability [7].

2.2 Equation of Lévy Flight

The main equation of the Lévy flight is:

$$\text{Lévy}(\beta) = \frac{\sigma(\beta) \times \text{randn}}{|\text{randn}|^{\frac{1}{\beta}}} \quad (1)$$

where β : It is a constant ($1 < \beta \leq 2$).

The standard deviation sigma is calculated with the following equation:

$$\sigma(\beta) = \left\{ \frac{\Gamma(1 + \beta) \times \sin(\frac{\pi\beta}{2})}{\Gamma\left[\frac{(1+\beta)}{2}\right] \times \beta 2^{(\frac{\beta-1}{2})}} \right\}^{\frac{1}{\beta}}, \quad \sigma(\beta) = 1 \quad (2)$$

The Gamma function is defined by the integral:

$$\Gamma(z) = \int_0^{\infty} t^{z-1} e^{-t} dt \quad (3)$$

The Lévy flight represents a random trajectory and it is the length of the step obtained by means of a Lévy distribution (Eq. 4):

$$\text{Lévy} \sim u = t^{-\beta}, \quad (1 < \beta \leq 2) \quad (4)$$

which has an infinite variance with an infinite mean.

2.3 Rules of Cuckoo Search

The breeding behavior of cuckoos, is described below in the basic steps of cuckoo search using the following three rules [11]:

1. Each cuckoo lays one egg at a time, and dumps its egg in a randomly chosen nest.
2. The best nests with high quality of eggs will carry over to the next iterations.
3. The number of available host nests is fixed, and the egg laid by a cuckoo is discovered by the host bird with a probability $Pa \in [0,1]$. In this case, the host bird can either throw the egg away or abandon the nest, and build a completely new nest. For simplicity, this last assumption can be approximated by the fraction Pa of the n nests that are replaced by new nests (with new random solutions).

Each egg in a nest represents a solution and the cuckoo egg represents a new solution. Therefore, there is no difference between an egg, a nest, and solution.

Based on these three rules, the basic steps of the Cuckoo Search (CS) can be summarized as the pseudo code shown in Sect. 2.4.

2.4 Pseudo Code for Cuckoo Search Algorithm

The basic steps of the Cuckoo Search Algorithm can be summarized as the pseudo code shown in Fig. 1.

2.5 Generating New Solutions in the Cuckoo Search Algorithm

When generating new solutions $x_i^{(t+1)}$, for say a cuckoo i , a Lévy flight is performed using Eq. 5:

$$x_i^{(t+1)} = x_i^t + \alpha \cdot \text{randn}(\text{size}(D)) \oplus \text{Lévy}(\beta) \oplus S_i \quad (5)$$

where

- $x_i^{(t+1)}$ The new position.
- x_i^t Current position.
- α is the step size which should be related to the scales of the problem of interests, where $\alpha \geq 0$.
- The product \oplus means entry-wise multiplications.
- $\text{randn}(\text{size}(D))$ random is a random scalar value.

```

Begin
  Objective function  $f(x)$ ,  $x = (x_1, \dots, x_d)^T$ 
  Generate initial population of  $n$  host nests  $x_i$  ( $i = 1, 2, \dots, n$ )
  while ( $t <$  Maximum iteration) or (stop criterion)
    Get a cuckoo randomly by Lévy flights evaluate its quality/fitness  $F_i$ 
    Choose a nest among  $n$  (say,  $j$ ) randomly
    if ( $F_i > F_j$ ),
      replace  $j$  by the new solution;
    end if
    A fraction ( $P_a$ ) of worse nests are abandoned and new ones are built;
    Keep the best solutions (or nests with quality solutions);
    Rank the solutions and find the current best
  end while
  Postprocess results and visualization
End

```

Fig. 1 Pseudo code of the cuckoo search algorithm

D	dimensions.
Lévy(β)	It is a Lévy flight.

$$S_i = (x_i^t - x_{best}) \quad (6)$$

where

x_i^t It is the position vector in the nest we are evaluating at the moment.

x_{best} It is a vector containing the best position found so far.

If $x_i^t = x_{best}$, then the solution is maintained.

3 Rosenbrock Function

The mathematical function was integrated directly into the code of the Cuckoo Search Algorithm.

The function was evaluated with 32 dimensions. The mathematical function is defined below.

Rosenbrock Function [9]:

$$f(x) = \sum_{j=1}^{n_z/2} [100(x_{2j} - x_{2j-1}^2)^2 + (1 - x_{2j-1})^2] \quad (7)$$

Witch $x_j \in [-5, 10]$ and $f^*(x) = 0.0$.

In the area of optimization, mathematical functions have been used to test different methods, like in: An improved evolutionary method with fuzzy logic for combining Particle Swarm Optimization and Genetic Algorithms [8], Optimal design of fuzzy classification systems using PSO with dynamic parameter adaptation through fuzzy logic [12], and other Parallel Particle Swarm Optimization with Parameters.

We are interested in studying the Rosenbrock function, because the CS algorithm when it was compared to the GA [4], the worst results were obtained when working with the Rosenbrock function.

To improve the results we plan to dynamically adjust the parameters using fuzzy logic, but it is necessary to know the ideal range for each parameter to help the original algorithm.

4 Results for the Rosenbrock Function with Pa Variation

In this Section the comparison study of Cuckoo Search with variation of the Pa , α and β parameters is presented. We study the behavior of the algorithm by varying the parameters rank.

The parameters of the algorithm that we can change are:

- Iteration. Number of times the algorithm repeats the basic steps
- $Pa \in [0,1]$: Probability Pa to be discovered by the host bird.
- α : Is the step size, where $\alpha \geq 0$.
- β : It is a constant ($1 < \beta \leq 2$) implicit in the Lévy flight.

The parameters for the Cuckoo Search Algorithm variation of Pa are shown below:

- Number of simulations: 30
- Number of nests: 100
- Number of iterations is variation (1000–6000).
- Pa we vary the value of [0.1–0.9]
- $\alpha = 0.05$ it's recommended by literature
- $\beta = 1.7$
- Dimensions = 32

The parameters for the Cuckoo Search Algorithm variation of α are shown below:

- Number of simulations: 30
- Number of nests: 100
- Number of iterations is variation (1000–6000).
- $Pa = 0.75$
- α We vary the value of [0.1–0.9] and [1.1–1.9]
- $\beta = 1.7$
- Dimensions = 32

The parameters for the Cuckoo Search Algorithm variation of β are shown below:

- Number of simulations: 30
- Number of nests: 100
- Number of iterations is variation (1000–6000).
- $Pa = 0.75$
- $\alpha = 0.05$
- β We vary the value of [1.1–1.9]
- Dimensions = 32

In the following figures we can find the average of 30 times (Average), best result (Best), the worst results (Worst) and the average of the averages for each number of iterations (Avg.), for the Pa parameter.

The results of the tests made with the Rosenbrock function for the CS with Pa variation are shown in Fig. 2.

In Fig. 2, we can find the result obtained of average of 30 tests, varying the Pa parameter of 0.1–0.9, obtained (AVERAGE), best result (BEST), the worst results (WORST) for the range of iterations (1000–6000). In the row (AVG.) we can find

Iterations	Variation Pa (0.1 to 0.9)									Avg.
	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	
1000	Average	28.10	28.78	28.79	28.97	29.23	29.38	28.84	28.04	25.69
	Best	14.55	25.01	24.60	26.22	24.50	27.16	18.81	21.13	11.37
	worst	44.93	31.80	32.14	31.73	33.70	32.44	32.62	32.09	31.65
2000	Average	11.48	17.53	18.41	18.86	18.10	17.57	16.49	16.31	11.21
	Best	4.68	10.31	10.82	11.95	10.08	5.02	4.51	7.96	0.44
	worst	20.10	22.93	23.60	24.62	21.73	23.99	23.96	21.74	20.66
3000	Average	6.08	13.46	15.23	15.29	15.18	13.52	9.71	8.50	6.18
	Best	0.84	4.93	6.83	6.86	6.36	6.75	3.65	0.61	0.16
	worst	13.36	18.17	21.94	19.56	22.18	21.26	16.84	17.64	14.75
4000	Average	3.11	10.52	13.10	12.92	12.92	12.42	8.81	3.82	1.73
	Best	0.18	0.97	4.90	2.12	2.12	3.09	1.01	0.42	8.31E-05
	worst	10.35	15.42	19.96	21.65	21.65	19.31	16.25	15.16	8.20
5000	Average	2.35	8.87	10.42	11.33	10.05	10.73	6.38	2.70	0.18
	Best	0.18	1.73	0.88	3.57	2.01	0.93	0.44	0.11	1.13E-07
	worst	8.11	16.24	20.20	17.55	18.24	18.28	13.96	8.63	1.61
6000	Average	2.08	8.73	10.04	10.67	9.17	8.69	3.64	1.18	1.20E-05
	Best	0.06	0.36	2.77	1.54	2.76	0.31	0.11	0.02	5.37E-13
	worst	10.49	18.18	15.14	16.02	20.56	13.80	9.57	5.68	1.51E-04

Fig. 2 Simulation results for the variation Pa parameter

the average of all the averages ranging Pa 0.1–0.9 corresponding to the number of iterations 1000–6000.

The best average of averages (AVG.) is in 6000 iterations with value of 6.02, and the best Average its 1.20E–05 with value Pa of 0.9, the Best value is of 5.37E–13.

In Fig. 3, we can find the behavior of the CS algorithm, for the X-axis we consider the variations of values in Pa (0.1–0.9) and in the Y-axis the Average of iterations in the range of (0–30).

The best solutions are in the range of 0.7–0.9, and these values are considered in the CS algorithm with dynamic parameter setting for Pa .

The best Average its 1.20E–05 with 6000 iterations and with value Pa of 0.9.

The results of the tests made with the Rosenbrock function for the α parameter in the range of 0.01–0.09 are shown in Fig. 4, α is implicit in Eq. 1, and is related to the step size. The best average of average (AVG.) it's in 6000 iterations with value of 3.81, and the best Average (AVERAGE) is with a value α in 0.05 with average is 2.17.

The Fig. 5, we can find the behavior of the CS algorithm. In X-axis is the variation values in α with a range of 0.01–0.09 and Y-axis is the average obtained for each iteration from 1000 to 6000 iterations.

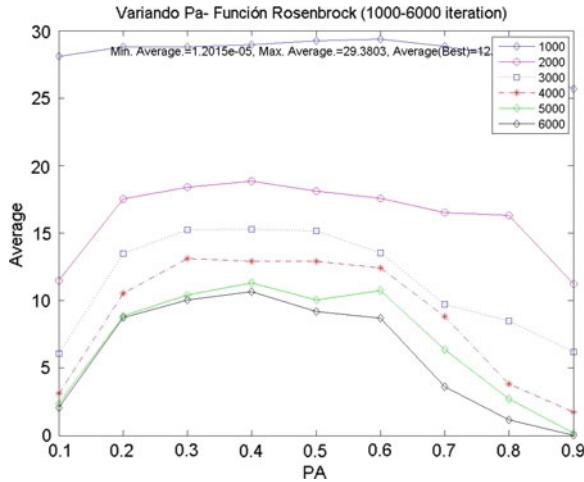


Fig. 3 Variation of Pa

Iterations		Variation α (0.01 to 0.09)									Avg.
		0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09	
1000	Average	52.25	38.30	33.10	29.94	27.79	27.57	27.25	26.06	25.81	32.01
	Best	34.83	32.53	26.94	24.10	24.45	22.64	23.79	10.46	18.95	
	worst	72.89	52.37	50.64	35.82	30.22	29.13	29.72	31.09	29.62	Avg.
2000	Average	23.04	17.99	16.55	16.45	14.17	16.27	16.09	15.37	15.80	16.86
	Best	15.44	7.55	5.17	6.30	6.85	4.85	5.43	3.43	1.06	
	worst	27.63	23.98	23.52	25.08	23.18	22.87	22.28	21.99	23.85	Avg.
3000	Average	17.81	13.31	11.72	11.01	10.48	11.23	10.05	10.94	11.46	12.00
	Best	6.54	5.73	1.26	1.91	3.63	3.60	0.92	1.11	0.64	
	worst	21.68	21.45	19.21	18.66	16.45	17.21	17.83	18.11	18.50	Avg.
4000	Average	12.80	8.71	8.02	8.83	7.66	6.93	6.76	7.77	7.11	8.29
	Best	2.12	2.39	1.39	0.50	0.14	0.47	0.26	0.28	0.53	
	worst	19.12	17.73	15.09	16.51	15.40	13.99	14.93	15.84	12.89	Avg.
5000	Average	8.51	5.28	6.23	5.74	5.33	3.98	6.28	5.35	5.52	5.80
	Best	0.42	0.43	0.05	0.52	0.51	0.10	1.09	0.25	3.50E-04	
	worst	14.74	12.29	13.24	13.82	12.67	10.64	11.14	11.26	10.98	Avg.
6000	Average	7.19	3.56	4.55	3.78	2.17	3.22	2.80	2.96	4.04	3.81
	Best	0.71	0.07	0.11	0.06	0.06	0.24	0.01	0.05	0.04	
	worst	12.32	10.56	8.62	8.60	6.81	9.27	8.86	10.11	9.12	

Fig. 4 Simulation results for the variation of α parameters in (0.01–0.09)

We can find the results obtained of variation α (0.1–0.9) in Fig. 6, the best average it's in 6000 iterations with value is 3.24. The best result is with α in 0.3, with average of 0.78.

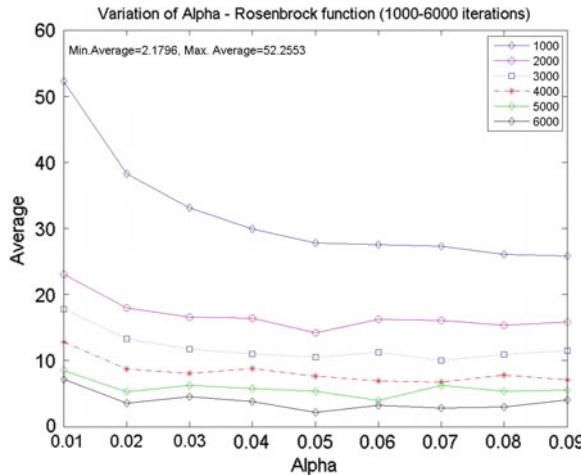


Fig. 5 Variation of α (0.01–0.09)

Iterations	Variation α (0.1 to 0.9)										
	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	Avg.	
1000	Average	25.7	25.82	25.88	26.68	27.4	28.2	29.17	30.97	35.4	28.35
	Best	20.2	22.23	24.16	24.97	25.7	26.5	28.44	29.63	31.10	
	worst	29.2	27.68	27.31	27.99	28.7	30.1	29.8	36.03	47.46	Avg.
2000	Average	16.9	18.13	19.77	20.47	21.2	22	23.21	24.37	25.07	21.23
	Best	5.93	3.92	17.93	19.06	12.4	16.5	19.9	22.77	22.52	
	worst	23.6	25.12	22.1	21.75	22.8	23.3	24.13	25.39	26.21	Avg.
3000	Average	12.9	13.39	13.21	14.78	15.6	17.2	18.32	19.43	20.36	16.13
	Best	2.83	8.65	1.85	13.53	13	16	14.26	18.09	19.22	
	worst	17.4	17.46	15.28	16.27	16.7	18.5	20.63	20.49	21.62	Avg.
4000	Average	7.83	7.13	8.40	9.01	10.6	12.2	13.3	14.94	15.67	11.01
	Best	0.35	0.99	5.83	2.62	7.19	10.9	6.29	13.86	10.35	
	worst	13.1	11.09	10.6	11.99	12.4	13.5	14.83	16.41	17.48	Avg.
5000	Average	5.53	3.86	3.68	4.60	6.30	7.33	9.34	10.54	11.93	7.01
	Best	0.68	0.09	0.63	1.65	4.81	0.40	7.54	9.12	10.30	
	worst	10.80	8.97	6.25	6.58	8.27	8.87	11.48	11.71	13.26	Avg.
6000	Average	2.44	0.92	0.78	0.93	1.70	3.20	5.18	6.30	7.75	3.24
	Best	0.02	2.52E-03	1.02E-03	1.03E-01	0.31	1.36	2.68	2.16	4.63	
	worst	6.98	4.82	5.24	4.26	4.37	6.10	11.77	8.39	9.62	

Fig. 6 Simulation results for the variation of α parameter (0.1–0.9)

In Fig. 7, we can find the behavior of the CS algorithm, for the X-axis we consider the variations of values in α (0.1–0.9) and Y-axis the Average of iterations (1000–6000), the Best Average is in $\alpha = 0.3$ for 6000 iteration with Min. Average is 0.78 and Max. Average is 35.40 for 1000 iteration with value in $\alpha = 0.9$.

Fig. 7 Variation of α (0.1–0.9)

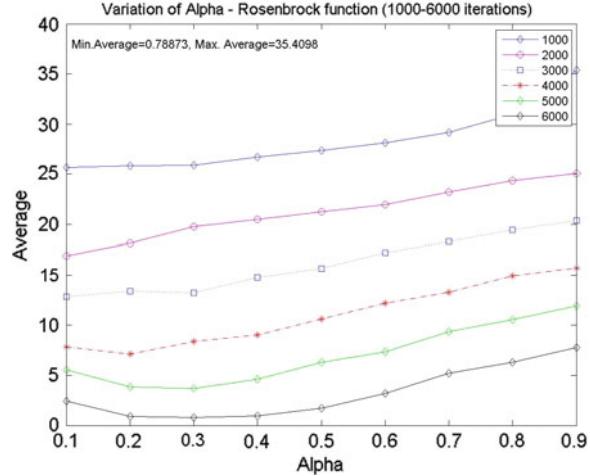


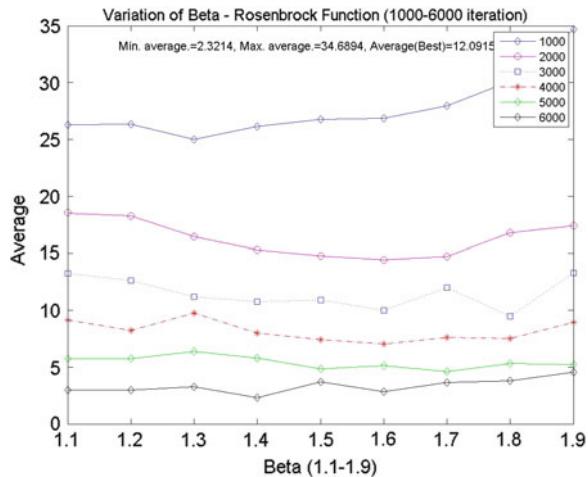
Fig. 8 Simulation results for the variation of β (1.1–1.9) parameter

Iterations	Variation β (1.1 to 1.9)										Avg.
	1.1	1.2	1.3	1.4	1.5	1.6	1.7	1.8	1.9	Avg.	
1000	Average	26.26	26.33	24.99	26.15	26.77	26.83	27.97	30.09	34.68	27.79
	Best	21.73	20.52	14.01	19.75	14.79	17.58	21.60	26.46	29.32	
	worst	28.36	27.93	28.87	29.53	30.48	29.22	31.69	35.05	62.96	Avg.
2000	Average	18.52	18.25	16.44	15.31	14.78	14.43	14.73	16.77	17.40	16.29
	Best	14.40	10.12	4.61	5.83	4.00	4.07	6.50	7.05	7.37	
	worst	21.48	23.60	21.88	22.91	21.24	20.94	23.78	22.37	27.00	Avg.
3000	Average	13.23	12.64	11.18	10.75	10.91	10.00	12.00	9.45	13.28	11.49
	Best	6.16	2.33	1.83	3.02	2.30	0.69	2.48	0.15	2.68	
	worst	17.64	18.10	17.37	18.29	20.45	17.64	20.90	17.36	20.90	Avg.
4000	Average	9.14	8.25	9.77	8.01	7.43	7.06	7.61	7.50	8.95	8.19
	Best	0.67	0.29	1.95	1.13	1.09	0.39	0.04	0.03	0.19	
	worst	12.91	14.24	18.91	13.70	12.24	15.31	17.00	15.59	15.65	Avg.
5000	Average	5.73	5.76	6.36	5.78	4.85	5.13	4.62	5.30	5.21	5.42
	Best	1.13	0.43	0.57	0.23	8.30E-03	0.14	0.02	0.32	0.84	
	worst	8.10	10.51	10.55	15.83	10.77	10.89	11.31	11.76	11.58	Avg.
6000	Average	2.99	2.99	3.26	2.32	3.69	2.82	3.65	3.79	4.56	3.34
	Best	0.04	0.04	0.06	0.12	0.15	0.06	0.11	0.17	0.43	
	worst	8.95	8.95	6.44	7.98	8.50	7.32	9.48	8.36	9.44	

The best average for Fig. 5 the Min. Average is 2.17 with value α in 0.05, in Fig. 7 the Min. Average is 0.78 for value α of 0.3 (Fig. 8).

In Fig. 9, we can find the behavior of the CS algorithm, for the X-axis we consider the variation values in β (1.1–1.9) and Y-axis the Average of iterations (1000–6000), the Min. Average is 2.32 with value the $\beta = 1.4$ and Max. The average is 34.68 for 1000 iterations with value in $\beta = 1.9$.

Fig. 9 Variation of β (1.1–1.9)



5 Conclusions

After having completed the study of the behavior of the parameters in the Cuckoo Search algorithm, we realize that the Rosenbrock function requires computing time to achieve good results and this can be achieved through a good balance in their P_a , α and β parameters.

In Sect. 4 the analysis for the comparative study of the CS algorithm variation parameters, we find that there are promising results.

The study of manual variation of the P_a parameter will help in designing the output and input ranges for the fuzzy system that will be built to dynamically adapt parameters in the CS algorithm.

The CS algorithm has successful applications to various problems of optimization [13], image processing [3], engineering design and data clustering [6], among other applications, and for this reason it is important to improve its results.

Acknowledgments We would like to express our gratitude to the CONACYT and Tijuana Institute of Technology for the facilities and resources granted for the development of this research.

References

1. Bo, X., Wen-Jing, G.: An overview of cuckoo-inspired intelligent algorithms and their applications. In: 2013 IEEE Symposium on Swarm Intelligence (SIS), pp. 85–89, 16–19 April 2013
2. Civicioglu, P., Besdok, E.: Comparative Analysis of the Cuckoo Search Algorithm, pp. 85–113. Springer, Berlin (2014)
3. Ghosh, S., Roy, S., Kumar, U., Mallick, A.: Gray Level Image Enhancement Using Cuckoo Search Algorithm, pp. 275–286. Springer, Berlin (2014)

4. Guerrero, M., Castillo, O., García, M.: Cuckoo search via lévy flights and a comparison with genetic algorithms. In: Fuzzy Logic Augmentation of Nature-Inspired Optimization Metaheuristics. Studies in Computational Intelligence, vol. 574, pp. 91–103. Springer, Berlin (2015)
5. Jamil, M., Zepernick, H.-J.: Multimodal function optimisation with cuckoo search algorithm. *Int. J. Bio-Inspired Comput.* **5**(2), 73–83 (2013)
6. Manikandan, P., Selvarajan, S.: Data Clustering Using Cuckoo Search Algorithm (CSA), Advances in Intelligent Systems and Computing, pp. 1275–1283. Springer, Berlin (2014)
7. Marichelvama, M.K., Prabaharan, T., Yang, X.-S.: Improved cuckoo search algorithm for hybrid flow shop scheduling problems to minimize makespan. *Appl. Soft Comput.* **19**, 93–101 (2014)
8. Melin, P., Olivas, F., Castillo, O., Valdez, F., Soria, J., Valdez, M.: Optimal design of fuzzy classification systems using PSO with dynamic parameter adaptation through fuzzy logic. *Expert Syst. Appl.* **40**(8), 3196–3206 (2013)
9. Molga, M., Smutnicki, C.: Test functions for optimization needs (2005)
10. Ong, P., Zainuddin, Z.: An efficient cuckoo search algorithm for numerical function optimization. In: AIP Conference Proceedings, vol. 1522, p. 1378 (2013)
11. Rajabioun, R.: Cuckoo optimization algorithm. *Appl. Soft Comput.* **11**, 5508–5518 (2011)
12. Valdez, F., Melin, P., Castillo, O.: Evolutionary method combining particle swarm optimization and genetic algorithms using fuzzy logic for decision making. In: Proceedings of the IEEE International Conference on Fuzzy Systems, pp. 2114–2119 (2009)
13. Yang, X.-S., Deb, S.: Engineering optimisation by cuckoo search. *Int. J. Math. Model. Numer.* **1**(4), 330–343 (2010)
14. Yang, X.-S., Deb, S.: Cuckoo search via Lévy flights. In: World Congress on Nature & Biologically Inspired Computing, 2009. NaBIC 2009, pp. 210–214 (2009)
15. Yang, X.-S.: Nature-Inspired Metaheuristic Algorithms, 2nd edn. Luniver Press (2010)

A New Bio-inspired Optimization Algorithm Based on the Self-defense Mechanisms of Plants

Camilo Caraveo, Fevrier Valdez and Oscar Castillo

Abstract In this work a new optimization algorithm inspired in the self-defense mechanisms of plants is presented. Through time the planet has gone through changes, so plants have had to adapt to these changes and adopt new techniques to defend from natural predators (herbivores). For the development of this algorithm we consider as a main idea the model of Lotka and Volterra predator prey model where two populations are taken and the objective is to maintain a balance between the two populations. Simulation results are presented to illustrate the potential of the proposed approach.

1 Introduction

Several meta-heuristic algorithms have been developed to solve various combinatorial optimization problems. These can be classified into different groups based on the criteria that are, for example, deterministic, iterative, population based, stochastic, etc. An algorithm that works with a set of solutions and the use of multiple iterations to approach the desired solution is called as iterative algorithm based on populations.

Throughout history there have been tested and developed multiple methods of search and optimization inspired by natural processes. This with the goal of solving particular problems in the area of computer science has tried different bio-inspired methods such as ACO, PSO, BCO, GA etc. [7, 8, 11]. Trying to get the solution of a specific problem with a smaller error.

C. Caraveo · F. Valdez (✉) · O. Castillo
Tijuana Institute of Technology, Tijuana, Mexico
e-mail: fevrier@tectijuana.mx

C. Caraveo
e-mail: Camilo.Caraveo@gmail.com

O. Castillo
e-mail: ocastillo@tectijuana.mx

In this work a new optimization algorithm inspired in the self-defense mechanisms of plants is presented. This in order to compete against the existing optimization methods in nature plants are exposed to many different pathogens in the environment. However, only a few can affect them. If a particular pathogen is unable to successfully attack a plant, it is said that it is resistant to it, in other words, cannot be affected by the pathogen. Although in nature this type of immunity stops almost all parasites, has received little investigation. Some scientists have uncovered the molecular components of the first line of defense. In their findings, they conclude parallels between plant and animal immune systems.

The proposed approach takes as its main basis the Lotka and Volterra predator-prey model, which is a system formed by a pair of first order differential equations nonlinear moderating the growth of two populations that interact with each other (predator and prey) [6].

2 Self-defense Mechanisms of the Plants

In nature, plants as well as animals are exposed to a large number of invading organisms such as insects, fungi, bacteria and viruses that can cause various types of diseases, and even death [10, 15].

Defense mechanisms (or coping strategies) are automatic processes that protect the individual against external or internal threats. The plant is able to react to external stimuli. When it detects the presence or attack of an organism triggers a series of chemical reactions that are released into the air that attracts natural predator of the assailant or cause internal damage to the aggressor [9].

The leaves normally release into the air small amounts of volatile chemicals, but when a plant is damaged by herbivorous insects, the amount of chemicals tends to grow. Volatile chemicals vary depending on the plant species and species of herbivorous insects [12]. These chemicals attract both predators and parasitic insects that are natural enemies of herbivores. Such chemicals, which work in the communication between two or more species, as well as those who serve as messengers between members of the same species are called semi-chemicals [1, 2, 5, 10].

These chemicals are also used to attract insects that are used for pollination of plants favoring the plant in the reproduction of their species, thus preventing their extinction.

3 Predator-Prey Model

One of the most remarkable features of life on our planet is the great diversity of issues and habits that have bodies that compose it. However, it is rather difficult to see that this manifestation of diversity does not happen in an arbitrary manner, but quite the opposite. The organisms live in communities, forming intricate

relationships of interaction, where each species directly or indirectly dependent on the presence of the other. One of the tasks of Ecology is to develop a theory of community organization for understanding the causes of diversity and mechanisms of interaction. In this paper we consider the interaction of two whose population size at time t is $x(t)$ and $y(t)$ species [2, 3]. Furthermore, we assume that the change in population size can be written as:

$$\frac{dy}{dt} = I(x, y) \quad (1)$$

$$\frac{dx}{dt} = P(x, y) \quad (2)$$

There are different kinds of biological interaction can be represented mathematically with this system of equations [3]. As $P(x, y)$ and $I(x, y)$ determining the growth rate of each of the populations; there is the case where one of these species is fed from the other, then the system of survival is given by: Eq. (3)

$$\begin{aligned} P_y(x, y) &< 0 \\ I_x(x, y) &> 0 \end{aligned} \quad (3)$$

That is, the change of the prey population relative to the predator decreases and the change of the predator population relative to the prey increases. These are some of the conditions that must meet a set of predator-prey equations [3, 14].

3.1 Analysis of the Lotka and Volterra Model

This model is based on the following assumptions.

1. The population grows proportionally to its size, and has enough space and food. If this happens and $x(t)$ represents the prey population (in the absence of predator), then the population growth is given by:

$$\begin{aligned} \frac{dx}{dt} &= ax, \quad a > 0, \\ x(t) &= x_0 e^{at}. \end{aligned} \quad (4)$$

The population of prey in the absence of the predator grows exponentially.

2. The predator $y(t)$ only feeds on the prey $x(t)$. Thus, if there is no prey, their size decreases with a rate proportional to its population is represented by Eq. (5)

$$\begin{aligned} \frac{dy}{dt} &= -dy, \quad d > 0, \\ y(t) &= y_0 e^{-dt} \end{aligned} \quad (5)$$

The population of predators in the absence of prey decreases exponentially to extinction.

3. The number of encounters between predator and prey is proportional to the product of their populations. Each of the number of encounters favors predators and reduces the number of prey.

The presence of prey helps the growth of the predator is represented by Eq. (6)

$$cxy, c > 0. \quad (6)$$

While the interaction between them, reduces the growth of prey is represented by Eq. (7)

$$-bxy, b > 0. \quad (7)$$

Under the above hypothesis we have a model of interaction between $x(t)$ and $y(t)$ is given by the following system: Eqs. (8) and (9)

$$\frac{dx}{dt} = Ax - Bxy \quad (8)$$

$$\frac{dy}{dt} = -Cxy + Dy \quad (9)$$

x is the number of prey.

y is the number of predator.

$\frac{dx}{dt}$ is the growth of the population of prey time t .

$\frac{dy}{dt}$ is the growth of the population of predator at time t .

A it represents the birth rate of prey in the absence of predator.

B it represents the death rate of predators in the absence of prey.

C measures the susceptibility of prey.

D measures the ability of predation.

4 Proposed Optimization Algorithm Based on the Self-defense Mechanisms of Plants

The proposed approach takes as its main basis the Lotka and Volterra predator-prey model, which is a system formed by a pair of differential equations of first order nonlinear moderating the growth of two populations that interact with each other (predator and prey).

To generate the initial population of the algorithm we use the equations of the model of Lotka and Volterra, the mathematical representation is shown in Sect. 3 Eqs. (8) and (9). Equation 8 is used to generate the population of prey (plants), and Eq. 9 is used to generate the population of depredators (herbivores), as mentioned above functions predator prey model is used to model our proposed model variables adapted to the proposal.

Figure 1 describes the steps of the optimization algorithm inspired by the defense mechanisms of plants.

The initial sizes of both populations (prey, predators) are defined by the user, the parameters (a , b , c , d) are also defined by the user, the model of Lotka and Volterra recommend the following parameters values $a = 0.4$, $b = 0.37$, $c = 0.3$, $d = 0.05$. The two populations interact with each other to generate the initial population of a plant,

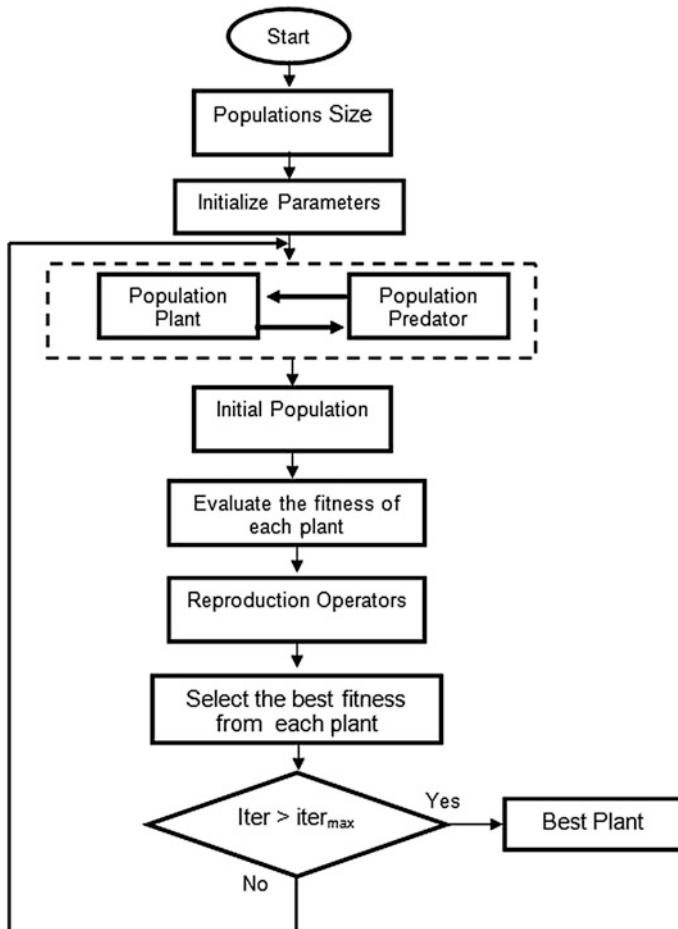


Fig. 1 Flowchart of the proposed algorithm

the plant population is evaluated reproduction in operators are applied to generate the plant offspring. The population is re-evaluated and if the stop criterion is not satisfied return the iterative cycle of the algorithm.

5 Simulation Results

In this section the results obtained are shown to test the performance of the algorithm, we used a set of benchmark functions, where the goal is to approximate minimize value of zero. 15 experiments were performed for the following mathematical functions the evaluation is for 1 Variable.

Table 1 shows the definitions of the mathematical functions used in this work.

5.1 Parameters Settings for the Algorithm

The parameters of common control in the algorithms are the population size and the number of maximum generation. The proposed the amount of parameters to defining the algorithm is more than in traditional algorithms. The configuration parameters are defined below: the parameters (a , b , c , d), the model of Lotka and Volterra recommends the following values for the variables $a = 0.4$, $b = 0.37$, $c = 0.3$, $d = 0.05$. We also need to define the size of prey populations (plants) and predators (herbivores), the model does not recommend optimal values for populations, and we use plants = 3000, Herbivores = 2000, must also define the time and maximum number of iterations, and we use iterations = 1000, time = iterations/2.

Table 2 shows the experimental results for the benchmark mathematical functions used in this research optimized with the proposed method. The Table shows the results of the evaluations for each function with 1 dimension; where it can be noted the best obtained values, and the average of 15 experiments.

Table 1 Mathematical functions

Function	Mathematical functions
Sphere	$f(x) = \sum_{i=1}^n x_i^2$
Griewank	$f(x) = \sum_{i=1}^d \frac{x_i^2}{4000} - \prod_{i=1}^d \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$
Rastrigin	$f(x) = 10n + \sum_{i=1}^n x_i^2 - 10 \cos(2\pi x_i)$
Ackley	$f(x) = -a \cdot \exp(-b \cdot \sqrt{\frac{1}{n} \sum_{i=1}^n x_i^2}) - \exp(\frac{1}{n} \sum_{i=1}^n \cos(cx_i)) + a + \exp(1)$

Table 2 Experimental result with 1 dimension

Function	Experiments with 1 dimensions			
	Optimization algorithm based on the self-defense mechanisms of plants			
	Best	Worse	Average	σ
Griewank	6.94E-11	1.15E-06	2.69E-07	3.63421E-07
Rastrigin	6.21E-08	0.000130732	2.06E-05	3.77699E-05
Ackley	1.05E-06	9.00E-04	3.49E-04	0.000269464
Sphere	3.67E-10	1.22E-06	1.62E-07	3.18435E-07

6 Conclusions

The proposal is to create and develop a new optimization algorithm inspired by the bio-defense mechanism of plants, the first challenge is to achieve the adaptation of the predator-prey model and test the algorithm on a problem, in this case we decided to try applying mathematics, and we have achieved acceptable results. This algorithm is in the early stages, we are researching necessary information about natural processes of plants to add adaptations of biological processes to the algorithm, in order to improve results and achieve our proposal can be make a competitive algorithm against methods bio-inspired existing.

References

1. Bennett, R.N., Wallsgrove, R.M.: Secondary metabolites in plant defense mechanisms. *New Phytol.* **127**(4), 617–633 (1994)
2. Berryman, A.A.: The origins and evolution of predator-prey theory. *Ecology* 1530–1535 (1992)
3. Cruz, J.M.L., González, G.B.: Modelo Depredador-Presa. *Revista de Ciencias Básicas UJAT* **7**(2), 25–34 (2008)
4. García-Garrido, J.M., Ocampo, J.A.: Regulation of the plant defence response in arbuscular mycorrhizal symbiosis. *J. Exp. Bot.* **53**(373), 1377–1386 (2002)
5. Law, J.H., Regnier, F.E.: Pheromones. *Annu. Rev. Biochem.* **40**(1), 533–548 (1971)
6. Lez-Parra, G.G., Arenas, A.J., Cogollo, M.R.: Numerical-analytical solutions of predator-prey models. *WSEAS Trans. Biol. Biomed.* **10**(2) (2013)
7. Melin, P., Olivas, F., Castillo, O., Valdez, F., Soria, J., Valdez, M.: Optimal design of fuzzy classification systems using PSO with dynamic parameter adaptation through fuzzy logic. *Expert Syst. Appl.* **40**(8), 3196–3206 (2013)
8. Neyoy, H., Castillo, O., Soria, J.: Dynamic fuzzy logic parameter tuning for ACO and its application in TSP problems. In: Recent Advances on Hybrid Intelligent Systems, pp. 259–271. Springer, Berlin (2013)
9. Ordeñana, K.M.: Mecanismos de defensa en las interacciones planta-patógeno. *Revista Manejo Integrado de Plagas. Costa Rica*, **63**, 22–32 (2002)
10. Paré, P.W., Tumlinson, J.H.: Plant volatiles as a defense against insect herbivores. *Plant Physiol.* **121**(2), 325–332 (1999)
11. Teodorovic, D.: Bee colony optimization (BCO). In: Lim, C.P., Jain, L.C., Dehuri, S. (eds.) Innovations in Swarm Intelligence, pp. 39–60. Springer, Berlin (2009). (65, 215)

12. Tollsten, L., Müller, P.M.: Volatile organic compounds emitted from beech leaves. *Phytochemistry* **43**, 759–762 (1996)
13. Vivanco, J.M., Cosio, E., Loyola-Vargas, V.M., Flores, H.E.: Mecanismos químicos de defensa en las plantas. *Investigación y ciencia* **341**(2), 68–75 (2005)
14. Xiao, Y., Chen, L.: Modeling and analysis of a predator–prey model with disease in the prey. *Math. Biosci.* **171**(1), 59–82 (2001)
15. Yoshida, T., Jones, L.E., Ellner, S.P., Fussmann, G.F., Hairston, N.G.: Rapid evolution drives ecological dynamics in a predator–prey system. *Nature* **424**(6946), 303–306 (2003)

Imperialist Competitive Algorithm Applied to the Optimization of Mathematical Functions: A Parameter Variation Study

Emer Bernal, Oscar Castillo and José Soria

Abstract This paper applies the imperialist competitive algorithm (ICA) to benchmark mathematical functions with the original method to analyze and perform a study of the variation of the results obtained with the ICA algorithm as we vary the parameters manually for 4 mathematical functions. The results demonstrate the efficiency of the algorithm to optimization problems and give us the pattern for future work in dynamically adapting these parameters.

1 Introduction

Swarm Intelligence techniques have become increasingly popular during the last two decades due to their capability to find a relatively optimal solution for complex combinatorial optimization problems. They have been applied in the fields of engineering, economy, management science, industry, etc. Problems that benefit from the application of Swarm Intelligence techniques are generally very hard to solve optimally in the sense that there is no such exact algorithm for solving them in polynomial time. These optimization problems are also known as NP-hard problems [6].

An algorithm that is well recognized in the domain of evolutionary computation is the imperialist competitive algorithm (ICA), which was introduced by Atashpaz-Gargari and Lucas in 2007. ICA has been inspired by the concept of imperialism; where in this case powerful countries attempt to make a colony of other countries. These algorithms have recently been used in several engineering applications [11].

E. Bernal · O. Castillo (✉) · J. Soria
Tijuana Institute of Technology, Tijuana, BC, México
e-mail: ocastillo@tectijuana.mx

We describe the imperialist competitive algorithm in its original form. The algorithm parameters adjustment is performed manually by varying the parameters to see how the behavior of the algorithm is affected. This algorithm was applied to benchmark mathematical functions, and the results of the ICA algorithm by varying the parameters are presented in Sect. 4.

The study of the algorithm is performed in order to show the effectiveness of the imperialist competitive algorithm (ICA) when applied to optimization problems, and to take this as a basis for future works.

Some of the papers that have applied the imperialist competitive algorithm can be described as follows. In [11] the Imperialist Competitive Algorithm Combined with Refined High-Order Weighted Fuzzy Time Series (RHWFTS–ICA) for short term loads forecasting. In this study, a hybrid algorithm based on a refined high-order weighted fuzzy algorithm and an imperialist competitive algorithm (RHWFTS–ICA) is developed. This method is proposed to perform efficiently under short-term load forecasting (STLF) [11]. In another paper [7] an Imperialist Competitive Algorithm With PROCLUS Classifier For Service Time Optimization In Cloud Computing Service Composition, CSSICA is proposed to make advances toward the lowest possible service time of composite service; in this approach, the PROCLUS classifier is used to divide cloud service providers into three categories based on total service time and assign a probability to each provider. An improved imperialist competitive algorithm is then employed to select more suitable service providers for the required unique services [7].

The paper is organized as follows: in Sect. 2 a description about the imperialist competitive Algorithm ICA is presented, in Sect. 3 a description of the mathematical functions is presented, in Sect. 4 the experiments results are described for we can to appreciate the ICA algorithm behavior by varying the parameters, in Sect. 5 the conclusions obtained after the study of the imperialist competitive algorithm versus mathematical functions are presented.

2 Imperialist Competitive Algorithm

In the field of evolutionary computation, the novel ICA algorithm is based on human social and political advancements [1], unlike other evolutionary algorithms, which are based on the natural behaviors of animals or physical events.

ICA starts with an initial randomly generated population, in which the individuals are known as countries. Some of the best countries are considered imperialists, whereas the other countries represent the imperialist colonies [7].

All the colonies of the initial population are divided among the mentioned imperialists based on their power. The power of an empire which is the counterpart of the fitness value in GA and is inversely proportional to its cost [1].

2.1 Forming Initial Empires (Initialization)

In order to represent an appropriate solution format, a $1 \times N_{var}$ array of variables represents a country, and a country is defined by [6]:

$$\text{Country} = [p_1, p_2, \dots, p_{N_{var}}] \quad (1)$$

where N_{var} is the number of variables to be considered of interest about a country and p_i is the value of i -th variable.

The variable values in the country are represented as floating point numbers. The cost of a country is found by evaluating the cost function f at the variables (p_1, p_2, \dots, p_n) then [1]:

$$\text{Cost} = f(\text{Country}) = f(p_1, p_2, \dots, p_n) \quad (2)$$

In the initialization step, we need to generate an initial population size of N_{pop} . Select N_{imp} of the most powerful countries to form empires. The remaining N_{col} population will be the colonies each of which belongs to an empire [1]

$$N_{coI} = N_{pop} - N_{imp} \quad (3)$$

To form empires, the colonies are divided among the imperialist countries according to the power of the imperialists. The normalized cost of each imperialist is determined by [6]

$$c_n = \max_i \{c_i\} - c_n \quad (4)$$

where, c_n is the n -th imperialist's cost, and C_n is the normalized cost of n -th imperialist.

Therefore, the power of each imperialist is calculated based on the normalized cost [6]:

$$p_n = \frac{C_n}{\sum_{i=1}^{N_{imp}} C_i} \quad (5)$$

where p_n is the power of n -th imperialist. The normalized power of n -th imperialist is the number of colonies that are possessed by that imperialist, calculated by:

$$NC_n = \text{Round}\{p_n N_{col}\} \quad (6)$$

where, NC_n is the number of initial colonies possessed by the n -th imperialist; N_{col} is the total number of colonies in the initial population, and round is a function that gives the nearest integer of a fractional number.

2.2 Moving the Colonies of an Empire Toward the Imperialist (Assimilation)

As shown in Fig. 1 the colony moves x distance along with the d direction towards its imperialist. The moving at x distance is a random number generated by random distribution within the interval $(0, \beta d)$ [6].

$$x \sim U(0, \beta d) \quad (7)$$

where β is a number greater than 1 and d is the distance between the colony and the imperialist.

As shown in Fig. 2, to search for different locations around the imperialist we add a random amount of deviation to the direction of motion, which is given by [1]:

$$\theta \sim U(-\gamma, \gamma) \quad (8)$$

where θ is a random number with uniform distribution and γ is a parameter that adjusts the deviation from the original direction.

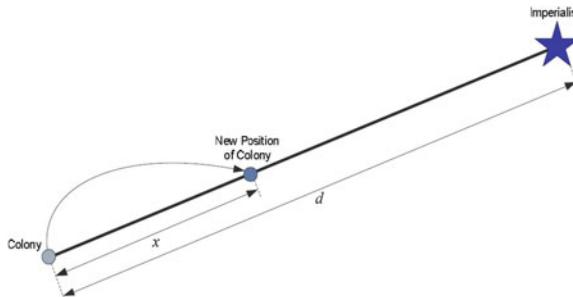


Fig. 1 Movement of the colonies toward the imperialist

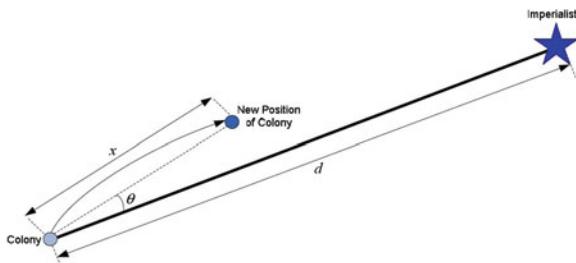


Fig. 2 Movement of the colonies toward their relevant imperialist in a randomly direction deviation

2.3 Exchanging the Positions of a Colony and an Imperialist

While moving towards the imperialist, a colony can reach a position of lower cost than the imperialistic. If so, the imperialist is moved to the position of that colony and vice versa. Then the algorithm will continue with the imperialist in a new position and the colonies begin to move toward this position [1]. In Fig. 3, the best colony of the empire is shown in darker color. This colony has a lower cost than imperialist. Figure 4 shows the whole empire after exchanging the position of the imperialist and the colony.

2.4 Total Power of an Empire

The power of an empire is calculated based on the power of its imperialist and a fraction of the power of its colonies. This fact has been modeled by defining the total Cost given by [6]:

$$TC_n = \text{Cost}(Imp) + \xi \text{mean}\{\text{Cost}(Col)\} \quad (9)$$

where TC_n is the total cost of n -th Empire and ξ is a positive number between 0 and 1.

Fig. 3 Position change between the imperialist and a colony

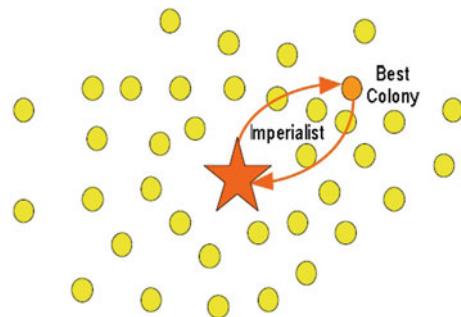
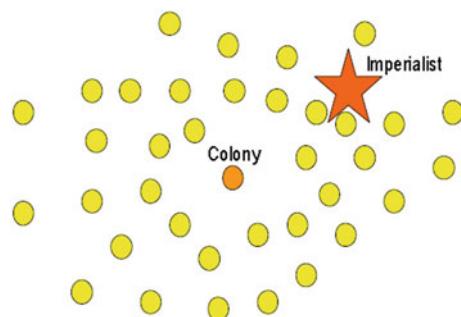


Fig. 4 Position after exchanging empire imperialist and the colony



2.5 Imperialist Competition

To start the competition, first, we find the probability of possession of each empire based on the total power. The normalized total cost is obtained by [1]:

$$NTC_n = \max_i\{TC_i\} - TC_n \quad (10)$$

where NTC_n is the Normalized total cost and TC_n is the total cost of empire. Then the probability of possessing a colony is computed by:

$$p_{p_n} = \frac{NTC_n}{\sum_{i=1}^{N_{imp}} NTC_i} \quad (11)$$

where $\sum_{i=1}^{N_{imp}} p_{p_i} = 1$.

2.6 Elimination of Weaker Empires

Weaker empires lose their colonies gradually to stronger empires, which in turn grow more powerful and cause the weaker empires to collapse over time. In Fig. 5, the weakest empire is eliminated by losing its last colony during the imperialist competition [6].

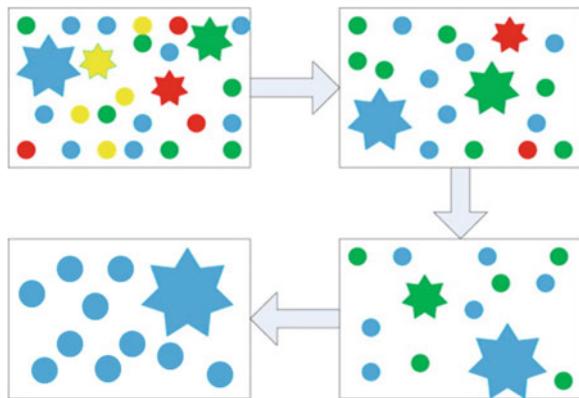
2.7 Convergence

Similar to other metaheuristic algorithms, ICA continues until a stopping criteria are met, such as predefined running time or a certain number of iterations. The ideal stopping criterion is when all empires have collapsed and only one (grand empire) remains (Fig. 6).



Fig. 5 Elimination of the weakest empire

Fig. 6 Representation of convergence in ICA



2.8 Pseudo Code of ICA

The pseudo code of ICA is defined as follows:

1. Select some random points on the function and initialize the empires.
2. Move the colonies toward their relevant imperialist (assimilation).
3. If there is a colony in an empire which has lower cost than that of imperialist, exchange the positions of that colony and the imperialist.
4. Calculate the total cost of all empires (related to the power of both imperialist and its colonies).
5. Pick the weakest colony (colonies) from the weakest empire and give it (them) to the empire that has the most likelihood to possess it (imperialistic competition).
6. Eliminate the powerless empires.
7. If there is just one empire, stop, if not go to step 2.

3 Benchmark Mathematical Functions

In this Section, the benchmark functions that are used are listed to evaluate the performance of the ICA algorithm by varying its parameters and to analyze the results obtained.

In the area of the metaheuristics for optimization the use of mathematical functions is common, and they are used in this work: consisting of an optimization algorithm based on imperialism in which the variation of its parameters will be analyzed for obtain its optimum values.

The mathematical functions are shown below:

- Sphere

$$f(x) = \sum_{j=1}^{n_x} x_j^2 \quad (12)$$

With $x_j \in [-100, 100]$ and $f^*(x) = 0.0$

- Quartic

$$f(x) = \sum_{j=1}^{n_x} x_j^4 + \text{random}(0, 1) \quad (13)$$

With $x_i \in [-1.28, 1.28]$ and $f^*(x) = 0.0 + \text{noise}$

- Rosenbrock

$$f(x) = \sum_{j=1}^{n_x/2} [100(x_{2j} - x_{2j-1}^2)^2 + (1 - x_{2j-1})^2] \quad (14)$$

With $x_j \in [-2.048, 2.048]$ and $f^*(x) = 0.0$

- Rastrigin

$$f(x) = \sum_{j=1}^{n_x} (x_j^2 - 10 \cos(2\pi x_j) + 10) \quad (15)$$

With $x_j \in [-5.12, 5.12]$ and $f^*(x) = 0.0$.

4 Simulation Results

In this Section the imperialist competitive algorithm (ICA) is implemented with 4 benchmark mathematical functions with 30 variables by varying their parameters and the results obtained by the ICA algorithm are shown in separate tables by parameter.

The parameters used in the imperialist competitive algorithm are:

- Number of variables: 30
- Number of countries: 200
- Number of imperialist: 10
- Number of decades: 3000

Table 1 shows that after executing the ICA Algorithm 10 times, by varying the revolution parameter, we can find the best, average and worst results for the different mathematical functions benchmark.

Table 1 Results by varying the revolution parameter

Function	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Sphere	Best	7.47E-10	3.24E-09	5.61E-10	2.52E-09	6.23E-09	2.74E-09	5.07E-10
	Worst	6.69E-08	1.57E-07	1.64E-07	1.51E-07	1.39E-07	7.40E-08	1.11E-07
	Mean	2.33E-08	2.72E-08	2.54E-08	4.63E-08	3.77E-08	2.65E-08	2.52E-08
Quartic	Best	2.34E-13	1.91E-13	8.45E-12	9.30E-13	3.58E-14	5.81E-12	3.87E-13
	Worst	3.22E-09	1.99E-09	2.87E-09	3.10E-08	6.04E-10	3.84E-10	2.88E-09
	Mean	3.68E-10	4.03E-10	9.12E-10	3.26E-09	1.42E-10	9.76E-11	5.83E-10
Rosenbrock	Best	2.51	11.74	19.10	1.39E-01	2.51	15.05	6.25
	Worst	81.26	83.71	80.05	84.89	88.59	85.24	94.01
	Mean	58.40	56.13	65.16	47.70	47.52	52.81	98.58
Rastrigin	Best	2.59	1.60E-03	1.12E-02	4.50E-03	3.30E-03	1.00E-03	59.32
	Worst	72.00	73.59	74.15	74.55	74.27	72.00	9.42E-04
	Mean	29.84	44.29	37.41	40.96	38.11	26.66	2.20E-02

Table 2 Results by varying the β (Beta) parameter

Function		2	1.8	1.6	1.5	1.4	1.3	1.2	1
Sphere	Best	8.12E-09	1.45E-28	5.03E-61	4.60E-83	2.03E-86	3.60E-25	6.16E-19	7.02E-18
	Worst	1.02E-08	3.97E-25	1.32E-58	1.46E-72	6.51E-73	2.08E-17	1.86E-15	1.54E-17
	Mean	6.83E-09	1.20E-25	1.29E-56	1.46E-73	1.30E-73	1.01E-17	3.81E-16	3.30E-17
Quartic	Best	2.34E-13	4.86E-41	8.15E-90	7.76E-132	8.03E-156	5.67E-37	9.75E-27	1.99E-23
	Worst	3.22E-09	8.48E-34	5.58E-85	1.13E-117	4.48E-143	1.79E-27	4.56E-22	2.40E-20
	Mean	3.68E-10	1.70E-34	1.12E-85	1.13E-118	4.48E-144	6.49E-28	9.16E-23	4.92E-21
Rosenbrock	Best	2.52	2.50	3.10E-02	7.67E-04	4.88E-04	6.68	12.36	19.06
	Worst	83.13	73.80	13.32	3.99	9.37	74.84	28.35	73.05
	Mean	59.48	24.15	5.28	1.95	2.11	28.14	22.29	32.60
Rastrigin	Best	1.60E-03	0.00	3.98	6.96	47.76	90.54	110.44	100.50
	Worst	73.59	36.00	80.95	74.80	120.46	120.39	163.54	138.30
	Mean	44.29	18.00	42.91	44.35	90.15	97.73	139.94	121.84

Table 3 Results by varying the γ (Gamma) parameter

Function		1	0.9	0.7	0.6	0.5	0.4	0.2	0.1
Sphere	Best	6.47E-09	4.01E-09	2.95E-09	5.51E-09	5.51E-09	2.05E-09	2.06E-09	6.15E-09
	Worst	1.40E-07	2.68E-08	1.63E-08	1.21E-07	1.21E-07	7.63E-08	5.21E-08	5.13E-08
	Mean	2.96E-08	1.70E-08	8.91E-09	3.71E-08	3.71E-08	2.37E-08	1.39E-08	2.06E-08
Quartic	Best	1.46E-13	1.10E-13	3.06E-14	1.19E-12	2.34E-13	2.10E-13	1.86E-12	2.06E-14
	Worst	1.33E-10	1.49E-11	5.42E-11	1.61E-09	3.22E-09	5.64E-10	5.28E-11	2.84E-08
	Mean	3.10E-11	9.46E-12	1.37E-11	3.51E-10	3.68E-10	1.17E-10	2.48E-11	5.78E-09
Rosenbrock	Best	8.45	36.00	36.01	1.46E-01	6.50E-03	6.50E-03	3.95	25.02
	Worst	38.05	72.00	72.00	72.00	72.00	36.00	72.00	72.00
	Mean	28.35	54.07	54.00	46.82	29.23	27.70	33.19	55.41
Rastrigin	Best	2.73	2.71	36.00	9.99E-01	9.70E-03	4.79E-04	6.20E-04	3.95
	Worst	72.00	72.00	72.00	72.00	72.00	72.00	72.00	36.00
	Mean	40.58	27.74	46.82	46.90	36.28	25.44	32.38	31.70

Table 4 Results by varying the ξ (Eta) parameter

Function		0.02	0.1	0.2	0.3	0.5	0.7	0.8	0.9
Sphere	Best	1.40E-09	3.49E-09	7.79E-09	4.54E-09	2.00E-09	4.90E-09	2.73E-09	3.53E-09
	Worst	4.71E-08	3.00E-08	3.82E-08	1.74E-08	1.68E-08	1.66E-08	1.98E-08	4.49E-08
	Mean	1.80E-08	1.32E-08	2.26E-08	1.01E-08	8.46E-09	1.12E-08	9.18E-09	1.67E-08
Quartic	Best	2.34E-13	6.15E-13	9.42E-14	7.89E-14	2.25E-13	1.51E-13	2.65E-13	2.06E-13
	Worst	3.22E-09	1.08E-11	2.37E-12	2.54E-11	3.69E-11	1.12E-08	2.97E-11	3.94E-09
	Mean	3.68E-10	5.72E-12	1.35E-12	3.13E-12	6.35E-12	1.58E-09	5.68E-12	5.34E-10
Rosenbrock	Best	6.50E-03	1.17E-01	2.85E-04	3.60E + 01	4.30E-03	4.30E-04	36.00	2.81
	Worst	72.00	72.00	108.00	72.00	108.00	72.00	72.00	71.82
	Mean	29.23	46.82	47.11	47.35	36.88	34.22	47.06	47.06
Rastrigin	Best	8.45	8.92E-02	1.17E-01	1.07	2.40E-03	18.12	5.20E-03	1.70E-03
	Worst	38.05	72.00	72.00	71.82	72.00	72.00	71.82	72.00
	Mean	28.17	36.06	39.79	47.13	26.01	40.78	36.93	32.40

Table 2 shows that after executing the ICA Algorithm 10 times, by varying the β (Beta) parameter, we can find the best, average and worst results for the different mathematical functions benchmark.

Table 3 shows that after executing the ICA Algorithm 10 times, by varying the γ (Gama) parameter, we can find the best, average and worst results for the different mathematical functions benchmark.

Table 4 shows that after executing the ICA Algorithm 10 times, by varying the ξ (Eta) parameter, we can find the best, average and worst results for the different mathematical functions benchmark

5 Conclusions

By analyzing the ICA algorithm by varying the parameters, we noticed that the parameter that affected most the operation is the β parameter in the range of 1.4–1.6 good results for the sphere and quartic functions.

For the Rosenbrock and Rastrigin functions the results are not good for 30 variables, but with fewer variables the algorithm performs better.

The remaining parameters though apparently do not affect significantly the operation of the algorithm these can be in the range of 0.2–0.5 in the revolution, from 0.4 to 0.6 for γ and small values of ξ the algorithm behaves better.

Acknowledgments We would like to express our gratitude to the CONACYT and Tijuana Institute of Technology for the facilities and resources granted for the development of this research.

References

1. Atashpaz-Gargari, E., Lucas, Y.C.: Imperialist competitive algorithm: an algorithm for optimization inspired by imperialistic competition. In: Evolutionary Computation, pp. 4661–4667 (2007)
2. Banaei, M., Seyed-Shenava, S., Farahbakhsh, Y.P.: Dynamic stability enhancement of power system based on a typical unified power flow controllers using imperialist competitive algorithm. Ain Shams Eng. J. **5**(3), 691–702 (2014)
3. Dallegrave Afonso, L., Cocco Mariani, V., dos Santos Coelho, Y.L.: Modified imperialist competitive algorithm based on attraction and repulsion concepts for reliability-redundancy optimization. Expert Syst. Appl. **40**(9), 3794–3802 (2013)
4. Duan, H., Huang, Y.L.Z.: Imperialist competitive algorithm optimized artificial neural networks for UCAV global path planning. Neurocomputing **125**, 166–171 (2013)
5. Goldansaz, S.M., Fariborz, J., Zahedi Anaraki, Y.A.H.: A hybrid imperialist competitive algorithm for minimizing makespan in a multi-processor open shop. Appl. Math. Model. **37** (23), 9603–9616 (2013)
6. Hosseini, S.M., Al Khaled, Y.A.: A survey on the imperialist competitive algorithm metaheuristic: implementation in engineering domain and directions for future research. Appl. Soft Comput. J. **24**, 1078–1094 (2014)

7. Jula, A., Othman, Z., Sundararajan, Y.E.: Imperialist competitive algorithm with PROCLUS classifier for service time optimization in cloud computing service composition. *Expert Syst. Appl.* **42**(1), 135–145 (2014)
8. Melin, P., Olivas, F., Castillo, O., Valdez, F., Soria, J., Valdez, M.: Optimal design of fuzzy classification systems using PSO with dynamic parameter adaptation through fuzzy logic. *Expert Syst. Appl.* **40**(8), 3196–3206 (2012)
9. Mozafaria, A., Behzad, Y., Ayoba, A.: Optimization of adhesive-bonded fiber glass strip using imperialist competitive algorithm. *Proc. Technol.* **1**, 194–198 (2012)
10. Nourmohammadia, A., Zandiehb, M., Tavakkoli-Moghaddamca, Y.R.: An imperialist competitive algorithm for multi-objective U-type assembly line design. *J. Comput. Sci.* **4**(5), 393–400 (2012)
11. Rasul, E., Javedani Sadaei, H., Abdullah, A.H., Gani, Y.A.: Imperialist competitive algorithm combined with refined high-order weighted fuzzy time series (RHWFTS–ICA) for short term load forecasting. *Energy Convers. Manage.* **76**, 1104–1116 (2013)
12. Shamshirband, S., Amini, A., Nor Badrul, A., Mat Kiah, M.L., Ying, W.T., Furnell, Y.S.: D-FICCA: a density-based fuzzy imperialist competitive clustering algorithm for intrusion detection in wireless sensor networks. *J. Int. Measur. Confederation* **55**, 212–226 (2014)

An Improved Intelligent Water Drop Algorithm to Solve Optimization Problems

Diana Martinez and Fevrier Valdez

Abstract The intelligent water drop algorithm (IWD), it is considerably a new algorithm based on nature, it was proposed in 2007 for the travelling salesman problem, and until today is still being in numerable investigations, and applications, is this paper the aim is to show as clearly as possible how does work the algorithm, the pseudo code, flowchart and some of its applications.

1 Introduction

Intelligent water drops is known as IWD for the initials, as the ant colony algorithm (ACO) and particle swarm optimization (PSO) is base on swarms inspirited in the nature. Contrary to what would be think, in this algorithm they're not swarms of animals, they are swarm of drops, just like the other algorithms swarms, every drop is an individual with the task of finding a route that is optimal for the algorithm.

In nature water drops are observed in rivers, as humans, rivers are seen as one unit, but this is made up of millions of drops and each has a task. In this algorithm it must take into account every drop, the velocity and the amount of soil that pull along, but must take into account that the droplets can move earth from one point to another and may leave land in its path [1].

It is important to note that although there is an algorithm considered new, there are many applications where this algorithm can be used as:

- **The salesman problem,** Algorithm used to find the shortest path in a sequence of nodes, or refrain from doing a route with higher cost and repeating nodes [2].
- **Job shop scheduling,** Algorithm which is important for the conservation of resources. It is mostly used to plot optimal routes in companies where shipments of commodity are made [3].

D. Martinez · F. Valdez (✉)
Tijuana Institute of Technology, Tijuana, BC, Mexico
e-mail: fevrier@tectijuana.edu.mx

- **Multidimensional knapsack,** It is a complicated combinatorial problem that seeks to maximize profits seeking to maximize profits without exceeding the capacity of each trip [4].
- **Robot Path Planning,** It is used for creating routes with fixed obstacles in autonomous robots [5].

And many more applications, all these algorithms have in common that they seek the best possible solutions or better ways all find the optimal result, so the IWD remains under investigation to continue finding new uses, or look for new optimization and exploit its full potential.

The paper is organized as follows: in Sect. 2 a description about how the algorithm works on a very general form, in Sect. 3 the explication of the algorithm, in Sect. 4 the flowchart is a visual form of the algorithm so it would be more easy to understand, Sect. 5 the pseudo code, in Sect. 6 the experiments and Sect. 7 where the conclusions are presented.

2 How It Works

The IWD algorithm has different phases but most important in this algorithm must be know, learn about the principal properties, is the velocity and the soil can carry on. These two properties can be varied over the entire life of the algorithm [6].

There are other important parts of the algorithm, which will be listed below.

- Number of drops
- Number of nodes
- Initial speed
- Initial Earth

In IWD there are two types of parameters they are static and dynamic, as its name implies static parameters are values that do not change throughout the algorithm, in this case they are the number of nodes that are in place to the problem, and the drops of water that will be used to find an optimal result. The dynamic parameters are values that can be modified during the algorithm such as the velocity and the soil, The velocity and soil are taken by the hand, as the velocity depends on how much soil takes every drop of water and soil ends being abandoned by the drop increasing its speed but can also take more soil which would reduce their velocity along of the course, this is how these two parameters go hand by hand with each other.

3 Algorithm

This is the algorithm used for the creation of the pseudo code and the flow chart is based on the article where the IWD was initially presented and is in a very basic form so it could be easily understated. It is form with four basic parts, the begging

where the problem is established, the second one where the static parameters set, the third one is like the second one but in this case the dynamic parameters is the one set, and the last one the fourth is the end where the solution is obtained.

1. Representation of the graph, where is establish the number of nodes of the problem, is where the water drop will visit and created a route.
2. Establish the number of iterations.
3. Representation of static parameters, number of drops, initial velocity.
4. Representation of dynamic parameters, the soil and velocity would change every time the drop moves across the nodes establish.
5. The condition is set did the drop visit all the nodes if the answers is node we go to the number four.
6. Save the best result.

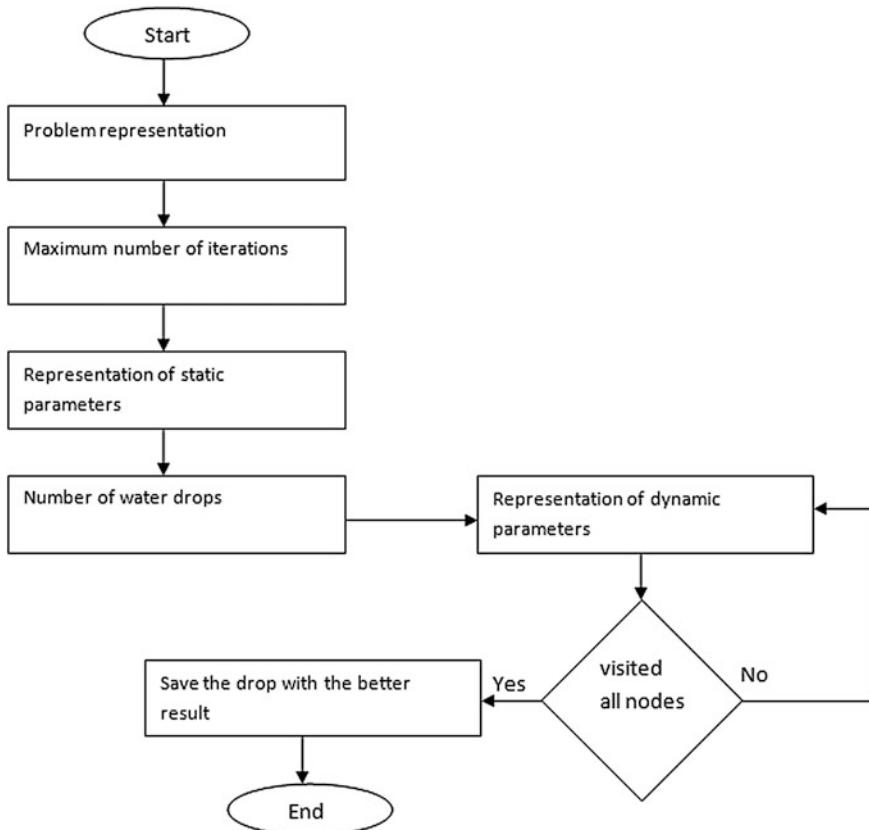


Fig. 1 Flow chart of the algorithm

4 Flowchart

Based on the original article of intelligent water drops we created a flowchart for a better representation of the algorithm [6].

In the first section of the flowchart the problem is created, is where the problem is represented in specific number of nodes.

The second part of the flowchart is where de static parameter is established, as the initial velocity and the initial soil, are the data that each drop begins with, number of drops per iteration. The third part of the flowchart is with dynamic parameters are establish is the soil and velocity but the difference is telling the algorithm how much soil with every drop whit take in the travel o how much with leave in the way, making the velocity change every time that the soil is change. In the fourth stage is where the condition are created so the algorithm can know when he needs to end with his work, if the drops has visit all the nodes it would save the best result, so it can be compare. We can observe in Fig. 1 the algorithm.

5 Pseudo Code

The pseudo code of the algorithm is as follows:

```

Begin
  Representation in the form of a graph
  Distribution of IWDs on the problem's graph
  Number of nodes
  Maxim numbers of iterations
    Representation of static parameters
    Setting values for static parameters
    Number of drops
    Initial Velocity
      Dynamic parameter initialization
      Soil
      Velocity
  Set condition
  Visited all nodes
  if no return to Dynamic parameter initialization
  if yes save the best result
  end

```

6 Experiments

Using an existent code, we took an existent program, with the IWD for the TSP (Travelling Salesman's Problem), which is called the IWD-TSP, the IWD-TSP was implemented by Visual Studio 2010 using the C# language [7].

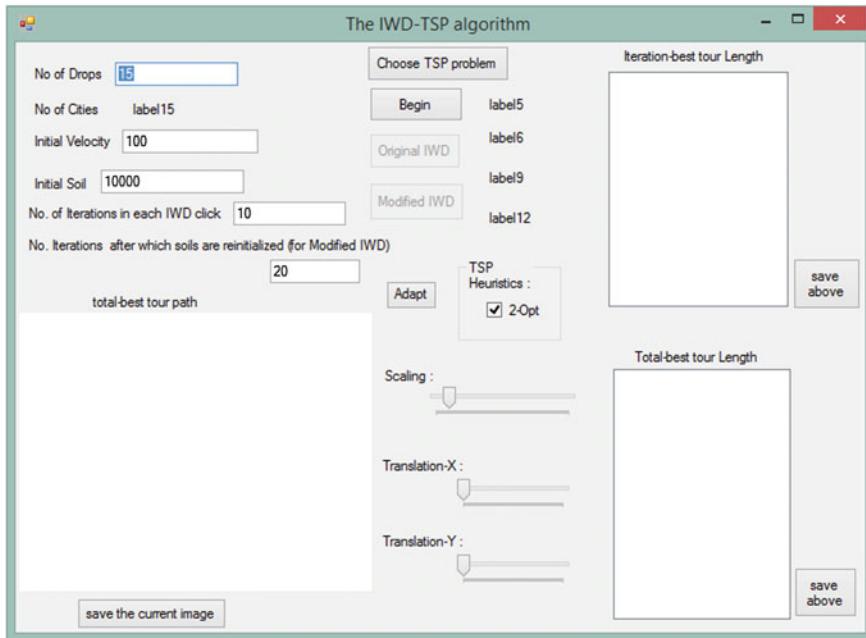


Fig. 2 The graphical interface of the used code

We can observed in Fig. 2 that this program is easy to use, we can modify the number of drops, the initial velocity, initial soil, and the number of cities can be modify with benchmarks that all ready exists. The program has default benchmarks but we use more so we can observe the behavior of the program with a big number of cities. The names of the benchmarks are list below.

- att48
- eil76
- kroA100
- d493
- rat783

The names of the benchmarks are added by the person who programmed this, always accompanied by the number of cities.

Each of the benchmarks has a best result they are listed below.

- att48 the best result 10,628
- eil76 the best result 538
- kroA100 the best result 21,282
- d493 the best result 35,002
- rat783 the best result 8806

Table 1 Results of the original method

No. of drops	No. of cities	Initial velocity	Initial soil	Best	Worst	Mean
30	48	100	10,000	33,988.807	38,371.764	46,570.715
15	76	100	10,000	573.79	617.413	594.575
20	100	100	10,000	22,931.594	24,479.56	23,845.191
50	493	100	10,000	39,489.548	41,750.364	40,790.046
80	783	150	10,000	9844.783	10,454.404	9895.283

Table 2 Results of the modify method

No. of drops	No. of cities	Initial velocity	Initial soil	Best	Worst	Mean
30	48	100	10,000	33,784.247	37,763.349	34,441.806
15	76	100	10,000	563.822	652.123	578.6062
20	100	100	10,000	22,657.693	23,090.157	22,615.8751
50	493	100	10,000	38,110.721	41,240.776	38,703.5591
80	783	150	10,000	9847.854	10,458.564	13,172.8739

At the beginning of this experiment we did not have available the data necessary to reach the optimal result set, so we had to start with trial and error experimentation, we failed to reach the optimum results that were established, but in some cases the results were very close to the optimal. In Table 1 we can see the result with the original water drops the number of water drops, number of cities, initial velocity and initial soil, that is the part that can be modify on the program, on the other side of the table we can observe the best result, the worst result and the mean of each experiment, we did 30 experiments for each benchmark.

In Table 2 we can see the result with the modify water drops the number of water drops, number of cities, initial velocity and initial soil, that is the part that can be modify on the program, on the other side of the table we can observe the best result, the worst result and the mean of each experiment, we did 30 experiments for each benchmark.

7 Conclusions

In this analysis of existing information, it was released more clearly, how this algorithm works, the algorithm was test with an existing program in which we had good results, were observed even when the number of cities increases, I was tested, the program has problems to reach an optimal solution for small cities and even this code works fine, it's remarkable that you can improve for large cities, it is as well as future work is expected to achieve a significant improvement in the existing program.

Acknowledgments We would like to express our gratitude to the CONACYT and Tijuana Institute of Technology for the facilities and resources granted for the development of this research.

References

1. Shah-Hosseini, H.: An approach to continuous optimization by the intelligent water drops algorithm. *Procedia-Soc. Behav. Sci.* **32**, 224–229 (2012). ISSN 1877-0428
2. Wang, X., Xu, G.: Hybrid differential evolution algorithm for traveling salesman problem. *Procedia Eng.* **15**, 2716 (2011)
3. Liu, Y., Yin, M., Gu, W.: An effective differential evolution algorithm for permutation flow shop scheduling problem. *Appl. Math. Comput.* **248**, 1 (2014)
4. Chih, M., Lin, C.-J., Chern, M.-S., Ou, T.-Y.: Particle swarm optimization with time-varying acceleration coefficients for the multidimensional knapsack problem. *Appl. Math. Model.* **38**(4), 15 (2014)
5. Mo, H., Xu, L.: Research of biogeography particle swarm optimization for robot path planning. *Neurocomputing* **148**, 19 (2015)
6. Shah-Hosseini, H.: The intelligent water drops algorithm: a nature-inspired swarm-based optimisation algorithm. *Int. J. Bio-Inspired Comput.* **1**(1/2), 71–79 (2009)
7. Industrial Engineering.: Faculty of Engineering, Naresuan University, Thailand (2012)

An Improved Simulated Annealing Algorithm for the Optimization of Mathematical Functions

Carolina Avila and Fevrier Valdez

Abstract In recent years the optimization methods are based on physical mimic natural biological processes among others, seek to strike a balance between exploitation and exploration of the search space to achieve good performance technical search and global optimization. In this work a simulated algorithm (SA) is described to analyze their convergence in mathematical functions, setting parameters were performed manually for Algorithm Benchmark 6 math functions, this algorithm apply high temperatures, the temperature in cooling process is reduced gradually.

1 Introduction

Since the simulated annealing algorithm developed by Scott Kirkpatrick, C. Daniel P. Gelatt and Vecchi Mario was introduced in 1983 [1, 2], has proved efficient to solve various optimization problems.

Simulated Annealing algorithm described in its original form, fit parameters to the existing literature was performed manually based algorithm has been applied to mathematics reference functions and tables showing the results Annealing optimization algorithm is presented I simulated.

The comparative study of the original algorithm with the algorithm and a fuzzy system is carried out in order to show the effectiveness of the simulated annealing algorithm address problems optimization just as effective against the diffuse test algorithm.

This paper is organized as follows: Sect. 2 describes the Simulated Annealing algorithm in Sect. 3 the description section mathematical functions, Sect. 4 the results of the simulations where you can see the modified algorithm SA and SA are described.

C. Avila · F. Valdez (✉)
Tijuana Institute of Technology, Tijuana, BC, Mexico
e-mail: fevrier@tectijuana.mx

2 Simulated Annealing

The simulated annealing algorithm appears to early 80s [3], is the resolution for the optimization of combinatorial problems with local minimum. Your approximation is to randomly generate a near current solution, based on the probability distribution proportional temperature scale solution. SA algorithm generates a new point if it is a better solution than the current point this process continues until an improvement is not generated, the algorithm to avoid being trapped in local minima not necessarily accept the overall allows elevating the objective points observed that local-search algorithms could often obtain better final objective criterion values if they were permitted to periodically accept solutions with worse criterion values [4].

The algorithm SA been used in different areas of research for solving different problems then it makes mention of some: Simulated annealing for the unconstrained quadratic pseudo-Boolean function [3], A medical diagnostic tool based on radial basis function classifiers and evolutionary simulated annealing [5], An approach based on simulated annealing to optimize the performance of extraction of the flower region using mean-shift segmentation [6], A simulated annealing algorithm for sparse recovery by l0 minimization Neurocomputing [7], An efficient simulated annealing algorithm for design optimization of truss structures [8].

Transition probability

Is the probability that the simulated annealing algorithm will move randomly indicate your current position, your changes will be accepted with probability p , [2] is determined by

$$p = e^{-\frac{\Delta E}{k_B T}} \quad (1)$$

Accepting a change to a random number r as a threshold is used, is determined by

$$p = e^{-\frac{\Delta f}{T}} > r \quad (2)$$

Temperature

The concept of TC in SA is analogous to the freezing point of a liquid, i.e., the temperature at which the liquid undergoes a phase change to form a solid [9]. The temperature is a parameter of the simulated annealing algorithm, the initial temperature T_0 defines when you start the cooling process to control general search results in each dimension is used to limit the scope of the search in that dimension, you must select an initial temperature adequate to prevent the algorithm from being trapped in a local minimum must be high enough to allow the system to have a high degree of freedom for changing neighboring solutions.

Step Walk

The random steps are widely used for randomization and local search in different algorithms, it is important to consider an appropriate step size which experienced search.

If the search step is too large, then the new solution x generated will be too far from the previous solution. Then, the measure is unlikely to be accepted. If the search step is too small, the change is too small to be significant, and therefore such a search is not efficient.

2.1 Pseudo Code for Simulated Annealing Algorithm

```

Objective function  $f(x); x = (x_1, \dots, x_p)^T$ 
Initialize initial temperature  $T_0$  and initial guess  $X^{(0)}$ 
Set final temperature  $T_f$  and max number of iterations  $N$ 
Define cooling schedule  $T \rightarrow \alpha T, (0 < \alpha < 1)$ 
While ( $T > T_f$   $\wedge n < N$ )
    Move randomly to new locations:  $x_{n+1} = x_n + \epsilon$  (random walk)
    Calculate  $\Delta f = f_{n+1}(x_{n+1}) - f_n(x_n)$ 
    Accept the new solution if better
        if not improved
            Generate a random number  $r$ 
            Accept if  $p = \exp[-\Delta f / T] > r$ 
        end if
        Update the best  $x_*$  y  $f_*$ 
     $n = n + 1$ 
end while

```

3 Benchmark Mathematical Functions

In this section the functions used to check the effectiveness of the algorithm SA is a series of tests were performed to estimate the processing time taken for the execution of the algorithm.

In the area of optimization Have Been used in different mathematical functions work Which Mentioned below: Optimal design of fuzzy classification systems using PSO With dynamic parameter adaptation-through fuzzy logic [10], A new gravitational search algorithm using fuzzy logic to Parameter adaptation [11], Parallel Particle Swarm Optimization With Parameters Adaptation Using Fuzzy Logic [12, 13].

The mathematical functions are shown below:

- **Ackley**

$$f(x) = \sum_{i=1}^n \frac{x_i^2}{4000} - \prod_{i=1}^n \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1 \quad (3)$$

- **Griewank**

$$(x) = \sum_{i=1}^n \frac{x_i^2}{4000} - \prod_{i=1}^n \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1 \quad (4)$$

- **Rastrigin**

$$f_2(x) = \sum_{i=1}^{n-1} 100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2 \quad (5)$$

- **Rosenbrock**

$$f_2(x) = \sum_{i=1}^{n-1} 100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2 \quad (6)$$

- **Shubert**

$$f(x) = \left(\sum_{i=1}^5 i \cos((i+1)x_1 + i) \right) \left(\sum_{i=1}^5 i \cos((i+1)x_2 + i) \right) \quad (7)$$

- **Sphere**

$$f(x) = \sum_{i=1}^n 2_i^2 \quad (8)$$

4 Fuzzy System Design

Design of the input variable can be seen in Fig. 1 showing iteration as input, three granulated triangular membership functions.

The fuzzy system has iteration as input, as shown in Fig. 2.

To the output variable, values between 0 and 0.001 so that the output variables were designed using this range are used. Granulated output three triangular membership functions, design variables can be seen in Fig. 3.

The rules for the fuzzy system are show in Fig. 4.

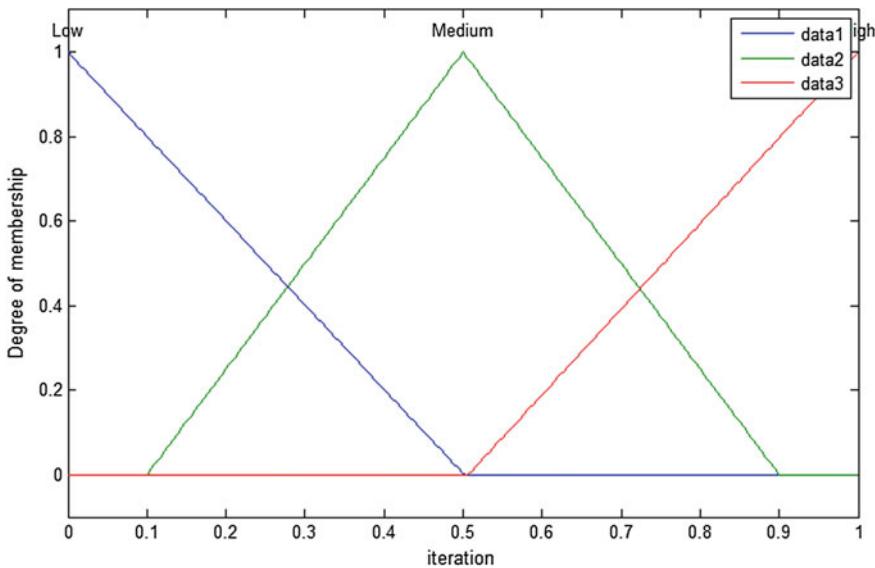


Fig. 1 Input 1: iteration

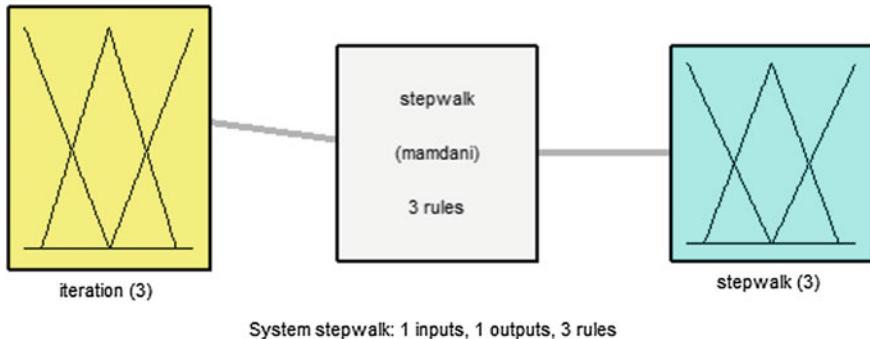


Fig. 2 Fuzzy system

5 Simulation Results

Comparison algorithm Simulated Annealing with the modification of search step parameter, each algorithm is presented 6 math functions were integrated in this section Benchmark was performed separately in different sizes 2, 4, 8, 16, 32, were handled 30 tests for each function.

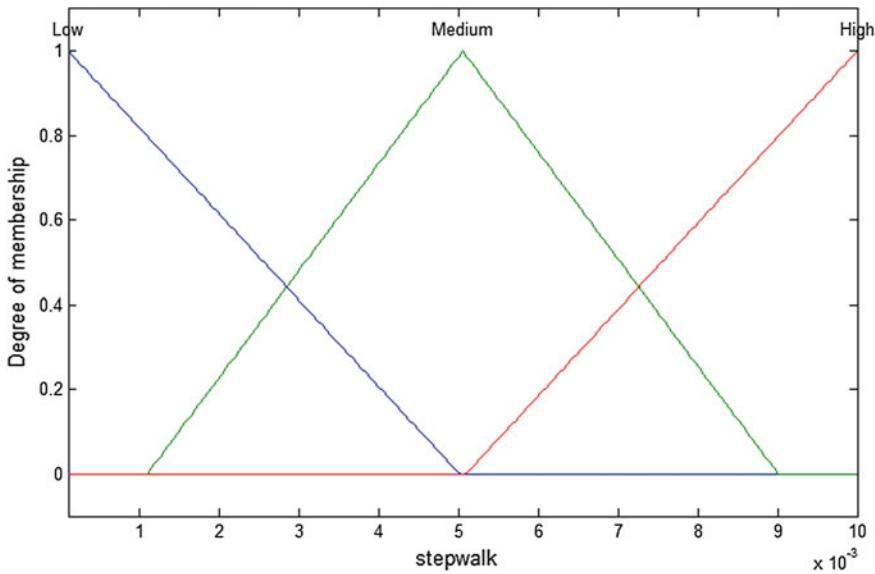


Fig. 3 Output: stepwalk

1. If (iteration is Low) then (stepwalk is Low) (1)
2. If (iteration is Medium) then (stepwalk is Medium) (1)
3. If (iteration is High) then (stepwalk is High) (1)

Fig. 4 Rules for fuzzy system

The parameters in the simulated annealing algorithm are:

- Initial temperature = 1,0
- Final stopping temperature = 1e-10
- Min value of the function = 0
- Maximum number of rejections = 1000
- Maximum number of runs = 500
- Maximum number of accept = 400
- Initial search period = 300

5.1 Simulation Results with Bat Algorithm

In this section the results obtained by the simulated annealing algorithm in separate tables of mathematical functions.

Table 1 Simulation results for the Ackley function

Dimension	Best	Worst	Average
2	5.25128E-06	7.18095165	2.193481482
4	0.000139	5.14199	3.389519
8	3.12685	7.590204	5.733816
16	3.529264	8.861	6.803242
32	3.815369	9.498121	6.680796

From Table 1 it can be appreciated that after executing the Simulated Annealing Algorithm 30 times, with different dimension, we can see the best, average and worst results for the Ackley function.

From Table 2 it can be appreciated that after executing the Simulated Annealing Algorithm 30 times, with different dimension, we can see the best, average and worst results for the Griewank function.

From Table 3 it can be appreciated that after executing the Simulated Annealing Algorithm 30 times, with different dimension, we can see the best, average and worst results for the Rastrigin function.

From Table 4 it can be appreciated that after executing the Simulated Annealing Algorithm 30 times, with different dimension, we can see the best, average and worst results for the Rosenbrock function.

Table 2 Simulation results for the Griewank function

Dimension	Best	Worst	Average
2	8.14577E-09	1.588546291	0.406811927
4	0.028927	2.316615	0.721329
8	0.117571	0.632723	0.401886
16	0.208112	0.688871	0.464005
32	0.313993	0.746776	0.527937

Table 3 Simulation results for the Rastrigin function

Dimension	Best	Worst	Average
2	3.36193E-11	4.974790251	1.663796622
4	3.11147E-11	4.974790248	1.504588318
8	1.11E-09	4.97479	1.855366
16	1.59528E-10	3.979831191	1.361478415
32	1.659E-11	3.9798312	1.4438547

Table 4 Simulation results for the Rosenbrock function

Dimension	Best	Worst	Average
2	5.02852E-12	8.96442E-10	3.25805E-10
4	4.10874E-06	163.5953252	37.64477198
8	6.79152E-05	0.000410012	0.000276813
16	0.001824	0.006152	0.003862
32	0.019168	0.134838	0.046564

Table 5 Simulation results for the Sphere function

Dimension	Best	Worst	Average
2	8.27898E-15	2.24479E-11	5.74437E-12
4	3.01018E-09	1.02496E-08	6.73109E-09
8	5.75131E-07	4.175587223	2.191320478
16	3.156019	8.021052	5.761168
32	5.499466	13.03222	10.15328

From Table 5 it can be appreciated that after executing the Simulated Annealing Algorithm 30 times, with different dimension, we can see the best, average and worst results for the Sphere function.

The results obtained with the Shuber function, we can see that after being executed 30 times the simulated annealing algorithm with 2 dimensions, could appreciate the following results: best—186.7308, worst—176.2034 and average—123.5715.

5.2 *Simulation Results with Simulated Annealing Algorithm with the Modification of Search Step*

In this section we show the results obtained by the genetic algorithm in separate tables of the math functions.

In Table 6 we can see that after running the Simulated Annealing algorithm 30 times, with different dimensions, we can see the best, worst and average results for the Ackley function.

In Table 7 we can see that after running the Simulated Annealing algorithm 30 times, with different dimensions, we can see the best, worst and average results for the Griewank function.

Table 6 Simulation results for the Ackley function

Dimension	Best	Worst	Average
2	1.1689E-05	0.00022057	0.00010497
4	0.00153	0.005424	0.003098
8	0.008422	0.020757	0.016022
16	0.023069	0.047725	0.038256
32	0.492734801	7.634892916	0.869987315

Table 7 Simulation results for the Griewank function

Dimension	Best	Worst	Average
2	1.78E-05	0.002682	0.000748
4	0.021917	0.132047	0.083161
8	0.31617	0.630594	0.463451
16	0.279919	0.647204	0.459059
32	0.250022	0.71744	0.51958

In Table 8 we can see that after running the Simulated Annealing algorithm 30 times, with different dimensions, we can see the best, worst and average results for the Rastrigin function.

In Table 9 we can see that after running the Simulated Annealing algorithm 30 times, with different dimensions, we can see the best, worst and average results for the Rosenbrock function.

In Table 10 we can see that after running the Simulated Annealing algorithm 30 times, with different dimensions, we can see the best, worst and average results for the Sphere function.

The results obtained with the Shuber function, we can see that after being executed 30 times the simulated annealing algorithm with 2 dimensions, could appreciate the following results: best—186.7309, worst—184.6257 and average—123.5767.

Table 8 Simulation results for the Rastrigin function

Dimension	Best	Worst	Average
2	8.12E-08	4.974795	1.835181
4	2.06E-06	4.97479	1.741946
8	1.45E-06	7.959668	1.695479
16	2.09E-06	4.974794	1.51286
32	8.4E-08	4.974793	1.570217

Table 9 Simulation results for the Rosenbrock function

Dimension	Best	Worst	Average
2	1.56E-08	4.92E-07	1.69E-07
4	0.01698	78.16866	7.832568
8	0.038923	4.130628	0.254744
16	1.439153	7.079002	3.595138
32	17.29083	82.81296	36.38108

Table 10 Simulation results for the Shere function

Dimension	Best	Worst	Average
2	3.8E-11	7.8E-09	1.79E-09
4	1.46E-07	3.83E-06	1.5E-06
8	4.19E-05	0.000156	9.85E-05
16	0.000714	0.001603	0.001208
32	0.005706	0.010623	0.0083

6 Conclusions

In the simulation analysis for the comparative study of genetic algorithms and the effectiveness of the bat algorithm, bat algorithm during the modification of parameters by trial and error is shown, the rate of convergence of the genetic algorithm is much faster, the comparison was made with the original versions of the two algorithms with the recommended literature parameters, this conclusion is based on 6 Benchmark math functions results may vary according to mathematical or depending on the values set in the parameters of the algorithm.

The application of various problems in bat algorithm has a very wide field where the revised items its effectiveness is demonstrated in various applications, their use can be mentioned in the processing digital pictures, search for optimal values, neural networks, and many applications.

Acknowledgments We would like to express our gratitude to the CONACYT and Tijuana Institute of Technology for the facilities and resources granted for the development of this research.

References

1. Jeong, S.-J., Kim, K.-S., Lee, Y.-H.: The efficient search method of simulated annealing using fuzzy logic controller. *Expert Syst. Appl.* **36**(3/2), 7099–7103 (2009). ISSN 0957-4174. <http://dx.doi.org/10.1016/j.eswa.2008.08.020>
2. Yang, X.: A new metaheuristic bat-inspired algorithm. Department of Engineering, University of Cambridge, Trumpington Street, Cambridge CB2 1PZ, UK (2010)
3. Alkhamis, T.M., Hasan, M., Ahmed, M.A.: Simulated annealing for the unconstrained quadratic pseudo-Boolean function. *Eur. J. Oper. Res.* **108**(3), 641–652 (1998). ISSN 0377-2217
4. Brusco, M.J.: A comparison of simulated annealing algorithms for variable selection in principal component analysis and discriminant analysis. *Comput. Stat Data Anal.* **77**, 38–53 (2014). ISSN 0167-9473
5. Alexandridis, A., Chondrodima, E.: A medical diagnostic tool based on radial basis function classifiers and evolutionary simulated annealing. *J. Biomed. Inf.* **49**, 61–72 (2014). ISSN 1532-0464. <http://dx.doi.org/10.1016/j.jbi.2014.03.008>
6. Bahadir, K.: An approach based on simulated annealing to optimize the performance of extraction of the flower region using mean-shift segmentation, *Appl. Soft Comput.* **13**(12), 4763–4785 (2013). ISSN 1568-4946
7. Du, X., Cheng, L., Chen, D.: A simulated annealing algorithm for sparse recovery by l0 minimization. *Neurocomputing* **131**, 98–104 (2014). ISSN 0925-2312
8. Lamberti, L.: An efficient simulated annealing algorithm for design optimization of truss structures. *Comput. Struct.* **86**(19–20), 1936–1953 (2008). ISSN 0045-7949. <http://dx.doi.org/10.1016/j.compstruc.2008.02.004>
9. Tavares, R.S., Martins, T.C., Tsuzuki, M.S.G.: Simulated annealing with adaptive neighborhood: a case study in off-line robot path planning. *Expert Syst. Appl.* **38**(4), 2951–2965 (2011). ISSN 0957-4174
10. Melin, P., Olivas, F., Castillo, O., Valdez, F., Soria, J., Garcia, J.: Optimal design of fuzzy classification systems using PSO with dynamic parameter adaptation through fuzzy logic. *Expert Syst. Appl.* **40**(8), 3196–3206 (2013)

11. Sombra, A., Valdez, F., Melin, P., Castillo, O.: A new gravitational search algorithm using fuzzy logic to parameter adaptation. In: IEEE Congress on Evolutionary Computation, pp. 1068–1074 (2013)
12. Valdez, F., Melin, P., Castillo, O.: Evolutionary method combining particle swarm optimization and genetic algorithms using fuzzy logic for decision making. In: Proceedings of the IEEE International Conference on Fuzzy Systems, pp. 2114–2119 (2009)
13. Valdez, F., Melin, P., Castillo, O.: Parallel particle swarm optimization with parameters adaptation using fuzzy logic. MICAI 2 374–385 (2012)

Optimization of Reactive Fuzzy Controllers for Mobile Robots Based on the Chemical Reactions Algorithm

**David de la O, Oscar Castillo, Abraham Meléndez, Patricia Melin,
Leslie Astudillo and Coral Sánchez**

Abstract This paper describes the optimization of a reactive controller for a mobile autonomous robot using the Chemical Reaction Algorithm (CRA) method to adjust the parameters of the membership functions of the fuzzy controller. The objective of reactive control is to apply the same ability that humans have when they are driving, i.e. to react to unexpected situations, traffic jams, traffic lights, etc. The CRA is a metaheuristic strategy that performs a stochastic search for optimal solutions, every solution is represented as an element, and the fitness of the element is evaluated in accordance with the objective function.

1 Introduction

A scenario characterized by both real and virtual, dynamic environment provides facilities to test different techniques of Artificial Intelligence (AI) with multiple autonomous robots, and these techniques include: Design of autonomous agents, multi-agent collaboration, strategy acquisition, real-time reasoning, learning, robotics and sensor fusion, among others. Other points constitutes the intentions,

D. de la O · O. Castillo · A. Meléndez · P. Melin (✉) · L. Astudillo · C. Sánchez
Tijuana Institute of Technology, Tijuana, Mexico
e-mail: pmelin@tectijuana.mx

D. de la O
e-mail: ddsh@live.com

O. Castillo
e-mail: ocastillo@tectijuana.mx

A. Meléndez
e-mail: abraham.ms@gmail.com

L. Astudillo
e-mail: leslie.astudillo@suntroncorp.com

C. Sánchez
e-mail: es.coral@gmail.com

plans and reasoning of autonomous agent to achieve its goal, robot navigation techniques and associated systems are key to the performance of autonomous robot agent.

Robotics is a science that is used in the design and construction of robots designed to perform a set of automated, precise and controlled tasks. Autonomous robotics in particular, is a sub-discipline of robotics focused on conventionally giving the robot a certain level of artificial intelligence that allows it to perform without human intervention. An autonomous robot is composed of a mechanical-electronic structure based on sensors and actuators and a controller responsible for managing them.

The key to autonomous robots is presented with the understanding that the controller, which allows, through the analysis of the signals obtained from the environment, plan their actions towards achieving the goals set for its design and conception. If an autonomous robot can be incorporated within the actuator assembly, a mechanical system of controlled movement, we would be in a system called Autonomous Mobile Robot.

One of the applications of fuzzy logic is the design of fuzzy control systems. The success of this control lies in the correct selection of the parameters of the fuzzy controller; it is here where the Chemical Reaction Algorithm (CRA) metaheuristics is applied, and this approach is based on a static population of metaheuristic which applies an abstraction of chemical reactions. Recent research on optimized membership functions have shown good results [1–31].

The present research work deals with the problem of mobile robot autonomy, for which a reactive controller for autonomous mobile robots navigation is developed using fuzzy logic and the CRA.

This paper is organized into four sections as follows: Sect. 2 describes the main problem, and progress of the research work is shown. Sections 3 and 4 describe the theory underlying the present work, in which issues such as fuzzy logic, CRA algorithm and a bit on the operation of autonomous mobile robot are discussed. Section 5 shows the results of the simulations.

2 Mobile Robot

The particular mobile robot considered in this work is based on the description of the Simulation toolbox for mobile robots [32], which assumes a wheeled mobile robot consisting of one conventional, steered, un-actuated and not-sensed wheel, and two conventional, actuated, and sensed wheels (conventional wheel chair model). This type of chassis provides two DOF (degrees of freedom) locomotion by two actuated conventional non-steered wheels and one un-actuated steered wheel. The Robot (see Fig. 1) has two degrees of freedom (DOFs): y-translation and either x-translation or z-rotation [5, 15–20].

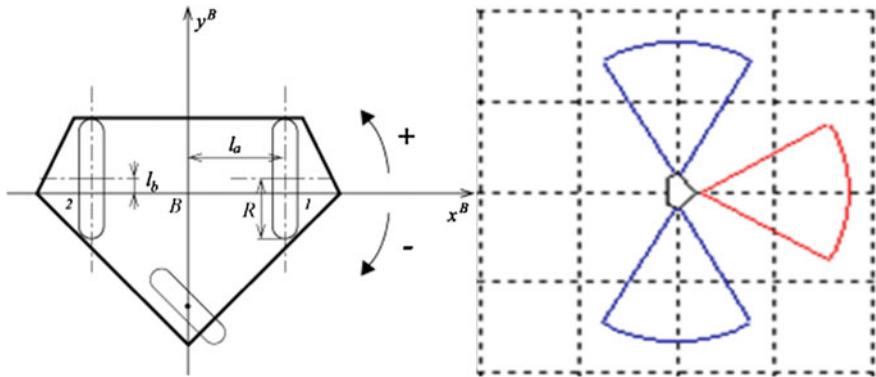


Fig. 1 Kinematic coordinate system assignments [17]

The kinematic equations of the mobile robot are defined as follows:

Equation 1: The sensed forward velocity solution [15]

$$\begin{pmatrix} V_{B_x} \\ V_{B_y} \\ \omega_{B_z} \end{pmatrix} = \frac{R}{2l_a} \begin{bmatrix} -l_b & l_b \\ -l_a & -l_a \\ -1 & -1 \end{bmatrix} \begin{pmatrix} \omega_{W_1} \\ \omega_{W_2} \end{pmatrix} \quad (1)$$

Equation 2: The Actuated Inverse Velocity Solution [15]

$$\begin{pmatrix} \omega_{W_1} \\ \omega_{W_2} \end{pmatrix} = \frac{1}{R(l_b^2 + 1)} \begin{bmatrix} -l_a l_b & -l_b^2 - 1 & -l_a \\ l_a l_b & -l_b^2 - 1 & l_a \end{bmatrix} \begin{pmatrix} V_{B_x} \\ V_{B_y} \\ \omega_{B_z} \end{pmatrix} \quad (2)$$

Under the Metric system are defined as:

- V_{B_x}, V_{B_y} Translational velocities [$\frac{m}{s}$],
- ω_{B_z} Robot z-rotational velocity [$\frac{rad}{s}$],
- $\omega_{W_1}, \omega_{W_2}$ Wheel rotational velocities [$\frac{rad}{s}$],
- R Actuated wheel radius [m],
- l_a, l_b Distances of the wheels from robot's axes [m].

3 The Chemical Optimization Paradigm

The proposed chemical reaction algorithm is a metaheuristic strategy that performs a stochastic search for optimal solutions within a defined search space. In this optimization strategy, every solution is represented as an element (or compound), and the fitness or performance of the element is evaluated in accordance with the objective function.

This algorithm has the advantage of not having external parameters (kinetic/potential energies, mass conservation, thermodynamic characteristics, etc.) as occurs in other optimization algorithms, this is a very straight forward methodology that takes the characteristics of the chemical reactions (synthesis, decomposition, substitution and double-substitution) to find the optimal solution [2, 21, 33, 34].

This approach is based on a static population metaheuristic, which applies an abstraction of chemical reactions as intensifiers (substitution, double substitution reactions) and diversifying (synthesis, decomposition reactions) mechanisms. The elitist reinsertion strategy allows the permanence of the best elements and thus the average fitness of the entire element pool increases with each iteration. Figure 2 shows the flowchart of the algorithm.

The main components of this chemical reaction algorithm are described below.

The synthesis and decomposition reactions are used to diversify the resulting solutions; these procedures show to be highly effective to rapidly lead the results to a desired value. They can be described as follows.

The single and double substitution reactions allow the algorithm to search for optima around a good previously found solution and they're described as follows.

The algorithm works mainly with chemical reactions with a change in the at least one substance (element or possible solution), that changes its composition and property sets, such reactions are classified into 4 types, which are described below:

3.1 Type 1: Combination Reactions

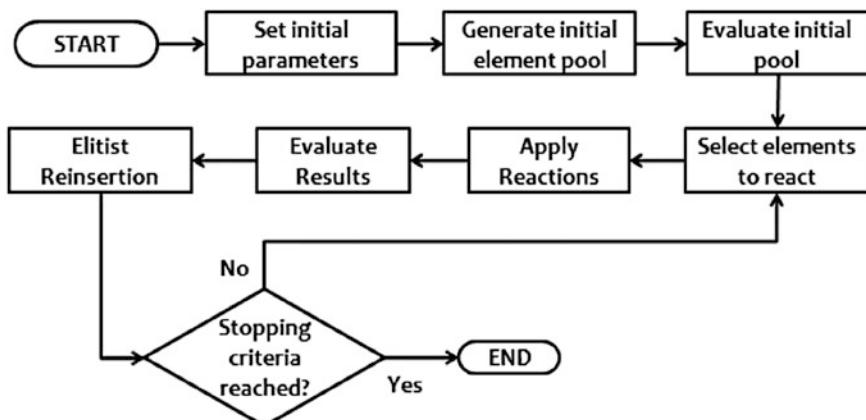


Fig. 2 General flowchart of the chemical reaction algorithm

A combination reaction is a reaction of two reactants to produce one product. The simplest combination reactions are the reactions of two elements to form a compound. After all, if two elements are treated with each other, they can either react or not.

3.2 Type 2: Decomposition Reactions



The second type of simple reaction is decomposition. This reaction is also easy to recognize. Typically, only one reactant is given. A type of energy, such as heat or electricity, may also be indicated. The reactant usually decomposes to its elements, to an element and a simpler compound, or to two simpler compounds.

Binary compounds may yield two elements or an element and a simpler compound. Ternary (three-element) compounds may yield an element and a compound or two simpler compounds (see Fig. 3).

3.3 Type 3: Substitution Reactions



Elements have varying abilities to combine. Among the most reactive metals are the alkali metals and the alkaline earth metals. On the opposite end of the scale of reactivities, among the least active metals or the most stable metals are silver and gold, prized for their lack of reactivity. Reactive means the opposite of stable, but means the same as active.

When a free element reacts with a compound of different elements, the free element will replace one of the elements in the compound if the free element is

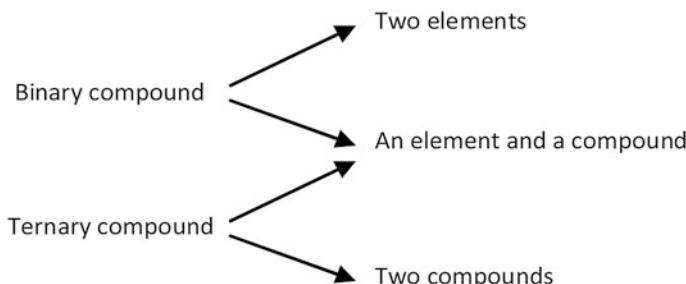


Fig. 3 Decomposition possibilities

more reactive than the element it replaces. In general, a free metal will replace the metal in the compound, or a free nonmetal will replace the nonmetal in the compound. A new compound and a new free element are produced.

3.4 Type 4: Double-Substitution Reactions



Double-substitution or double-replacement reactions, also called double-decomposition reactions or metathesis reactions, involve two ionic compounds, most often in aqueous solution [14].

In this type of reaction, the cations simply swap anions. The reaction proceeds if a solid or a covalent compound is formed from ions in solution. All gases at room temperature are covalent. Some reactions of ionic solids plus ions in solution also occur. Otherwise, no reaction takes place.

Just as with replacement reactions, double-replacement reactions may or may not proceed. They need a driving force.

In replacement reactions the driving force is reactivity; here it is insolubility or covalence.

At first sight, chemical theory and definitions may seem complex and none or few related to optimization theory, but only the general schema will be considered here to focus on the control application.

The steps to consider in this optimization method are illustrated in Fig. 4 and given as follows.

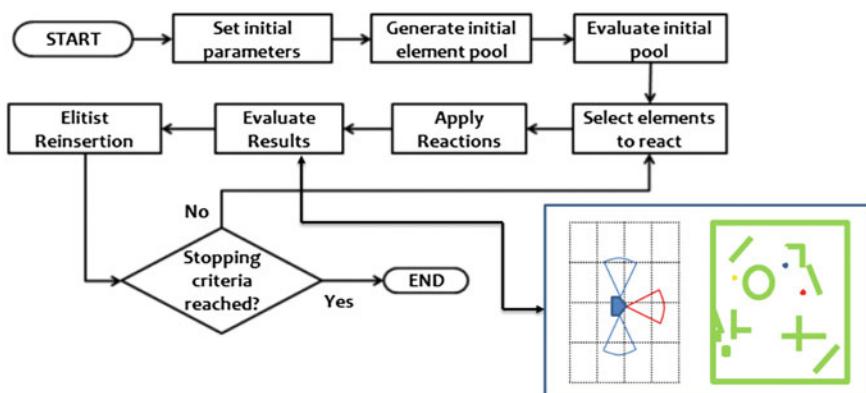


Fig. 4 Chemical reactions algorithm process

1. First, we need to generate an initial pool of elements/compounds.
2. Once we have the initial pool, we have to evaluate it.
3. Based on the previous evaluation, we will select some elements/compounds to “induce” a reaction.
4. Given the result of the reaction, we will evaluate the obtained elements/compounds.
5. Repeat the steps until the algorithm meets the criteria (desired result or maximum number of iterations is reached) [2, 21, 33, 34].

4 Experimental Approach

The Chemical Reaction Algorithm (CRA) is used to find the best fuzzy reactive controllers, which help the navigation system of the robot.

The purpose of using a metaheuristic strategy is to find the best possible controllers and this can be obtained using the CRA (see Fig. 4), as it searches along the solution space, combining the attributes from the best controllers in generating new ones. The idea is to optimize the parameters in the Membership Functions.

The CRA module is responsible for the initialization of the parameters and elements, select the elements and chemical reactions applied, evaluates the results, using the simulation performs an elitist reinsertion and put them back in the population.

4.1 Element Encoding Approach

The elements consist of 25 real-valued vectors, representing the parameters for the membership functions; we use 5 triangular membership functions for each variable, it is represented with 5 real valued for each linguistic variable, of is form the membership functions are symmetric, we take as a basic the works [5, 16–20, 32, 35]. This encoding is shown Fig. 5.

The role of reactive control is to apply the same ability that we use when we are driving, i.e. to react to unexpected situations, traffic jams, traffic lights, etc., but in a more basic concept. The objective is to guide the robot through the maze avoiding any collision. It is our goal to optimize the robot to find the exit of the maze, we used a maze to optimize the reactive control because the feature that gives the simulation, i.e. it is a closed space in which the robot cannot move easily and each wall is considered an obstacle to the robot to avoid while moving. The FIS is of Mamdani type; each consisting of 3 inputs is the distances obtained by the sensors of the robots described in Sect. 2, and 2 outputs that control the speed of the servo motors on the robot, and all this information is encoded in each element.

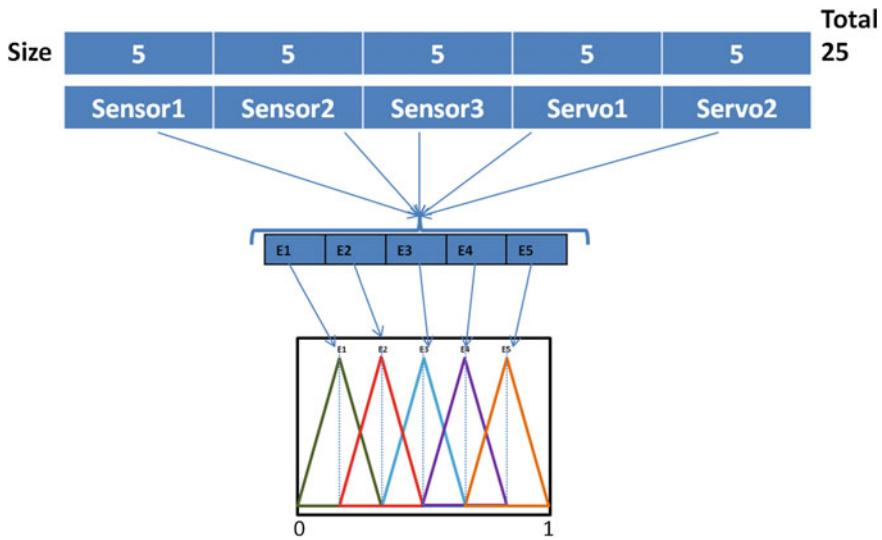


Fig. 5 Element encoding reactive controller

4.2 Objective Function

The CRA will be generating elements that will need to be evaluated and assigned a crisp value that will represent the controller performance on each of the criteria that we want to improve. For this, we need to provide the CRA with a good evaluation scheme that will penalize unwanted behaviors and reward with higher fitness values those individuals that provide the performance we are looking for in the controller; if we fail to provide a proper evaluation method we can guide the population to suboptimal solutions or non solution at all [5, 16–20, 36–38].

4.3 Reactive Controller Objective Function

The criteria used to measure the Reactive controller performance takes into account the following [5, 16–20, 39, 40]:

- Covered Distance
- Time used to cover the distance
- Battery life.

A Fitness FIS will provide the desired fitness value, adding very basic rules that reward the controller that provided the longer trajectories and smaller times and higher battery life. This seems like a good strategy that will guide the control population into evolving and provide the optimal control, but this strategy on its

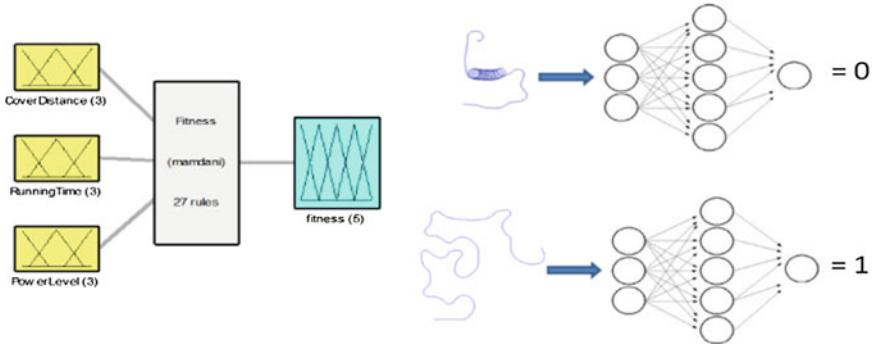


Fig. 6 Fitness function for the reactive controller

own it's not capable of doing just that, it needs to have a supervisor on the robots trajectory to make sure is a forward moving trajectory and that they don't contain looping parts, For this, a Neural Network (NN), is used to detect cycle trajectories that don't have the desired forward moving behavior by giving low activation value and higher activation values for the ones that are cycle free. The NN has two inputs and one output, and 2 hidden layers, see Fig. 6.

The evaluation method for the reactive controller has integrated both parts the FIS and the NN, where the fitness value for each individual is calculated with Eq. 3, based on the response of the NN the peak activation value is set to 0.35, this meaning that any activation lower than 0.35 will penalize the fitness given by the FIS [5, 16–20].

$$f(i) = \begin{cases} fv * nnv, & nnv < 0.35 \\ fv, & nnv \geq 0.35 \end{cases} \quad (7)$$

where

- fi Fitness value of the i th individual,
- fv Crisp value out of the fitness FIS,
- nnv Looping trajectory activation value.

4.4 Fuzzy Rules

We are not optimizing rules, therefore, a complete rule-base is considered. A rule-base is considered complete when all possible combinations of input membership functions of all the input variables participate in fuzzy rule-base formation. With Eq. 8 we obtain the size of the fuzzy rules population, where m , n , p represent the maximum number of membership functions we allow for the input variables, and k

is the maximum number of membership function for the output variables, in our case ($m = n = p = k = 5$). Equation 8 gives a total of 625 fuzzy rules subsets [41].

$$rp = m * n * p * k \quad (8)$$

5 Simulation Results

This section presents the results of experiments performed for the reactive controller.

The tools that are used to perform the experiments are Matlab and the Simrobot simulation tool.

In this section, we show tests of reactive controller, which includes the creation of a CRA algorithm to optimize the controller. The fitness of each controller is determined by their performance in the simulation tool. The robot should react in a closed environment (maze), avoiding obstacles present (walls) and must perform movements and avoid repeated.

Table 1 shows the configuration of CRA and displays the results of the approach, where we have the fitness value obtained in each experiment. It also shows the mean, variance, the best and worst value obtained.

Figure 7 shows the convergence of chemical reaction algorithm along some of the experiments. We can see that the algorithm converges slowly or falls into local maxima.

Table 1 Summary of reactive control results

Element	Iteration	RateSynt	RateDeco	RateSubst	RsteDobSubst
20	3000	0.2	0.2	0.2	0.2
	Rank	Fitness		Rank	Fitness
	1	0.20719		11	0.2034
	2	0.10004		12	0.1942
	3	0.18247		13	0.20435
	4	0.20541		14	0.20719
	5	0.16627		15	0.20681
	6	0.1813		16	0.20278
	7	0.18152		17	0.20316
	8	0.2049		18	0.20609
	9	0.2047		19	0.20719
	10	0.204		20	0.19664
				Best	0.20719
				Poor	0.10004
				Average	0.19025
				Std Dev	0.02885

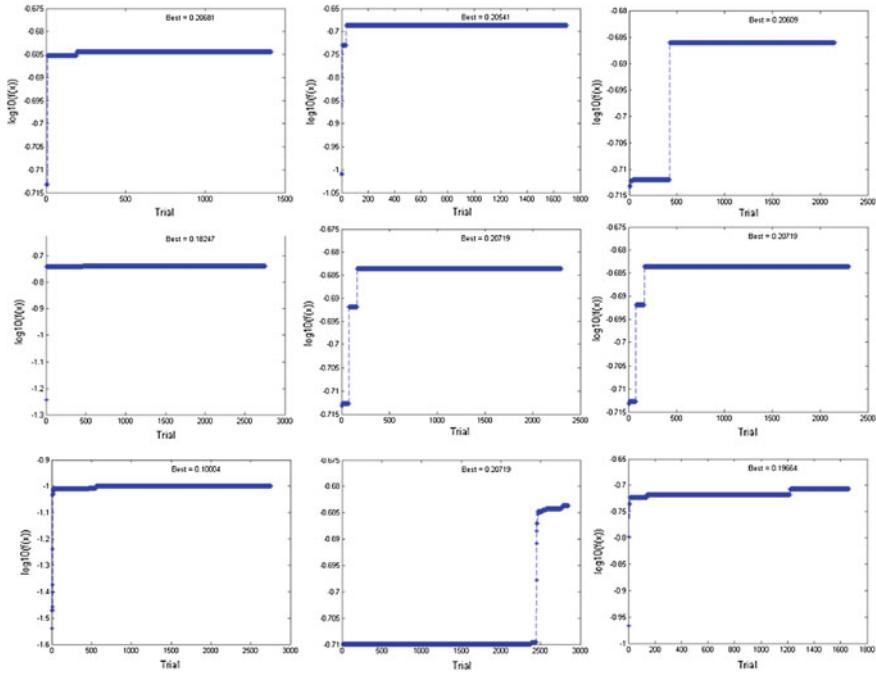


Fig. 7 Best simulation of experiments a with the chemical optimization algorithm

6 Conclusions

In this paper, the CRA algorithm is used to tune the parameters of the fuzzy controller for the Reactive Controller of a Mobile Robot.

Until now we have achieved the goal of avoiding the sliding block of the robot for period of time, even working on achieving complete the course. Based on the experiments we observed that the algorithm converges slowly and therefore the expected results are not obtained; we think the optimization of the rules as a possible solution. We propose the following improvements to the algorithm: (1) Using fuzzy control to ensure that the parameters of the reactions in dynamic change in each iteration. (2) Modify the operations of the reactions, which utilize a single random value to generate a possible solution, and we propose using a table of random numbers, equal in size to the vector of the possible solution.

Acknowledgments We would like to express our gratitude to CONACYT, and Tijuana Institute of Technology for the facilities and resources granted for the development of this research.

References

1. Astudillo, L., Castillo, O., Aguilar L., Martínez, R.: Hybrid control for an autonomous wheeled mobile robot under perturbed torques. In: Melin, P., Castillo, O., Aguilar, L., Kacprzyk, J., Pedrycz, W. (eds.) *Foundation of Fuzzy Logic and Soft Computing*, vol. 4529, pp. 594–603. Springer, Berlin (2007)
2. Astudillo, L., Melin, P., Castillo, O.: Nature optimization applied to design a type-2 fuzzy controller for an autonomous mobile robot. In: 2012 Fourth World Congress on Nature and Biologically Inspired Computing (NaBIC) (2012)
3. Bingül, Z., Karahan, O.: A fuzzy logic controller tuned with PSO for 2 DOF robot trajectory control. *Expert Syst. Appl.* **38**(1), 1017–1031 (2011)
4. Cervantes, L., Castillo, O.: Design of a fuzzy system for the longitudinal control of an F-14 airplane. In: Castillo, O., Kacprzyk, J., Pedrycz, W. (eds.) *Soft Computing for Intelligent Control and Mobile Robotics*, vol. 318, pp. 213–224. Springer, Berlin (2011)
5. de la O D., Castillo, O., Meléndez, A.: Optimization of fuzzy control systems for mobile robots based on PSO of recent advances on hybrid approaches for designing intelligent systems. In: Castillo, O., Melin, P., Kacprzyk, J. (eds.) *Recent Advances on Hybrid Approaches for Designing Intelligent Systems*, vol. 547, pp. 191–208. Springer, Berlin (2014)
6. De Santis E., Rizzi, A., Sadeghiany, A., Mascioli, F.: Genetic optimization of a fuzzy control system for energy flow management in micro-grids. In: IFSA World Congress and NAFIPS Annual Meeting (IFSA/NAFIPS), 2013 Joint (2013)
7. Esmin, A., Aoki, A.R., Lambert-Torres, G.: Particle swarm optimization for fuzzy membership functions optimization. In: 2002 IEEE International Conference on Systems, Man and Cybernetics (2002)
8. Fang, G., Kwok, N.M., Ha, Q.: Automatic Fuzzy Membership Function Tuning Using the Particle Swarm Optimization. In: PACIA '08. Pacific-Asia Workshop on Computational Intelligence and Industrial Application (2008)
9. García, M., Montiel, O., Castillo, O., Sepúlveda, R.: Optimal path planning for autonomous mobile robot navigation using ant colony optimization and a fuzzy cost function evaluation. In: Melin, P., Castillo, O., Ramírez, E., Kacprzyk, J., Pedrycz, W. (eds.) *Analysis and Design of Intelligent Systems using Soft Computing Techniques*, vol. 41, pp. 790–798. Springer, Berlin (2007)
10. Garcia, M., Montiel, O., Castillo, O., Sepúlveda, R., Melin, P.: Path planning for autonomous mobile robot navigation with ant colony optimization and fuzzy cost function evaluation. *Appl. Soft Comput.* **9**(3), 1102–1110 (2009)
11. Martínez, R., Castillo, O., Aguilar, L.T.: Optimization of interval type-2 fuzzy logic controllers for a perturbed autonomous wheeled mobile robot using genetic algorithms. *Inf. Sci.* **179**(13), 2158–2174 (2009)
12. Martínez-Marroquín, R., Castillo, O., Soria, J.: Particle swarm optimization applied to the design of type-1 and type-2 fuzzy controllers for an autonomous mobile robot. In: Melin, P., Kacprzyk, J., Pedrycz, W. (eds.) *Bio-inspired Hybrid Intelligent Systems for Image Analysis and Pattern Recognition*, vol. 256, pp. 247–262. Springer, Berlin (2009)
13. Martinez-Soto, R., Castillo O., Aguilar, L., Melin, P.: Fuzzy logic controllers optimization using genetic algorithms and particle swarm optimization. In: Sidorov, G., Hernández, A., Aguirre, L., Reyes García, C. (eds.) *Advances in Soft Computing*, vol. 6438, pp. 475–486. Springer, Berlin (2010)
14. Martinez-Soto, R., Castillo O., Aguilar, L., Baruch, I.S.: Bio-inspired optimization of fuzzy logic controllers for autonomous mobile robots. In: Fuzzy Information Processing Society (NAFIPS), 2012 Annual Meeting of the North American (2012)
15. Measurement and Instrumentation.: Faculty of Electrical Engineering and Computer Science, Brno University of Technology, Czech Republic Department of Control. Autonomous Mobile Robotics ToolboxFor Matlab 5. Online (2001)

16. Melendez, A., Castillo, O.: Optimization of type-2 fuzzy reactive controllers for an autonomous mobile robot. In: 2012 Fourth World Congress on Nature and Biologically Inspired Computing (NaBIC) (2012)
17. Meléndez, A., Castillo, O.: Evolutionary optimization of the fuzzy integrator in a navigation system for a mobile robot. In: Castillo, O., Melin, P., Kacprzyk, J. (eds.) Recent Advances on Hybrid Intelligent Systems, vol. 451, pp. 21–31. Springer, Berlin (2013)
18. Meléndez, A., Castillo, O.: Hierarchical genetic optimization of the fuzzy integrator for navigation of a mobile robot. In: Melin, P., Castillo, O. (eds.) Soft Computing Applications in Optimization, Control, and Recognition, vol. 294, pp. 77–96. Springer, Berlin (2013)
19. Meléndez, A., Castillo, O., Soria, J.: Reactive control of a mobile robot in a distributed environment using fuzzy logic. In: Fuzzy Information Processing Society, 2008. NAFIPS 2008. Annual Meeting of the North American (2008)
20. Meléndez, A., Castillo, O., Garza, A.A., Soria, J.: Reactive and tracking control of a mobile robot in a distributed environment using fuzzy logic. In: FUZZ-IEEE (2010)
21. Melin, P., Astudillo, L., Castillo, O., Valdez, F., Garcia, M.: Optimal design of type-2 and type-1 fuzzy tracking controllers for autonomous mobile robots under perturbed torques using a new chemical optimization paradigm. *Expert Syst. Appl.* **40**(8), 3185–3195 (2013)
22. Rajeswari, K., Lakshmi, P.: PSO optimized fuzzy logic controller for active suspension system. In: 2010 International Conference on Advances in Recent Technologies in Communication and Computing (ARTCom) (2010)
23. Ross, O., Camacho, J., Sepúlveda, R., Castillo O.: Fuzzy system to control the movement of a wheeled mobile robot. In: Castillo, O., Kacprzyk, J., Pedrycz, W. (eds.) Soft Computing for Intelligent Control and Mobile Robotics, vol. 318, pp. 445–463. Springer, Berlin (2011)
24. Amin, S., Adriansyah, A.: Particle swarm fuzzy controller for behavior-based mobile robot. In: ICARCV '06. 9th International Conference on Control, Automation, Robotics and Vision (2006)
25. Singh, R., Hanumandlu, M., Khatoon, S., Ibraheem, I.: An adaptive particle swarm optimization based fuzzy logic controller for line of sight stabilization tracking and pointing application. In: 2011 World Congress on Information and Communication Technologies (WICT) (2011)
26. Valdez, F., Melin, P., Castillo, O.: Fuzzy control of parameters to dynamically adapt the PSO and GA Algorithms. In: 2010 IEEE International Conference on Fuzzy Systems (FUZZ) (2010)
27. Venayagamoorthy, G.K., Doctor, S.: Navigation of mobile sensors using PSO and embedded PSO in a fuzzy logic controller. In: Industry Applications Conference, 2004. 39th IAS Annual Meeting. Conference Record of the 2004 IEEE (2004)
28. Venayagamoorthy, G.K., Doctor, S.: Navigation of mobile sensors using PSO and embedded PSO in a fuzzy logic controller. In: Industry Applications Conference, 2004. 39th IAS Annual Meeting. Conference Record of the 2004 IEEE (2004)
29. Wang, D., Wang, G., Hu, R.: Parameters optimization of fuzzy controller based on PSO. In: 3rd International Conference on Intelligent System and Knowledge Engineering, ISKE (2008)
30. Wang, L., Kang, Q., Qiao, F., Wu, Q.: Fuzzy logic based multi-optimum programming in particle swarm optimization. In: 2005 IEEE Networking, Sensing and Control, 2005. Proceedings (2005)
31. Yang, M., Wang, X.: Fuzzy PID controller using adaptive weighted pso for permanent magnet synchronous motor drives. In: Second International Conference on Intelligent Computation Technology and Automation, ICICTA '09 (2009)
32. Adika, C.O., Wang, L.: Short term energy consumption prediction using bio-inspired fuzzy systems. In: North American Power Symposium (NAPS) (2012)
33. Astudillo, L., Melin, P., Castillo, O.: Optimization of a fuzzy tracking controller for an autonomous mobile robot under perturbed torques by means of a chemical optimization paradigm. In: Castillo, O., Melin, P., Kacprzyk, J. (eds.) Recent Advances on Hybrid Intelligent Systems, vol. 451, pp. 3–20. Springer, Berlin (2013)

34. Sanchez, C., Melin, P., Astudillo, L.: Chemical optimization method for modular neural networks applied in emotion classification of recent advances on hybrid approaches for designing intelligent systems. In: Castillo, O., Melin, P., Kacprzyk, J. (eds.) Recent Advances on Hybrid Approaches for Designing Intelligent Systems, vol. 547, pp. 381–390. Springer, Berlin (2014)
35. Tsai, C.-C., Tsai, K.-I., Su, C.-T.: Cascaded fuzzy-PID control using PSO-EP algorithm for air source heat pumps. In: 2012 International Conference on Fuzzy Theory and It's Applications (IFUZZY) (2012)
36. Chen, J., Xu, L.: Road-junction traffic signal timing optimization by an adaptive particle swarm algorithm. In: ICARCV '06. 9th International Conference on Control, Automation, Robotics and Vision (2006)
37. Dorrah, H.T., El-Garhy, A.M., El-Shimy, M.E.: PSO-BELBIC scheme for two-coupled distillation column process. *J. Adv. Res.* **2**(1), 73–83 (2011)
38. Vazquez, J., Valdez, F., Melin, P.: Comparative study of particle swarm optimization variants in complex mathematics functions. In: Castillo, O., Melin, P., Kacprzyk, J. (eds.) Recent Advances on Hybrid Intelligent Systems, vol. 451, pp. 223–235. Springer, Berlin (2013)
39. Castillo, O., Soria, J., Arias, H., Morales, J., Inzunza, M.: Intelligent control and planning of autonomous mobile robots using fuzzy logic and multiple objective genetic algorithms. In: Melin, P., Castillo, O., Ramírez, E., Kacprzyk, J., Pedrycz, W. (eds.) Analysis and Design of Intelligent Systems using Soft Computing Techniques, vol. 41, pp. 799–807. Springer, Berlin (2007)
40. Castillo, O., Martínez-Marroquín, R., Melin, P., Valdez, F., Soria, J.: Comparative study of bio-inspired algorithms applied to the optimization of type-1 and type-2 fuzzy controllers for an autonomous mobile robot. *Inf. Sci.* **192**, 19–38 (2012)
41. John, Y., Reza, L.: Fuzzy Logic: Intelligence, Control, and Information. Prentice Hall, New Jersey (1999)

Part V

Nature Inspired Optimization

Applications

Segmentation of Coronary Angiograms Using a Vesselness Measure and Evolutionary Thresholding

Ivan Cruz-Aceves and Arturo Hernández-Aguirre

Abstract This paper presents a new two stage automatic vessel segmentation method of coronary arteries in X-ray angiograms. In the first stage of the method, a multiscale vesselness measure based on the properties of the eigenvalues of the Hessian matrix is used to detect vessel structures. This vesselness measure was compared with three vessel detection methods using the area under the receiver operating characteristic curve. In the second stage, the magnitude response of the vesselness measure is segmented by a new evolutionary thresholding method using the weighted sum method for multi-objective optimization. This evolutionary method was compared with five classical thresholding methods using five statistical measures and a training set of 20 angiograms. Finally, the proposed method was compared with four state-of-the-art vessel segmentation methods. Experimental results provided an area of 0.8970 with the training set, and an average performance of statistical measures of 0.8302 with a test set of 20 angiograms.

1 Introduction

Automatic segmentation of coronary vessels in X-ray angiography images plays a significant role in the diagnosis and treatment of cardiovascular disease. Hence accurate segmentation of coronary arteries has become essential for computer-assisted diagnosis (CAD) systems. The major disadvantages in the analysis of coronary angiograms are the nonuniform illumination in vessel segments and the

I. Cruz-Aceves

Cátedras-CONACYT at Center for Research in Mathematics (CIMAT),
A.C., Jalisco S/N, Col. Valenciana, C.P. 36000 Guanajuato, Mexico
e-mail: ivan.cruz@cimat.mx

A. Hernández-Aguirre (✉)

Center for Research in Mathematics (CIMAT), A.C., Jalisco S/N, Col. Valenciana,
C.P. 36000 Guanajuato, Mexico
e-mail: artha@cimat.mx

weak contrast between coronary vessels and image background. Due to these drawbacks, the histogram of an angiogram presents multiple peaks and valleys, which pose problems in the correct classification of vessel and nonvessel pixels.

In literature, a number of methods have been proposed for enhancement and automatic segmentation of blood vessels in medical images of different clinical studies. Among the solutions proposed for automatic vessel segmentation are the methods based in mathematical morphology. The method of Eiho and Qian [1] represents one of the most commonly used coronary artery segmentation techniques, because its efficiency and ease of implementation. This method consists on the application of the top-hat operator as vessel enhancement method, followed by the morphological erosion, thinning, and watershed transformation. Maglavera et al. [2] proposed another morphological-based method, which uses the skeleton, thresholding and connected-component operator for automatic detection of vessels in angiograms. Bouraoui et al. [3], introduced a two steps morphological-based method, which utilizes the gray-level hit-or-miss transform and the region-growing strategy. On the other hand, Lara et al. [4] used the region-growing strategy with a differential-geometry approach for semiautomatic segmentation of vessels in angiograms. In general, these morphological-based methods do not perform well when vessels have low contrast and different diameters.

In order to overcome the shortcomings of the morphological-based methods for automatic vessel segmentation, some strategies using spatial filtering have been used. The most commonly spatial filtering technique is the method introduced by Chaudhuri et al. [5] known as Gaussian matched filters (GMF). This method tries to approximate the shape of blood vessels through a Gaussian curve as the matching template. The Gaussian kernel is rotated at different orientations and convolved with the original image. From the filter bank of oriented responses, the maximum response at each pixel is conserved to obtain the filtered image. Generally, the GMF method is used as a preprocessing technique. Chanwimaluang and Fan [6, 7] proposed an automatic retinal fundus segmentation method by thresholded the GMF response with the local entropy thresholding technique, and also they used their segmentation method for a retinal vessel registration system. Kang et al. [8] proposed a fusion strategy consisting of the top-hat operator and the GMF as vessel enhancement methods. For each method the strategy obtains an enhanced image. Both images are thresholded by the maximizing entropy thresholding method. The final segmentation result is obtained by pixel-wise multiplication of the two thresholded images. Usually, the method of Gaussian matched filters presents low efficiency when images contain occlusions or small vessels to be detected.

More recently, vessel enhancement methods using the properties of the Hessian matrix have been introduced. The Hessian matrix is calculated by convolving the second-order derivatives of a Gaussian kernel with the original image. One of the advantages of these methods is the ability of detecting vessels with different diameters, although they are highly sensitive to noise because of the second-order derivative. Frangi et al. [9] proposed a vesselness measure by using the properties of the eigenvalues calculated from the Hessian matrix. This vesselness measure is able to classify vessel and nonvessel pixels accurately. Different vesselness

measures have been proposed using the Hessian matrix and eigenvalues methodology [10–12], and also some works have combined the vesselness measure of Frangi et al. [9] with additional methods to segment blood vessels. M’hiri et al. [13] used the vesselness measure with an interactive random walk formulation, and Li et al. [14] developed a region-growing procedure using the pixels with the maximum feature response obtained from the vesselness measure.

This paper presents a new automatic segmentation method for coronary arteries in X-ray angiograms. The foundation of this method is formed by the steps of enhancement and segmentation. In the enhancement step, the method of Frangi et al. is used to deal with noise and nonuniform illumination. The segmentation step presents a new evolutionary thresholding method based on Differential Evolution and the weighted sum method. The performance of the enhancement step is assessed using the area under the ROC curve and compared with different vesselness measures. The evolutionary thresholding method is compared with five automatic thresholding methods using the statistical measures of sensitivity, specificity, accuracy, positive predictive value and negative predictive value. In addition, the proposed method is compared with four state-of-the-art vessel segmentation methods using statistical measures and the ground-truth images outlined by a specialist.

The organization of this paper is as follows. In Sect. 2, the dataset of coronary angiograms is introduced. The blood vessel enhancement methods are presented in Sect. 3. The proposed vessel segmentation method and the performance metrics are presented in Sects. 4 and 5, respectively. The experimental results are discussed in Sect. 6, and conclusions are given in Sect. 7.

2 Database of Coronary Angiograms

The database used in the present work consists of 40 X-ray coronary angiograms of 27 different patients. Each image is of size 300×300 pixels. In order to assess the performance of the segmentation methods, the holdout method has been adopted, where twenty of the 40 images have been used as training set primarily for tuning purposes of the methods and the remaining 20 images have been used as test set for evaluation of vessel segmentation methods. The ground-truth images were outlined by a specialist and the ethics approval was provided by the Mexican Social Security Institute, UMAE T1 León.

3 Blood Vessel Enhancement

Coronary arteries in X-ray angiographic images can be described as dark tubular structures with an uneven illumination, different diameters and diverse orientations. Due to these factors, a vessel enhancement procedure is commonly employed as a

preprocessing step before using a specific segmentation method. In this section, we review and implement some vessel enhancement methods based on the matrix of second order derivatives of the Gaussian kernel, which has been applied in different vesselness measures.

3.1 Vesselness Measure of Frangi et al.

The vesselness measure of Frangi et al. [9], uses the properties of the eigenvalues of the Hessian matrix (matrix of second order derivatives of the Gaussian kernel) computed at each pixel of the intensity image $L(x, y)$ as follows:

$$H = \begin{bmatrix} L_{xx}(x, y) & L_{xy}(x, y) \\ L_{yx}(x, y) & L_{yy}(x, y) \end{bmatrix}, \quad (1)$$

where L_{xx} , L_{xy} , L_{yx} , and L_{yy} denote the second order derivatives of the intensity image $L(x, y)$,

$$L_{xx} = \frac{\partial^2 L}{\partial x^2} = L(x, y)^* \sigma^2 G_{xx}, \quad (2)$$

$$L_{yy} = \frac{\partial^2 L}{\partial y^2} = L(x, y)^* \sigma^2 G_{yy}, \quad (3)$$

$$L_{xy} = \frac{\partial^2 L}{\partial x \partial y} = L(x, y)^* \sigma^2 G_{xy}, \quad (4)$$

where the symbol $*$ denotes a convolution operator and G_{xx} , G_{xy} , and G_{yy} are the second order derivatives of the Gaussian kernel G at different scales of σ , which is defined as

$$G(x, y; \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right). \quad (5)$$

Since the Hessian matrix is symmetric, the large eigenvalue λ_1 and the small eigenvalue λ_2 of the intensity image $L(x, y)$ can be computed as follows:

$$\alpha = \sqrt{(L_{xx} - L_{yy})^2 + 4L_{xy}^2}, \quad (6)$$

$$\lambda_1 = \frac{L_{xx} + L_{yy} + \alpha}{2}, \quad (7)$$

$$\lambda_2 = \frac{L_{xx} + L_{yy} - \alpha}{2}. \quad (8)$$

Finally, the vesselness measure of Frangi et al. is calculated as

$$V_F = \begin{cases} \exp\left(-\frac{R_\beta^2}{2\beta^2}\right) \left[1 - \exp\left(-\frac{s^2}{2\gamma^2}\right)\right] & \text{if } (\lambda_1, \lambda_2 < 0) \\ 0, & \text{otherwise,} \end{cases} \quad (9)$$

where $R_\beta = \lambda_1/\lambda_2$, $s = \sqrt{\lambda_1^2 + \lambda_2^2}$, $\beta = 0.5$, and γ represents one-half of the maximum value of s . The final enhanced image is obtained by selecting the maximum value of V_F over the different scales of σ at each pixel.

3.2 Vesselness Measure of Salem et al.

Salem et al. [10], used the Hessian matrix on the intensity image to compute the largest eigenvalue and vessel orientation over all scales. The largest eigenvalue λ_{\max} at each image pixel at different scales is calculated as

$$\lambda_{\max} = \max_{\sigma} [\lambda_1(\sigma)]. \quad (10)$$

The vessel orientation is computed from the eigenvectors of the Hessian matrix as

$$\theta_- = \tan^{-1} \left[\frac{-2L_{xy}}{L_{yy} - L_{xx} + \alpha} \right], \quad (11)$$

and subsequently the standard deviation of orientation values θ_- over scales is obtained as

$$\theta_{std} = std_{\sigma} [\theta_-(\sigma)]. \quad (12)$$

Finally, the proposed vesselness measure of Salem et al., can be calculated as follows:

$$V_{SSN} = \frac{\lambda_{\max}}{1 + \theta_{std}}, \quad (13)$$

where λ_{\max} is the local maximum eigenvalue over scales, θ_{std} is the standard deviation, and the value of 1 is used to avoid singularities.

3.3 Vesselness Measure of Wang et al.

The vesselness measure proposed by Wang et al. [11], computes the dominant eigenvalue to each pixel in a 2D image at different scales, since its eigenvector indicates direction of maximal curvature. This measure can be used for single-scale and multiscale analysis. The response function for single-scale analysis is defined as follows:

$$Z(x, y, \lambda_1) = \begin{cases} \log(|\lambda_1| + 1), & \lambda_1 < -1 \\ 0, & \text{otherwise.} \end{cases} \quad (14)$$

For multiscale analysis, the values of σ are determined by the predefined range of maximum and minimum scales of vessel diameters. The response function for multi-scale analysis is defined as

$$Z(x, y, \lambda_1(\sigma)) = \begin{cases} \log(|\lambda_1(\sigma)| + 1), & \lambda_1(\sigma) < -8 \log(\sigma) \\ 0, & \text{otherwise,} \end{cases} \quad (15)$$

The final enhanced image is acquired through the application of the following vesselness measure, which obtains local maximum pixel value over scales.

$$V_{WLZ} = \max_{\sigma=\sigma_1, \sigma_2, \dots, \sigma_n} Z(x, y, \lambda_1(\sigma)). \quad (16)$$

3.4 Vesselness Measure of Tsai et al.

The vessel enhancement method proposed by Tsai et al. [12] is composed of three fundamental steps. First, a contrast enhancement function is performed on the original image in order to reduce the gray-level of vessel pixels as follows:

$$L_e(x, y) = \begin{cases} L(x, y) + 30, & \text{if } L(x, y) > T, \\ L(x, y) - 30, & \text{else,} \end{cases} \quad (17)$$

where T is a threshold value, and $L(x, y)$ the original image. Subsequently, the eigenvalues of the Hessian matrix are computed over the resulting image $L_e(x, y)$, and the large positive eigenvalue λ_2 is considered as a vessel feature to be used in the adaptive feature transform function as follows:

$$V(x, y; \sigma) = \begin{cases} 1 - \exp\left(-\frac{s^2}{c}\right), & \text{if } \lambda_2(x, y; \sigma) > 0, \\ 0, & \text{else,} \end{cases} \quad (18)$$

where c is a constant value and s^2 is defined as

$$s^2 = \left(\frac{1}{w^2 \times L_e(x, y)} \right)^2 \times \lambda_2^2(x, y; \sigma), \quad (19)$$

and w is a weighting factor. Finally, the resulting enhanced image is acquired by applying the vessel filter response over all scales.

$$V_{TLC} = \max(V(x, y; \sigma)), \sigma = \sigma_1, \dots, \sigma_n \quad (20)$$

The previously described vesselness measures are used as vessel enhancement or vessel detection methods, and the magnitude response of these methods is in general, segmented through thresholding methods. In the following section, a new evolutionary thresholding method is proposed and described in detail.

4 Proposed Evolutionary Thresholding Method

This section presents a new evolutionary thresholding method based on Differential Evolution and the weighted sum method, which is used to segment the magnitude response of the vesselness measure of Frangi et al. [9].

4.1 Differential Evolution

Differential Evolution (DE) is a stochastic real-parameter method proposed by Storn and Price [15, 16] for global optimization problems. DE is similar to standard evolutionary algorithms since it uses a set of randomly initialized solutions, which are known as individuals $X = \{x_1, x_2, \dots, x_{N_p}\}$, where N_p represents the population size. This population is gradually improved through different variation operators and with an objective function, where the final solution is the individual with the best fitness in the whole population.

The DE algorithm consists of three evolutionary stages: mutation, crossover, and selection. The mutation process is used to create a mutant vector $V_{i,g+1}$ for each generation g based on the distribution of the population, which is defined as

$$V_{i,g+1} = X_{r1,g} + F(X_{r2,g} - X_{r3,g}), \quad r1 \neq r2 \neq r3 \neq i, \quad (21)$$

where r_1, r_2 , and r_3 are the indexes of three mutually different and uniformly selected individuals of the population, and F is the differentiation factor. Consequently, the crossover operator is performed to create a trial vector $U_{i,g+1}$ as

$$U_{i,g+1} = \begin{cases} V_{i,g+1}, & \text{if } r \leq CR, \\ X_{i,g}, & \text{if } r > CR, \end{cases} \quad (22)$$

where r represents a uniform random threshold on the range $(0, 1)$, which is used to be compared with the crossover rate (CR). If the threshold value is bigger than the crossover rate, the current information of the individual $X_{i,g}$ is preserved, otherwise the current information of the mutant vector $V_{i,g+1}$ is transcribed to the trial vector $U_{i,g+1}$. Finally, the selection operator is applied to obtain the best solution between the trial vector and the current individual according to a fitness function. The selected vector is then used to replace the current individual in the next generation, as it is shown below

$$X_{i,g+1} = \begin{cases} U_{i,g+1}, & \text{if } f(U_{i,g+1}) > f(X_{i,g}), \\ X_{i,g}, & \text{otherwise.} \end{cases} \quad (23)$$

The application of the three evolutionary operators is performed until a stopping criterion is satisfied (e.g., number of generations), and the final solution is chosen to be the individual with the best fitness in the whole process.

4.2 Weighted Sum Method

Multi-objective optimization (MOO) is the process of optimizing systematically and simultaneously a number of objective functions [17]. One of the most commonly used strategies for solving MOO problems is the weighted sum method (WSM). The fundamental idea behind WSM consists in the arrangement of the objective functions and a particular set of weights in order to form a single function as follows:

$$U = \sum_{i=1}^k w_i F_i(\mathbf{x}), \quad (24)$$

where k is the number of objective functions of the problem, w is the weighting factor, and $F(\mathbf{x})$ represents the objective function to be evaluated. Although a multi-objective optimization problem has multiple solutions, the weighted sum method provides a single solution point according to the set of weights. In general, the cumulative sum of the weights has to be equal to 1, and the weights are usually established by the preference of the user.

4.3 Inter-class Variance Criterion

The inter-class variance criterion used in the Otsu method [18], is applied on the histogram of intensity levels of an image. This criterion computes the probability

distribution (p_i), using the number of pixels n_i of intensity level i , and the total number of pixels N as

$$P_i = \frac{n_i}{N}. \quad (25)$$

The mean intensity of the given image can be determined by the following equation:

$$\mu = \sum_{i=1}^L iP_i, \quad (26)$$

where L represents the number of intensity levels ($\{0, 1, 2, \dots, L - 1\}$) in the image. The probability distribution for each class of pixels having a threshold intensity value t is calculated as

$$w_j = \sum_{i=t_{j-1}+1}^{t_j} P_i. \quad (27)$$

The mean intensity for each class of pixels can be computed as

$$\mu_j = \sum_{i=t_{j-1}+1}^{t_j} iP_i / w_j. \quad (28)$$

The inter-class or between-class variance can be calculated using the following equation:

$$\sigma^2 = \sum_{j=1}^n w_j (\mu_j - \mu)^2. \quad (29)$$

The intensity level corresponding to the maximum inter-class variance gives the optimal threshold value in order to classify foreground and background pixels. This threshold value is acquired by maximizing the following objective function:

$$\phi = \max_{1 < t_1 < \dots < t_{n-1} < L} \{\sigma^2(t)\}. \quad (30)$$

4.4 Two-Dimensional Entropy Criterion

The two-dimensional entropy or local entropy thresholding technique was proposed by Pal and Pal [19]. This criterion is based on the co-occurrence matrix of the image

and the second-order entropy. The second-order entropy corresponding to the foreground image can be defined as

$$H_A(s) = -\frac{1}{2} \sum_{i=0}^s \sum_{j=0}^s P_{ij}^A \log_2 P_{ij}^A, \quad (31)$$

and similar, the second-order entropy of the background image as

$$H_C(s) = -\frac{1}{2} \sum_{i=s+1}^{L-1} \sum_{j=s+1}^{L-1} P_{ij}^C \log_2 P_{ij}^C, \quad (32)$$

where, s is a threshold value, L represents the number of intensity levels, P_A and P_C are the probability distributions of the foreground and background pixels calculated using the probability of co-occurrence p_{ij} of intensity levels as

$$P_A = \sum_{i=0}^s \sum_{j=0}^s p_{ij}, \quad (33)$$

$$P_C = \sum_{i=s+1}^{L-1} \sum_{j=s+1}^{L-1} p_{ij}. \quad (34)$$

The total second-order entropy of the image can be written as

$$H_T = H_A(s) + H_C(s). \quad (35)$$

The corresponding optimal threshold of the image is that intensity value which maximizes the total second-order local entropy. The threshold value is obtained through maximizing the following objective function:

$$\varphi = \max_{1 < t_1 < \dots < t_{n-1} < L} \{H_T(t)\}. \quad (36)$$

4.5 Proposed Method

The proposed evolutionary thresholding method addresses the segmentation problem as a multi-objective optimization process by using Differential Evolution and the weighted sum method.

Since the weighted sum method uses a composite objective function to solve a multi-objective problem. In our work, the composite objective function is composed of the inter-class variance criterion and the two-dimensional entropy

criterion. Each criterion is normalized in the range [0, 1], and they are weighted with scalar values as

$$U = w_1 F_1(\mathbf{x}) + w_2 F_2(\mathbf{x}) \quad (37)$$

where w_1 and w_2 are the scalar factors, $F_1(\mathbf{x})$ is the inter-class variance criterion, and $F_2(\mathbf{x})$ is the two-dimensional entropy criterion. The advantage of using the weighted sum method instead Pareto front method resides in the fact that it provides a single threshold solution automatically and also that it makes possible to apply the mono-objective Differential Evolution algorithm preserving the properties of low computational time, efficiency and ease of implementation.

According to the previous description, the proposed evolutionary thresholding method using Differential Evolution can be implemented by using the following procedure:

1. Initialize number of generations G , population size N_p , differentiation factor F , and the crossover rate CR .
2. Initialize each individual X_i by generating random candidate solutions in the range of $[0, 255]$.
3. For each individual $X_{i,g}$ where $g = \{1, \dots, G\}$:
 - (a) Compute $V_{i,g+1}$ through mutation step (21);
 - (b) Assign $U_{i,g+1}$ using the crossover operator (22);
 - (c) Evaluate $U_{i,g+1}$ in fitness function (37);
 - (d) Update $X_{i,g+1}$ according to the selection operator (23).
4. If stopping criterion is satisfied (e.g., number of generations), then stop.

5 Performance Metrics

In order to evaluate the segmentation results of the proposed method in X-ray coronary angiograms, measures of sensitivity, specificity, accuracy, positive predictive value (PPV), negative predictive value (NPV) and the area under the receiver operating characteristic (ROC) curve have been adopted.

These measures are based on the classification of vessel and nonvessel pixels as:

- True-Positive (TP): Vessel pixel correctly classified as vessel pixel.
- False-Positive (FP): Nonvessel pixel incorrectly classified as vessel pixel.
- True-Negative (TN): Nonvessel pixel correctly classified as nonvessel pixel.
- False-Negative (FN): Vessel pixel incorrectly classified as nonvessel pixel.

In this work, the area under the ROC curve (A_z) is used to assess vessel enhancement methods through true-positive and false-positive rates according to the vessel pixels labeled by a specialist in the ground-truth images.

The other five measures are used to evaluate the vessel segmentation methods, and they are defined as follows:

$$Sensitivity = \frac{TP}{TP + FN} \quad (38)$$

$$Specificity = \frac{TN}{TN + FP} \quad (39)$$

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \quad (40)$$

$$PPV = \frac{TP}{TP + FP} \quad (41)$$

$$NPV = \frac{TN}{TN + FN} \quad (42)$$

The six statistical measures are defined in the range [0, 1], where the ideal result is equal to 1, and zero is the worst result. In the following section, the segmentation results obtained from the proposed method are presented and analyzed using the evaluation metrics.

6 Results and Discussion

The computational experiments were tested and implemented on a computer with an Intel Core i3, 2.13 GHz, and 4 GB of RAM using the Matlab software.

As part of the vessel detection step, four methods based on the Hessian matrix have been compared using the training set of 20 angiograms. The best set of parameters for each method was acquired via ROC curve analysis. The set of Frangi et al. method parameters were obtained by varying the range of sigma and the step size value. The best parameters were determined to be $\sigma = [1, 12]$, and $\Delta = 0.5$. The best parameters of the method of Salem et al. were determined to be $\sigma = [1, 20]$, and $\Delta = 0.5$. The best parameters of the method of Wang et al. were determined to be $\sigma = [1, 19]$, and $\Delta = 0.5$. The best parameters of the method of Tsai et al. were determined to be $\sigma = [1, 10]$, and $\Delta = 0.5$, $c = 1$, and $w = 0.5$.

In order to compare the performance of the aforementioned methods, Fig. 1 presents the area under the ROC curve (A_z) for each vessel detection method, where the best performance is obtained by the Frangi et al. method according to the training set of 20 angiograms. The vessel detection results are visualized in Fig. 2, where one angiogram of the training set is presented along with its ground-truth image. The resulting images show that the Frangi et al. method can detect in a more suitable way the vessel structures than the other three Hessian matrix methods.

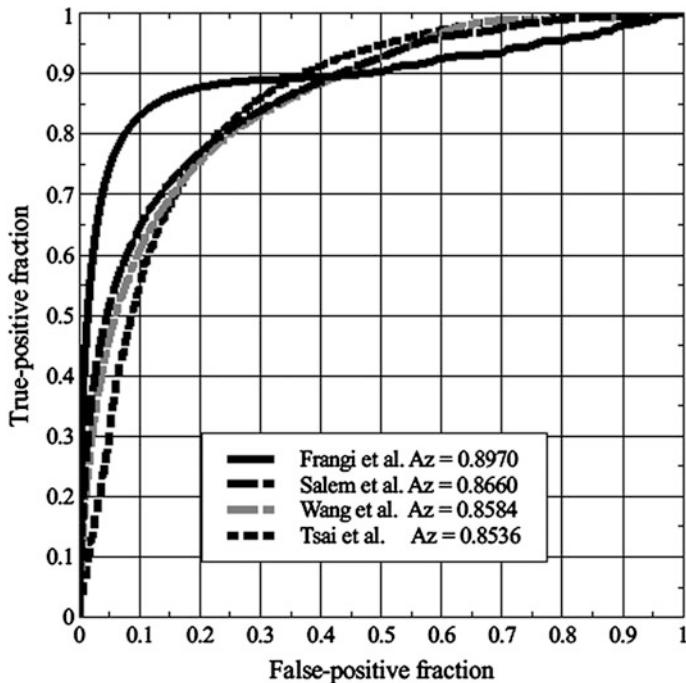


Fig. 1 Comparison of ROC curves for vessel detection using the training set of angiograms with the methods of Frangi et al., Salem et al., Wang et al., and Tsai et al., respectively

Since the selection of the optimal parameters for vessel detection methods plays an essential role. Figure 3 presents how can affect these parameters the detection step and hence the final segmentation result. This figure shows the detection results of the method of Frangi et al. for different parameters of σ (average width of vessels) and step size, where the best parameters were set according to ROC analysis as $\sigma = [1, 12]$, and using a step size $\Delta = 0.5$.

The multiscale feature analysis is one of the advantages of the Hessian matrix methods regarding to the methods of mathematical morphology. For further analysis, the method of Frangi et al. [9] taking into account the previously mentioned set of parameters, was used to our proposed vessel segmentation method.

Moreover, the training set of 20 angiograms is also employed to evaluate the performance of the evolutionary thresholding method. This method is compared with five state-of-the-art automatic thresholding methods along with the statistical measures of sensitivity, specificity, accuracy, positive predictive value and negative predictive value. Table 1 presents the segmentation results of the six thresholding methods, where the average value of the five statistical measures is used to determine the best performance in the training set.

Figure 4 presents the segmentation results of an image of the training set, which are acquired by thresholding the magnitude response of the multiscale method of

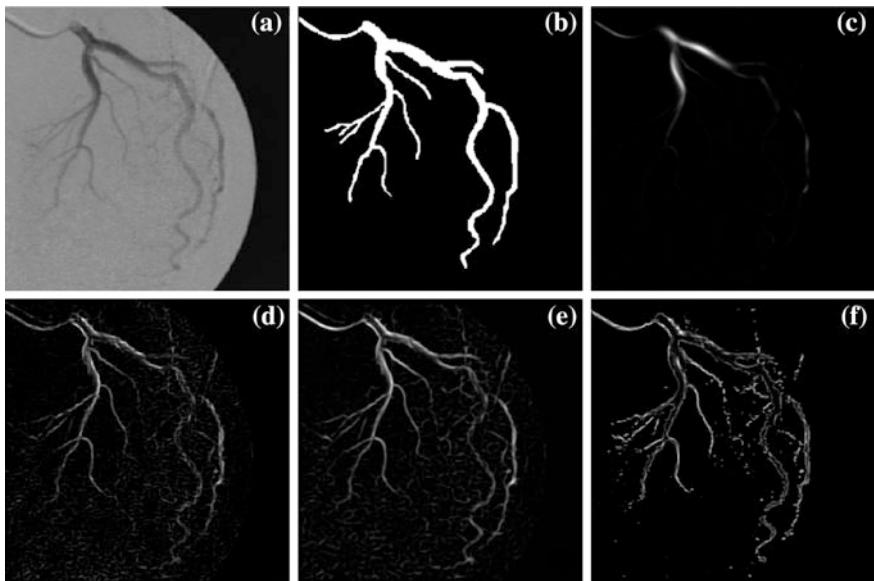


Fig. 2 **a** Original angiographic image, **b** ground-truth of (a) outlined by an specialist, **c** magnitude response of the Frangi et al. method, **d** magnitude response of the Salem et al. method, **e** magnitude response of the Wang et al. method, and **f** magnitude response of the Tsai et al. method

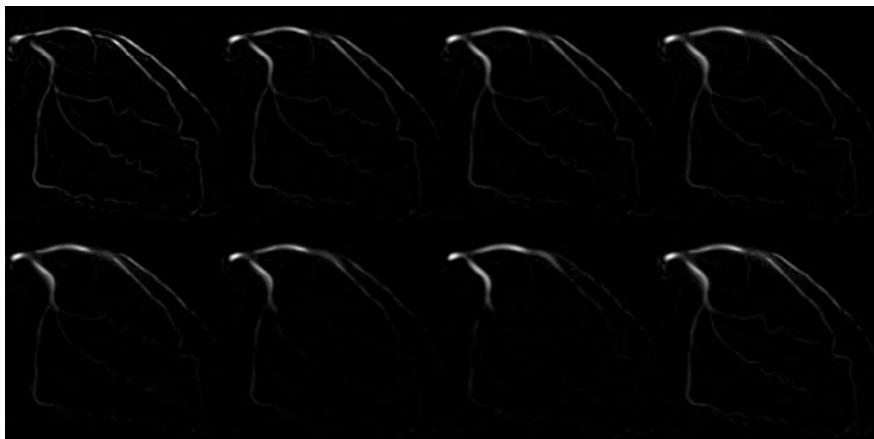


Fig. 3 Vessel detection results for different parameters: *First row*, from *left to right*, first image with range = [1, 3] and step size = 0.5, second image with range = [1, 5] and step size = 0.5, third image with range = [1, 7] and step size = 0.5, last image with range = [1, 10] and step size = 0.5. *Second row* from *left to right*, first image with range = [1, 12] and step size = 2, second image with range = [1, 12] and step size = 4, third image with range = [1, 12] and step size = 6, and last image with range = [1, 12] and step size = 0.5

Table 1 Comparative analysis of the evolutionary thresholding method with five state-of-the-art thresholding methods using the training set of 20 angiograms

Thresholding method	Sensitivity	Specificity	Accuracy	PPV	NPV	Average
Proposed method	0.7703	0.9764	0.9664	0.6256	0.9881	0.8654
Pal and Pal [19]	0.8171	0.9690	0.9616	0.5739	0.9904	0.8624
RATS [20, 21]	0.3646	0.9965	0.9657	0.8410	0.9684	0.8272
Otsu [18]	0.3144	0.9976	0.9643	0.8682	0.9660	0.8221
Moments [22]	0.2434	0.9986	0.9618	0.8958	0.9627	0.8125
Kapur et al. [23]	0.2126	0.9988	0.9605	0.8995	0.9612	0.8065

Frangi et al. The best segmentation results are obtained by our proposed evolutionary method and by the second-order entropy of Pal and Pal [19]. Both methods show similar performance in the similarity measures; however, our proposed method is able to detect small vessels in a more stable and accurate way, since it can be observed in Fig. 4 according to the ground-truth image of the original angiogram.

Since the proposed evolutionary thresholding method uses the evaluation of the second-order entropy of Pal and Pal [19] and the inter-class variance of Otsu [18] method, the proposed segmentation method can lead to more efficiency and accuracy than the state-of-the-art methods. In Fig. 5 the process of threshold value selection by the Pal and Pal method, Otsu method, and our proposed method in the angiogram of Fig. 4 is illustrated. The threshold value obtained by Otsu method is acquired by maximizing the inter-class variance of the image, which for that particular image is equal to 51 with unsatisfactory results. The threshold value obtained by the Pal and Pal method is acquired by maximizing the second-order entropy using the co-occurrence matrix of the image, where the optimal value was determined to be equal to 4 with suitable results. Using the combination of the two above mentioned criteria evaluated as a multi-objective optimization problem through the weighted sum method, the proposed thresholding method is able to obtain better segmentation results than the five thresholding methods. In our proposed evolutionary method, the weighting factors were statistically determined as $w_1 = 0.2$ for the inter-class variance criterion and $w_2 = 0.8$ for the second-order entropy criterion using the training set of 20 angiograms and the average of the five statistical measures.

Finally, the proposed automatic vessel segmentation method is compared with four specialized vessel segmentation methods. The methods of Wang et al. [11] and Tsai et al. [12] are based on the vessel enhancement and segmentation steps. The enhancement step uses the properties of the eigenvalues calculated from the Hessian matrix, and then the resulting image is segmented by using different procedures. The method of Eiho and Qian [1] uses the top-hat operator as vessel enhancement method, and then a morphology-based procedure is applied to obtain the final coronary artery tree in the angiogram. The Chanwimaluang and Fan method [6] uses the Gaussian matched filters as vessel enhancement method, and then the

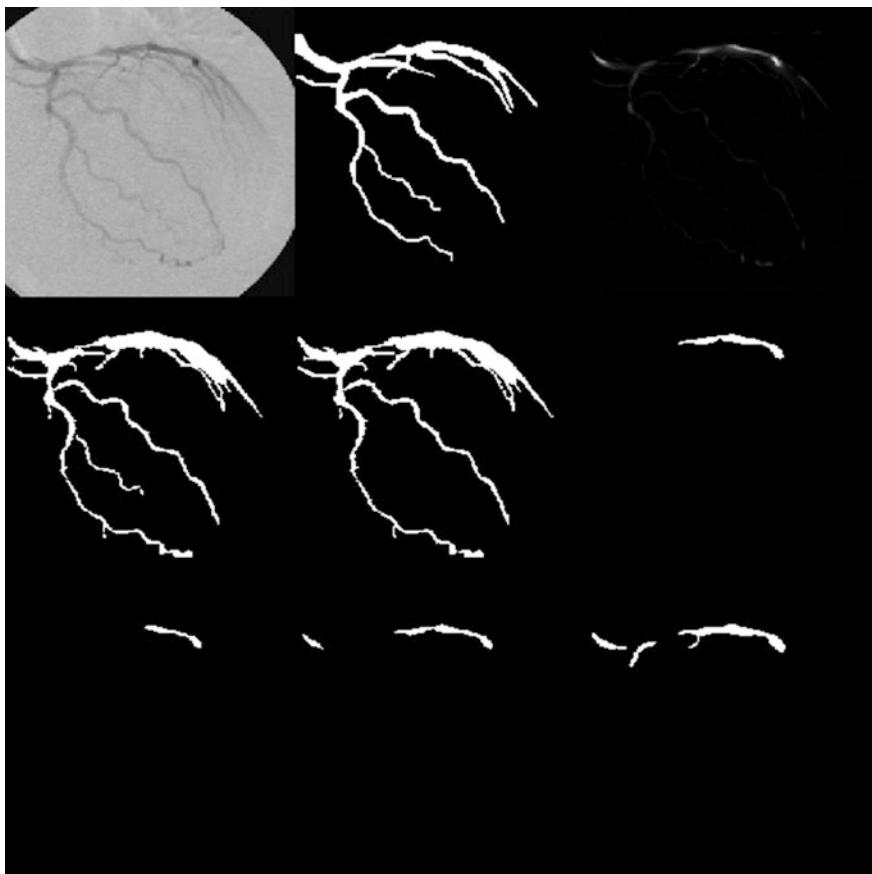


Fig. 4 Segmentation results of the magnitude response of the Frangi et al. method using the six thresholding methods. From *left* to *right*, *first row*, original angiographic image, ground-truth, and magnitude response of the Frangi et al. to be thresholded. *Second row*, segmentation results of the proposed method, Pal and Pal method, and Otsu method. *Last row*, segmentation results of the Kapur et al. method, Moments method and RATS method

resulting filtered image is thresholded by using the second-order entropy method of Pal and Pal [19] to obtain the final vessel segmentation result. The optimal parameters for the vessel enhancement step of the four comparative methods were acquired using ROC curve analysis. In Table 2, a comparative analysis of the proposed method with the four specialized vessel segmentation methods using five statistical measures is presented. The average of the statistical measures shows that the proposed method provides the highest rate of blood vessel segmentation using the test set of 20 angiograms.

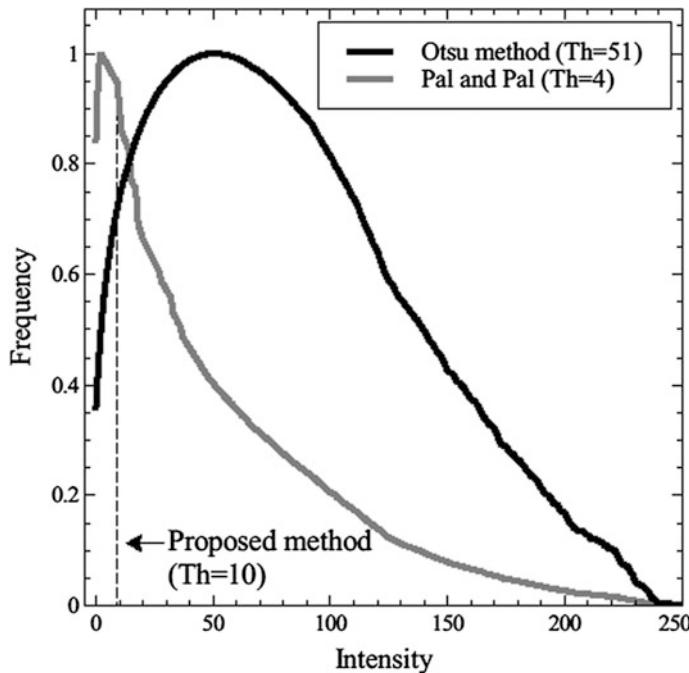


Fig. 5 Process of threshold value selection of the proposed evolutionary method using the second-order entropy and the inter-class variance as objective functions evaluated through the weighted sum method

Table 2 Comparative analysis of the proposed method with the Wang et al. method, morphological-based method of Eiho and Qian, Chanwimaluang and Fan method, and the Tsai et al. method, using the test set of 20 angiograms

Thresholding method	Sensitivity	Specificity	Accuracy	PPV	NPV	Average
Proposed method	0.8744	0.9579	0.9557	0.3664	0.9964	0.8302
Wang et al. [11]	0.6186	0.9743	0.9516	0.3221	0.9740	0.7681
Eiho and Qian [1]	0.7930	0.8483	0.8447	0.2630	0.9836	0.7465
Chanwimaluang and Fan [6]	0.8363	0.8260	0.8266	0.2471	0.9866	0.7445
Tsai et al. [12]	0.2989	0.9612	0.9188	0.3447	0.9525	0.6952

The proposed method outperforms the comparative methods in terms of sensitivity, accuracy, positive predictive value and negative predictive value, and obtaining a competitive result in terms of specificity measure.

In order to visualize the vessel segmentation results obtained from the proposed method and comparative methods, a subset of angiograms of the test set of images is presented in Fig. 6.

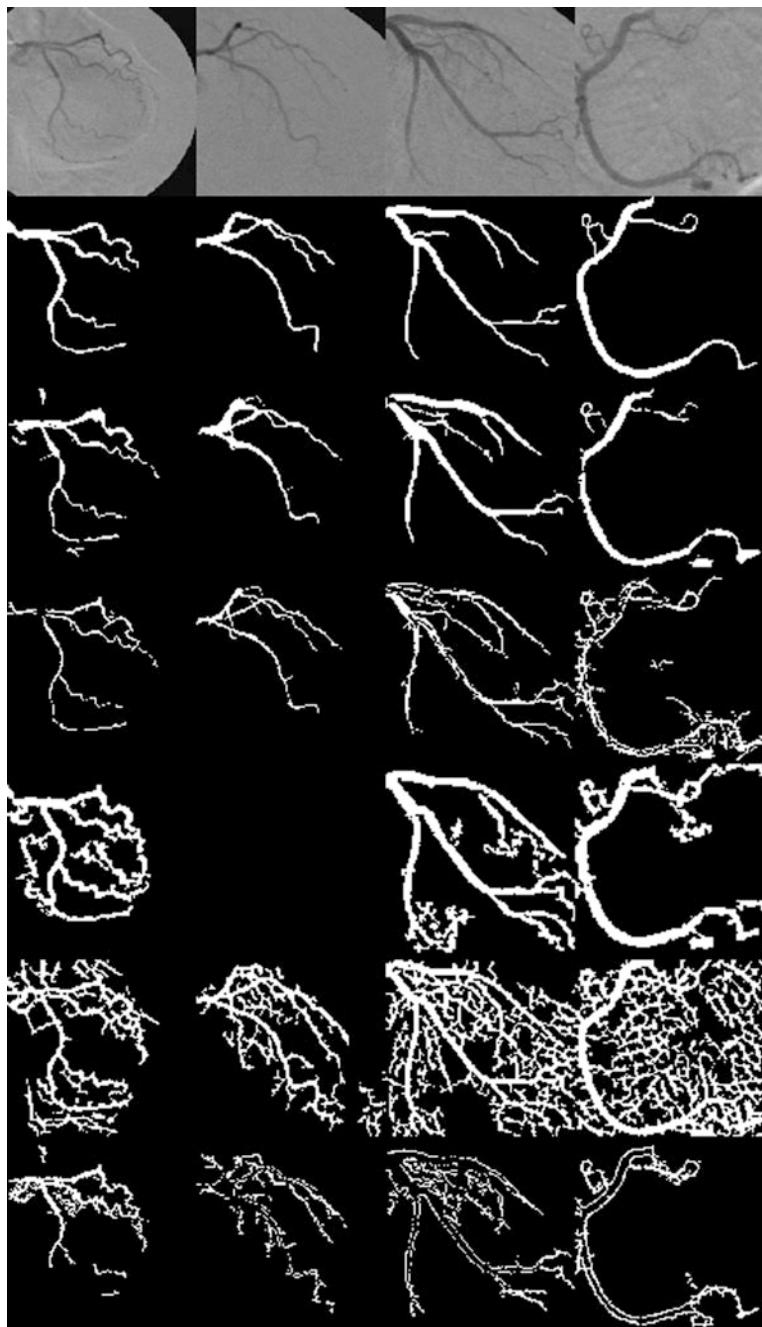


Fig. 6 Segmentation results for a subset of angiograms from the test set. *First row* shows original angiograms. *Second row* presents the ground-truth of the images of the *first row*. The remaining five rows presents the segmentation results of the proposed method, Wang et al. method, Eiho and Qian method, Chanwimaluang et al. method and Tsai et al. method, respectively

According to the segmentation results, the methods of Eiho and Qian, Chanwimaluang and Fan, and Tsai et al. present low quality results. The method of Wang et al. presents competitive results compared with the ground-truth images and with the proposed method in terms of specificity; however, it does not perform well in the remaining four statistical measures including the final average measure.

The use of the vessel enhancement method of Frangi et al. in the proposed method provides robustness and accuracy in the detection of blood vessels. Furthermore, the evolutionary thresholding method based on Differential Evolution and the weighted sum method can overcome the problems of detecting small vessels, different vessel diameters, and nonuniform illumination along vessel structures in an appropriate way regarding the comparative vessel segmentation methods. The experimental results show that the accuracy in coronary artery segmentation obtained from the proposed method is suitable for cardiac angiography systems.

7 Conclusion

In this paper, a new automatic vessel segmentation method in X-ray coronary angiograms has been presented. The proposed method consists of the steps of detection and segmentation. In the detection step, the multiscale method of Frangi et al. is used to enhance the coronary vessels in the original angiogram. This enhancement method was compared with three vesselness measures using the area under the ROC curve. The method of Frangi et al. outperforms the comparative methods achieving an area under ROC curve of 0.8970 with the training set of 20 angiograms. The segmentation step consists on the selection of the optimal threshold value from the magnitude response of the method of Frangi et al. In this step a new evolutionary thresholding method was introduced, which uses the weighted sum method to evaluate the inter-class variance criterion and the second-order entropy criterion simultaneously. This evolutionary method was compared with five automatic thresholding methods using five statistical measures, obtaining the best average performance (0.8654). According to the experimental results, the proposed method, consisting of the application of the method of Frangi et al. for detection of vessels, and an evolutionary thresholding method for segmentation, can lead to higher average efficiency than four specialized vessel segmentation methods using a test set of 20 angiograms. In addition, the experimental results have also shown that the proposed method is highly appropriate for clinical decision support systems.

Acknowledgments This research has been supported by the National Council of Science and Technology of Mexico (Project Cátedras-CONACYT No. 3150-3097). The authors would like to thank the cardiology department of the Mexican Social Security Institute, for clinical advice and assistance in data collection.

References

1. Eihö, S., Qian, Y.: Detection of coronary artery tree using morphological operator. *Comput. Cardiol.* **24**, 525–528 (1997)
2. Maglavera, N., Haris, K., Efstratiadis, S., Gourassas, J., Louridas, G.: Artery skeleton extraction using topographic and connected component labeling. *Comput. Cardiol.* **28**, 17–20 (2001)
3. Bouraoui, M., Ronse, C., Baruthio, J., Passat, N., Germain, P.: Fully automatic 3D segmentation of coronary arteries based on mathematical morphology. In: 5th IEEE International Symposium on Biomedical Imaging (ISBI): From Nano to Macro, pp. 1059–1062 (2008)
4. Lara, B.D., Faria, A., Araujo, A., Menotti, D.: A semi-automatic method for segmentation of the coronary artery tree from angiography. In: XXII Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI), pp. 194–201 (2009)
5. Chaudhuri, S., Chatterjee, S., Katz, N., Nelson, M., Goldbaum, M.: Detection of blood vessels in retinal images using two-dimensional matched filters. *IEEE Trans. Med. Imaging* **8**, 263–269 (1989)
6. Chanwimaluang, T., Fan, G.: An efficient blood vessel detection algorithm for retinal images using local entropy thresholding. In: Proceedings of the IEEE International Symposium on Circuits and Systems, vol. 5, pp. 21–24 (2003)
7. Chanwimaluang, T., Fan, G., Fransen, S.: Hybrid retinal image registration. *IEEE Trans. Inf Technol. Biomed.* **10**, 129–142 (2006)
8. Kang, W., Wang, K., Chen, W., Kang, W.: Segmentation method based on fusion algorithm for coronary angiograms. In: 2nd International Congress on Image and Signal Processing (CISP), pp. 1–4 (2009)
9. Frangi, A., Niessen, W., Vincken, A., Viergever, M.: Multiscale vessel enhancement filtering. In: Medical Image Computing and Computer-Assisted Intervention (MICCAI'98), vol. 1496, pp. 130–137. Springer, Berlin (1998)
10. Salem, N.M., Salem, S.A., Nandi, A.K.: Segmentation of retinal blood vessels based on analysis of the hessian matrix and clustering algorithm. In: 15th European Signal Processing Conference (EUSIPCO 2007). Springer Lecture Notes in Computer Science, pp. 428–432 (2007)
11. Wang, S., Li, B., Zhou, S.: A segmentation method of coronary angiograms based on multi-scale filtering and region-growing. In: International Conference on Biomedical Engineering and Biotechnology, pp. 678–681 (2012)
12. Tsai, Y.C., Lee, H.J., Chen, M.Y.C.: Adaptive segmentation of vessels from coronary angiograms using multi-scale filtering. In: International Conference on Signal-Image Technology and Internet-Based Systems, pp. 143–147 (2013)
13. M'hiri, F., Duong, L., Desrosiers, C., Cheriet, M.: Vesselwalker: Coronary arteries segmentation using random walks and Hessian-based vesselness filter. In: IEEE 10th International Symposium on Biomedical Imaging (ISBI): From Nano to Macro, pp. 918–921 (2013)
14. Li, Y., Zhou, S., Wu, J., Ma, X., Peng, Y.: A novel method of vessel segmentation for X-ray coronary angiography images. In: Fourth International Conference on Computational and Information Sciences (ICCIS), pp. 468–471 (2012)
15. Storn, R., Price, K.: Differential evolution—a simple and efficient adaptive scheme for global optimization over continuous spaces. Technical Report TR-95-012, International Computer Sciences Institute, Berkeley, Calif, USA (1995)
16. Storn, R., Price, K.: Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. *J. Global Optim.* **11**(4), 341–359 (1997)
17. Marler, R.T., Arora, J.S.: The weighted sum method for multi-objective optimization: new insights. *Struct. Multi. Optim.* **41**, 853–862 (2010)

18. Otsu, N.: A threshold selection method from gray-level histograms. *IEEE Trans. Syst. Man Cybern.* **9**, 62–66 (1979)
19. Pal, N.R., Pal, S.K.: Entropic thresholding. *Sig. Process.* **16**, 97–108 (1989)
20. Kittler, J., Illingworth, J., Föglein, J.: Threshold selection based on a simple image statistic. *Comput. Vision Grap. Image Process.* **30**, 125–147 (1985)
21. Illingworth, J., Kittler, J.: A parallel threshold selection algorithm. In: *Proceedings of SPIE Architectures and Algorithms for Digital Image Processing*, vol. 596, pp. 129–134 (1985)
22. Tsai, W.: Moment-preserving thresholding: a new approach. *Comput. Vision Grap. Image Process.* **29**, 377–393 (1985)
23. Kapur, J., Sahoo, P., Wong, A.: A new method for gray-level picture thresholding using the entropy of the histogram. *Comput. Vision Grap. Image Process.* **29**, 273–285 (1985)

Exploring the Suitability of a Genetic Algorithm as Tool for Boosting Efficiency in Monte Carlo Estimation of Leaf Area of Eelgrass

Cecilia Leal-Ramirez, Héctor Echavarría-Heras and Oscar Castillo

Abstract Eelgrass is an important seagrass species that grants ecological services of valuable economic relevance for mankind. On spite of its significance, human activities have infringed a notorious threat to the permanence of this species. Transplanting plays a fundamental role in restoration strategies, and assessing the success of concomitant plots requires non-invasive techniques. The use of allometric scaling and leaf area estimations derived from digital images, have provided simplified and accurate proxies for both standing crop and productivity. For regularly shaped leaves which produce nearly rectangular images, one key approach for the estimation of the involved areas is the use of the Monte Carlo method. Nevertheless, irregularities in the contour of leaves could make the computational time required to achieve a certain accuracy threshold burdensome. In this paper we explored the potential of a genetic algorithm as an agent for boosting the efficiency of the aforementioned technique. Our results show that the addition of the proposed code was able to markedly increasing the accuracy of regular Monte Carlo estimations while simultaneously boosting the computational efficiency of these procedures in a significant way.

1 Introduction

It is a widely acknowledged fact that ecosystems services provide a relevant contribution to human welfare. Constanza et al. [1] estimated the average economic value of ecosystem services at the level of the whole biosphere to be of the order of US\$33 trillion per year. Marine systems were found to contribute about 63 % of the

C. Leal-Ramirez (✉) · H. Echavarría-Heras
Centro de Investigación Científica y de Educación Superior de Ensenada,
Carretera Ensenada-Tijuana, No 3918, Zona Playitas, 22860 Ensenada, B.C., Mexico
e-mail: cleal@cicese.mx

O. Castillo
Tijuana Institute of Technology, Calzada Tecnológico s/n, Tijuana, B.C., Mexico

annual value, with almost half deriving from coastal systems. Approximately, 25 % of the overall contribution of coastal ecosystems corresponded to algal beds and seagrasses, which are submerged angiosperms that are considered a characteristic feature of shallow environments in tropical and temperate zones. *Zostera marina* commonly known as eelgrass is a particularly relevant seagrass species that distributes worldwide in shallow and nearshore environments, sustaining communities that are recognized as among the richest and most varied in the abundance of sea life [10]. Moreover, eelgrass is the dominant species along the coasts of both the North Pacific and North Atlantic [11]. This cosmopolitan and highly productive macrophyte was found to contribute up to 64 % of the total primary production of an estuarine system [15]. Eelgrass forms dense meadows that provide nursery habitats for many commercially important fish species [3]. Eelgrass beds also contribute to the remediation of contaminated sediments [16] filter and retain nutrients from the water column [12] help in the stabilization of sediments [14] and reduce erosion forces by stumping wave energy, thus promoting the stabilization of adjacent shorelines [4]. Nevertheless, in spite of these ecologically relevant services, the permanence of eelgrass beds is nowadays threatened by deleterious anthropogenic influences, and special restoration efforts must be enforced in order to ensure the conservation of this significant species. One commonly used restoration strategy is transplanting, and the monitoring of these efforts through time allows the evaluation of the restoration of the ecological properties that characterize natural populations. The noticeable growth form of eelgrass makes the biomass of the leaves a determinant of aboveground biomass, and the associated rate of growth of the leaves a consistent measure of productivity. Therefore, measurements of leaf biomass and its dynamics provide important information for the assessment of the overall status of an eelgrass population. Traditional, methods for the estimation of these variables require sampling procedures that include the removal of a representative number of shoots. Even though these invasive endeavors do not infringe harm to natural populations, the destruction of shoots during early stages of growth might provoke undesirable alterations in transplanted populations. Therefore, in order to eliminate these inconveniences within the assortment of tools for the conservation of eelgrass it is important to include procedures aimed to the non-destructive estimation of the biomass of the leaves, as well as, for the corresponding rates of growth.

Allometric models have been conventionally used in seagrass research, in order to adapt nondestructive proxies for biomass and productivity. Examples of these surrogates are the relationship between the width of the leaves and dry weight in *talassia testudinum* [9], the relationship between stem length and density in eelgrass [6], the relationship between the length of the leaves and their dry weight in eelgrass [7], and the relationship between the dry weight of eelgrass leaves and their length and width [5]. Moreover, Echavarria-Heras et al. [2] demonstrated that an allometric surrogate for eelgrass leaf-growth rates can be easily obtained in terms of allometric parameters for the scaling of leaf biomass and corresponding area. These authors also found that whenever allometric parameters for the scaling of eelgrass leaf dry weight in terms of leaf area are available, then simple leaf area measurements can provide accurate, cost-effective and non-destructive alternatives to

assessments based on traditional methods. But allometric scaling relationships are markedly sensitive to variation in the parameters involved. Even slight numerical differences in fitted values for these allometric parameters, along with errors of aggregation in calculating growth rates could induce important deviations between the observed and projected rates. Yet another important source of error relates directly to leaf area measurements. Therefore, for consistent allometric estimations of leaf biomass or productivity of eelgrass it is convenient to rely on accurate estimations of leaf area.

The observed ribbon-like appearance of the leaves in *Zostera marina* is a feature that permits obtaining direct estimations of observed blade length l and width h . These variables provide opportune estimations of the corresponding blade area AL_{lh} through the leaf length times width approximation [3], that is,

$$AL_{lh} = l \cdot h. \quad (1)$$

Nevertheless, leaf area estimates obtained by commercial electronic scanning devices like the LI-3000C portable leaf area meter confirm that the fairness and accuracy of the above approximation depends sensibly on leaf shape, that is, the closer the leaf contour is to a rectangle the higher the accuracy of the length times width approximation will be. Therefore, for the sake of accuracy we should rely on the use of automatic leaf area meters. Unfortunately, these scanning devices are expensive enough as to be unavailable for most eelgrass assessments practitioners and the need for non invasive methodologies justifies the quest for cost effective alternatives. For most eelgrass leaves the use of direct digital image processing or the merging of these techniques with the Monte Carlo method could deliver leaf area estimations that are simpler to obtain and also of a higher accuracy than the length times width approximation [8]. Nevertheless, the precision of estimations of leaf area of eelgrass through direct digital image processing depends on the resolution of the scanning device used to produce the images, and to this we add that the Monte Carlo method itself can also present inconveniences. Indeed, in order that this procedure delivers fast and accurate leaf area estimations, it is desirable that the area to be estimated could be enclosed by a piecewise continuous arrangement of smooth curves. Nevertheless, when the contour of the objective region is irregular, the computational procedures required to obtain the enclosed area could be burdensome. Particularly, for eelgrass leaves, observations show that the effects of herbivory or drag forces can induce noticeable changes in shape (see Fig. 1).

These alterations could be of such an extent that it might be difficult to represent the peripheral contour of the associated digital image through an analytical arrangement of smooth curves. This effect is mainly present in older and longer leaves, which have been exposed to environmental influences for larger amounts of time than newly produced ones. Examinations demonstrate that the pool of irregular or damaged leaves could even reach 40 % of available data. Yet another inconvenience related to size is that some newly formed leaves may have reduced enough dimensions as to induce ambiguity in the identification of the contour of images. Therefore, when an automatic leaf area meter is not available and a cost-effective

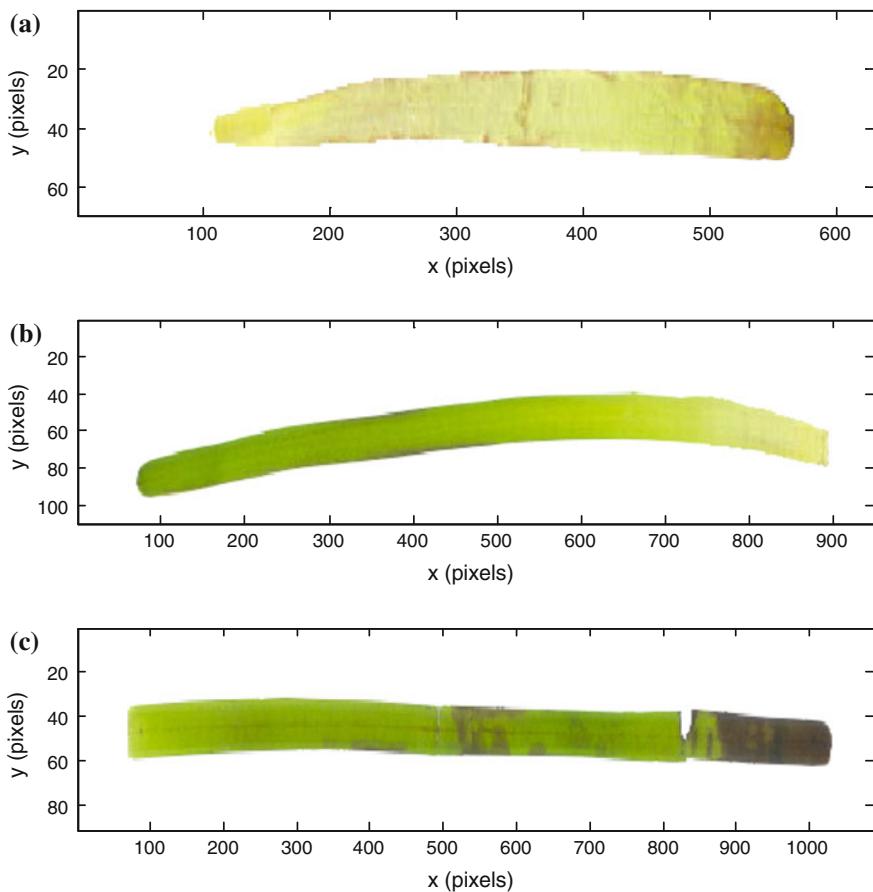


Fig. 1 **a** An eelgrass leaf exhibiting the smallest width at the base. **b** A leaf with marked curvature. **c** The image of the leaf shows the effects of herbivory

and non-invasive alternative for the estimation of the area of leaves in eelgrass is required, we could still rely on the use of combined digital imagery and the Monte Carlo method, but the chore requires the adaptation of methodologies aimed to improve the efficiency and accuracy of the linked estimation of area. In this work we explore the capabilities of a genetic algorithm as a tool to achieve the aforementioned task.

2 Methods

The aim of this section is the description of a genetic algorithm as conceived in order to improve the accuracy of estimations of eelgrass leaf area. Before we proceed into the subject we briefly explain the fundamental ideas behind the

Monte Carlo method for the estimation of the area enclosed by an arrangement of regular curves.

2.1 Monte Carlo Estimation of Plane Areas

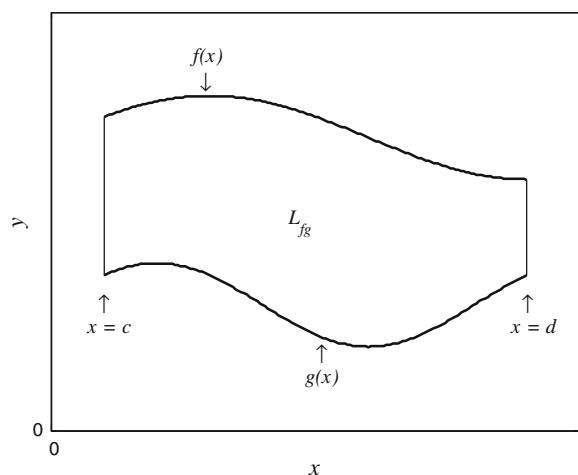
Without loss of generality we will focus in a plane figure contained in the Cartesian plane R^2 and bounded above and below by the plots of two non-intersecting smooth functions. In order to describe the assemble formally, let's consider a couple of real valued and continuous functions $f(x)$ and $g(x)$ depending on an independent variable x defined on a domain $D = \{x|c \leq x \leq d\}$ contained in the set of real numbers R . Then, if additionally, the inequality $f(x) > g(x)$ holds through D , the area AL_{fg} of the domain $L_{fg} = \{(x, y)|c \leq x \leq d, f(x) \leq y \leq g(x)\}$ (see Fig. 2) is given by,

$$AL_{fg} = \int_c^d (f(x) - g(x))dx \quad (2)$$

However, if the form of the function $f(x) - g(x)$ is complicated, then it could be difficult or even impossible to obtain the integral (2) analytically. The Monte Carlo method estimates the area of the region L_{fg} by means of repeated random tests. The method firstly draws a rectangle $\Omega = \{(x, y)|x_a \leq x \leq x_b, y_a \leq y \leq y_b\}$ that encloses the domain L_{fg} whose area AL_{fg} needs to be calculated. Since the length and width of the rectangle Ω are known, its area $A\Omega$ is given by (see Fig. 3)

$$A\Omega = (x_b - x_a) * (y_b - y_a). \quad (3)$$

Fig. 2 The domain L_{fg} bounded above and below by the plots of the non-intersecting functions $f(x)$ and $g(x)$ and that extends from $x = c$ through $x = d$



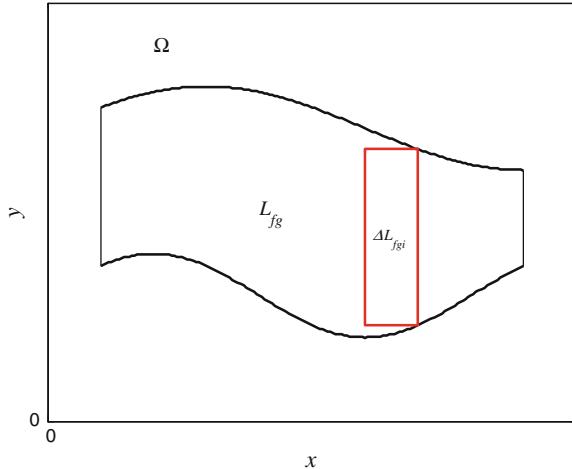


Fig. 3 The goal area AL_{fg} of a domain L_{fg} bounded above and below by the plots of $f(x)$ and $g(x)$, respectively for $c \leq x \leq d$ is placed within a reference rectangle $\Omega = \{(x, y) | x_a \leq x \leq x_b, y_a \leq y \leq y_b\}$; with $x_a < c < d < x_b$. The region L_{fg} can be approximated by a collection of rectangles $\Delta L_{fgi}, 1 \leq i \leq k$

Assume that a number $np(\Omega)$ of points in Ω are randomly chosen, and that we record the number $np(L_{fg})$ of these points that were selected from the region L_{fg} . Suppose that the experiment is repeated m times and that we allow $np(\Omega)$ to become arbitrarily large. Assume that, the sampled points were drawn from a uniform distribution. Then, as m steadily increases, we will observe that the ratio $np(L_{fg})/np(\Omega)$ approaches a fixed number $P(L_{fg})$ and we can state that the limiting value of $P(L_{fg})$ is the probability that a point lies in the region L_{fg} , which is given by the ratio $AL_{fg}/A\Omega$, that is, we can formally establish the result,

$$AL_{fg} = P(L_{fg})A\Omega. \quad (4)$$

Moreover, since $P(L_{fg})$ can be approximated by the ratio $np(L_{fg})/np(\Omega)$, we can write Eq. (4) in the equivalent form,

$$AL_{fg} = AL_{fgm} + e_m, \quad (5)$$

where

$$AL_{fgm} = \frac{np(L_{fg})}{np(\Omega)} A\Omega, \quad (6)$$

is the Monte Carlo approximation to AL_{fg} obtained when a number $np(\Omega)$ of points is randomly drawn from Ω and e_m stands for the involved approximation error.

The procedure used to obtain an average value AL_{fgm} for the ratio AL_{fgm} taken over a number of simulation trials, will be called through, Monte Carlo method to calculate the area enclosed by a domain $L_{fg} = \{(x, y) | c \leq x \leq d, f(x) \leq y \leq g(x)\} \subset R^2$, or simply Monte Carlo method. The overall approximation error e_a is given by the absolute deviation between the real or observed area AL_{fgo} and its Monte Carlo proxy AL_{fgm} that is,

$$e_a = |AL_{fgo} - \langle AL_{fgm} \rangle| \quad (7)$$

If we want to estimate the area of an eelgrass leaf by using the Monte Carlo method we will firstly need to adapt the dimensions of the reference rectangle Ω , and for effectiveness to embed as it is explained above the target leaf image within boundaries described by curves $f(x)$ and $g(x)$ of a known mathematical form. This characterizes the leaf image as a region L_{fg} contained in the reference rectangle Ω . Whenever, the closed form of $f(x)$ and $g(x)$ is known it is possible to assess in an automatic way, if a randomly generated point lies in the interior of the leaf image L_{fg} . Nevertheless, when the leaf presents irregularities like those shown in Fig. 1, the task of characterizing $f(x)$ and $g(x)$ analytically could not be achieved in a simple way. In that event, we will depend on placing a suitably large amount $np(\Omega)$ of randomly generated points in the reference rectangle Ω , and then counting in a direct way the number $np(L_{fg})$ that fall within the leaf image L_{fg} itself, which becomes a difficult task. Moreover, achieving this task will require slicing the leaf image by means of a partition of disjoint rectangles ΔL_{fgi} as shown in Fig. 4b.

A suitable number k of screen rectangles is a prime requirement for efficiency, because the larger the number of cells included in the partition, the more accurate the estimation of the upper and lower boundaries of L_{fg} will become. But even though an optimal partition has been chosen, given the amount of points that must be placed in the interior of a leaf for an efficient estimation of its area, if a leaf has a long length and presents as well, irregularities like those shown in Fig. 1, the involved processing time of Monte Carlo procedures could be costly. It is worthwhile to stress, that besides the computational burden of testing the placement of points within the image in every single run, the Monte Carlo method requires also, a large number of replicates for averaging purposes, which could rise the overall processing time in a significant way.

2.2 Genetic Algorithm for Numerical Operations and Processing Time Optimization in the Monte Carlo Method

A genetic algorithm is a searching process inspired by the laws of natural selection and genetics in which a simulated process of evolution is used to obtain the solution to a certain optimization problem. At first, a population of candidate solutions is

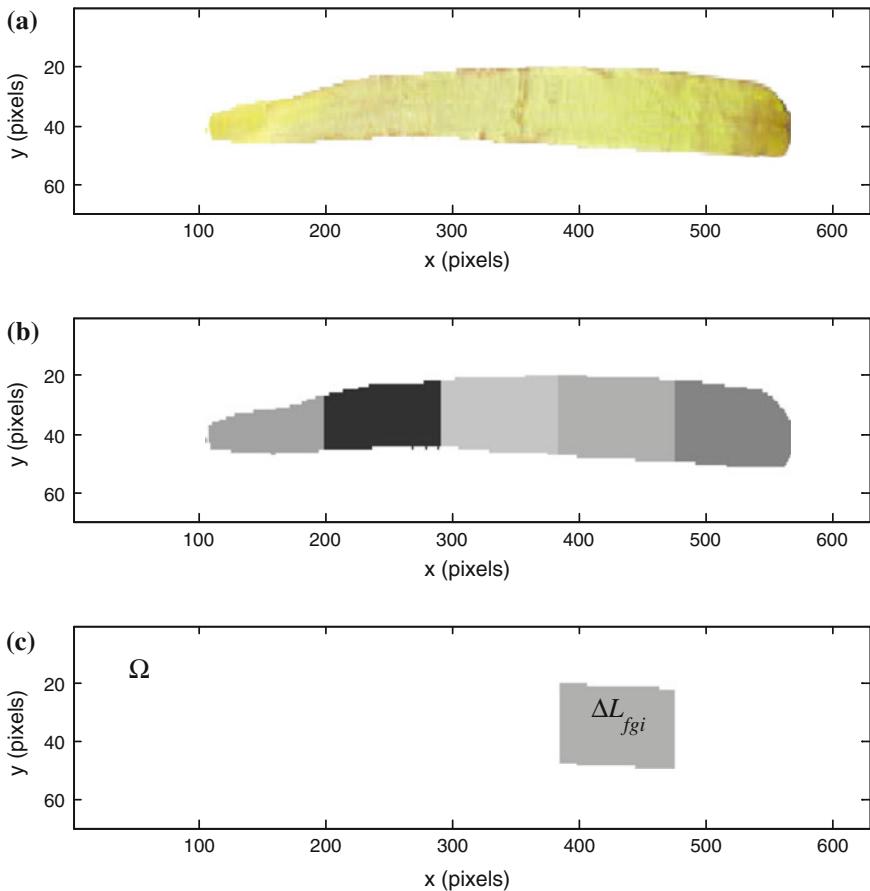


Fig. 4 **a** A leaf image. **b** Partition of the leaf image using rectangular cells ΔL_{fgi} . **c** A single partition cell ΔL_{fgi} in Ω is exhibited. Using the Monte Carlo method to approximate the area of the leaf image shown in **b** requires counting the number of points randomly selected out of the region Ω that were picked up from this cell. In order to estimate the leaf area, this procedure must be performed recursively until all partition cells covering the leaf image L_{fg} are exhausted

generated randomly. A candidate solution is also called individual, creature or phenotype. Each individual has a genotype or set of chromosomes which define its properties. Then, by evaluating the objective function for each one of the individuals, the algorithm obtains the fitness values of the corresponding genotypes. In adapting a genetic algorithm for an optimal performance of the Monte Carlo method for the estimation of the area of the region L_{fg} , it will be required to consider a fixed number of individuals N_{ind} , each individual representing a number $np(\Omega)$ of points (i, j) to be randomly generated a number N_{sample} of times over Ω . Next the algorithm scales the raw fitness scores to convert them into a more usable range of values, and selects members, called parents, based on their fitness. The processes of

mutation and crossover are applied to the parents to generate new individuals. Children with better scores are selected and used to replace their parents in forming a new generation; this with the aim of having a population evolved toward better performance. Here, we symbolized a mutation rate by means of m_r and a crossover rate through c_r . By applying an iterative cycle, the genetic algorithm repeatedly modifies the population of candidate solutions improving fitness. The algorithm stops when one of the stopping criteria, that could be either a value e_a of the approximation error or a number of generations N_{gen} , is met.

3 Results

The genetic algorithm as conceived here is intended to optimize both the number of iterations and the overall processing time required by the standard Monte Carlo method to count the number of random points distributed over a collection of rectangles or cells that approximate the target area L_{fg} (Fig. 4b). If the procedure is able to optimize the computing task required to estimate the area of a particular cell (Fig. 4b), then optimizing the number of calculations required to estimate the whole ensemble of cells that approximate L_{fg} will be a matter of parallel processing. This problem will be addressed elsewhere; meanwhile we focus here in the problem of optimizing the time of execution of the Monte Carlo method over an individual rectangular cell (Fig. 4c). The size of the cell selected for illustrating the performance of the genetic algorithm is similar to the size of a small newly produced eelgrass leaf L_{lhs} . Moreover, for a small blade, typical observed dimensions are $l = 1.35$ cm and $h = 0.19$ cm. The area AL_{lhs} as given by Eq. (1) is $AL_{lhs} = 0.257$ cm². To ascertain the relative dimensions of this leaf we might quote that the length of a *Zostera marina* leaf can be expected to of the order of 1141 mm [13] while average width could be 5.5 mm [3]. Therefore, if we consider another leaf L_{lhb} , longer and older than L_{lhs} , having the above quoted dimensions, its area AL_{lhb} , when calculated using Eq. (1) becomes $AL_{lhb} = 62.755$ cm². It turns out that $AL_{lhb} = 244.18AL_{lhs}$. Then, the area of a small leaf L_{lhs} could be comparable to the area of one of the rectangles in a partition to approximate the image of L_{lhb} . Therefore, in order to test the performance of the considered genetic algorithm we will focus in a rectangular region L_{fgo} having an area comparable to AL_{lhs} . Indeed, for this example we choose, $x_a = 0.0$, $x_b = 1.29$, $y_a = 0.0$, $y_b = 0.25$. Therefore, we have $\Omega = \{(x, y) | 0.0 \leq x \leq 1.29, 0.0 \leq y \leq 0.25\}$, with the variation of x and y expressed in cm. In order, to characterize L_{fgo} we chose $f(x) = 0.254$, and $g(x) = 0.0254$, that is, we have $L_{fgo} = \{(x, y) | 0.0254 \leq x \leq 1.2954, 0.0254 \leq y \leq 0.254\}$. Hence, in order to determine if a point (x, y) lies within the target area AL_{fgo} we verified the following conditions,

$$0.0254 \leq i \leq 1.2954 \text{ and } 0.0254 \leq j \leq 0.254 \quad (8)$$

The exact area to be estimated is $AL_{fgo} = 0.290322 \text{ cm}^2$. For $np(\Omega) = 4730$ the regular Monte Carlo method obtained an estimation AL_{fgm} with an approximation error of $e_a = 1.0 \times 10^{-3}$ and with a processing time of $t = 2 \text{ min}$ (Fig. 5). Then, using this approach to estimate the area AL_{lhb} of a typically long leaf would require a total processing time of $t = 524 \text{ min}$. In order to test the performance of the device resulting by merging the Monte Carlo technique and the conceived genetic algorithm, we performed several simulation runs setting a value of $e_a = 1.0 \times 10^{-3}$ as a search stopping condition. This stopping condition set also the number of generations N_{gen} required by a simulation to estimate AL_{fgo} . Figure 6 portrays the results of the simulation that shows that when utilizing $np(\Omega) = 5875$, the stopping condition was attained at the 35th generation, elapsing a total processing time of $t = 12 \text{ s}$. In turn, a recursive application of the amended method will require a total processing time of $t = 52 \text{ min}$, which amounts to a considerable reduction in processing time relative to the normal Monte Carlo approach.

It is a well-known fact, that the conceived values of the m_r and c_r rates, as well as, the methods involved in the generation of new individuals, could boost the efficiency of a genetic algorithm. In our study case when we set $m_r = 0.3$ and $c_r = 0.3$. We obtained better results (see Fig. 6). This result undoubtedly exhibits the advantage of using a genetic algorithm for boosting efficiency in calculating the area of the chosen cell, AL_{fgo} and ultimately in the overall parallel processing time required to estimate the many areas of the cells composing a whole eelgrass leaf.

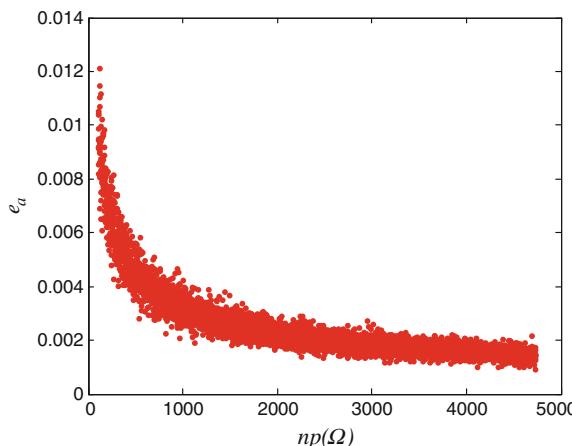


Fig. 5 When approximating the objective area AL_{fgo} the regular Monte Carlo method elapsed a total processing time of $t = 2 \text{ min}$ before reaching the stopping condition $\epsilon_a = 1.0 \times 10^{-3}$. The task required a number $np(\Omega) = 4730$ of randomly placed points over Ω

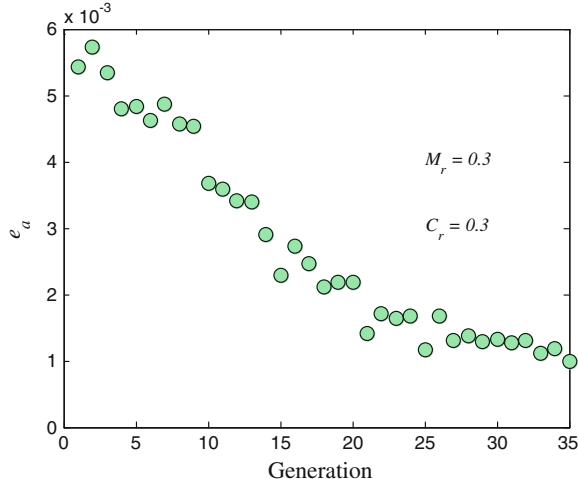


Fig. 6 When estimating AL_{fgo} , the arrangement resulting by merging the regular Monte Carlo method and the conceived genetic algorithm required a total processing time of $t = 12$ s before the stopping condition $e_a = 1.0 \times 10^{-3}$ was achieved. This stopping condition was attained when a number $N_{gen} = 35$ of generations was reached. The task required a total of $np(\Omega) = 5875$, randomly selected points from Ω

4 Discussion and Conclusion

Eelgrass leaves exhibit a characteristic belted shape that can be usually approximated by means of a long and thin rectangle. This has justified the widespread use of Eq. (1) to estimate the associated area [3]. Nevertheless, Fig. 1 shows images of leaves that do not conform to the regularly observed belted shape. The difficulties that the estimation of the area of these irregular leaves poses have stimulated the quest for alternative efficient, simplified and non-destructive estimations. By counting the number of pixels contained in the image of a leaf and knowing the image resolution, that is the number of pixels in a squared centimetre, leaf area estimation reduces to calculating a simple proportion. Hence, direct Image processing techniques could potentially become a convenient cost effective and non-invasive approach [8]. Nevertheless, in practice the amount of pixels that an image can enclose will be limited by the technical features of the device used to produce it. Most commercial scanners sustain a maximum resolution which seems unsuitable for highly precise estimation, and as a result counting pixels in an image could underestimate the real area in the leaf. Therefore, heightening precision in the direct digital image processing method requires increasing the resolution of the leaf image which could eventually lead to acquiring a costly automatic leaf area meter. The Monte Carlo method has been traditionally used to estimate the area of plane regions and can be adapted to produce estimates of the area of images of eelgrass leaves [3]. In comparison with the direct image processing method in which

boosting precision implies increasing the resolution of the leaf image, when using the Monte Carlo method enhancing precision will entail increasing the number of points randomly distributed over the leaf image. This could increase processing time in such a way that this procedure would become computationally inefficient. As a matter of fact, our results show that in order to produce a reasonably accurate estimation of the area of certain leaves (e.g. AL_{lhb}) the regular Monte Carlo method could be expected to elapse a processing time up to $t = 524$ min, which is onerous, considering that in eelgrass a regular assessment task, in some cases, would require to analyze a sampling batch up to 809 leaves [3]. The approach we present here can overcome the inherent difficulties by allowing the Monte Carlo method to become an efficient device when estimating the areas of eelgrass leaves from images. Indeed our results show that the genetic algorithm addendum to regular Monte Carlo technique produces an outstanding reduction in the essential processing time for estimating AL_{lhb} , this even applying the device in a recursive form as it happens in the regular operational mode of the Monte Carlo approach. Moreover, by considering that a genetic algorithm allows the consideration of parallel processing, the benefits in computational time reduction derived from its used could be expected to be extraordinary. Consequently, the construct presented here can be considered a valuable tool that permits digital images of eelgrass leaves taken by commercially available scanners to provide reliable estimations of the associated area. Since, these estimations could reduce the uncertainty in allometric estimations of the pertinent aboveground biomass and leaf growth rates. Hence, the present techniques can greatly contribute to the conservation of the considered seagrass species.

References

1. Costanza, R., d'Arge, R., de Groot, R., de Groot, R., Farber, F., Garsso, M., Hannon, B., Limburg, K., Naeem, S., O'Neill, R.V., Paruelo, J., Raskin, R.G., Sutton, P., van den Belt, M.: The value of the world's ecosystem services and natural capital. *Nature* **387**, 253–259 (1997)
2. Echavarria-Heras, H., Solana-Arellano, E., Franco-Vizcaino, E.: An allometric method for the projection of eelgrass leaf biomass production rates. *Math. Biosci.* **223**, 58–65 (2009)
3. Echavarria-Heras, H., Solana-Arellano, E., Leal-Ramirez, C., Franco-Vizcaino, E.: The length-times-width proxy for leaf area of eelgrass: criteria for evaluating the representativeness of leaf-width measurements. *Aquat. Conserv. Mar. Freshw. Ecosyst.* **21**(7), 604–613 (2011)
4. Fonseca, M.S., Fisher, J.S.: A comparison of canopy friction and sediment movement between four species of seagrass with reference to their ecology and restoration. *Mar. Ecol. Prog. Ser.* **29**, 15–22 (1986)
5. Hamburg, S.P., Homman, P.S.: Utilization of Growth parameters of eelgrass *Zostera marina* for productivity estimation under laboratory and insitu conditions. *Mar. Biol.* **93**, 299–303 (1986)
6. Jacobs, R.P.W.M.: Distribution and aspects of the production and biomass of eelgrass, *Zosteramarina L.* at Roscoff, France. *Aquat. Bot.* **7**, 151 (1979)
7. MacRoy, C.P.: Standing stock and other features of eelgrass (*Zostera marina*) populations on the coast of Alaska. *J. Fish. Res. Bd. Canada* **27**, 1811–1812 (1970)

8. Leal-Ramirez, C., Echavarria-Heras, H.: A method for calculating the area of *Zostera marina* leaves from digital images with noise induced by humidity content. *Sci. World J.* **2014**, 11 (2014)
9. Patriquin, D.G.: Estimation of growth rate, production and age of the marine angiosperm, *Thalassiatestudinum*. *Konig. Carib. J. Sci.* **13**, 111–123 (1973)
10. Phillips, R.C.: Temperate grass flats. In: Odum, H.T., Copeland, B.J., Mc Mahan, E.A. (eds.) *Coastal Ecological Systems of the United States: A Source Book for Estuarine Planning*, vol. 2, pp. 737–773. F.W.P.C.A. Contract RFP 68–1282, USA (1969)
11. Short, F.T., Coles, R.G., Pergent-Martini, C.: Global seagrass distribution. In: Short, F.T., Coles, R.G. (eds.) *Global Seagrass Research Methods*, pp. 5–30. Elsevier Science B.V., Amsterdam, The Netherlands (2001)
12. Short, F.T., Short, C.A.: The seagrass filter: purification of coastal water. In: Kennedy, V.S. (ed.) *The Estuary as a Filter*, pp. 395–413. Academic Press, Massachusetts (1984)
13. Solana-Arellano, M.E., Echavarria-Heras, H.A., Ibarra-Obando, S.E.: Leaf-size dynamics for *Zostera marina* L. in San Quintin Bay, México: a theoretical study. *Estuar. Coast. Shelf Sci.* **44**, 351–359 (1997)
14. Ward, L.G., Kemp, W.M., Boynton, W.R.: The influence of waves and seagrass communities on suspended particulates in an estuarine embayment. *Mar. Geol.* **59**(1–4), 85–103 (1984)
15. Williams, R.B.: Nutrient level and phytoplankton productive in the estuary. In: Chabreck, R. A. (ed.) *Proceedings of the coastal marsh and estuary management symposium*, pp. 59–89. Louisiana State University, Baton Rouge, USA (1973)
16. Williams, T.P., Bubb, J.M., Lester, J.N.: Metal accumulation within salt marsh environments. *Mar. Pollut. Bull.* **28**(5), 277–289 (1994)

Obtaining Pharmacokinetic Population Models Using a Genetic Algorithm Approach

Oscar Montiel, J.M. Cornejo, Carlos Sepúlveda
and Roberto Sepúlveda

Abstract Nonlinear mixed effect model is the most used technique when developing a pharmacokinetic population model (PopPK), the characterization of a drug disposition into the body and taking decisions related to the dose adjustments. The covariate model is used to establish a relationship between the model parameters and the characteristics of the patients, and it helps to explain sources of variability in the PopPK. A known problem in the development of a covariate model is to decide which covariates should or should not be included in the model. In this work, a genetic algorithm (GA) was used to decide which covariates contribute in a major degree prediction of the variability in a PopPK model.

1 Introduction

The implementation of new mathematical methods into biology and medicine has been increasing in the last three decades. This owes in most, to the rise of the speed in the computers, which has allowed novel developments of biological system models [1]. PopPKs are a good example of the aforementioned; they are designed to study the drug behavior into a group of individuals. Clinical observations have demonstrated that the efficiency and toxicity of a drug depend on plasmatic concentration (C_P) [2];

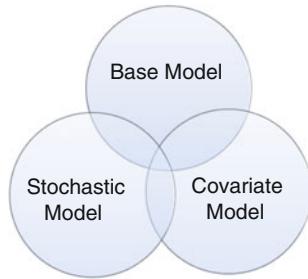
O. Montiel (✉) · J.M. Cornejo · C. Sepúlveda · R. Sepúlveda
Instituto Politécnico Nacional, Centro de Investigación y Desarrollo de
Tecnología Digital (CITEDI-IPN), Av. Del Parque no. 1310, Mesa de Otay,
22510 Tijuana, B.C., Mexico
e-mail: oross@ipn.mx

J.M. Cornejo
e-mail: jmcornejo@uabc.edu.mx

C. Sepúlveda
e-mail: csepulveda@citedi.mx

R. Sepúlveda
e-mail: rsepulvedac@ipn.mx

Fig. 1 Components of a population model



for this reason, mathematical models have been designed to describe the relation among *Dose – C_p – Effect* in an individual. Nevertheless, these models require of optimal inferences over their parameters. Mixed effects modeling is the main approach for the development of PopPKs. These models are comprised of three main components: a base model, a stochastic model, and a covariate model, Fig. 1, [3].

The developing of a covariate model is one of the most important tasks in PopPKs, it gives us the opportunity to understand sources of variability and explain them in terms of parameters and response [4]. However, the identification of potential covariates may be time-consuming, and several statistical methods have been proposed for this task. The aim of this chapter is to provide a genetic algorithm (GA) approach for the identification of potential covariates, and compare it with statistical methods used in mixed effects modeling. In order to achieve this aim, we develop the PopPK of Tobramycin in Matlab in order to carry out the covariate model analysis and the identification of their potential covariates.

2 The Mixed Effects Model

In mixed effects models, the data are in the form of repeated measurements on the same subject (using the word subject as an experimental unit) or longitudinal data [5]. For the reader, the subject will be the population under study, and the sample will be the quantity of individuals that we select to represent this population, see Fig. 2.

As we mentioned, mixed effects modeling is the main approach for the development of PopPKs; the term ‘mixed’ refers to the effects of random quantities (e.g. between subject variability, residual variability, etc.), and ‘fixed’ effects (e.g. population parameters) which are typical values of parameters. To derive these parameters, we need to choose a base PK model:

$$Y = f(x, \beta) + \varepsilon, \quad (1)$$

where Y is a vector of observations, $\mathbf{Y} = \{Y_{1,1}, Y_{1,2}, \dots, Y_{n,1}, Y_{n,2}, \dots, Y_{ni,nj}\}^T$, x is a matrix of design variables (including time), β is the vector of model parameters, f is a function that depends on β in a nonlinear manner, and ε is the residual error that

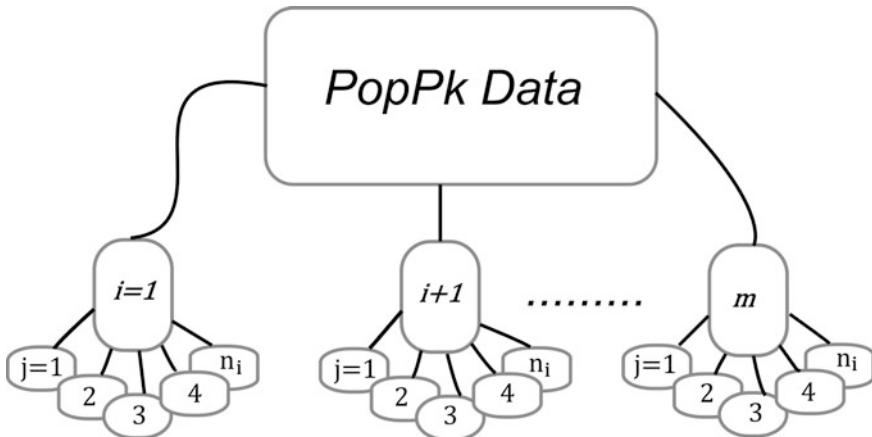


Fig. 2 Population data in pharmacokinetics

refers to the deviation of measured drug concentration from the predicted level in a specific time, and it is considered to be a variable with $\varepsilon \sim N(0, \sigma^2)$ [1].

A base PK model refers to a specific compartmental PK model, where compartmental models represent the body as a number of well-stirred compartments. An example of this is shown in Fig. 3, where a one-compartment model with a drug being administered as a single intravenous dose is illustrated. An example of a base model is shown in (2).

$$C_p = \frac{D}{V_d} \cdot e^{-\frac{Cl}{V_d} t}, \quad (2)$$

The model represents the relationship between the dependent variable drug concentration (C_p), and the independent variable time (t), whereas volume of distribution (V_d) and clearance (Cl) are fixed parameters that describe the effect of a given dose (D) [6, 7]. The covariate model represents relationships between covariates and model parameters using fixed effects parameters, that is, we can explain population PK parameters in terms of covariates,

Equation	Representation	Base Model in Matlab
$C_p = \frac{D}{V_d} \cdot e^{-\frac{Cl}{V_d} t}$		<pre> pkmd = PKModelDesign; pkmd.addCompartment('Central', 'Bolus', 'linear-clearance', true); [model, modelMap] = pkmd.construct;</pre>

Fig. 3 One compartment model, and to the right side the mathematical expression representing the processes of absorption, distribution, and elimination of a drug into the body

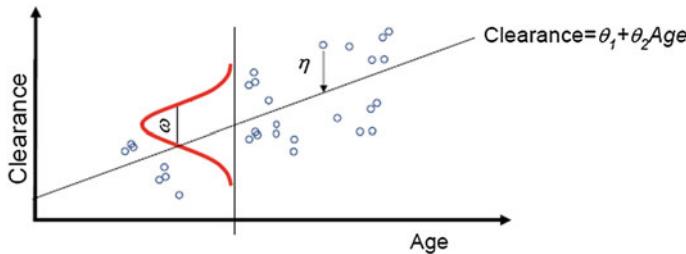


Fig. 4 Distribution of interindividual variability

$$Cl = \theta_1 + \theta_2 Age + \eta, \quad (3)$$

where Cl is defined in terms of linear function of Age; θ_1, θ_2 are the intercept and slope respectively; η is considered to be a variable that describes inter individual variability with $\eta \sim N(0, \omega^2)$, see Fig. 4.

2.1 Estimation of Population Parameters

Linear mixed effect models are not commonly seen in population pharmacokinetic data. Instead of them, nonlinear mixed effects models are common in the analysis of pharmacokinetic data, and their parameter estimation is tied to the methods of maximum likelihood. As we have seen, the random effects act as parameters, and for instance, they need to be estimate together with fixed effects. These parameters are estimated by maximum likelihood estimation based on marginal density of Y ,

$$p(Y_i|\beta, \sigma^2, \Omega) = \int p(Y_i|\beta, \sigma^2, \eta_i) p(\eta_i|\Omega) d\eta_i, \quad (4)$$

where Y_i is the vector of drug concentrations of i th subject, $p(Y_i|\beta, \sigma^2, \eta_i)$ is the conditional probability density of observed data, while $p(\eta_i|\Omega)$ is the conditional density of η_i , where Ω is the variance covariance matrix for the random effects. The integral (4) does not have a closed form, so that different approximation methods can be apply (e.g. first order methods). The objective functions for each approximation method is numerically minimized with regard to the parameters $(\beta, \sigma^2, \Omega)$ [8].

2.2 Covariate Model Development

The term “covariate” is very important in PopPK modelling [9], and it is any variable that is specific to an individual and may influence the pharmacokinetics of a drug (e.g. Age, Weight, Height, etc.). The covariate model development can be a

difficult task, especially if there are a large number of covariates to consider. Besides selecting important covariates, we need to specify the function form of the covariate model. Some of the covariate models often used are [1]:

- Linear additive : $Cl = \theta_1 + \theta_2 \cdot (\text{covariate})$ (5)

- Exponential Model : $Cl = \theta_1 e^{\theta_2 \cdot (\text{covariate})}$ (6)

- Exponential Ln – Transformed $\ln Cl = \ln \theta_1 + \theta_2 \cdot (\text{covariate})$ (7)

The selection of covariate models depends on the modeler and there is no consensus regarding who is the best model.

2.3 Covariate Screening Methods in PopPK

One of the most important aspects in the construction of a covariate model is to identify, which covariates are significant. Several statistical methods for covariate screening have been proposed, and particularly, they are used when there is a large number of covariates. Nevertheless, there is a lot of works regarding the statistical methods for the covariate selection and their performance, and among these methods, there are less based on the theory of natural selection like GAs. In this section, we present a brief review of one of the frequently used statistical methods for covariate selection, in the analysis of nonlinear mixed effects models; as well as a GA for covariate selection and the theory behind it.

2.4 Stepwise Multiple Linear Regression for Covariates Selection

In this method, the individual parameter estimates (β_i), obtained from a model without covariates, are regressed on the individual covariates (x_i). Its main features are that the models for the covariates are restricted to being linear.

$$\beta_i = \theta_0 + \theta_1 \cdot x_1 + \theta_2 \cdot x_2 + \cdots + \theta_n \cdot x_n \quad (8)$$

where θ_0 is the intercept, and $\theta_1, \theta_2 \dots \theta_n$ are linear regression coefficients. The model is built in a stepwise addition/deletion approach, that is, it is a combination of forward selection, and backward elimination, where each variable is tested to be included or excluded in the model. The added term that decreases the Akaike information criteria (AIC) to the greatest extent is retained in the model.

$$AIC = n \log(2\pi) + n \log \widehat{\sigma^2} + n + 2(p + 1) \quad (9)$$

AIC is proportional to the residuals sum of squares $\widehat{\sigma^2}$, plus a penalty for the number of parameters p . The procedure stops when no further decrease in AIC value is achievable [10].

3 Genetic Algorithms

The GAs are algorithms that are used for optimization, search, and learning tasks, which are inspired in the natural evolution processes. The concept of GAs was proposed first by Holland in the 1960s [11]. In the simple GA, an individual is represented by a fixed-length bit string, and the population is a collection of N chromosomes (set of possible solutions) that are randomly initialized. Each position in the string is assumed to represent a particular feature of an individual (bits, or genes), and the value stored in that position represents how that feature is expressed in the solution [12]. The solutions to a given problem are evaluated by a fitness level, and the fitter individuals (parents) are randomly selected; then a reproduction operator is used to generate new individuals. The main reproduction operator is the bit-string crossover, in which two strings are used as parents and new individuals are formed by swapping a sub-sequence between the two strings.

The main components of a genetic algorithm can be summarized as follows [13]:

- Initial population: Usually consist of a random generation of solutions to the given problem.
- Representation: Correspondence between the feasible solutions (phenotype) and the coding of the variables or representation (genotype).
- Evaluation function: Determines the quality of the individuals of the population.
- Operators: To promote evolution.

3.1 Genetic Algorithm Operators

The GAs are probabilistic methods that get new individuals; they tend to be dependent on the representation, usually, the operators are used for selection, crossover and mutation [13].

3.1.1 Selection

The selection is a method that allows you to choose a set of individuals of the population with higher fitness as parents of the next generation. The selection criterion is usually assigned to individuals with a probability proportional to its

quality. There are two basic types of selection scheme in common usage: proportionate and ordinal selection. Proportionate selection picks out individuals based on their fitness values relative to the fitness of the other individuals in the population [13]. Examples of such a selection type include roulette-wheel selection which can be visualized as spinning a one-armed roulette wheel, where the sizes of the holes reflect the selection probabilities [14]. Ordinal selection selects individuals based upon their fitness, but upon their rank within the population. The individuals are ranked according to their fitness values [13].

3.1.2 Crossover Operator

Crossover exchanges and combines a set of genes from the parents to generate new individuals (offspring), according to a crossover probability. The simplest way how to perform crossover is to choose randomly some crossover point and copy everything before this point from the first parent and then copy everything after the crossover point from the other parent [13, 15], see Fig. 5.

3.1.3 Mutation Operator

This operator is used to maintain genetic diversity among generations. The idea is to alter one or more gene values in the chromosome randomly. This operator provides to the algorithm with exploratory properties, see Fig. 6.

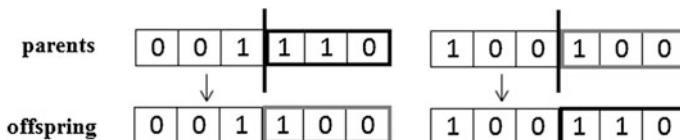


Fig. 5 Example of one-point crossover

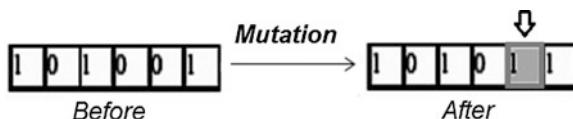


Fig. 6 Example of mutation

4 Population Pharmacokinetics of Tobramycin

The PopPK covariate model of Tobramycin and its analysis was developed in Matlab. The identification of the potential covariates and the reduction of variability in the population model were achieved through a stepwise method and a GA. The process is shown in Fig. 7. An example of a final model is shown in the equation below.

$$C_p = \frac{D}{V_d} \cdot e^{\left[-\frac{(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \eta)t}{V_d} \right]} + \varepsilon \quad (10)$$

4.1 Population Data Base

The database used in this work consists of 250 samples taken from 78 individuals, see Table 1. The purpose of this analysis was to characterize the population pharmacokinetics of Tobramycin, which is an antibiotic used to treat infections, and it is notorious for their narrow therapeutic index.

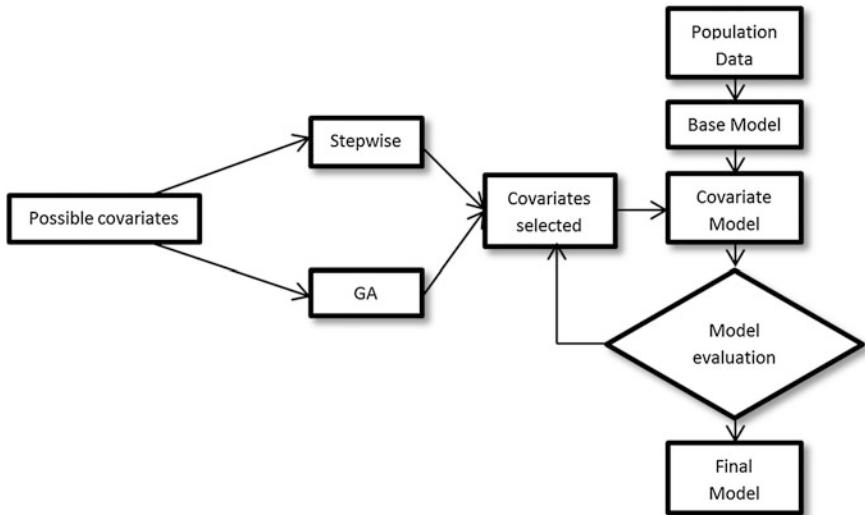


Fig. 7 Process for developing a PopPK model

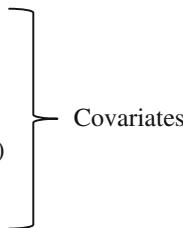
Table 1 Database sample from the individual 21. After importing the data for the individual 21, only one concentration was obtained

ID	TIME	CP	DOSE	WT	AGE	HT	BSA	BMI	IBW	LBW
21	0	-	120	71.3	39	170	1.83	24.67	66.02	55.04
21	8	-	60	71.3	39	170	1.83	24.67	66.02	55.04
21	16	-	60	71.3	39	170	1.83	24.67	66.02	55.04
21	24	-	60	71.3	39	170	1.83	24.67	66.02	55.04
21	32	-	60	71.3	39	170	1.83	24.67	66.02	55.04
21	40	-	40	71.3	39	170	1.83	24.67	66.02	55.04
21	48	-	40	71.3	39	170	1.83	24.67	66.02	55.04
21	56	-	40	71.3	39	170	1.83	24.67	66.02	55.04
21	64	-	40	71.3	39	170	1.83	24.67	66.02	55.04
21	72	-	40	71.3	39	170	1.83	24.67	66.02	55.04
21	75	-	-	71.3	39	170	1.83	24.67	66.02	55.04
21	80	1.1	40	71.3	39	170	1.83	24.67	66.02	55.04

Note that we have written a dash where no information was available

The database consists of the following information:

- ID: Is the number that identifies each individual.
- TIME: Time in hours (hrs)
- DOSE: Dose in milligrams (mgs)
- CP: Plasmatic concentration
- WT: Weight (kg)
- AGE: (years)
- HT: Height (cm)
- BSA: Body surface area (m^2)
- BMI: Body mass index (kg / m^2)
- IBW: Ideal body weight (kg)
- LBW: Lean body weight (kg)



4.2 Base Model Development

The single compartment model was used to obtain the covariate model, it was implemented in Matlab, as it is shown in Fig. 3.

The base model for parameter Cl was defined as linear. The main reason to focus in the parameter clearance is because it is the most important of all the pharmacokinetic parameters [2]. The initial values of the fixed effects were 0.01 for Cl , and 0.01 for V_d , with a maximum number of iterations of $1e^{-04}$. Additionally, an exponential error model was used to model random effects parameters including inter individual variability and residual error. The final model was estimated using restricted maximum likelihood (REML) and quasi-Newton algorithm for the optimization process.

The Tables 2 and 3 show the results obtained without using any covariate.

To develop the covariate model of Cl against covariates, a multiple linear regression was performed without any previous analysis. Then, the covariates were screened using the stepwise method, where Age was identified as the most important covariate for the Cl model, the results are shown in Tables 4 and 5.

4.3 Genetic Algorithm for Covariates Selection

AGA can be implemented using the model of (7) as a general model where there are 2^p different possible sets for n variables. In order to achieve satisfactory results of a GA, it is important to have a correct encoding, which can obey the following rule:

- If i th position is 0 the i th explanatory variable is not included in the model.
- If i th position is 1 the i th explanatory variable is included in the model.

The first always will be 1 because it represents the intercept θ_0 .

As an example of a model, the binary structure 110011 would represent:

$$\beta_i = \theta_0 + \theta_1 \cdot x_1 + \theta_4 \cdot x_4 + \theta_5 \cdot x_5 + \varepsilon \quad (11)$$

Table 2 Results of fixed effects estimates

Fixed effects parameters	Estimated value	Standard error
$V_d = \text{Central}$	3.1608	0.065
$Cl = Cl_Central$	1.3685	0.070653

Table 3 Results of random effects estimates

$V_d = \text{Central}$	$Cl = Cl_Central$
0.1108	0
0	0.35813

Table 4 Parameters estimated using all the covariates

Name	Description	Estimate	Standard Error
‘theta1’	‘Cl_Central’	7.3008	5.9892
‘theta2’	‘Cl_Central/WT’	-0.1270	0.0612
‘theta3’	‘Cl_Central/AGE’	-0.0184	0.0033
‘theta4’	‘Cl_Central/HT’	-0.0310	0.0412
‘theta5’	‘Cl_Central/BSA’	10.2341	5.1410
‘theta6’	‘Cl_Central/IBW’	-0.0312	0.0340

Table 5 The stepwise method only identified AGE as the most statistically significant

Name	Description	Estimate	Standard Error
‘theta1’	‘Cl_Central’	8.3434	0.67
‘theta3’	‘Cl_Central/AGE’	-0.0722	0.012

To find the fittest chromosome, the GA can use (9) as the fitness function.

In this case, a population of eight chromosomes was generated randomly in Matlab, and the codification was made in range of five bits for each individual, where the first bit of each string is always 1. See Algorithm 1:

Algorithm 1.

```

for i=1:n
    for j=1
        p(i,j)= (1);
    for j=2:5
        p(i,j)= randi([0 1],1);
    end
    end
end

```

The amount of parameters in the fitness function depends on how many bits equals to 1 are part of an individual, for example: (11100) = 3 parameters. Additionally, to identify the covariates, the decoding changes the chromosome content to the decimal base:

where the individual (11100) = 28

corresponds to: $Cl = \theta_0 + \theta_1 \cdot WT_1 + \theta_2 \cdot Age_2 + \theta_3 \cdot HT_3 + \varepsilon$.

From here, the equation is used as the fitness function, and roulette wheel selection is used for selecting potentially useful solutions for recombination. Only a single crossover point in the third position was used with a probability of 0.8 and a Gaussian mutation operator. Table 6 shows that WT and HT were selected by the GA.

5 Results

Finally the identified covariates on Cl were added into the Pharmacokinetic model. Table 7 shows three different evaluation criteria on the covariates selected by stepwise screening and by the genetic algorithm.

It can be seen, than the proposed genetic algorithm shows better performance than the stepwise method. The genetic algorithm has received a better AIC value

Table 6 WT, and HT arising from using the GA

Name	Description	Estimate
‘theta1’	‘Cl_Central’	6.9452
‘theta2’	‘Cl_Central/WT’	0.2745
‘theta4’	‘Cl_Central/HT’	0.1571

Table 7 Comparative results between the stepwise method and the genetic algorithm

Description	logl	Mse	AIC
Stepwise	-325.6623	0.0940	673.3246
GA	-325.1470	0.0933	648.357

(6.48.357), This also influenced the quality of the estimators (logl) and by consequence in the reduction of the mean square error.

6 Conclusions

The main purpose of this work was to show that using genetic algorithms is possible to develop PopPK models as an alternative to the statistical methods; particularly, for the selection of a set of covariables that contribute with a major degree prediction of the variability inside a PopPK model. The comparison of the obtained results in this work exposes the possibility of incorporating methods of intelligent computation for the development of PopPK models.

Acknowledgments We thank to Instituto Politécnico Nacional (IPN), to the Comisión de Fomento y Apoyo Académico del IPN (COFAA), and the Mexican National Council of Science and Technology (CONACYT) for supporting our research activities.

References

1. Bonate, P.L.: Pharmacokinetic-Pharmacodynamics Modeling and Simulation. Springer, Berlin (2011)
2. Rosenbaum, S.: Basic pharmacokinetics and pharmacodynamics. Wiley, Hoboken (2011)
3. Katarina, V., Branislava, M., Sandra, V., Zoran, T., Milica, P., Iztok, G.: Population Pharmacokinetic Analysis of Therapeutic Drug Monitoring Data in Optimizing Pharmacotherapy of Antiepileptic Drugs. In: Foyaca-Sibat, H. (ed.) Novel Treatment of Epilepsy, pp. 96–106. In Tech, Croatia (2011)
4. Khandelwal, A., Harling, K., Jonsson, E.N., Hooker, A.C., Karlsson, M.O.A.: A fast method for testing covariates in population PK/PD models. AAPS J. **13**(3), 464–472 (2011)
5. Davidian, M., Giltinan, D.M.: Nonlinear models for repeated measurement data: an overview and update. J. Agric. Biol. Environ. Stat. **8**(4), 387–419 (2003)
6. Ulrika, W., Jonson, E.N., Karlsson, M.O.: Comparison of stepwise model building strategies in population pharmacokinetic-pharmacodynamic analysis, Oct 2002
7. Gabrielsson, J., Weiner, D.: Pharmacokinetic & Pharmacodynamic Data Analysis: Concepts and Applications. Apotekarsocieteten, Stockholm (2006)
8. Kim, S., Li, L.: A novel global search algorithm for nonlinear mixed-effects models. Springer, Berlin (2011)
9. Mould, D.R., Upton, R.N.: Basic concepts in population modeling, simulation, and model-based drug development (2012)
10. Bozdogan, H.: Model Selection and AKAIKE'S information criterion (AIC): the general theory and its analytical extensions. Psychometrika **52**(3), 345–370 (1987)
11. Holland, J.H.: Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor (1975)

12. Sivanandam, S.N., Deepa, S.N.: *Introduction to Genetic Algorithms*. Springer, Berlin (2008)
13. WookAhn, C.: *Advances in Evolutionary Algorithms. Theory, Design and Practice*. Springer, Berlin (2006)
14. Eiben, A.E., Smith, J.E.: *Introduction to Evolutionary Computing*. Springer, Berlin (2003)
15. Lowen, R., Verschoren, A.: *Foundations of Generic Optimization*. Springer, Berlin (2008)

Parallel Evolutionary Artificial Potential Field for Path Planning—An Implementation on GPU

Ulises Orozco-Rosas, Oscar Montiel and Roberto Sepúlveda

Abstract This work presents the implementation of the Parallel Evolutionary Artificial Potential Field (PEAPF) on a Graphics Processing Unit (GPU) as an improvement to speed up the path planning computation in mobile robot navigation. Simulation results to validate the analysis and implementation are provided; they were specifically made to show the effectiveness and the efficiency of the proposal.

1 Introduction

Path planning is one of the fundamental tasks in mobile robotics. As stated by Latombe [1], the capability of effectively planning its motions is “eminently necessary since, by definition, a robot accomplishes a task by moving in the real world” [2]. The path must be collision-free (feasible) and satisfy certain optimization criteria [3]. In this work, path planning is generating a viable path, in an environment with obstacles without colliding with them, and optimizing the path with respect to its length.

The main objective in this work is the implementation of the PEAPF on a GPU to speed up the path planning computation through parallel computation, since during the past several decades, there has been an increasing interest on this topic. The primary goal of parallel computing is to improve the speed of calculus. From pure calculation perspective, parallel computing can be defined as a form of

U. Orozco-Rosas · O. Montiel (✉) · R. Sepúlveda

Instituto Politécnico Nacional, Centro de Investigación y Desarrollo de Tecnología Digital (CITEDI-IPN), Av. Del Parque No. 1310, Mesa de Otay, 22510 Tijuana, B.C., México
e-mail: orross@ipn.mx

U. Orozco-Rosas
e-mail: uorozco@citedi.mx

R. Sepúlveda
e-mail: rsepulvedac@ipn.mx

computation in which many calculations are carried out simultaneously, operating on the principle that large problems can often be divided into smaller ones, which are then solved concurrently [4].

From the programmer's perspective, a natural question is how to map the concurrent calculations onto computers. Suppose having multiple computing resources, parallel computing can then be defined as the simultaneous use of multiple computing resources (cores or computers) to perform the concurrent calculations. A large problem is broken down into smaller ones, and each one is afterwards solved concurrently on different computing resources. The software and hardware aspects of parallel computing are closely intertwined together [4].

This work is organized as follows: Sect. 2 presents the fundamentals of the PEAPF method and a brief description of its core components [13]. These components are the Artificial Potential Field (APF) method, the Evolutionary Algorithms (EAs) in specific a Genetic Algorithm (GA), and the Parallel Computing. Section 3 is dedicated to the implementation of the PEAPF. In Sect. 3.1, the PEAPF algorithm is described, where the fitness function based on the APF method represents the most expensive process in terms of computation time. Section 3.2 presents the PEAPF implementation on GPU. We have chosen MATLAB with CUDA kernel integration to take advantage of the CUDA GPU computing technology and the benefits of the MATLAB software platform. In Sect. 4, the results from the experiments and the performance evaluation are presented. The experimental framework consists of two tests: off-line and on-line path planning. They were designed to demonstrate the relevance of the PEAPF proposal implemented on a GPU. Finally, in Sect. 5 the conclusions are given.

2 Parallel Evolutionary Artificial Potential Field

Numerous algorithms have been developed over the last few years to create real time path planning systems for autonomous robots. There are three tasks that must be carried out by a self-governing robot to enable the execution of the robot navigation [5, 6]. These activities are: world perception consisting in mapping and modeling the environment; path planning for obtaining an ordered sequence of objective points and convert this sequence into a path; the path tracking in controlling the robot to follow the path [7]. The selection of an appropriate algorithm in every stage of the path planning is very important to ensure that the navigation process will run smoothly [5].

Here, we used the PEAPF method [13], implemented on GPU to obtain a feasible, safe and efficient path in a moderate run time to drive the mobile robot in an autonomous form to its goal. The PEAPF algorithm is the integration of the Artificial Potential Field method with Evolutionary Algorithms and Parallel Computation for taking advantages of novel processors' architectures (in this case GPUs), to obtain a flexible path planner capable of performing the on-line and off-line path planning.

The APF method was proposed by Khatib [8] for local planning. At the beginning, this method was used for obstacle avoidance in path planning for robot manipulators and making obstacle avoidance in real time [9]. The main idea behind the APF method is to establish an attractive potential field force, around the goal position, as well as to establish a repulsive potential field force around obstacles. The two potential fields together integrate the total potential field [10].

The traditional APF approaches are often purely reactive in nature, and they do not optimize the path arrived [11]. When the APF has been used as a planning method, it has well known problems to obtain the “global” optimal path, because it is not able to discard all local minimum problems [9]; hence, new methods that overcome these problems have been developed [12]. One of them is where the APF is blended with EAs to obtain a different potential field methodology named Evolutionary Artificial Potential Field (EAPF) [11, 13], here the APF method was combined with GAs to derive optimal potential field functions [12]. The variational planning approach uses the potential as a cost function, and attempts to find a path to the goal position that minimizes this cost [14]. In the EAPF method, only the information regarding to the positions of the robot, obstacles and the goal are required to plan the path [11]. An improvement of the EAPF method is PEAPF [13], where it employs parallel evaluation of the fitness function using the CPU (Central Processing Unit) threads.

In this work, the PEAPF algorithm employs parallel evaluation of the fitness function on GPU, the reasons are founded in taking advantages of novel processors’ architectures to counteract the expensive computational time required in the evaluation of the fitness function. The implementation of the PEAPF algorithm is in MATLAB, and for the parallel evaluation, a GPU is achieved using CUDA programming. CUDA is a parallel computing technology from NVIDIA that consists of a parallel computing architecture and developer tools, libraries, and programming directives for GPU computing [15].

There are two fundamental types of parallelism in applications: task parallelism and data parallelism. Task parallelism arises when there are many tasks or functions that can be operated independently and largely in parallel. Task parallelism focuses on distributing functions across multiple cores. Data parallelism arises when there are many data items that can be operated on at the same time. Data parallelism focuses on distributing the data across multiple cores [4]. In this work, data parallelism was employed because there were many possible paths to evaluate, and each path is composed by data containing the values of the proportional gains. The major focus of this work is how to solve a data-parallel problem with CUDA programming. The reason is founded in that CUDA programming is especially well-suited to address problems that can be expressed as data-parallel computations. Many applications that process large data sets can use a data-parallel model to speed up the computations [4].

3 Implementation

In this section, the PEAPF implementation on GPU is described. The PEAPF algorithm is a high-performance hybrid metaheuristic that integrates the APF method with EAs and Parallel Computation for taking advantages of novel processors' architectures, to obtain a flexible path planner capable to perform the on-line and off-line path planning.

3.1 PEAPF Algorithm

In Fig. 1, the control flow of the PEAPF algorithm is shown. The control flow diagram is organized in two parts, the CPU and the GPU. All the work related to the APF (fitness function) is executed on the GPU in parallel form (inside the dashed rectangle), and all the processes related to the evolutionary algorithm are executed on the CPU in the sequential form. The PEAPF starts with the creation of an initial random population $P(t)$ of chromosomes on the CPU, where the population size is N . Each chromosome codifies a pair of proportional gains, so each pair is composed by one attraction gain k_a and one repulsion gain k_r . The path generated with the solution given by the algorithm will be composed by the best pair found.

On the GPU, the first process is to calculate the potential field $U(q)$ using Eq. 1 for each chromosome created (path) in parallel.

$$U(q) = \frac{1}{2} \left(k_a (q - q_f)^2 + k_r \left(\frac{1}{\rho} - \frac{1}{\rho_0} \right)^2 \right) \quad (1)$$

where, q is the robot position and q_f is the goal position, k_a and k_r are scalar variables that represent the attractive and repulsive proportional gains respectively; ρ_0 is the limit distance of influence of the potential field, and ρ is the shortest distance to the obstacle. The second part of the Eq. 1 is zero when $\rho > \rho_0$.

Continuing on the GPU, the second process is to calculate the potential force $F(q)$ using Eq. 2 for each chromosome in parallel.

$$F(q) = -\nabla U(q) \quad (2)$$

The last step on the GPU is the evaluation of the length of the path given by each chromosome in parallel. The evaluation is performed using the objective function S given by Eq. 3 which is the cumulative Euclidean distance from point to point in the path, where, each path consists of n points.

$$S = \sum_{i=0}^n (q_{i+1}^2 - q_i^2)^{1/2} \quad (3)$$

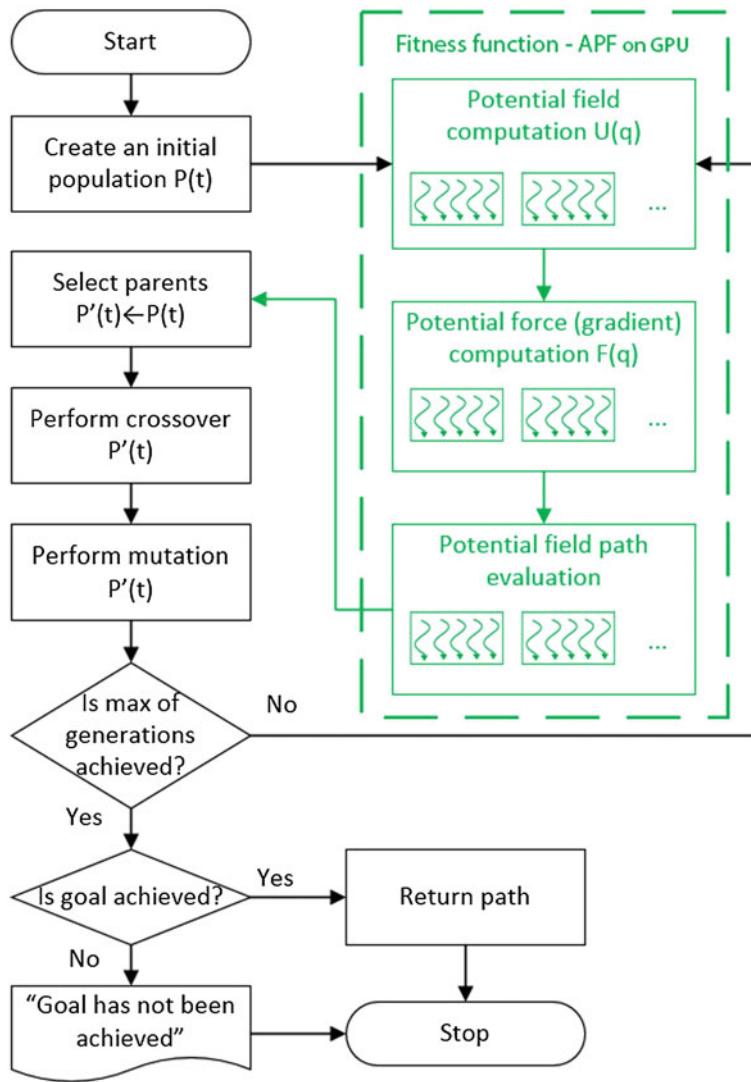


Fig. 1 Control flow of the PEAPF. The processes inside of the dashed rectangle represent the fitness function that is implemented on GPU

Returning to the CPU, the selection process where the chromosomes are chosen according to their fitness value (calculated previously on the evaluation step on GPU) is performed. The selection process drives the EA to improve the population fitness over the successive generations [16]. Next, the crossover is performed. The crossover operation roughly mimics biological recombination between two single-chromosomes (haploid) organisms [17]. Then, the mutation process is performed, where, random mutations alter a certain percentage of the bits in the list of chromosomes.

The mutation process tends to distract the evolution from converging on a popular solution [18]. For a detailed description of the EAs operators refers to [10].

The PEAPF algorithm shown in the control flow of Fig. 1 iterates until the maximum number of generation N_{gen} is reached. The PEAPF algorithm evolves the parameters k_a and k_r to obtain the corresponding best values. If the goal was achieved ($goal = 1$), then the algorithm returns the path (best path generated through the parameters k_a and k_r). Otherwise, it returns a message “Goal has not been achieved”.

3.2 PEAPF Implementation on GPU

We have chosen MATLAB as a platform to implement the PEAPF algorithm because it is a programming language widely used in scientific research. On the last years, the scientific community has been exploring new ways to speed up MATLAB functions and programs; one way is using GPU computing.

It is possible to take advantage of the CUDA GPU computing technology with the MATLAB language throughout different strategies. One strategy or way is using GPU-enabled MATLAB functions. Another way is using GPU-enabled functions in toolboxes or using CUDA kernel integration in MATLAB applications.

In this work, we have chosen the CUDA kernel integration in MATLAB as the strategy to implement the PEAPF algorithm. A kernel is code written for execution on the GPU. Kernels are functions that can run on a large number of threads. The parallelism arises from each thread independently running the same program on different data. Before continuing, we have to consider that a GPU can accelerate an application only if it fits with the next criteria [15]:

- Computationally intensive—The time spent on computation significantly exceeds the time spent on transferring data to and from GPU memory.
- Massively parallel—The computation can be broken down into hundreds or thousands of independent units of work.

Applications that do not satisfy these criteria might actually run slower on a GPU than on a CPU.

The CUDA kernel integration in MATLAB requires the creation of an executable kernel CU or PTX (parallel thread execution) files, and running that kernel on a GPU from MATLAB. The kernel is represented in MATLAB by a `CUDAKernel` object, which can operate on MATLAB array or `gpuArray` variables [19]. The following steps describe the `CUDAKernel` general workflow: First, use compiled PTX code to create `CUDAKernel` object, which contains the GPU executable code, an overview of this implementation is shown in Fig. 3, where, with “**Device**” we are referring to the GPU and its system memory. Next, set properties on the `CUDAKernel` object to control its execution on the GPU. And finally, call `feval` on the `CUDAKernel` with the required inputs, to run the kernel on the GPU, an overview of the MATLAB code for this implementation is shown in Fig. 2, where, with “**Host**” we are referring to the CPU and its system memory.

Host:

```
% Create CUDAKernel object.
kernel_APF_Potential = parallel.gpu.CUDAKernel('apf_gpu_eval.ptx', 'apf_gpu_eval.cu',
                                                'APF_Potential');

...
% Call feval with defined inputs.
[device_variables] = feval(kernel_APF_Potential, ... );
...

% Create CUDAKernel object.
kernel_APF_Gradient = parallel.gpu.CUDAKernel('apf_gpu_eval.ptx', 'apf_gpu_eval.cu',
                                                'APF_Gradient');
...

% Call feval with defined inputs.
[device_variables] = feval(kernel_APF_Gradient, ... );
...

% Create CUDAKernel object.
kernel_APF_Evaluation = parallel.gpu.CUDAKernel('apf_gpu_eval.ptx', 'apf_gpu_eval.cu',
                                                'APF_Evaluation');
...

% Call feval with defined inputs.
[device_variables] = feval(kernel_APF_Evaluation, ... );
...
```

Fig. 2 Overview of the code section to call the CUDA kernel from MATLAB for the PEAPF implementation

Device:

```
// CUDA kernel for potential field computation.
__global__ void APF_Potential(...)
{
    int id = (blockIdx.x * blockDim.x) + threadIdx.x;
    ...
}

// CUDA kernel for potential force computation.
__global__ void APF_Gradient(...)
{
    int id = (blockIdx.x * blockDim.x) + threadIdx.x;
    ...
}

// CUDA kernel for potential field path evaluation.
__global__ void APF_Evaluation(...)
{
    int id = (blockIdx.x * blockDim.x) + threadIdx.x;
    ...
}
```

Fig. 3 Overview of the code in the CUDA kernel for the PEAPF implementation, this code represents the processes inside the dashed rectangle in Fig. 1

4 Experiments and Results

Path planning can be divided into two kinds of problems: off-line and on-line [20]. The off-line path planning for mobile robots in structured environments is where the perception process allows to construct maps or models of the world that are used for robot localization and robot motion planning. The on-line path planning for mobile robots in unstructured environments is where the robot has to learn how to navigate. In this section, two experiments are presented; they embrace the off-line and on-line path planning cases.

For consistency among the experiments, we used the same PEAPF parameter values in all the experiments. The next parameters yielded the best results for this proposal:

- A population size of $N = 128$ chromosomes. Each chromosome consists of two genes, k_a and $k_r \in \mathbb{N}$, codified using 8-bit. Note: the value of N is fixed for the two experiments, but it will be varied in the performance evaluation.
- The proportional gains are constrained, so k_a and $k_r \leq 49$.
- Single point crossover is used.
- Mutation rate of 15 %.
- Elitist selection with a rate of 50 %.
- Maximum number of generations $N_{gen} = 10$.

4.1 Experiment 1: Off-line Path Planning

As mentioned before, the off-line path planning is presented when the environment is totally known. For this experiment, the start position of the mobile robot is at coordinate (5.50, 9.00), and the goal position of the mobile robot is at (5.00, 3.50); moreover, we know the position and size of the obstacles in the environment (map). There are two blocks of five obstacles, each one forming an L-shaped obstacle to recreate a real-world corridor stage, and one more block of three obstacles forming a rectangular obstacle in the center of the environment, as it is shown in Fig. 4a.

The center position coordinates for the upper L-shaped block of obstacles are (2.00, 5.50), (2.00, 6.50), (2.00, 7.50), (3.00, 7.50), and (4.00, 7.50). For the lower L-shaped block of obstacles are (6.00, 2.50), (7.00, 2.50), (8.00, 2.50), (8.00, 3.50), and (8.00, 4.50). Finally, the center position coordinates for the centered rectangular block of obstacles are (4.00, 5.00), (5.00, 5.00), and (6.00, 5.00), where, each obstacle in the blocks has a radius of 0.50 units.

The off-line path planning can be performed by the PEAPF algorithm when the environment information is loaded. In Fig. 4a it can be seen that the mobile robot has many different possible paths (ways) to reach the goal. Figure 4b shows few of these possible paths to reach the goal; it can be seen that some of them are better in terms of path distance. Now the PEAPF has to find the best pair of proportional

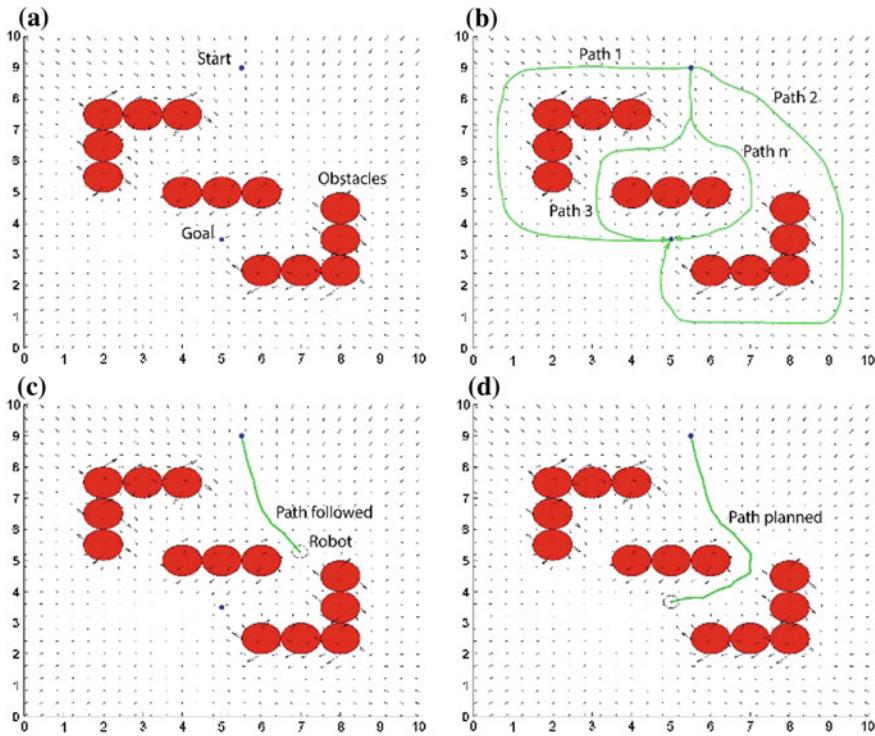


Fig. 4 **a** Known environment. **b** A few main possible paths. **c** Path followed during the navigation. **d** Best path founded by the PEAPF

gains (k_a and k_r) to generate the shortest path to reach the target; this path must be feasible for the navigation of the mobile robot, as it is shown in Fig. 4c. The result of the PEAPF is shown in the Fig. 4d where the PEAPF algorithm successfully found the optimal path for the mobile robot navigation.

4.2 Experiment 2: On-line Path Planning

An advantage of the off-line path planning case is that the environment is known; however, real-world applications frequently face the mobile robot to environments partially known or totally unknown, where the already planned path needs to be adjusted for reaching the goal fulfilling the obstacle avoidance and finding a feasible path for the navigation of the robot. This case is referred as on-line path planning.

In this experiment, we are going to test the PEAPF algorithm by placing an obstacle at random position, in such a way that it blocks the already found best path,

interfering with the mobile robot trajectory (during navigation). In this case, the random obstacle must remain static once it was put. To recreate a real-world situation; i.e. when an obstacle such as a trash can or a planter is put in some place where before it was nothing there.

The initial conditions for this experiment are: start position of the mobile robot at coordinate (2.00, 7.00), and goal at (8.00, 3.00). At the beginning, we know the position and size of certain obstacles in the environment. There are two blocks of five obstacles, each one forming an L-shaped obstacle, and one more block of two obstacles forming a rectangular obstacle in the lower left of the environment, as it is shown in Fig. 5a. The centers position coordinates for the left upper L-shaped block of obstacles are (1.50, 8.25), (2.50, 8.25), (3.50, 8.25), (3.50, 7.25), and (3.50, 6.25). For the right lower L-shaped block of obstacles the coordinates are (6.50, 3.75), (6.50, 2.75), (6.50, 1.75), (7.50, 1.75), and (8.50, 1.75). Finally, the center position coordinates for the left lower rectangular block of obstacles are (3.75, 1.75), and (3.75, 2.75), where each obstacle in the blocks has a radius of 0.50 units.

Actually, in this experiment we are testing both cases off-line and on-line path planning. Using the above world configuration the experiment was conducted as

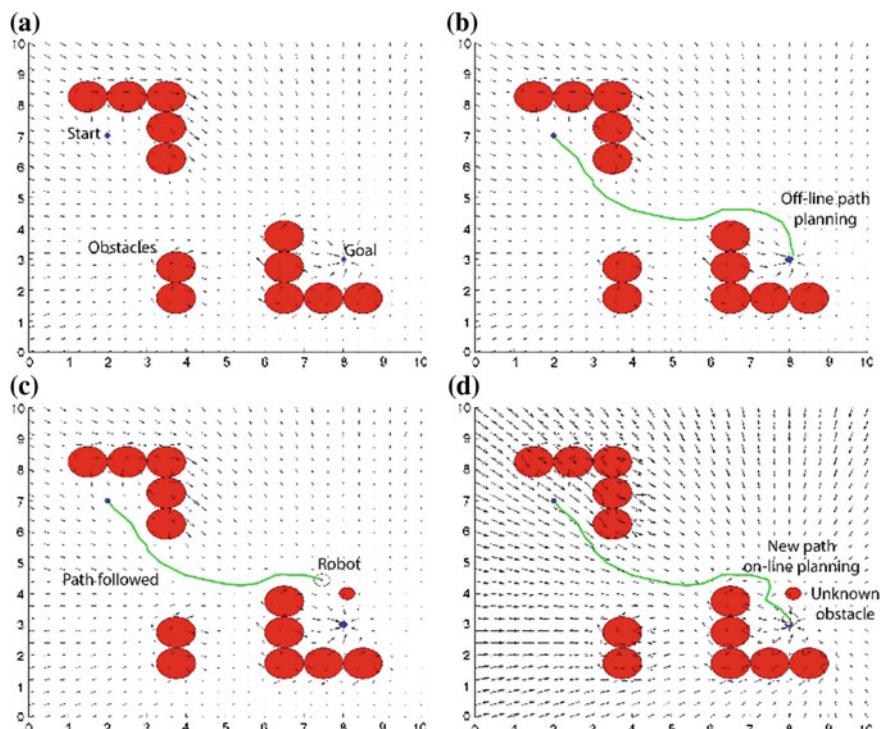


Fig. 5 **a** Known environment. **b** Off-line path planning with the PEAPF. **c** Unknown obstacle detection during the navigation. **d** On-line path planning with the PEAPF

follows: First, the path planning is achieved in the off-line mode because the environment is known at the beginning, as it is shown in Fig. 5a. Then, we switch to the on-line mode, and the navigation of the mobile robot starts. It will follow the path planned in off-line mode, this path is shown in Fig. 5b. After that, during the navigation an obstacle is added at position (8.10, 4.00) to make a change in the environment. Next, the mobile robot at the position (7.46, 4.43) senses the new obstacle, and it calculates the obstacle position to update the world configuration (environment), as it is shown in Fig. 5c. Later, the PEAPF algorithm recalculates the new path using the new configuration. Finally, the mobile robot follows the latest path to reach the goal, as it is shown in Fig. 5d.

4.3 Performance Evaluation of the PEAPF

For this performance evaluation, we used the off-line experiment exposed before in Sect. 4.1. To compare the CPU implementation versus GPU implementation we carried out thirty tests for each population size with the aim to record the execution times. The average time and standard deviation of each sample is shown in Table 1.

In this performance evaluation we refer to “CPU” as the implementation of the algorithm totally in sequential form using only the MATLAB, and we refer to “GPU” as the implementation of the PEAPF algorithm as it is shown in the Fig. 1, where only the fitness function is executed in parallel on the GPU and the rest of the algorithm (EA) is executed in the sequential form on the CPU.

In parallel computing, speedup refers to how much a parallel program is faster than its equivalent in the sequential form. To know the benefits of using the GPU in the APF evaluation (fitness function), we measured the amount of time that the PEAPF program took to execute for population sizes of (32, 64, 128, 256, 512, and 1024) \times 10 on an Intel i7 Processor, and then using an NVIDIA GeForce GTX 860 M, the results are shown in Table 1. For example, for a population size of 32 chromosomes, during ten generations ($N \times 10$), actually, we have evaluated 320 chromosomes.

Table 1 Average time, standard deviation and speedup of each population size evaluated in the sequential form on the CPU and in the parallel form on the GPU

Population size (N \times 10)	CPU		GPU		Speedup
	Mean (μ) seconds	Std. Dev. (σ)	Mean (μ) seconds	Std. Dev. (σ)	
32	4.491	1.128	5.902	0.985	0.761
64	8.255	0.770	5.784	0.068	1.427
128	16.895	1.165	6.120	0.033	2.761
256	33.832	1.478	10.762	0.230	3.144
512	67.957	2.430	17.402	0.257	3.905
1024	134.701	3.548	32.003	0.528	4.209

The last column of Table 1 shows the speedup, based on the population size evaluated. Among the results we can observe how the computation time decreases when the population size is incremented. For a population size of 1024×10 , the computation time is decreased from more than two minutes on the CPU to almost thirty seconds on the GPU. The speedup is 4.21 for this population size demonstrating that the PEAPF on GPU outperforms the CPU implementation.

In Fig. 6, we can observe the computation time in seconds for the evaluation of each population size, where the GPU implementation shows its advantage through the increment of the population size (data). On the other hand, it can be seen that the plot shows the CPU is actually faster for very small population sizes. However, as the technology evolves and matures, GPU solutions are increasingly able to handle smaller problems, a trend that is expected to continue as it is mentioned by [15].

Population sizing has been one of the important topics to consider in evolutionary computation. Researchers usually argue that a “small” population size could guide the algorithm to poor solutions and that a “large” population size could make the algorithm expend more computation time in finding a solution [21].

In this work, we have seen how the PEAPF algorithm solves the path planning for off-line and on-line cases. The results demonstrated that the GPU implementation is very helpful to accelerate the evaluation of the solutions, because the problem is stated as a data-parallel problem. Making a compromise between solution quality and computation time, we have found that for the experiments evaluated in this work, the best performance on GPU is obtained when the population size is of 128 chromosomes and the number of generations is 10.

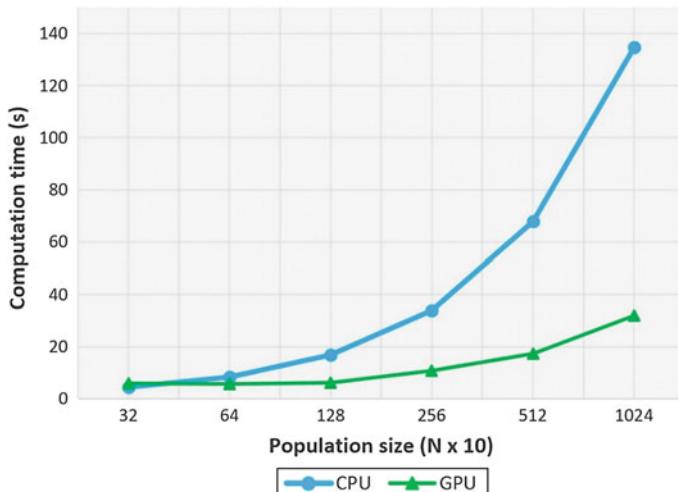


Fig. 6 Plot of the PEAPF computation time results showing the time required to evaluate the possible paths for different population size

5 Conclusions

In this chapter, we have presented the PEAPF on GPU as a novel implementation for off-line and on-line path planning for mobile robots. Through the integration of the APF method with EAs and Parallel Computation, it has been presented a path planning system capable of obtaining good solutions (even the global optimum) in a moderate run time. Due to the simulation results, we can conclude that PEAPF on GPU can be capable of facing more complex and bigger planning problems, making the implementation suitable for applications in the real world in big sceneries.

Acknowledgments We thank to Instituto Politécnico Nacional (IPN), to the Comisión de Fomento y Apoyo Académico del IPN (COFAA), and the Mexican National Council of Science and Technology (CONACYT) for supporting our research activities.

References

1. Latombe, J.C.: *Robot Motion Planning*. Kluwer Academic Publishers, Norwell (1991)
2. Murray, R.M., Li, Z., Sastry, S.S.: *A Mathematical Introduction to Robotic Manipulation*. CRC Press, USA (1994)
3. Sedighi, K., Ashenayi, K., Manikas, T., Wainwright, R., Tai, H.M.: Autonomous local path planning for a mobile robot using a genetic algorithm. In: Congress on evolutionary computation, CEC 2004, vol. 2, pp. 1338–1345 (2004)
4. Cheng, J., Grossman, M., McKercher, T.: *Professional CUDA C programming*. Wrox—Wiley, Indianapolis (2014)
5. Sariff, N., Buniyamin, N.: An overview of autonomous mobile robot path planning algorithms. In: 4th student conference on research and development, SCOReD 2006, pp. 183–188 (2006)
6. Shin, D.H., Singh, S.: Path generation for robot vehicle using composite clothoid segments. Tech. Rep. CMU-RI-TR-90-31, The Robotic Institute, Carnegie-Mellon University, Pittsburgh, Pennsylvania 15213 (1990)
7. Montiel, O., Sepúlveda, R., Castillo, O., Melin, P.: Ant colony test center for planning autonomous mobile robot navigation. *Comput. Appl. Eng. Educ.* **21**(2), 214–229 (2013)
8. Khatib, O.: Real-time obstacle avoidance for manipulators and mobile robots. In: Proceedings of robotics and automation. IEEE international conference on, vol. 2, pp. 500–505 (1985)
9. Zhang, Q., Chen, D., Chen, T.: An obstacle avoidance method of soccer robot based on evolutionary artificial potential field. *Energy Procedia* **16**(Part C(0)), 1792–1798 (2012)
10. Orozco-Rosas, U., Montiel, O., Sepúlveda, R.: High-performance navigation system for mobile robots. In: Montiel, O., Sepúlveda, R.: (eds.) *High Performance Programming for Soft Computing*, chap. 12, pp. 258–281. CRC Press, Boca Raton (2014)
11. Vadakkepat, P., Tan, K.C., Ming-Liang, W.: Evolutionary artificial potential fields and their application in real time robot path planning. In: Proceedings of the 2000 congress on evolutionary computation, vol. 1, pp. 256–263 (2000)
12. Vadakkepat, P., Lee, T.H., Xin, L.: Application of evolutionary artificial potential field in robot soccer system. In: IFSA world congress and 20th NAFIPS international conference, 2001. Joint 9th, vol. 5, pp. 2781–2785 (2001)
13. Montiel, O., Sepúlveda, R., Orozco-Rosas, U.: Optimal path planning generation for mobile robots using parallel evolutionary artificial potential field. *J. Intell. Robot. Syst.* **1**–21 (2014)
14. Goerzen, C., Kong, Z., Mettler, B.: A survey of motion planning algorithms from the perspective of autonomous uav guidance. *J. Intell. Rob. Syst.* **57**(1–4), 65–100 (2010)

15. Reese, J., Zarnek, S.: Gpu programming in matlab. <http://www.mathworks.com/company/newsletters/articles/gpu-programming-in-matlab.html>
16. Sivanandam, S.N., Deepa, S.N.: Introduction to Genetic Algorithms. Springer, Heidelberg (2008)
17. Mitchell, M.: An introduction to Genetic Algorithms. Bradford, Cambridge, Massachusetts, USA (2001)
18. Haupt, R.L., Haupt, S.E.: Practical Genetic Algorithms. Wiley, Hoboken (2004)
19. Mathworks: Run cuda or ptx code on gpu. <http://www.mathworks.com/help/distcomp/run-cuda-or-ptx-code-on-gpu.html>
20. Aghababa, M.P.: 3D path planning for underwater vehicles using five evolutionary optimization algorithms avoiding static and energetic obstacles. *Appl. Ocean Res.* **38**, 48–62 (2012)
21. Roeva, O., Fidanova, S., Paprzycki, M.: Influence of the population size on the genetic algorithm performance in case of cultivation process modelling. In: 2013 federated conference on computer science and information systems (FedCSIS), pp. 371–376 (2013)

Design and Acceleration of a Quantum Genetic Algorithm Through the Matlab GPU Library

Oscar Montiel, Ajelet Rivera and Roberto Sepúlveda

Abstract The potential processing power of a quantum computer is quantum parallelism, but significant disadvantages of quantum simulators are processing speed and memory. In this work, we illustrate with a Quantum Genetic Algorithm (QGA) the advantages of using the software platform of Compute Unified Device Architecture (CUDA) from NVIDIA, in special, the Matlab Graphic Processing Unit (GPU) library was used. The original software for Matlab named Quack!, which is a quantum computer simulator, was modified with the aim of speeding up a QGA. Experimental results that show advantages of using a QGA, as well as comparative experiments of the sequential implementation versus implementations that use the CUDA cores for different NVIDIA cards are presented.

1 Introduction

Quantum computing refers to compute with quantum systems called quantum computers (QC). A QC is a physical device that makes use of the quantum mechanic phenomena as superposition, entanglement and interference to perform operations on data. It is a device based on quantum bits (qubits) which purpose is to process information by executing algorithms, differently from classical computers

O. Montiel (✉) · A. Rivera · R. Sepúlveda

Centro de Investigación y Desarrollo de Tecnología Digital (CITEDI-IPN),
Instituto Politécnico Nacional, Av. Del Parque No. 1310, Mesa de Otay, 22510,
Tijuana, BC, México
e-mail: orross@ipn.mx

A. Rivera
e-mail: arivera@citedi.mx

R. Sepúlveda
e-mail: rsepulvedac@ipn.mx

based on binary bits. One of the essential characteristics of a QC is the reduction of complexity order of some classic algorithms; for example, in the case of a search algorithm that looks for any item in an unsorted database with N entries, a classical computer can take $O(N)$ steps, but the Grover's quantum searching algorithm takes only $O(\sqrt{N})$ steps [1].

Quantum computing was introduced by Richard Feynman in 1982 in the paper entitled "Simulating Physics with Computers" [2]. In 1985, David Deutsch in the paper "Quantum theory, the Church-Turing principle and the universal quantum computer" introduced the concept of Universal Turing Machine (UTM), and the interest in the field grew up when Shor [3] in 1994 published the paper entitled "Algorithms for Quantum Computation: Discrete Logarithms and Factoring", where quantum algorithms for finding discrete logarithms and factoring integers are described. After that, thousands of papers in different fields based on QC concepts have been written.

In the last decade, there has been a big progress in the experimental development of a QC that can exploit the full complexity of quantum physics [4]. Up to date, the company D-Wave Systems Inc., claims to be the first commercial quantum computing company, and that it has developed the world's first real QC [5]. While there are controversial opinions about the veracity of that information [6], scientists are still working in the development of new algorithms. They usually use a simulation platform for testing them on a conventional computer; although it is well known that simulating an entirely general quantum system on a classical binary computer is possible only for very small systems, because of the exponential scaling of the Hilbert space with the size of the quantum system.

In this work, we present the development and test of a quantum genetic algorithm (QGA) [7] for function optimization. We also show advantages of using the CUDA cores for speeding up the original Quack! quantum simulator. This is important since the potential of a QC is the quantum parallelism, which allows simultaneous operations with a set of superposed states, and the main disadvantages of simulation are high running times and memory usage, which increases exponentially with the number of qubits used. This problem can be minimized by using high-performance computing platforms and programming models in parallel, one of them is the CUDA, which allows increased performance by taking advantage of the power of the graphics processing unit (GPU).

The organization of this work is as follows: In Sect. 2, the quantum computing basis are provided. In Sect. 3, the QGA is explained; we have used the Classic Genetic Algorithm (CGA) to compare the genetic operations and process. In Sect. 4, the CGA and QGA are compared to show the advantages of the QGA, moreover, speeding up results of using the CUDA cores for processing the QGA are given. Finally, in Sect. 5, the conclusions of this work are presented.

2 An Overview of Quantum Computing

In quantum computing the Dirac notation is used to distinguish vectors from scalars; so, the symbol to identify the \mathbf{u} vector is the “ket” and it looks like $|u\rangle$ and the dual vector of \mathbf{u} is represented using a “bra” as $\langle u|$ hence, the inner product of \mathbf{u} and \mathbf{v} is written in “bra-kets” as $\langle u|v\rangle$.

For computer simulation, any physical system needs to be discretized, and a quantum system is not the exception; hence, let us consider that a subatomic particle can only be detected in any of the equidistant points $\{x_0, x_1, \dots, x_{n-1}\}$, where $x_1 = x_0 + \delta_x, x_2 = x_1 + \delta_x, x_3 = x_2 + \dots$ being δx a tiny increase, such as it is illustrated in Fig. 1. Using n large and δx very small, a reasonable good approximation of a continuous real system can be obtained. If we associate to the set of equidistant points x_i for $0 \leq i \leq n - 1$, the n -dimensional complex vector $[c_0, c_1, \dots, c_{n-1}]^T$, and we denote the x_i particle using the Dirac notation, i.e., $|x_i\rangle \in \mathbb{C}^n$ to represent vectors forming the canonical basis of \mathbb{C}^n then we have

$$\begin{aligned} |x_0\rangle &\mapsto [1, 0, \dots, 0]^T \\ |x_1\rangle &\mapsto [0, 1, \dots, 0]^T \\ &\vdots \\ |x_{n-1}\rangle &\mapsto [0, 0, \dots, 1]^T \end{aligned} \tag{1}$$

In this way, any arbitrary state $|\psi\rangle \mapsto [c_0, c_1, \dots, c_{n-1}]^T$ can be represented as an element of \mathbb{C}^n where the c_i 's are known as complex amplitudes; i.e., $|\psi\rangle$ is a linear combination of the $|x_i\rangle$'s and c_i 's. Therefore, $|\psi\rangle$ is a superposition of all the basis states, representing that the particle can be simultaneously in all the positions x_i 's, or a mixture of all the $|x_i\rangle$'s, which proportions will depend on the complex amplitude values c_i 's; hence, the state $|\psi\rangle$ can be written as,

$$|\psi\rangle = c_0|x_0\rangle + c_1|x_1\rangle + \dots + c_{n-1}|x_{n-1}\rangle \tag{2}$$

A qubit is a vector in \mathbb{C}^2 , where the basis vectors denoted by $|0\rangle = [1 \ 0]^T$ and $|1\rangle = [0 \ 1]^T$ form an orthonormal basis which in turn forms the computational basis. Any arbitrary state of a qubit can be expressed as:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \tag{3}$$

where α and β are the complex amplitudes that specify the probability amplitudes of the corresponding states, $|\alpha|^2$ is the probability that the qubit can be found in the

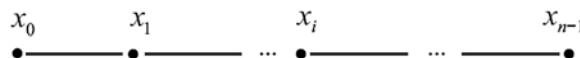
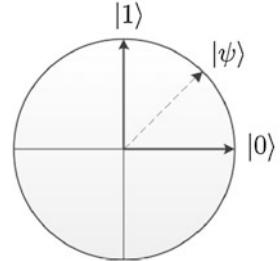


Fig. 1 Subatomic particle

Fig. 2 Qubit with real amplitudes



“0” state, and $|\beta|^2$ is the probability that the qubit can be found in the “1” state. Since the states are normalized using $|\psi\rangle/\|\psi\rangle$, where $|\psi\rangle = \sqrt{|\alpha|^2 + |\beta|^2} = 1$ the equation $|\alpha|^2 + |\beta|^2 = 1$ is satisfied. Figure 2 illustrates the computational basis for the case where the amplitudes are real.

A system with n qubits is known as a quantum register $|\Psi\rangle$, and it consists of a set of individual qubits that can represent the state vector of the register when the tensorial product is applied to them. For example, if we consider a quantum register with three qubits, we will have 2^3 computational basic states, they are $|000\rangle, |001\rangle, |010\rangle, |011\rangle, |100\rangle, |101\rangle, |110\rangle$, and $|111\rangle$; for the particular case of $|100\rangle$, the first and second qubits are in the state $|0\rangle$ and the third qubit is in the state $|1\rangle$; hence, using the tensorial product, we have the next equivalence $|100\rangle = |1\rangle \otimes |0\rangle \otimes |0\rangle$, this is a classic state. For a quantum state, now we have three qubits, but the first and the third qubit are in a superposition state:

$$\left(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle \right) \otimes |0\rangle \otimes \left(\frac{1}{2}|0\rangle + \frac{\sqrt{3}}{2}|1\rangle \right) \quad (4)$$

where, after realizing the tensorial product, some terms are 0, obtaining:

$$|\Psi\rangle = \frac{1}{2\sqrt{2}}|000\rangle + \frac{\sqrt{3}}{2\sqrt{2}}|001\rangle + \frac{1}{2\sqrt{2}}|100\rangle + \frac{\sqrt{3}}{2\sqrt{2}}|101\rangle \quad (5)$$

therefore, the three-qubits system has four information states which values are $1/8$, $3/8$, $1/8$, and $3/8$, respectively.

The Schrodinger equation describes how any isolated quantum system evolves over time, as a register of qubits is a system of this nature, it is also described by this first order partial differential equation

$$i\hbar \frac{\partial |\psi(t)\rangle}{\partial(t)} = H|\psi(t)\rangle \quad (6)$$

This equation involves an initial state of the register, a time independent Hermitian matrix H , called the Hamiltonian of the system, which is the observable of the total system energy and normalized plank constant \hbar . Since the equation is a

linear sum of solutions, it is also a solution, which is fundamental to the principle of superposition.

Knowing that the Hamiltonian equation can be solved, the simplest case is that the Hamiltonian is independent of time and the solution is

$$U(t) = e^{\frac{-iHt}{\hbar}} \quad (7)$$

This means that if the initial state of the system is known $|\psi(0)\rangle$ we can determine the state at the time t after acting in the initial state with the operator $\exp(-iHt/\hbar)$ then the time evolution of a quantum system is described as:

$$|\psi(t)\rangle = U(t)|\psi(0)\rangle = e^{\frac{-iHt}{\hbar}}|\psi(0)\rangle \quad (8)$$

The physical phenomenon used for manipulating a quantum state varies. For example if a qubit is encoded in the spin of an electron, this is modified by applying magnetic fields in various orientations, and its evolution is given by the Schrodinger equation, where H specifies the fields and forces acting on the system, these physical processes that perform an action on a quantum states are called quantum gates, and in quantum computation they are described mathematically as unitary matrices.

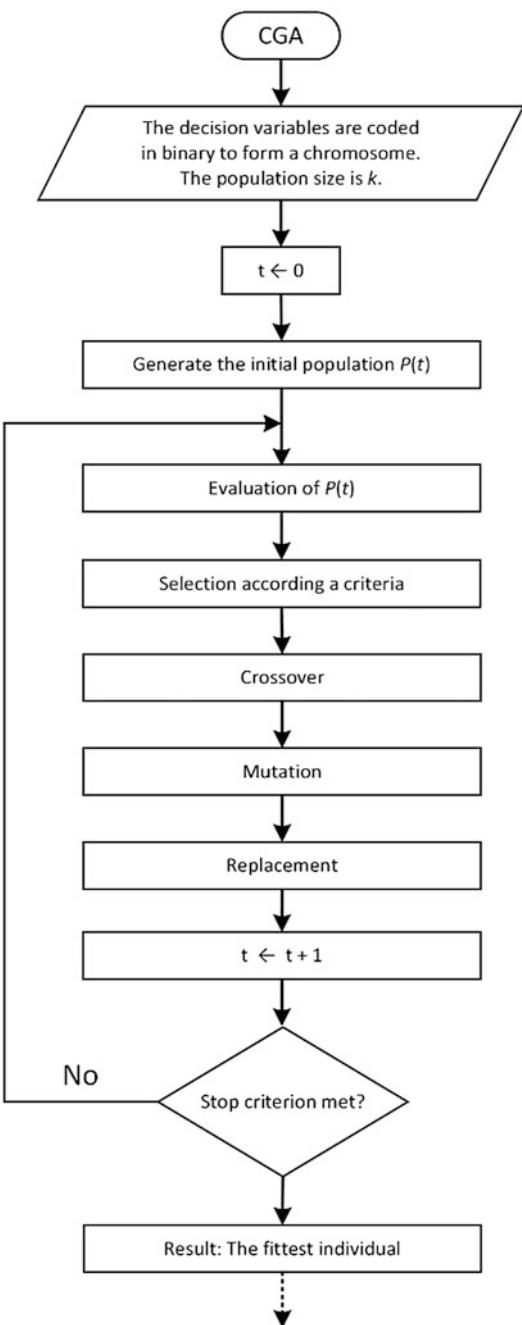
The evolution in time t , of an isolated quantum system is described by a unitary transformation from an initial state $|\psi(0)\rangle$ to a final state $U(t)|\psi(0)\rangle = U|\psi(0)\rangle$ this means that a quantum logic gate operates in an isolated quantum computer, transforming the state to the point to which a measurement is made of the state.

3 Quantum Genetic Algorithms

The Genetic Algorithm (GA) is an optimization and search technique based on the principles of genetics that mimics the process of natural selection. It is an exploratory algorithm that manipulates a population of individuals called chromosomes; in essence, all the GAs programs are coded for a conventional binary computer. On the other hand, QGA is a computational method that combines the concepts and principles of quantum computing and GAs, it uses quantum registers to represent a chromosome (individual) and quantum gates as genetic operators. QGAs are characterized for having rapid convergence and good global search capabilities. Similar to the conventional Genetic Algorithm (GA), the QGA is a probabilistic algorithm, but since it is based on quantum registers, they can represent a linear superposition of solutions. With the aim of extracting some computational advantages of the QGA in contrast with the conventional GA, a short comparison between them is given.

Figure 3 describes the block diagram of a conventional GA; in the simplest way, the decision variables (genes) of a problem are coded in binary to form a chromosome. In the algorithm five important operations can be identified, they are: Initialization, evaluation, selection, variation which includes mutation and crossover

Fig. 3 Block diagram of a conventional genetic algorithm



operations, and replacement. At iteration $t = 0$, the initial population $P(t)$ of size k is created at random, uniformly distributed over the entire search space. Next, $P(t)$ is evaluated according to a criteria (fitness function), and the promising solutions are selected; which in turn are varied to generate new candidate solutions sharing similarities with the original ones, but they are novel in some way. The new solutions replace the last population and the iteration counter is updated ($t \leftarrow t + 1$). Operations from evaluation to replacement are iterated until the stop criterion is met.

Figure 4 shows the block diagram of a QGA. Here, the chromosomes are made up by quantum registers, hence a chromosome can represent the superposition of all possible states as it is shown in Fig. 5. In the algorithm the first step is to generate the initial population $Q(t)$, to do this, a quantum register $|\Psi\rangle$ is created and initialized with the $|00\dots0\rangle$ value, then it is put in the superposition state to take measurements and forming a population $P(t)$ of size k . All the chromosomes are evaluated using a cost function, the population is sorted using an elitist strategy, the best individual b is saved, new quantum registers are created using the best individual in the actual population, and a new population $P(t)$ is created by taking measurements of the new quantum registers. The process stops when a finish criteriais fulfilled.

4 Experimental Results

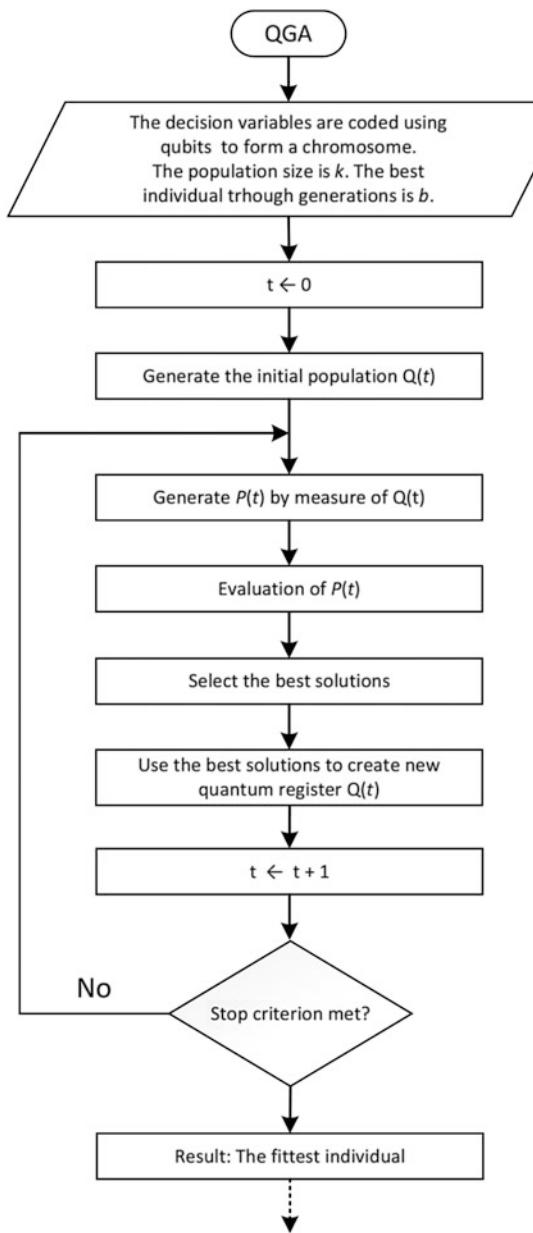
We performed several experiments to show the advantages of the QGA, they were achieved using the next computer equipment and software:

- A desktop computer based on the Intel Core i7 Processor (i7-3930 K).
- An Nvidia GPU card GTX760.
- An Nvidia GPU card GTX440.
- The Ubuntu 14.10 LTS GNU/Linux distribution operating system.
- The quantum computer simulator for Matlab Quack! by Rohde [8].
- Our GPU_Quack! implementation, which is a modification of the original Quack! that exploits the advantages of the Nvidia GPU cards.
- Matlab R2012.

The goal of the experiments was to find the global maximizer of the single variable multimodal function $f(x) = 0.1x \sin(0.1x) \in [0, 250]$. In the experiments different amount of bits and qubits to represent real numbers were used.

In the first two experiments, the idea was to observe the distribution of solutions for different run of the CGA and QGA, as well as the behavior of the population in one run for both algorithms. The CGA and QGA run during 30 generations in each experiment. The last two experiments are devoted to illustrate the speed advantages of running the QGA using the GPU_Quack! that take advantage of using the CUDA cores.

Fig. 4 Block diagram of a quantum genetic algorithm



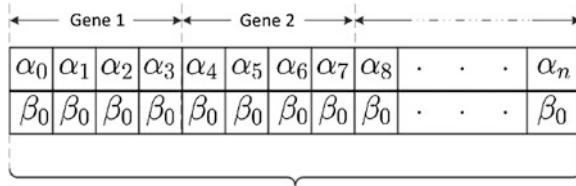


Fig. 5 Quantum chromosome structure, each gene represents a decision variable

The conducted experiments were:

- Experiment 1: Classic Genetic Algorithm (CGA) versus Quantum Genetic Algorithm (QGA). Here, we used a resolution of 10 bits and qubits for the CGA and QGA, respectively; eight bits/qubits for the integer part, and two bits/qubits to represent the fractional part. For the interval $[0, 250]$ the maximal resolution is 0.25. To take statistical values, each algorithm run 100 times, during 30 generations. Particular details of the parameters of each algorithm are shown in Table 1, Fig. 6a, b show the distribution of solutions of the 100 runs for the CGA and QGA, respectively. Table 2 shows the statistical values of the experiment. Figure 7a, b illustrate the best individual, as well as the mean population behavior of one run.
- Experiment 2: Classic Genetic Algorithm (CGA) versus Quantum Genetic Algorithm (QGA). Here, we used a resolution of 10 bits and qubits for the CGA

Table 1 Experiment 1. CGA and QGA parameters

Parameter CGA	Value	Parameter QGA	Value
Number of generation	30	Number of generation	30
Cross-over probability	50 %	Population for evolve	50 %
Mutation probability	1 %	Population size	8
Population size	8		

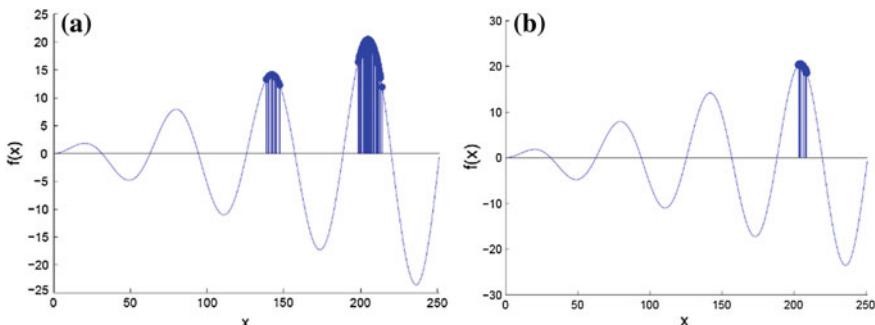


Fig. 6 Experiment 1. Optimal values found after 100 run. **a** CGA, **b** QGA

Table 2 Experiment 1. CGA and QGA statistical results of 100 runs

Result	CGA	QGA
Standard deviation best individual	2.318	0.5274
Media best individual	19.672	20.441
Mean best individual	18.706	20.1284
Standard deviation mean individual	2.8128	3.7564
Media mean individual	17.248	15.9562
Mean of the mean individual	16.792	15.2859
Standard deviation min individual	10.973	25.6832
Media min individual	12.601	-18.8874
Mean min individual	9.0114	-15.1958

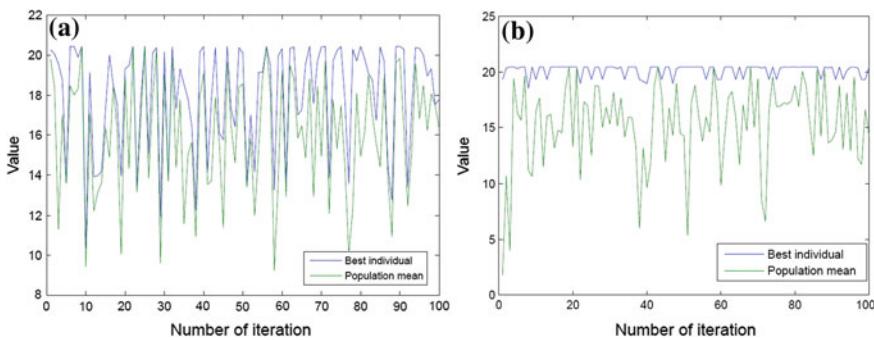


Fig. 7 Experiment 1. Behavior of individuals in the population in one run through 100 iterations. **a** CGA, **b** QGA

Table 3 Experiment 2. Classical and quantum genetic algorithm parameters, second test

Parameter CGA	Value	Parameter QGA	Value
Number of generation	30	Number of generation	30
Cross-over probability	50 %	Population for evolve	50 %
Mutation probability	10 %	Population size	8
Population size	8		

and QGA, respectively; nine bits/qubits for the integer part, and one bits/qubits to represent the fractional part. For the interval [0, 500] the maximal resolution is 0.5. Similarly than in Experiment 1, we have illustrated in Table 3 particular implementation details of the CGA and QGA algorithms. Figure 8a, b show the distribution of solutions of 100 runs of the CGA and QGA, respectively. In Table 4 are contained the statistical values of the experiment. Figure 9a, b illustrate the best individual, as well as the mean population behavior of one run.

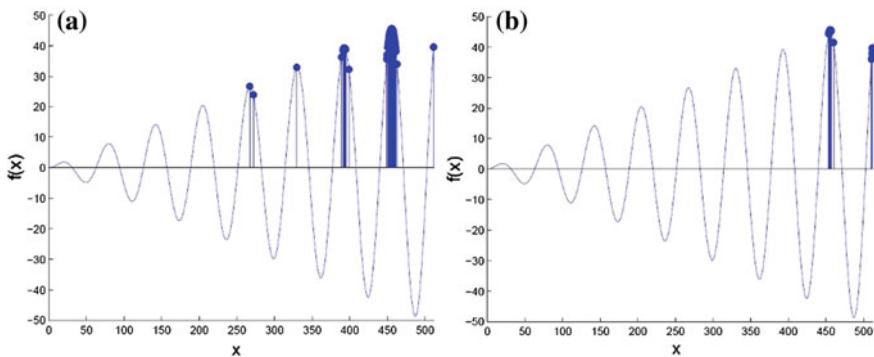


Fig. 8 Experiment 2. Optimal values found after 100 run. **a** CGA, **b** QGA

Table 4 Experiment 2. Classical and quantum genetic algorithm results of 100 runs

Result	CGA	QGA
Standard deviation best individual	4.0876	2.204
Media best individual	44.411	45.55
Mean best individual	42.231	44.746
Standard deviation mean individual	7.8336	7.9429
Media mean individual	26.531	34.655
Mean of the mean individual	26.686	3.772
Standard deviation min individual	26.23	34.341
Media min individual	-17.626	-1.231
Mean min individual	-14.45	-6.5304

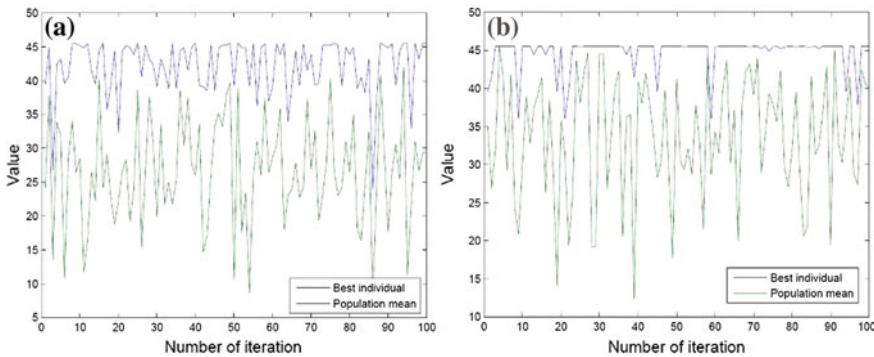


Fig. 9 Experiment 2. Behavior of individuals in the population in one run through 100 iterations. **a** CGA, **b** QGA

Table 5 CPU versus GPU

Amount of qubits	CPU Intel core I7 (s)	GPU GT 440 (s)	Speedup	GPU GTX 760 (s)	Speedup
8	9.42	32.39	0.29	34.03	0.27
9	51.95	47.55	1.09	43.01	1.2
10	341.52	205.31	1.66	54.35	6.28
11	2911.78	1733.11	1.68	275.96	10.55
12	19,882	14,945.67	1.33	1810.91	10.97

- Experiment 3: QGA programmed with the Quack! versus QGA programmed with the GPU_Quack! running on the GT440 Nvidia card. Here, we used different amount of bits and qubits for the CGA and QGA, respectively. Table 5 illustrates the processing speed advantages of the GPU_Quack! for different amount of qubits.
- Experiment 4: QGA programmed with the Quack! versus QGA programmed with the GPU_Quack! running on the GTX760 Nvidia card. Table 5 illustrates the processing speed advantages of the GPU_Quack! for different amount of qubits.

5 Conclusions

With the first two experiments the advantages of the QGA over the CGA were illustrated using plots of the solutions distribution of 100 runs of each algorithm, it can be observed that the QGA has better exploratory characteristics than the CGA. In the first experiment, the CGA, an 85 % of times found values close to the global maximizer, and the 15 % of times found local optimal values; on the other hand, the QGA, a 100 % of times found values close to the optimal maximizer. Similarly, in the second experiment, the CGA 79 % of times found values near the optimal maximizer, whereas the QGA, a 91 % of times locates the values close to the optimal value. The difference is because in both experiments the search space was increased. The third and fourth experiment demonstrated that the GPU_Quack! outperforms the original Quack! since modifications makes that the quantum simulator takes advantage of using the massively processing capabilities of the CUDA cores.

In Table 5, were shown different performance values for different GPUs compared against a high-end CPU. A good performance was obtained when a fair comparison of a high-end CPU with a high-end GTX 760 GPU was achieved; differently, when a low-end GT 440 GPU was compared against the CPU, a notable speedup was obtained for small amounts of qubits, but the speedup decreased when the amount of qubits was increased, this was due to the low-end GPU that was used, compared to the high-end CPU.

Acknowledgments We thank to Instituto Politecnico Nacional (IPN), to the Comisión de Fomento y Apoyo Académico del IPN (COFAA), and the Mexican National Council of Science and Technology (CONACYT) for supporting our research activities.

References

1. Fujisaki, M., Miyajima, H., Shigei, N.: Data search algorithms based on quantum walk **1**, 164–169 (2012)
2. Feynman, R.P.: Simulating physics with computers. *Int. J. Theor. Phys.* **21**(6–7), 467–488 (1982)
3. Shor, P.W.: Algorithms for Quantum Computation: Discrete Logarithms and Factoring. AT&T Bell Labs, New Jersey (1995)
4. Ladd, T.D., Jelezko, F., Laflamme, R., Nakamura, Y., Monroe, C., O'Brien, L.: Quantum computers. *Nat. Int. Wkly. J. Sci.* **464**, 45–53 (2010)
5. D-Wave Systems: Commercial quantum computing company (2014). <http://www.dwavesys.com/>
6. Johnston, H.: Is d-wave's quantum computer actually a quantum computer? (2014). <http://physicsworld.com/cws/article/news/2014/jun/20/is-d-wave-quantumcomputer-actually-a-quantum-computer>
7. Han, K.-H., Kim, J.-H.: Genetic quantum algorithm and its application to combinatorial optimization problem. In: Proceedings of the 2000 congress on evolutionary computation, pp. 1354–1360 (2000)
8. Rohde, P.P.: Quack! a quantum computer simulator for matlab. Centre for Quantum Computer Technology, Department of Physics, University of Queensland, Brisbane, Australia (2005)

Implementing Pool-Based Evolutionary Algorithm in Amazon Cloud Computing Services

Rene Márquez Valenzuela and Mario García Valdez

Abstract In this article, we describe the implementation of a pool-based evolutionary algorithm on a cloud computing environment. We use the EvoSpace model for pool-based evolutionary algorithms (Pool-EA), which is designed to exploit cloud computing resources in order to generate better results than using a simple GA. The platform has been deployed in the cloud using Amazon services (Amazon EC2).

1 Introduction

A distributed algorithm is an algorithm designed to take advantage of computational resources of a computer, or in this case, a series of computers to work on finding an optimal solution in less time. This algorithm design is very useful because, as the internet a massive network of computers connected together, they can leverage resources of other computers for solving complex problems without the need to hire specialized services; perhaps they can't reach the level of processing of a supercomputer, but this concept can be a choice between conventional computer and the use of supercomputers to solve problems. Distributed algorithms are used in many varied application areas of distributed computing, Such as telecommunications, scientific computing, distributed information processing, and Real-time process control [1].

Distributed algorithms are a sub-type of parallel algorithms, typically executed concurrently, with separate parts of the algorithm being run simultaneously on independent processors, and having limited information about what the other parts of the algorithm are doing. One of the major challenges in developing and implementing distributed algorithms is successfully coordinating the behavior of the independent parts of the algorithm in the face of processor failures and unreliable communications links. The choice of an appropriate distributed algorithm to

R.M. Valenzuela · M.G. Valdez (✉)
Tijuana Institute of Technology, Tijuana, BC, Mexico
e-mail: Mario@tectijuana.edu.mx

solve a given problem depends on both the characteristics of the problem, and characteristics of the system the algorithm will run on, such as the type and probability of processor or link failures, the kind of inter-process communication that can be performed, and the level of timing synchronization between separate processes [1].

These concepts are important to the field of evolutionary algorithms, because such algorithms may need much computing power to perform their tasks, so divide the problem into tasks and solve each one in a node of a network of computers concurrently can give better results in less time and with much less cost.

There are some models of Distributed Evolutionary Algorithms, such as Island-based models (Island-model distributed GA's), Distributed PSO's, among others; in this paper the case of Pool-based evolutionary algorithms is studied.

The pool-based evolutionary algorithms (Pool-EA's) basically work with the principle that a central population exist, and that all the participants in the process can access that population, take a sample, and apply some series of tasks or operations, and return the results back to the population, in this case, an evolved series of individuals.

In this paper, we worked using The EvoSpace Model for Pool-based Evolutionary algorithms [2] which is a model built based on a central repository where a population of individuals is concentrated and can be accessed from multiple computers to perform evolutionary processes on their individuals. This model incorporates some of the main principles of the tuple-space model, and incorporates additional mechanisms and concepts to address some of the main issues for Pool-EAs, like starvation of the population and work redundancy.

The remainder of this paper is organized as follows. Section 2 contains important information about the topics we are addressing in this paper. Section 3 has the explanation of the model we are going to be implementing. In Sect. 4 the architecture of the model and the cloud implementation. Section 5 contains the conclusions.

2 Evolutionary Algorithms

Evolutionary algorithms are a branch of artificial intelligence, which began to emerge in the early '60s, with the emergence of evolutionary programming, introduced by Lawrence J. Fogel; also the creation of genetic algorithms, proposed by John Henry Holland. And a different approach, created by Ingo Rechenberg and Hans-Paul Schwefel called Evolution Strategies. These three techniques were developed separately, but then combined, with some others, to form the field of evolutionary computing, which combines all these techniques, based on the principles of Charles Darwin, about the survival of the fittest and the evolution of species. Since then, the field of artificial intelligence has grown exponentially since added many more methods, also based on the nature and behavior of the species to find solutions to computationally complex problems.

This type of algorithms and combinatorial optimization involves continuous optimization problems. Its algorithms can be considered overall optimization methods with a metaheuristic or stochastic optimization character and are mostly applied for black box problems (Derivatives not known). Often in the context of expensive optimization.

Evolutionary computation uses iterative progress, such as growth or development in a population. This population is then selected in a guided random search using parallel processing to achieve the desired end. Such processes are often inspired by biological mechanisms of evolution.

As evolution can produce highly optimized processes and networks, it has many applications in computer science. In this paper, we use Genetic algorithms as the main technique to test with some benchmark functions, using the Pool-based model, to analyze its performance.

2.1 Genetic Algorithms

Genetic algorithms are a heuristic search technique, part of the evolutionary algorithms, using the theory of natural selection of Charles Darwin, to find optimal solutions in a search space. This technique is used to generate useful solutions to optimization problems and search problems. Such algorithms include processes such as selection, mutation and crossover, which is applied to a population of individuals, which represent possible solutions to a problem, and these individuals are evolved to find an optimal solution, this using a fitness function, which is used to evaluate each individual and get to the best solution. Since Holland's work in 1975 [3], several variations to the original genetic algorithm have been developed, using different techniques of selection, mutation, crossover, and to find the best parameters.

2.1.1 Genetic Representation

Some of the most important elements of genetic algorithms are its populations and individuals. Individuals represent one possible solution to the problem that needs to be solved. The individual needs a representation that can be understood by the GA and that can be evolved by its mechanisms. Genetic representation can encode appearance, behavior, physical qualities of individuals. Genetic algorithms use linear binary representations. The most standard one is an array of bits. Arrays of other types and structures can be used in essentially the same way. The main property that makes these genetic representations convenient is that their parts are easily aligned due to their fixed size. This facilitates simple crossover operation. Variable length representations were also explored in Genetic algorithms, but crossover implementation is more complex in this case [4].

2.1.2 Selection

Selection is the process of selecting a portion of the initial population of a genetic algorithm, usually the fittest of the individuals, to breed a new generation of individuals. Individual solutions are selected through a fitness-based process, measuring each individual with a fitness function. Not necessarily needs to be the fittest of individuals, but there are many other metrics to select certain individuals, dependant on the type of problem to solve. The fitness function is defined over the genetic representation and measures the quality of the represented solution. There are many types of selection; there's no "best" method per say, each method of selection can work best with certain type of problem:

- **Roulette:** A selection operator in which the chance of a chromosome getting selected is proportional to its fitness (or rank). This is where the concept of survival of the fittest comes into play.
- **Tournament:** A selection operator which uses roulette selection N times to produce a tournament subset of chromosomes. The best chromosome in this subset is then chosen as the selected chromosome. This method of selection applies addition selective pressure over plain roulette selection.
- **Best:** A selection operator which selects the best chromosome (as determined by fitness). If there are two or more chromosomes with the same best fitness, one of them is chosen randomly.
- **Random:** A selection operator which randomly selects a chromosome from the population.

2.1.3 Crossover

The crossover is a genetic operator which its main function is to combine individuals, to get a new individual, containing elements from its parents, and possibly, being a better solution for the problem that the genetic algorithms is trying to solve. It is analogous to reproduction and biological crossover. As the selection operator, there are many different methods to choose from, and also there is no best method for all types of problems:

- **One-point crossover:** A single crossover point on both parents' organism strings is selected. All data beyond that point in either organism string is swapped between the two parent organisms. The resulting organisms are the children.
- **Two-point crossover:** calls for two points to be selected on the parent organism strings. Everything between the two points is swapped between the parent organisms, rendering two child organisms.
- **Cut and splice:** results in a change in length of the children strings. The reason for this difference is that each parent string has a separate choice of crossover point.

2.1.4 Mutation

Mutation is a genetic operator which main purpose is to add diversity to the individuals in a population; sometimes, crossover is not enough to find the best solutions. Mutations help this process by changing one or more chromosomes of the individual and possibly making it a better solution from one generation to another. This helps maintaining diversity, by preventing the populations of individuals from becoming too similar to each other, thus slowing or even stopping evolution.

There are different types of mutations:

- **Bit string mutation:** Suitable for binary string; flips one bit at random from the chromosome. The probability of a mutation of a bit is $1/t$, where t is the length of the binary vector.
- **Flip bit:** Takes the chosen chromosome and inverts the bits (i.e. if the chromosome bit is 0, it is changed to 0)
- **Non-uniform:** The probability that amount of mutation will go to 0 with the next generation is increased by using non-uniform mutation operator. It keeps the population from stagnating in the early stages of the evolution. It tunes solution in later stages of evolution. This mutation operator can only be used for integer and float genes.
- **Uniform:** This operator replaces the value of the chosen gene with a uniform random value selected between the user-specified upper and lower bounds for that gene. This mutation operator can only be used for integer and float genes.
- **Gaussian:** This operator adds a unit Gaussian distributed random value to the chosen gene.

3 EvoSpace Model

EvoSpace is a model for the development and execution of pool-based evolutionary algorithms (Pool-EA) [2]. This model was developed to experiment with these kind of distributed algorithms, and to be able to exploit available cloud computing services; using this alternative, we don't need to physically build a network of computers that work in parallel in search for a solution of a problem. With EvoSpace, we can make all that work in the cloud. In this case, we implemented the architecture using Amazon EC2 services, for the deployment of virtual machines in the cloud, which are going to be doing the genetic algorithm work. In the original work, EvoSpace was using different services, like Heroku and PiCloud. The full architecture is explained in the next pages.

3.1 *EvoSpace Components*

This model needs three main components for it to function; these components work in conjunction with each other and are explained in the next subsections.

3.1.1 *EvoStore Container*

This is a memory space shared by several processes following the principles of the tuple-space model [2]. In this shared space, the population of individuals (our possible solutions to a problem) is stored, but the GA operations are not going to be applied here. Here, the main operations that are applied are to initialize a population, insert and withdraw individuals from the population. This is based primarily on the Pool-based principle, that the population should be stored in a centralized repository, while distributed clients asynchronously extract a sample of the population and apply genetic operations to it, then return the evolved sample to the central repository.

3.1.2 *Individuals (Population)*

This is the set of possible solutions for the problem we are trying to solve. The set of individuals that are going to be evolving in different clients asynchronously and then getting returned to the *EvoStore* repository, possibly being fittest to the solution we are trying to find. Individuals are stored as dictionaries, an abstract data type that represents a collection of unique keys and values and being associated one to another. A dictionary representing a individual in this model is composed of these fields: the **id** string which helps to identify each individual from one to another; a **chromosome** string, which length and data depends on the problem, and a **fitness** value.

3.1.3 *Workers*

The workers are a group of clients (computers) that are going to be accessing the central repository, getting samples of the population, and applying genetic operators to it, looking for optimal solutions for the problem that it is trying to solve. They do not communicate with each other; each worker's job is independent from the other ones. If the model is applied locally in a computer, and it has hardware capable of multithreading, each core available of the processor can act as an individual worker, each core doing genetic operations to a sample of the population. In this case, as the model is deployed in the cloud, we can deploy certain number of virtual machines, and each one would represent a worker. This is important due to the easy scalability that the model allows.

4 Cloud Architecture

The implementation of the EvoSpace model for pool-based evolutionary algorithms in a cloud computing environment is presented here. All of the components were developed using Python programming language (Fig. 1).

In this diagram we can see the different components previously explained, and we can see a general look of their interaction with each other. Previously we explained what is each component, but now we need to explain the tools that we use for their deployment and different libraries for the environment.

- **Individuals (Population):** for the creation of the individuals (population) we use the Distributed Evolutionary Algorithms in Python (DEAP) library [5], as the other components, such as EvoSpace, is completely developed in Python language. The generated population is sent to EvoSpace and stored there for the workers to access in later in the process.
- **Workers:** for the creation of the workers, we used *Celery* (<http://www.celeryproject.org/>) which is a very useful and powerful library for Python; Celery is an asynchronous task queue/job queue based on distributed message passing. This helps us to make certain tasks, such as initialization of the population, and evolution of it, be available to the number of workers we need to deploy for a experiment. Using Celery, in conjunction with Redis as the link between EvoSpace and the workers, each worker can take a sample from the population and apply genetic operations to it, then send back the results to the central repository, as explained before.

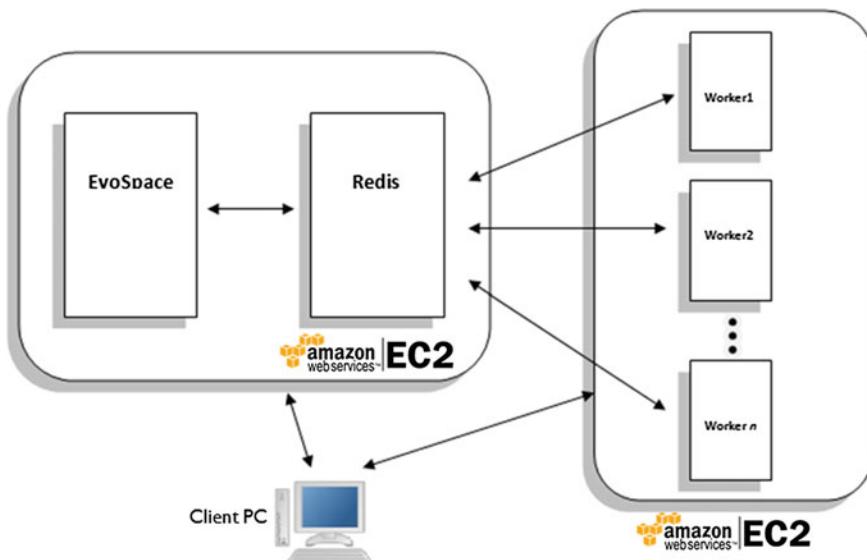


Fig. 1 Evospace cloud architecture

For the hosting and deployment of workers and EvoSpace server, we used Amazon Elastic Compute Cloud (EC2) service; which is a web service that provides resizable compute capacity in the cloud. It is designed to make web-scale cloud computing easier for developers. It enables you to increase or decrease capacity within minutes, not hours or days. You can commission one, hundreds or even thousands of server instances simultaneously. Of course, because this is all controlled with web service APIs, your application can automatically scale itself up and down depending on its needs.

We implemented the scalability factor directly in a script, which allow us to choose how many workers we will need for the experiment and then launched them and get them ready for when we run the program.

There were previously made configurations for the virtual machines. On Amazon, we need to create a virtual machine, and configure it for our needs; in our case, we used virtual machines running Ubuntu 14.04 operating system, and then download all the require software to run the project, like Celery, Redis and a lot more libraries. When the virtual machine is completely configured, we can create images of that virtual machine, which in Amazon are called AMIs, and save it to run more instances of the same image when needed. This is very important for the launch of the machines that will do the worker function, as it is pretty scalable for what we need.

Having all this, we wrote a script that lets us run the number of instances we specify; the launch of the instances in Amazon takes some minutes to complete, so in the script we used a library to be checking the status of each instance; this library is called *Boto* (<https://github.com/boto/boto>) and this library lets us connect to Amazon services and a lot of functionalities like launch and terminate instances. When the instances are launched, we send a configuration file to each of the instances. In this configuration file, we have important parameters for the execution of the algorithm, like number of workers, size of population, size of the sample for each worker, length of chromosome, max number of samples for each worker, etc. The script then initialize Celery as a daemon in each of the worker instances and points them to the Redis and Evospace server, which should be initialize in other server preferably. Now that the workers are listening for petitions, the script runs a specified file in which petitions are then sent. First, the one to initialize a population, then, the one that main purpose is to get samples and evolve them; this task is run for every worker we specified. For example, if we run a experiment with 4 workers, the task will be called 4 times, and as we have 4 workers listening, each worker will be taking one of those task and completing it. Then each worker will be evolving the samples and returning them to the central repository, till we find the optimal solutions, or till the max samples for each worker ends. When the algorithm ends, the best individuals for each worker are saved in a text file, then automatically downloaded to a folder in our computer. In this file we have information as the total time of execution, number of evaluations, the best individuals and their fitness value, etc. This file saves information from every worker; it is very useful because we can see the evolution of the population and how every worker took samples and reached the optimal solutions.

5 Conclusions

With this implementation of the EvoSpace model on a different service as Amazon EC2, we can conclude that it is a pretty adaptable platform and not hard to implement in different services; Amazon EC2 is a pretty reliable service for this kind of project, as we haven't got any problems initiating and running experiments as of yet. One of the negatives of this service is obviously the cost; it isn't as high as other services, but it is not totally free. Implementing the platform on a local network with many computers can be an alternative, but the scalability factor would not be easy to take advantage of.

Acknowledgment We would like to express our gratitude to the CONACYT and Tijuana Institute of Technology for the facilities and resources granted for the development of this research.

References

1. Lynch, N.: Distributed algorithms. In: *Distributed Algorithms*. San Francisco, California (1996)
2. Garcia Valdez, M.: The EvoSpace model for pool-based evolutionary algorithms. *J. Grid Comput.* Nov 2014
3. Holland, J.H.: *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Michigan (1975)
4. Engelbrecht, A.: *Fundamentals of Computational Swarm Intelligence*, pp. 25–26. Wiley, New York (2005)
5. Fortin, F.A.: DEAP: evolutionary algorithms made easy. *J. Mach. Learn. Res.* (2012)

An Ant Colony Algorithm for Solving the Selection Portfolio Problem, Using a Quality-Assessment Model for Portfolios of Projects Expressed by a Priority Ranking

S. Samantha Bastiani, Laura Cruz-Reyes, Eduardo Fernandez, Claudia Gómez and Gilberto Rivera

Abstract One of the most important problems faced by any organization is make decisions about how to invest and manage the resources to get more benefits; however, the organizations resources are not enough to support all portfolios proposals. To these problems that face the executives of the big organizations, is known as Select Portfolio Problem. In this work is developed an ant colony algorithm, which is an especially effective meta-heuristic, this meta-heuristic is hybridized with a multi-objective local search, this strategy allows using knowledge of the ant, to build potential solutions, knowledge is obtained through the pheromone trail left by ants when find good solutions, for that the algorithm does not converge prematurely an evaporation strategy is implemented. The strategy meta-heuristic include an optimization model for portfolio selection called discrepancies model, this model is implemented when the information concerned to the quality of the projects is in form of ranking, besides help to evaluate portfolios through ten criteria to maximize the impact of the portfolio. This approach allowed reaching privileged areas of Pareto's front, where identified solutions that reflect the

S. Samantha Bastiani · L. Cruz-Reyes (✉) · C. Gómez · G. Rivera
Instituto Tecnológico Nacional de México, 1ro. de Mayo y Sor Juana I. de la Cruz s/n,
CP 89440 México, TAMPs, Mexico
e-mail: lauracruzreyes@itcm.edu.mx

S. Samantha Bastiani
e-mail: b_shulamith@hotmail.com

C. Gómez
e-mail: cggs71@hotmail.com

G. Rivera
e-mail: riveragil@gmail.com

E. Fernandez
Universidad Autónoma de Sinaloa (UAS). Justicia Social SN, Ciudad Universitaria, 81229
Los Mochis, Sinaloa, Mexico
e-mail: eddyf@uas.edu.mx

preferences of the decision maker. The experimental tests show the advantages of our proposal, providing reasonable evidence of its potential for solving the select portfolio problems with many objectives.

1 Introduction

The business and government organizations face problems of resource allocation and selection of the best portfolio, this problem is a common and often critical management issue. Besides to be faced with these problems in selecting project portfolios, the decision maker (DM) usually starts with limited information about projects and portfolios and His/her time is often the most critically scarce resource. To solve the optimization problem when the DM only has the information a ranking of priorities have been developed some multi-criteria approaches, which comprises a variety of techniques for exploring the preferences of concerned DMs as well as models for analyzing the complexity inherent to real decisions. Some of the broadly known multi-criteria approaches are MAUT (cf. [3]), AHP (cf. [19]), outranking methods (cf. [2, 19]), and rule-based approaches (cf. [7]). But, in many situations the DM feels more comfortable employing simple decision procedures, because of the lack of available information, lack of time, aversion to more elaborate decision methods, and even because of his/her fondness for established organizational practices. Multi-criteria analysis constitutes a good option to evaluate public projects because it may handle intangibles, ambiguous preferences, qualitative and imprecise information, and veto conditions. Different multi-criteria methods have been proposed for addressing project evaluation and portfolio selection (e.g. [1, 5, 8, 11, 13, 14, 16, 18, 20, 21]), their advantages are documented in the specialized literature (e.g. [15, 17]). These methods in conjunction with a meta-heuristic provide a best compromise solution. One the meta-heuristics used to solve the optimization problem is the ant colony system, as has been found to give good results. This approach has been used to solve the problem of portfolio selection, some work that has used this approach related to the problem of portfolio selection are [15, 17].

Therefore in this paper suggests a new approach called ACO-SPRI (ant colony algorithm), that with information of the selection portfolio problem, allows the ant leaves a pheromone trail which memorize the characteristics of “good” generated solutions and thus guide the construction of new solutions by a rule called roulette wheel selection technique. In conjunction with the algorithm, it integrates a new model for selection of portfolios, which makes use of the ranking of a set of projects according to the preferences of a decision maker. Joining these approaches, help to explore the solution space and thus find the best compromise solution. The contribution is structured as follows: a criticism of previous related approaches is given in Sect. 2. In Sect. 3, an ant colony meta-heuristic method for solving the multi-objective problem is described. In Sect. 4 details the proposed multi-objective optimization model of project portfolios and on this background several computer experiments are presented in Sect. 5. Finally, we draw some conclusions in Sect. 6.

2 Related Works

The most solution methods seek to generate the Pareto frontier, and later, by some interactive method, multi-criteria procedure or heuristic, try to identify the best compromise. This way to identify the best solution is commonly referred to as a posteriori preferences modeling. However, most recent works show the advantages of use of a priori articulation, as preferences by creating a fuzzy outranking relation, followed by a generating process of a subset of the Pareto frontier [4].

One of these methods is in [4], that introduce a new approach called Non-Outranked Ant Colony Optimization, that in conjunction with a local search and a Roulette Wheel Technique to constructed the best solution, optimizes project portfolios with a priori articulation of decision-maker preferences based on an outranking model called fuzzy preference model, this meta-heuristic approach provides an approximation to the Pareto frontier which reduces only to areas of interest.

Other approach proposed is of the [6], which requires priori preference information. The aim of this work is at providing a heuristic approach in the field of multi-objective portfolio selection by introducing Ant Colony Optimization which in conjunction with a multi-objective portfolio selection model and a pseudo-random proportional rule to construct the best solution, provides an approximation the Pareto frontier. This approached differs not only in the modeling of the problem but also in the number of optimization criteria, e.g., five to ten objectives in the numerical examples.

One of the principals methods discussed in the present research is the ranking methods, these have a high popularity in the R&D project management, and in most government organizations are used, Cooper et al. [3]. This was approached by Fernandez and Olmedo [10] under the assumption that “the portfolio impact on a decision maker’s mind is determined by the number of supported projects and their particular rank”. In this paper propose a method for optimizing the portfolio, in addition of evaluate the disagreement of the portfolio as discrepancies, that in conjunction with a Genetic algorithm and a binary tournament technique with construct good solutions, will help to explore the solutions space and provides an approximation the Pareto frontier.

Other of the major methods based in ranking is proposed in [9], in this paper develops a model for composing social-oriented project portfolios. The information concerning the quality of the projects is in the form of a project ranking, which can be obtained by the application of a proper multi-criteria method. Inspired by outranking methods, the model provides a preference relation over project portfolios related. A best portfolio is primarily found through a multi-objective optimization that regards violations to pre-established decision makers preferences and the competing portfolios’ cardinalities. The overall solution is obtained by developing a Genetic Algorithm that include a Binary Tournament technique to construction the good solutions, which found an approximation to the Pareto frontier. Some of the characteristics of these works are shown in Table 1.

Table 1 Some of the characteristics of the work of state of the art

Related works	Algorithm	Pareto frontier	Construction of the solution	Model	LS
[9]	Genetic algorithm	Complete	Binary tournament	Model of discrepancies	X
[10]	Genetic algorithm	Complete	Binary tournament	Model of discrepancies	X
[4]	ACO	Complete and simplified	Roulette wheel technique	Fuzzy preference model	✓
[6]	ACO	Complete	Pseudo-random proportional rule	Multi-objective portfolio selection model	X
PR	ACO	Complete and simplified	Roulette wheel technique	Model of discrepancies	✓

ACO ant colony algorithm, LS local search, PR present research

3 An Ant Colony Optimization Method

The important features of the meta-heuristic techniques are satisfactory resolution of the problems of multi-objective optimization. Compared with multi-objective optimization methods based on mathematical programming, meta-heuristic approaches exhibit relevant advantages:

1. They have the ability to deal with a set of solutions (called a population) at the same time, allowing for the efficient frontier to be approximated in a single algorithm run, and
2. They are less sensitive to the mathematical properties of objective functions and problem constraints.

However, many researchers have argued that, when the number of objective functions increases, the selection of appropriate individuals for conducting the population towards the Pareto frontier becomes more difficult. In presence of many objectives, there are important concerns associated with the a posteriori articulation of preferences, but despite having many objectives, judging solutions to the problem of portfolio selection is not very difficult because the preferences among categories obey a lexicographic priority. That is, the set of objectives corresponding to the first category has a sort of pre-emptive priority with respect to the set of the second category; and this latter with respect to the set of objectives in the third category. Therefore, it is not a hard task for the *DM* to make judgments among non-dominated solutions.

In [4], good results were reported in problems with 9 and 16 objective functions working with an ant-colony based optimization method. Our proposed algorithm, ACO-SPRI (Ant Colony Optimization for Solving the Portfolio problem from project Ranking Information) takes ideas from Dorigo's ACS [7] and Cruz et al. [4], but incorporates changes in the construction and update phases to take into account

the features of the problem of portfolio selection. As most ACO algorithms, ACO-SPRI has two phases: construction and update. ACO-SPRI is shown in Algorithm 1.

Algorithm 1: ACO-SPRI

```

1: Data:  $N$ (All Projects),  $B$ (Budget)
2: IP ( $Tot\_iter \leftarrow 200$ ,  $Iter \leftarrow 0$ ,  $Num\_Ants \leftarrow 300$ )
3: Repeat
4:    $C \leftarrow \emptyset$ 
5:   for each ant in the colony do
6:      $A' \leftarrow \text{projects\_candidates}(N, C)$ 
7:     Repeat
8:        $k \leftarrow \text{Rule\_Selection}(N')$ 
9:        $C_k \leftarrow 1$ 
10:       $A' \leftarrow \text{projects\_candidates}(N, C)$ 
11:      until  $C \leq budget \leq B_{\text{Total}}$ 
12:       $C \leftarrow \text{Evaluation of fitness}(C)$ 
13:       $C \leftarrow \text{Local\_Search}(C)$ 
14:       $FNS \leftarrow \text{Fast non-dominated sorting}(C)$ 
15:      Pheromone deposit ( $FNS$ )
16:      Pheromone Evaporation ( $FNS$ )
17: until ( $iter = tot\_iter$ )

```

There are three main characteristics of ant-colony learning algorithms: a heuristic measure η that provides information related with the problem under study, and is used to calculate portfolio preferences. The second characteristic is the bi-dimensional measurement τ that records learning of all ants and is used to find the best solutions. Finally, the process of intensification carried out through a local search; this process acquires information from the portfolios previously constructed by ants to explore the nearest neighbors and thus find the quality solutions. These characteristics are further detailed in the following sections.

3.1 Pheromone Representation

The pheromone is the main component, since the pheromone trails memorize the characteristics of good generated solutions, which will guide the construction of new solutions by the ants. The update of the pheromone is carried out using the generated solutions, so the pheromone trails change dynamically during the search, to reflect the acquired knowledge. This component is represented by the letter τ and is modeled in ACO-SPRI as a bi-dimensional array of size $N \times N$, where N is the total number of applicant project proposals. The pheromone between two projects i and k is represented as $\tau_{i,k}$, and indicates how good it is that both projects receive financial support. The pheromone matrix acts as a reinforcement learning structure reflecting the knowledge gained by the ants that formed high-quality portfolios. At the beginning of the first iteration of ACO-SPRI, the pheromone matrix is

initialized to $\tau_{i,k} = 0.09$ for all $(i; k) \in Pr \times Pr$ (P_r , set of projects). After that, each ant constructs a feasible portfolio. In a colony with n_a ants, new solutions are generated after each iteration, and these solutions act are subdivided into domination fronts. The fronts are obtained by considering the objectives of the selection portfolio problem, Eq. 8. The set composed by these fronts is denoted by $F_{act} = \{F_0, F_1, F_2, \dots, F_j, F_{j+1}, \dots\}$, where F_0 contains then on-dominated solutions, F_1 contains the portfolios that are dominated by only one solution, F_2 those dominated by two solutions, and so forth. In general, the portfolios dominated by j solutions are in F_{j+1} . The set F will be pheromone intensification in order to increase the selective pressure towards the best compromise. Each pair of projects (i, k) for each solution C intensifies the pheromone trail according to Eq. (1).

$$\Delta\tau_{i,k} = \frac{|F_{act}| - f_{act}(C) + 1}{|F_{act}|} (1 - \tau_{i,k}) \quad (1)$$

where $f_{act}(C)$ is a function that indicates the front containing the portfolio C in the solution set constructed by the ants (and improved by local search).

To prevent premature convergence, the colony includes a strategy known as evaporation that reduces pheromone trails for fixed periods of time; this strategy is modeled by Eq. (2).

$$\tau_{i,k} = (1 - \rho)(\tau_{i,k}) + ((\rho)(|C|)) \quad (2)$$

where ρ is a constant value (0.63) and $|C|$ is the number of projects in the portfolio.

3.2 Selection Rule

The construction of solutions is done according to a selection rule. Each ant builds its portfolio by selecting the projects one by one, taking into account the local knowledge (heuristic). This considers the benefits provided by the project to the portfolio and how many resources the project consumes. Local knowledge for the k th project is denoted by η_k and is calculated by Eq. (3).

$$\eta_k = \frac{\frac{1}{r_k}}{c_k} \quad (3)$$

where for each Project k , c_k its budget requirement and r_k its assigned rank. If C is a partially-constructed portfolio, one or more projects may be included in its. From among all the project proposals, only those that are not part of C and whose inclusion favors the fulfillment of budgetary constraints should be considered. This set is known as the candidate project list and is denoted by N' , which is a subset of P_r (set of projects). The choice of which $k \in N'$ will be added is made by using the selection rule Eq. (4)

$$k = \begin{cases} \arg \max_{k \in N'} \{\Omega(C, k)\} & \text{if } p \leq \alpha_1, \\ S_{k \in N'} \{\Omega(C, k)\} & \text{if } \alpha_1 < p \leq \alpha_2, \\ s_{k \in N'} & \text{otherwise} \end{cases} \quad (4)$$

where α_1 is the intensification threshold, α_2 is the diversification threshold, p is a pseudorandom number between zero and one, $\Omega(C, k)$ is the intensification effect, $S_k \in N'$ is a middle state effect between intensification and diversification, and $s_k \in N'$ is the diversification effect. The value of $\Omega(C, k)$, which is the intensification criteria, is calculated according to Eq. (5), where w represents the weight given to the preference between η_k and $\tau_{i,k}$; where j is the consider project.

$$\Omega(C, k) = (w)(\eta_k) + (1 - w)(\tau_{i,k}) \quad (5)$$

The selection scheme $S_k \in N'$ is based on the known roulette wheel technique [12], with the probability of each project calculated as in Eq. (6). Note that $\alpha_1 - \alpha_2$ is the probability of triggering a middle state between intensification and diversification.

$$S_k = \frac{\Omega(C, k)}{\sum_{k \in N'} \Omega(C, k)} \quad (6)$$

Lastly, the diversification effect $s_k \in N'$ that is also a selection scheme, but exerted by choosing a project uniformly at random. Once that the ants have constructed the portfolios a local search process is used with the purpose achieving better resource. This process is described in detailed in the next section.

3.3 Multi-objective Local Search

The algorithm intensification is promoted by a greedy variable-neighborhood local search that is carried out on the solutions constructed by the ants. This search explores regions near the best known solutions by a simple scheme consisting of randomly selecting v projects, and generating all possible combinations of them for each solution. Small values for v provoke behavior that is too greedy, whereas large values produce intolerable computation times. In our experiments we obtained a good balance between these by using $v = \lceil \ln N \rceil$. The algorithmic outline for the local search is illustrated in Algorithm 1. As observed in Algorithm 1, the search starts by choosing v projects at random (Line 2), and generating all combinations of those projects (Line 3–4). Every combination is set for each portfolio (Lines 5–15). In Lines 10–13, the repairing procedure has two main goals: (1) improving clearly suboptimal portfolios and (2) bringing unfeasible portfolios to the feasible region. Thus, it has two conditions to check:

- If the generated solution is partially constructed; then the repairing process adds projects to portfolio, according to the selection rule (Eq. 4), but respecting the bits assigned by the current combination. This is done until no project can be added to the portfolio.
- If the generated solution is unfeasible; then there pairing process removes projects at random until the portfolio does not surpass the budget. The probability of removing a project is inversely proportional to its expected benefits. No project chosen by the current combination can be removed. In the generated instances, most solutions could become feasible by the repairing procedure.

Finally, the solution set is grouped by fronts (Line 16).

Algorithm 2. ACO-SPRI local search algorithm

```

1:  $SC \leftarrow \text{Solution\_built}()$ 
2:  $\lceil \ln N \rceil$ 
3:  $SP \leftarrow \text{Select\_projects}(v, N)$ 
4:  $Tot\_com \leftarrow \text{generate\_combinations}(SP)$ 
5: for each  $C \in SC$  do
6:   for  $j = 0$  until  $Total\_SC$  do
7:     for  $i = 0$  until  $Total\_com$  do
8:        $IC[i] \leftarrow \text{Insert\_combinations}(SC, Tot\_com)$ 
9:        $\text{Evaluation\_budget}(IC[i])$ 
10:      if  $IC[i].budget > budgetTot$  then
11:         $\text{Remove\_projects}(IC[i])$ 
12:      elseif  $IC[i].budget < budgetTot$  then
13:         $\text{Append\_projects}(IC[i])$ 
14:      end if
15:    end for
16:  end for
17: end for
18:  $FNS \leftarrow \text{fast non-dominated sorting}(IC)$ 
19: return  $FNS$ 

```

4 Proposed Model

Here, we present an improved model to reflect the portfolio's impact and the *DM* reluctance to reject relatively well-ranked projects.

Let us assume that the set of projects can be separated in $M \geq 2$ ordered categories. The impact of the portfolio can be modeled by several measures of the projects' quality. We suggest to use $M + 1$ “proxy” variables N_1, \dots, N_M , and P , where: N_i is the number of projects assigned to the i th category belonging to the portfolio and P is an indicator of the average rank of the projects in the portfolio, defined as shown in Eq. (7)

$$P = \frac{\sum_{i=1}^N x_i(N_i - r_i + 1)}{|C|} \quad (7)$$

where r_i is the project rank, $r = 1$ corresponds to the highest ranked project (the best one); $|C|$ denotes the total amount of projects in the portfolio; x_i is the indicator function of the portfolio. Comparing feasible portfolios, the vector (N_1, \dots, N_M, P) contains the arguments favoring a particular one in terms of projects' quality. The model should be completed by introducing some measures reflecting the *DM* disappointment. So, we introduce a new concept of discrepancy.

The term discrepancy reflects the negative effect exerted on the *DM*'s mind by the fact that one project, as compared to others, appears to have merits to belong to the portfolio and yet is not included in it. To model this concept we must introduce the reference portfolio that is created using the available budget and strictly following the priorities expressed in the ranking of projects. Let C_{ref} be a reference portfolio and a and b two P_r projects. It holds that Rank(a) better than Rank(b) $\Rightarrow (b \in C_{ref} \Rightarrow a \in C_{ref})$. A distinction should be made between three types of discrepancies:

Weak discrepancy: There is a weak discrepancy in the portfolio C generated by the project a if the following conditions are met: (i) $a \in C_{ref}$; (ii) $a \notin C$; (iii) the budget necessary for the project a is much higher than the average budget of the same-category projects belonging to C .

Strong discrepancy: There is a strong discrepancy in the portfolio C generated by the project a if the following conditions are met: (i) $a \in C_{ref}$; (ii) $a \notin C$; (iii) the budget necessary for the project a is significantly higher than the average budget of the same-category projects belonging to C , but not much higher.

Unacceptable discrepancy: There is an unacceptable discrepancy in the portfolio C generated by the project a if any of the following conditions are met:

First: (i) $a \in C_{ref}$; (ii) $a \notin C$; (iii) the budget necessary for the project a is not significantly higher than the average budget of the same-category projects belonging to C .

Second: There is $b \in C$ such that: rank (a) better than rank (b); (ii) $a \notin C$; the budget for a is not significantly higher than the budget for b .

Weak and strong discrepancies are separated taking into account the category of the discrepant project. Let us denote by $n_{sd,i}$ and $n_{wd,i}$ the number of strong and weak discrepancies in the i th category. Let us also denote by UND the set of portfolios that contain unacceptable discrepancies and by R_F the region defined by budgetary constraints in the portfolio space. So, we propose to solve the selection portfolio problem the following vector optimization, Eq. (8):

$$\begin{aligned} &\text{Optimize } (N_1, n_{sd,1}, n_{wd,1} \dots N_M, n_{sd,M}, n_{wd,M}, P) \\ &C \in R_F - UND \end{aligned} \quad (8)$$

where (N_1, \dots, N_M, P) contains the objectives to be maximized; the remaining objectives should be minimized.

The formulation in (8) is a clearly improved preference model. Its main disadvantage is a greater number of objective functions, which increases linearly with the number of categories. We suggest using 2–3 categories, which leads to 7–10 objective functions in the new model of discrepancies.

5 Computational Experiments

The meta-heuristic ACO-SPRI algorithm was implemented to solve the problem of portfolio selection. The objective of the tests was to validate the results obtained by integrating the new proposed model in ACO-SPRI algorithm; and then to verify the quality of solutions by comparing them with benchmark works [9, 10].

5.1 Experimental Environment

The following configuration corresponds to experimental conditions required for tests described in this paper:

1. Software: Operating System, Mac OS X Lion 10.7.5 (11G63b) Java Programming Language, Compiler NetBeans 7.2.1.
2. Hardware: computer equipment, Intel Core i7 2.8 GHz CPU and 4 GB of RAM.
3. Instances: An instance used for this study was taken from state of the art, reported in Fernandez [9, 10].
4. Performance Measure: In this case the performance is measured through the aforementioned objectives.

5.2 Some Illustrative Examples

The test instances were taken from [9] with the following characteristics:

- 25 projects with a total budget of 1.2 billion dollars to be distributed,
- 40 projects with a total budget of 5 billion dollars to be distributed,

These instances can be seen in Table 2. The algorithm yield was measured in accordance with the number of non-dominated solutions, that is, if a solution $\vec{x}^* \in \Omega$ it is a non-dominated solution, if there is no $\vec{x} \in \Omega$ so that: $f(\vec{x}) \leq f(\vec{x}^*)$ for $i = 1, \dots, k$ and for at least one value $i, f(\vec{x}) < f(\vec{x}^*)$, where Ω is the set of solutions (e.g. [19]).

Initially, a heuristic based on dividing the set of projects in three categories is applied. The categories are labeled: (1) priority, (2) satisfactory and (3) acceptable.

Table 2 Information of test instances: (a) 25 (b) 40

1a		1b	
<i>r</i>	Project cost	<i>r</i>	Project cost
1	100	1	450
2	80	2	230
3	50	2	300
4	90	3	150
5	80	4	193
6	150	5	198
7	90	5	283
8	160	6	300
9	130	7	194
10	200	7	210
11	50	8	262
11	50	9	196
11	50	10	150
12	30	11	82
12	30	11	284
12	30	12	74
13	20	13	151
14	60	14	198
15	90	15	83
16	80	15	194
17	90	16	184
18	100	17	194
19	10	18	283
20	70	19	139
20	70	20	184
<i>N</i> = 25		21	121
		22	193
		23	194
		23	262
		24	210
		25	200
		26	284
		26	151
		27	139
		28	230
		29	196
		30	192
		31	121
		32	208
		33	211
		<i>N</i> = 40	

Table 3 Some results

Solution	Priority category										Satisfactory category							Acceptable category			Card					
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	
ACO-SPRI_1	1	1	0	1	0	1	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	0	19
ACO-SPRI_2	1	1	1	1	0	1	1	0	0	0	1	1	1	1	1	1	1	1	1	0	0	0	1	1	1	19
ACO-SPRI_3	1	1	1	1	1	1	1	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	1	1	0	18
ACO-SPRI_4	1	1	1	1	1	1	1	0	1	0	1	1	1	1	1	1	1	1	1	0	0	0	1	0	0	18
ACO-SPRI_5	1	1	1	1	1	1	0	0	0	0	1	1	1	1	1	1	1	1	0	1	0	0	1	0	1	17
Solution from [9]	1	1	1	1	0	1	0	0	0	1	1	1	1	1	1	1	1	1	1	1	0	1	1	0	19	

The traditional heuristics for solving this problem considers giving support by following the rank order until the available resources run out.

Below, we present a group of experiments to verify the validity and advantages of our approach for solving different case studies. They prove that our approach has good potential for solving real resource-allocation problems.

(a) First example: 25 projects

Let us suppose that the DM is not averse to make contradictory decisions with respect to the ranking information, thus being prepared to accept some discrepancies in order to increase the number of supported projects. Hence, (s)he wants to evaluate non-dominated solutions to selection portfolio problem, Eq. 8.

Tables 3 and 4 show several non-dominated solutions to selection portfolio problem, Eq. 8, obtained by ACO-SPRI in the project and objective space, respectively. The average computer time was 78 s. The last row of these tables shows the best solution from [9].

Regarding the reference portfolio, all the non-dominated solutions obtained using ACO-SPRI algorithm drastically increase the number of supported projects whose cardinality varies between 17, 18 and 19 projects.

The solution with the least significant number of discrepancies and the maximum number of projects in priority category is denoted by ACO-SPRI_3. On the other hand, ACO-SPRI_2 is the solution supporting the highest number of projects and second in supporting priority projects. As compared to the result from [9], ACO-SPRI_2 supports more priority projects with less discrepancy. This better distribution of support is reflected in a better average of project rankings.

On comparing ACO-SPRI_2 and ACO-SPRI_3, only a DM with a marked fondness for increasing the total amount of supported projects would prefer the first over the second. By analyzing the structure of solutions, we can see that ACO-SPRI_2 substitutes the project 8 (priority) by projects 19 and 25, which generates a discrepancy in the category that should be prioritized.

Table 4 Results in the objective space

Solution	N_1	$n_{wd,1}$	$n_{sd,1}$	N_2	$n_{wd,2}$	$n_{sd,2}$	N_3	$n_{wd,3}$	$n_{sd,3}$	P
ACO-SPRI 1	6	2	0	7	1	0	6	0	0	11.68421053
ACO-SPRI 2	7	1	0	6	2	0	6	0	0	11.68421053
ACO-SPRI_3	8	0	0	6	2	0	4	0	0	12.66666667
ACO-SPRI_4	7	1	0	7	1	0	4	0	0	12.88888889
ACO-SPRI_5	8	0	0	4	3	0	5	0	0	12.35294118
Solution from [9]	6	2	0	6	2	0	7	0	0	11.3

Table 5 Some results

Solution	Priority category										Satisfactory category						Card				
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	
ACO-SPRI_1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
ACO-SPRI_2	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
ACO-SPRI_3	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1
Solution from [9]	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1
Solution	Satisfactory category										Acceptable category						Card				
	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	
ACO-SPRI_1	1	1	0	0	1	1	0	0	0	0	0	1	0	0	0	0	1	0	0	0	26
ACO-SPRI_2	1	1	0	0	0	1	0	1	0	0	1	0	0	0	0	0	0	1	1	0	27
ACO-SPRI_3	1	0	0	0	0	1	0	1	0	0	1	0	0	0	0	0	0	1	0	0	25
Solution from [9]	1	1	0	1	1	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	26

Table 6 Results in the objective space

Solution	N_1	$n_{sd,1}$	$n_{wd,1}$	N_2	$n_{sd,2}$	$n_{wd,2}$	N_3	$n_{sd,3}$	$n_{wd,3}$	P
ACO-SPRI_1	13	0	0	11	1	0	2	0	0	21.19230769
ACO-SPRI_2	12	1	0	10	1	0	5	0	0	19.48148148
ACO-SPRI_3	13	0	0	8	3	0	4	0	0	21.04
Solution from [9]	13	0	0	11	2	0	2	0	0	20.2309

(b) Second example: an instance with 40 projects

As shown above, Tables 5 and 6 display some non-dominated solutions fondness to the selection portfolio problem, Eq. 8, obtained by ACO-SPRI, in the project and objective space. The average computer time was 5 min 56 s. The last row in these tables shows the best solution obtained from [9]. The solution denoted by ACO-SPRI_1 is the one having the smallest number of discrepancies and the maximum number of projects supported in the priority category. If it is compared with the solution from [9], the projects 15 and 33 in ACO-SPRI_1 replace the pair (24, 34), which improves the power indicator and reduces the number of discrepancies.

ACO-SPRI_2 is the solution supporting the highest number of projects. If it is compared with ACO-SPRI_1, we can see that the projects in ACO-SPRI_2 (28, 30, 39) replace the pair (1, 25). Unless the DM has a very marked fondness for supporting the total amount of projects, (s)he should not accept the best project from the portfolio is excluded to make room for other much inferior to it. Such decision produces discrepancies in priority category and clear reduction of P indicator.

6 Conclusions

Multi-objective combinatorial optimization plays a decisive role in the decision-making process on the strategic management level. Our paper introduces a new discrepancies model and an ant colony algorithm with a local search multi-objective, which in conjunction shows as a solid method to provide an approximation of Pareto Frontier. As shown in the experimentation, the quality of the results reflects the preferences of the DM. Compared with previous approaches, the main features of the new method are:

- (a) An improved multi-criteria description of the portfolio quality, which takes into account the number and level of priority of the supported projects, discrepancies with respect to the portfolio derived from the rank ordering, and an indicator of the average rank of the supported projects;
- (b) An ant colony multi-objective meta-heuristic to solve the optimization problem and find the best compromise.

As a consequence, our proposal outperforms previous approaches concerning the quality of solutions. Although the experimental evidence is still limited, it seems that the advantage of our results increases with the complexity of the problem.

References

1. Bertolini, M., Braglia, M., Carmignani, G.: Application of the AHP methodology in making a proposal for a public work contract. *Int. J. Project Manage.* **24**(5), 422–430 (2006)
2. Brans, J.P., Mareschal, B.: PROMETHEE methods. In: Figueira, J., Greco, S., Erghott, M. (eds.) *Multiple Criteria Decision Analysis: State of the Art Surveys*, pp. 163–190. Springer Science+Business Media, New York (2005)
3. Cooper, R., Edgett, S., Kleinschmidt, E.: Portfolio management for new product development: results of an industry practices study. *R&D Manag.* **31**(4), 361–380 (2001)
4. Cruz, L., Fernandez, E., Gomez, C., Rivera, G., Perez, F.: Many-objective portfolio optimization of interdependent projects with ‘a priori’ incorporation of decision-maker preferences. *Appl. Math.* **8**(4), 1517–1531 (2014)
5. Dey, P.K.: Integrated project evaluation and selection using multiple-attribute decision-making. *Int. J. Prod. Econ.* **103**(1), 90–103 (2006)
6. Doerner, K., Gutjahr, W.J., Hartl, R.F., Strauss, C., Stummer, C.: Pareto ant colony optimization: a metaheuristic approach to multiobjective portfolio selection. *Ann. Oper. Res.* **131**(1–4), 79–99 (2004)
7. Dorigo, M., Gambardella, L.M.: Ant colony system: a cooperative learning approach to the traveling salesman problem. *IEEE Trans. Evolut. Comput.* **1**(1), 53–66 (1997)
8. Duarte, B., Reis, A.: Developing a projects evaluation system based on multiple attribute value theory. *Comput. Oper. Res.* **33**(5), 1488–1504 (2006)
9. Fernandez, E., Felix, L.F., Mazcorro, G.: Multi-objective optimisation of an outranking model for public resources allocation on competing projects. *Int. J. Oper. Res.* **5**(2), 190–210 (2009)
10. Fernandez, E., Olmedo, R.: Public project portfolio optimization under a participatory paradigm. *Appl. Comput. Intell. Soft Comput.* **2013**, Article ID 891781, 13 p. doi:[10.1155/2013/891781](https://doi.org/10.1155/2013/891781) (2013)
11. Gabriel, S., Kumar, S., Ordoñez, J., Nasserian, A.: A multiobjective optimization model for project selection with probabilistic consideration. *Socio-Econ. Plann. Sci.* **40**(4), 297–313 (2006)
12. Gómez Santillán, C., Cruz Reyes, L., Meza Conde, E., Schaeffer, E., Castilla Valdez, G.: A self-adaptive ant colony system for semantic query routing problem in P2P networks. *Computación y Sistemas (CyS)*, **13**(4) (2010)
13. Greco, S., Matarazzo, B., Slowinski, R.: Rough sets theory for multicriteria decision analysis. *Eur. J. Oper. Res.* **129**(1), 1–47 (2001)
14. Halouani, N., Chabchoub, H., Martel, J.M.: PROMETHEE-MD-2T method for project selection. *Eur. J. Oper. Res.* **195**(3), 841–849 (2009)
15. Kaplan, P., Ranjithan, S.R.: A new MCDM approach to solve public sector planning problems. In: Proceedings of the 2007 IEEE Symposium on Computational Intelligence in Multi Criteria Decision Making, pp. 153–159 (2007)
16. Khalili-Damghani, K., Sadi-Nezhad, S.: A hybrid fuzzy multiple criteria group decision making approach for sustainable project selection. *Appl. Soft Comput.* **13**(1), 339–352 (2013)
17. Liesio, J., Mild, P., Salo, A.: Preference programming for robust portfolio modeling and project selection. *Eur. J. Oper. Res.* **181**(3), 1488–1505 (2007)
18. Mavrotas, G., Diakoulaki, D., Caloghirou, Y.: Project prioritization under policy restrictions. A combination of MCDA with 0-1 programming. *Eur. J. Oper. Res.* **171**(1), 296–308 (2006)

19. Roy, B.: The outranking approach and the foundations of ELECTRE methods. In: Bana e Costa, C.A. (ed.) *Reading in Multiple Criteria Decision Aid*, pp. 155–183. Springer, Berlin (1990)
20. Saaty, T.L.: The analytic hierarchy and analytic network processes for the measurement of intangible criteria and for decision-making. In: Figueira, J., Greco, S., Erghott, M. (eds.) *Multiple Criteria Decision Analysis: State of the Art Surveys*, pp. 345–407. Springer Science +Business Media, New York (2005)
21. Sugrue, P., Mehrotra, A., Orehovec, P.M.: Financial aid management: an optimisation approach. *Int. J. Oper. Res.* 1, 267–282 (2006)
22. Yang, Y., Yang, S., Ma, Y.: A literature review on decision making approaches for research and development project portfolio selection. In: CSAMSE Conference (2012)

Part VI

Optimization: Theory and Applications

A Comparison Between Memetic Algorithm and Seeded Genetic Algorithm for Multi-objective Independent Task Scheduling on Heterogeneous Machines

Héctor Joaquín Fraire Huacuja, Alejandro Santiago,
Johnatan E. Pecero, Bernabé Dorronsoro, Pascal Bouvry,
José Carlos Soto Monterrubio, Juan Javier Gonzalez Barbosa
and Claudia Gómez Santillan

Abstract This chapter is focused on the problem of scheduling independent tasks on heterogeneous machines. The main contributions of our work are the following: a linear programming model to compute energy consumption for the execution of independent tasks on heterogeneous clusters, a constructive heuristic based on local search, and a new benchmark set. To assess our approach we compare the performance of two solution methods: a memetic algorithm, based on population search and local search, and a seeded genetic algorithm, based on NSGA-II.

H.J.F. Huacuja (✉) · A. Santiago · J.C.S. Monterrubio · J.J.G. Barbosa · C.G. Santillan
Tecnológico Nacional de Mexico, Instituto Tecnológico de Ciudad Madero, Ciudad Madero,
Mexico
e-mail: automatas2002@yahoo.com.mx

A. Santiago
e-mail: alx.santiago@gmail.com

J.C.S. Monterrubio
e-mail: soto190@gmail.com

J.J.G. Barbosa
e-mail: jjgonzalezbarbosa@hotmail.com

C.G. Santillan
e-mail: cggs71@hotmail.com

J.E. Pecero · B. Dorronsoro · P. Bouvry
University of Luxembourg, Luxembourg, Luxembourg
e-mail: johnatan.pecero@uni.lu

B. Dorronsoro
e-mail: bernabe.dorronsoro@uni.lu

P. Bouvry
e-mail: pascal.bouvry@uni.lu

A Wilcoxon rank-sum test shows significant differences in the diversity of solutions found but not in hypervolume. The memetic algorithm gets the best diversity for a bigger instance set from the state of the art.

1 Introduction

High Performance Computing (HPC) requires high amount of power [1], HPC systems are fundamental in a wide variety of scenarios as Wall Street or NASA [2]. HPC systems also known as super computers are formed by a set of machines interconnected over a bus or network platform. With more complex applications these systems are growing in the number of cores (machines). The National University of Defense Technology (NUDT) in China has a HPC system with 3,120,000 cores with a performance of 33.9 Pflops and energy consumption of 17.9 MW [3]. Although more cores provide more computing power, this has implications in energy consumption. The more energy used to compute applications the economic costs and heat problems increase.

This chapter is focused in the problem of scheduling independent tasks on heterogeneous machines. Lower voltage configurations in machines, a technology known as Dynamic Voltage and Frequency Scaling already implemented in most of nowadays computing machines, allow lower energy consumption at expense of computing times, therefore a trade-off between maximum completion time (make span) and energy consumption has to be found. As objectives are in conflict we deal with a multi-objective (MO) problem in which more than one optimal solution exists, known as Pareto optimal solutions. We propose a MO Memetic algorithm and a Seeded MO algorithm based on NSGA-II [4] to address the mentioned problem; the memetic algorithms are based on population search and local search [5], while seeded genetic algorithms are based on genetic algorithms beginning with quality individuals [6–9].

To the best of our knowledge there is not a set of instances with optimal values for the multi-objective problem treated. The compilation of instances used for scheduling with precedence constraints in [10], is used as a benchmark. The optimal values of makespan are obtained with the linear programming model in [11], based on it we developed a linear programming model to compute energy. Finally, a comparison for bigger instances is performed. The main contributions of our work are the following:

- A linear programming model to compute energy consumption for tasks without precedence constraints in heterogeneous clusters.
- A constructive heuristic based on Local Search (Constructive Local Search).

The rest of this chapter is organized as follows. Section 2 contains the basic concepts in multi-objective optimization. The description of the scheduling problem is treated in Sect. 3. Section 4 details the linear programming model proposed to

compute energy. The values found by the linear programming models of makespan and energy are in Sect. 5. The Sect. 6 describes the structure of the Memetic and Seeded MO algorithm implemented. The experimental setup is detailed in Sect. 7. A comparison between the proposed algorithms with the benchmark values and bigger instances is shown in Sect. 8. Finally, Sect. 9 presents our conclusions and lines of future research.

2 Background Concepts

This section presents the basic concepts used in multi-objective optimization based on Pareto optimality.

Definition 1 A Multi-objective optimization problem (MOP).

Given a vector function $\vec{f}(\vec{x}) = [f_1(\vec{x}), f_2(\vec{x}), \dots, f_k(\vec{x})]$ and its feasible solution space Ω , the MOP consists in finding a vector $\vec{x} \in \Omega$ that optimizes the vector function $\vec{f}(\vec{x})$. Without loss of generality we will assume only minimization functions.

Definition 2 Pareto dominance.

A vector \vec{x} dominates \vec{x}' (denoted by $\vec{x} \prec \vec{x}'$) if $f_i(\vec{x}) \leq f_i(\vec{x}')$ for all i functions in \vec{f} and there is at least one i such that $f_i(\vec{x}) < f_i(\vec{x}')$.

Definition 3 Pareto optimal.

A vector \vec{x}^* is Pareto optimal if not exist $\vec{x}' \in \Omega$ such that $\vec{x}' \prec \vec{x}^*$.

Definition 4 Pareto optimal set.

Given a MOP, the Pareto optimal set is defined as $P^* = \{\vec{x}^* \in \Omega\}$.

Definition 5 Pareto front.

Given a MOP and its Pareto optimal set P^* , the Pareto front is defined as $PF^* = \left\{ \vec{f}(\vec{x}) \mid \vec{x} \in P^* \right\}$.

3 Problem Description

Our case of study occurs in HPC clusters, where machines are used to compute big loads of tasks. The following conditions are assumed: the machines are DVFS capable [12], the machines are heterogeneous, only tasks without precedence constraints are received (i.e., independent tasks), the performance of a machine is not affected by any other machine (i.e., unrelated machines). Different voltage configurations in machines produce different execution times.

Given a set of tasks $T = \{t_1, t_2, \dots, t_n\}$, a set of heterogeneous machines $M = \{m_1, m_2, \dots, m_k\}$, the execution times of every task in every machine

$P = \{p_{1,1}, p_{1,2}, \dots, p_{n,k}\}$. The minimum makespan produced by the assignation of machine/task that minimizes:

$$\text{MAX}_{j=1}^k \left(\sum P_{ij} \quad \forall t_i \in m_j \right) \quad (1)$$

Since machines are DVFS capable and heterogeneous, different voltage levels exist for each machine m_j with a relative speed associated. When the highest voltage is selected *speed* is equal to 1 (the normal execution time in P), with a lower voltage selected the relative speed decrease for example when *speed* equal to 0.5 (50 % of the normal speed), there is a relative execution time P'_{ij} calculated as follows.

$$P'_{ij} = \frac{P_{ij}}{\text{speed}} \quad (2)$$

Then given a set of tasks T , a set of heterogeneous machines M and a set of voltages $V_j = \{v_{i,1}, v_{i,2}, \dots, v_{i,l}\} \forall m_j \in M$ of different l sizes. The minimum energy is produced by the assignation of machine/voltage/task that minimizes:

$$\sum_{j=1}^k \left(\sum V_{jp}^2 p'_{ij} \quad \forall t_i \in m_j \right) \quad (3)$$

Where p is the index of the selected voltage in V_j , since DVFS machines capabilities are being used in the energy consumption, the objective function for makespan is modified as follows.

$$\text{MAX}_{j=1}^k \left(\sum P'_{ij} \quad \forall t_i \in m_j \right) \quad (4)$$

The MOP studied in this work consists in finding the assignation of machine/voltage/task that minimizes (3) and (4).

4 Two Phases Mixed Integer Linear Programming Model

This section presents a Mixed Integer Linear Programming model (MILP) of two phases. The first phase computes the optimal makespan with the model in [11]. The second phase computes the energy with the optimal makespan as constraint, although this no guarantees the optimal values for energy.

4.1 Energy Model

The proposed model uses a binary variable $x_{i,j,p}$, which takes the value of 1 if the task i is assigned to the machine j with the voltage configuration p . The y variable represents the current makespan in the model. The constants $V_{j,p}$ and $s_{j,p}$ represents the voltage and the speed used by the machine j in the configuration p . Finally the constant *makespan* represents the optimal makespan from first phase. The formulation of the energy model is the following:

$$\min \sum_{j=1}^k \sum_{i=1}^n \sum_{p=1}^l \frac{P_{ij}}{s_{jp}} V_{j,p}^2 x_{i,j,p} \quad (5)$$

Subject to

$$\sum_{j=1}^k \sum_{p=1}^l x_{i,j,p} = 1, \quad i = 1, \dots, n. \quad (6)$$

$$\sum_{i=1}^n \sum_{p=1}^l \frac{P_{ij}}{s_{jp}} x_{i,j,p} \leq y, \quad j = 1, \dots, k. \quad (7)$$

$$y \leq \text{makespan}. \quad (8)$$

$$x_{i,j} \in \{0, 1\} \quad (9)$$

$$y \in \{\mathbb{R}^+\} \quad (10)$$

where (5) defines the objective function to minimize (the energy). The constraint (6) assigns only one task i to one machine j with one configuration p . The constraint (7) represents the upper bound of the makespan. The constraint (8) establishes the *makespan* obtained by the phase one as a limit in the phase two. Constraints (9) and (10) establish the range of variables.

5 Values Found by the Two Phases

The proposed MILP model of two phases performs an exhaustive search, because of that a set of small instances is required, we use the benchmark set from [10] removing the precedence constraint. The voltage configurations are taken from [13] and are configured using the round robin principle; the first machine gets the first configuration, when no more configurations left the next machine gets the first configuration and so on. The name of the instances is split with underline as author_machines_tasks_number. The following table shows the obtained values in every instance (M for makespan, E for energy).

6 Memetic and Seeded Algorithms

Memetic algorithms are based on a random initial population search with Local Search and Seeded algorithms are based on initial heuristic methods to speed up the search [6]. In consideration to study the effect of local improvements versus seeded solutions, both implementations use NSGA-II [4] as a basis.

In Genetic algorithms, the vector of decision variables is named chromosome. The oldest concepts of genetic algorithms map the set of decision variables (phenotype) into a binary string (genotype). Nowadays the chromosome is represented by a particular data structure. The chromosome representation in Fig. 1 sets a machine and voltage configuration for every task in T and is the one used in this work.

6.1 Random Local Search

Local search methods perform improvements over a current solution using the concept of neighborhood. By definition, the neighbor solutions are one step from the current solution and one step is defined as one particular movement (neighborhood operator) like *insertion* or *swap* [13–15]. The neighborhood operator used in this work is the *boundary* operator, where a gene of the chromosome is randomly set between the upper and lower bound feasible. The next algorithm in Fig. 2 shows the general framework for Local Search.

The implementations consider a gene is compound of two values, machine and the voltage configuration. Then a neighbor of the current solution is formed by randomly select one task and changes it to a random machine with a random feasible voltage configuration.

6.2 Constructive Local Search

Some objective functions can be evaluated partially; these objective functions can be used to construct solutions from scratch. The HEFT method in [16] is a fast heuristic used in scheduling that uses this approach, adding decision variables to

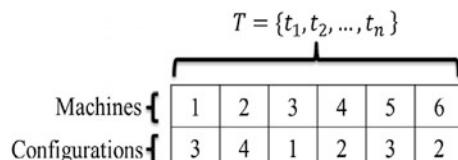


Fig. 1 Chromosome representation

Fig. 2 Local search

Purpose: Locally improve a solution Data structures: \vec{x} : A vector of decision variables Functions: $neighbor(\vec{x})$: Returns a neighbor solution \vec{x}' from \vec{x} Input: A solution to improve \vec{x} Output: The improved solution \vec{x}

```

for  $i = 1$  to  $MAXSTEPS$  do
     $\vec{x}' \leftarrow neighbor(\vec{x})$ 
    if  $\vec{x}' \prec \vec{x}$  then
         $\vec{x} \leftarrow \vec{x}'$ 
    end if
end for

```

construct a quality solution. Not only fast heuristic methods are developed with this approach, the exact method known as Branch and Bound (B&B) [17] is possible because objective functions are partially evaluated and incremental with every decision variable. The disadvantage of fast heuristic methods or B&B is that the final solution is always deterministic. Instead, we propose a more general approach based on Random Local Search [18], using stochastic solutions is possible to achieve different regions of the Pareto front. We named this approach Constructive Local Search (CLS).

Purpose: Construct a quality solution Data structures: \vec{x} : A vector of decision variables Functions: $PartialMakespan(\vec{x}, i)$: Evaluate makespan objective function until ith task $PartialEnergy(\vec{x}, i)$: Evaluate energy objective function until ith task $neighbor(\vec{x})$: Returns a neighbor solution \vec{x}' from \vec{x} Input: An empty vector solution \vec{x} Output: A constructed solution \vec{x}

```

for  $i = 1$  to  $n$  do
     $steps \leftarrow 0$ 
     $\vec{x} \leftarrow \vec{x} + x_i$ 
     $\vec{x}.makespan \leftarrow PartialMakespan(\vec{x}, i)$ 
     $\vec{x}.energy \leftarrow PartialEnergy(\vec{x}, i)$ 
    while  $steps < MAXSTEPS$  do
         $\vec{x}' \leftarrow neighbor(\vec{x})$ 
         $\vec{x}'.makespan \leftarrow PartialMakespan(\vec{x}', i)$ 
         $\vec{x}'.energy \leftarrow PartialEnergy(\vec{x}', i)$ 
        if  $\vec{x}' \prec \vec{x}$  then
             $\vec{x} \leftarrow \vec{x}'$ 
        end if
         $steps += 1$ 
    end while
end for

```

Fig. 3 Constructive local search

Purpose: Find the best set of non-dominated solutions
 Data structures:
 P_t List of parent solutions
 Q_t List of offspring solutions
 R_t List of temporal solutions
 Functions:
Constructive_Local_Search: Returns a set of solutions constructed using CLS
Sort(R_t, \geq_n): Sort solutions R_t using the crowded comparator \geq_n
make_new_pop(P_t): Use selection, crossover and mutation over P_t to create a new population
 Input: Seeded solutions
 Output: A set of non-dominated solutions

```

 $R_t \leftarrow \text{Constructive\_Local\_Search}()$ 
 $Q_t \leftarrow \text{Constructive\_Local\_Search}()$ 
repeat
   $R_t \leftarrow P_t \cup Q_t$ 
  Sort( $R_t, \geq_n$ )
   $P_{t+1} \leftarrow R_t[0 : N]$ 
   $Q_{t+1} \leftarrow \text{make\_new\_pop}(P_{t+1})$ 
   $t++$ 
until  $t = \text{MAXGENERATIONS}$ 
```

Fig. 4 Seeded algorithm

The main idea of CLS in Fig. 3 is increment a single decision variable at once; with every new decision variable at the current decision vector a Random Local Search is performed, until the maximum number of search steps MAXSTEPS is reached.

6.3 Seeded Genetic Algorithm and Memetic Algorithm

There are two differences between the Memetic and Seeded algorithms implemented. The Memetic algorithm is initialized with initial random solutions, while the Seeded genetic algorithm is initialized with solutions constructed with CLS. The second difference is the Seeded algorithm does not use local improvements. The differences are shown in the algorithms from Figs. 4 and 5; for details of NSGA-II see [4].

7 Experimentation

In the experimental executions both algorithms proposed are configured with the same genetic operators: single point crossover and 10 % of the solution is mutated with the boundary operator, the crossover probability is 100 %. The assessment of the algorithms was measured in two manners; first the set of instances in Sect. 5 is used to evaluate how far both algorithms are from the optimal makespan and best energy found. Secondly we used a set of bigger instances with 512 tasks and 16 machines from [13]. The voltage configurations in the machines are the ones in [12] using the round robin principle. 50 independent runs were performed for every

```

Purpose: Find the best set of non-dominated solutions
Data structures:
 $P_t$  List of parent solutions
 $Q_t$  List of offspring solutions
 $R_t$  List of temporal solutions
Functions:
 $Sort(R_t, \geq_n)$ : Sort solutions  $R_t$  using the crowded comparator  $\geq_n$ 
 $make\_new\_pop(P_t)$ : Use selection, crossover and mutation over  $P_t$  to create a new population
 $Local\_Search(Q_t)$ : Improve solutions in  $Q_t$  using Local Search
Input: Random solutions
Output: A set of non-dominated solutions

```

```

 $R_t \leftarrow$  Random solutions
 $Q_t \leftarrow$  Random solutions
repeat
   $R_t \leftarrow P_t \cup Q_t$ 
   $Sort(R_t, \geq_n)$ 
   $P_{t+1} \leftarrow R_t[0 : N]$ 
   $Q_{t+1} \leftarrow make\_new\_pop(P_{t+1})$ 
   $Local\_Search(Q_{t+1})$ 
   $t++$ 
until  $t = MAXGENERATIONS$ 

```

Fig. 5 Memetic algorithm

instance in both sets with a 200 population, 300 generations and the MAXSTEPS parameter is set to 100 for the CLS and Random Local Search. The Multi-objective quality indicators of hypervolume [19] (HV), generational distance [20] (GD) and generalized spread [21] (GS) are computed for both instances sets.

8 Results

From the 50 independent runs for every instance in Sect. 5 the best value found in makespan and energy is saved, the error generated with the values from Table 1 .

As shown in Table 2 both implementations produce about 20 % of error from the optimal makespan and Memetic algorithm produce the best result. On the other hand, SGA produce the best energy. Is normal that energy be an easier objective because is not the optimal value in energy.

Table 1 Values obtained by the MILP of two phases

Instance	M	E	Instance	M	E
Ahmad_3_9_28	13	79.493	Kang_3_10_84	36	202.629
Hsu_3_10_84	36	218.643	Kuan_3_10_28	11	66.561
Eswari_2_11_61	55.5	186.300	Liang_3_10_80	36	202.629
Hamid_3_10	58	341.789	Daoud_2_11_64	55.5	186.300
Heteropar_4_12_124	35	280.843	YCLee_3_8_80	29	161.836
Ilavarasan_3_10_77	36	202.629	Sample_3_8_100	31	165.471
Kang_3_10_76	36	202.629	SampleFig_3_8_89	29	161.836

Table 2 Error percentage to makespan optimal and energy found

	SGA (%)	Memetic (%)
Makespan	20.564	20.205
Energy	0.000	0.056

From the Brauntset [13], the MO quality indicators were computed in every of the 50 independent runs and the average values are shown in Table 3. The hypervolume reference point is selected from the worst value found for every instance plus one unit and the reference front used was constructed with the non-dominated solutions of the 50 executions. Results are shown in the following table.

From the above results, it is clear that Memetic algorithm outperforms the seeded algorithm (bigger values better for hypervolume, lower values better for the others quality indicators). A similar behavior is presented from the set in Sect. 5, but the best generalized spread is produced by the Seeded algorithm, the summarize results are shown in Table 4.

Table 3 Experiments results hypervolume (HV), generational distance (GD) and generalized spread (GS)

Instance	HV		GD		GS	
	SGA	Memetic	SGA	Memetic	SGA	Memetic
u_c_hihi	2.1E+11	2.3E+11	0.0721	0.0389	0.9219	0.9132
u_c_hilo	937.3E+11	1.1E+11	0.0159	0.0128	0.9012	0.6938
u_c_lohi	2.0E+11	2.2E+11	0.0693	0.0457	0.8981	0.9126
u_c_lolo	1.1E+11	1.3E+11	0.0154	0.0128	0.8887	0.6874
u_i_hihi	71.3E+11	55.3E+11	0.0049	0.0115	0.9916	0.9124
u_i_hilo	31.3E+11	29.3E+11	0.0100	0.0175	0.9683	0.8454
u_i_lohi	95.2E+11	65.3E+11	0.0056	0.0121	0.9384	0.9054
u_i_lolo	37.7E+11	34.2E+11	0.0108	0.0144	0.9514	0.8582
u_s_hihi	252.9E+11	183.2E+11	0.0182	0.0256	0.8937	0.9338
u_s_hilo	86.0E+11	74.6E+11	0.0125	0.0113	0.9393	0.8769
u_s_lohi	262.6E+11	194.9E+11	0.0177	0.0224	0.9119	0.9384
u_s_lolo	123.5E+11	120.2E+11	0.0159	0.0154	0.9340	0.8662
Total	2.39E+15	2.58E+15	0.2683	0.2404	11.1384	10.3438

Table 4 Summarize results from both instance set

	Braunt		Section 5	
QI	SGA	Memetic	SGA	Memetic
HV	2.39E+15	2.58E+15	1.71E+05	1.73E+05
GD	0.2683	0.2404	0.1096	0.0590
GS	11.1384	10.3438	11.4416	16.4848

Table 5 p -values computed by Wilcoxon test

Instance set	HV	GD	GS
Section 5	0.662385	0.004072	1.14E-05
Braunt	0.887386	0.629726	0.020489

The non-parametric statistical test of Wilcoxon rank sum test [22] is performed to evaluate if the differences in the performance of the algorithms is statistical significant. The p -values computed by the Wilcoxon test are show in Table 5.

p -value equal or lower to 0.5 are accepted with a 95 % level of confidence, from Sect. 5 instances no significant hypervolume difference were produce, only convergence (generational distance) and diversity (generalized spread). When the problem dimension increases, as in the Braunt set, only generalized spread difference is significant. The explanation for this is the crowding distance in the NSGA-II manages the diversity, while Local Search only near the solutions to the real Pareto Front. This result explains why Memetic algorithm gets a better global performance.

9 Conclusions and Future Work

This chapter provides a small benchmark for energy-aware scheduling independent tasks in heterogeneous machines, which is feasible to compute in time with the linear programming model in [11] and the two-phase model proposed.

The experimental results show when instance dimension increases the diversity is the only significant improvement of the Memetic algorithm over the Seeded algorithm and for the smaller set the Seeded algorithm produces the best diversity. The difference in diversity is explained by the parameter MAXSTEP. When the number of tasks in T increases, the initial seeded solutions came from a more intensive search, thus less diverse. Otherwise with few tasks in T , the initial solutions came from a less intensive search, so more diverse. Then parameter configuration would impact directly in the final performance.

The concept of Constructive Local Search was introduced, which is a novel approach and a lot of research can be done, for example in neighborhoods swap, insertion, reverse and variable neighborhood search. CLS is easily hybridizable with population searches as genetic algorithms, memetic algorithms and particle swarm. Also is easily hybridizable with GRASP in [23, 24]; CLS could be used to speed up several optimization methods. The use of stochastic search and different neighborhoods in CLS allows explore different regions of the search space and not only deterministic initial solutions as in common constructive methods. Is possible to apply this concept not only in multi-objective optimization, it could be implemented in single objective optimization as well. For the instance set used in this work please contact the authors, also it will be available in some authors' personal web.

Acknowledgments A. Santiago would like to thank CONACyT Mexico, for the support no. 360199.

References

1. Feng, W.-C.: The importance of being low power in high performance computing. *CTWatch Q.* **1**(3), 11–20 (2005)
2. Center, J.W.I.R., Feitelson, D.: A survey of scheduling in multiprogrammed parallel systems. Research report, IBM T.J. Watson Research Center (1994)
3. TOP500.org.: The 43rd top 500 list published during isc14 in Leipzig, germany (2014)
4. Deb, K., Agrawal, S., Pratap, A., Meyarivan, T.: A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: Nsga-II. In: Proceedings of the 6th International Conference on Parallel Problem Solving from Nature, vol. 1917 (2000)
5. Moscato, P., Cotta, C.: A modern introduction to memetic algorithms. In: Gendreau, M., Potvin, J.-Y. (eds.) *Handbook of Metaheuristics*. International Series in Operations Research & Management Science, vol. 146, pp. 141–183. Springer, US (2010)
6. Meadows, B., Riddle, P., Skinner, C., Barley, M.: Evaluating the seeding genetic algorithm. In: Cranefield, S., Nayak, A. (eds.) *AI 2013: Advances in Artificial Intelligence*. Lecture Notes in Computer Science, vol. 8272, pp. 221–227. Springer International Publishing, Berlin 2013
7. Julstrom, B.A.: Seeding the population: improved performance in a genetic algorithm for the rectilinear steiner problem. In: *Proceedings of the 1994 ACM Symposium on Applied Computing*, SAC '94. New York, NY, USA, pp. 222–226. ACM, New York (1994)
8. Ponterosso, P., Fox, D.S.J.: Heuristically seeded genetic algorithms applied to truss optimisation. *Eng. Comput.* **15**(4), 345–355 (1999)
9. Oman, S., Cunningham, P.: Using case retrieval to seed genetic algorithms. *Int. J. Comput. Intell. Appl.* **1**, 71–82 (1997)
10. Pineda, A.A.S., Pecero, J.E., Huacuja, H.J.F., Barbosa, J.J.G., Bouvry, P.: An iterative local search algorithm for scheduling precedence-constrained applications on heterogeneous machines. In: *6th Multidisciplinary International Conference on Scheduling: Theory and Applications (MISTA 2013)*, pp. 472–485 (2013)
11. Mokotoff, E., Jimeno, J.L.: Heuristics based on partial enumeration for the unrelated parallel processor scheduling problem. In: *Annals of Operations Research*, pp. 133–150. Kluwer Academic Publishers, Dordrecht (2002)
12. Pecero, J.E., Bouvry, P., Barrios, C.J.: Low energy and high performance scheduling on scalable computing systems. In: *Latin-American Conference on High Performance Computing*, pp. 1–8 (2010)
13. Braunt, T.D., Siegel, H.J., Beck, N., Boloni, L.L., Maheswarans, M.: A comparison study of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems. *J. Parallel Distrib. Comput.* **22**, 810–837 (2001)
14. Schiavinotto, T., Sttzle, T.: The linear ordering problem: instances, search space analysis and algorithms. *J. Math. Model. Algorithms* **3**(4), 367–402 (2005)
15. Villanueva, D., H. Fraire Huacuja, A. Duarte, R. Pazos R., J. Carpio Valadez, and H. Puga Soberanes, “Improving iterated local search solution for the linear ordering problem with cumulative costs (lopcc),” in *Knowledge-Based and Intelligent Information and Engineering Systems*(R. Setchi, I. Jordanov, R. Howlett, and L. Jain, eds.), vol. 6277 of *Lecture Notes in Computer Science*, pp. 183–192, Springer Berlin Heidelberg, 2010
16. Topcuoglu, H., Hariri, S., Wu, M.-Y.: Performance-effective and low-complexity task scheduling for heterogeneous computing. *IEEE Trans. Parallel Distrib. Syst.* **13**, 260–274 (2002)
17. Lawler, E.L., Wood, D.E.: Branch-and-bound methods: a survey. *Oper. Res.* **14**(4), 699–719 (1966)

18. Villanueva, J., Huacuja, H., Rangel, R., Valadez, J., Soberanes, H., Barbosa, J.: Iterated local search algorithm for the linear ordering problem with cumulative costs (lopcc). In: Castillo, O., Kacprzyk, J., Pedrycz, W. (eds.) Soft Computing for Intelligent Control and Mobile Robotics. Studies in Computational Intelligence, vol. 318, pp. 395–404, Springer, Berlin (2011)
19. Auger, A., Bader, J., Brockhoff, D., Zitzler, E.: Theory of the hypervolume indicator: optimal μ -distributions and the choice of the reference point. In: Foundations of Genetic Algorithms (FOGA 2009). ACM, New York (2009)
20. Veldhuizen D.A.V., Lamont, G.B.: Evolutionary computation and convergence to a Pareto front. Stanford University, California, pp. 221–228, Morgan Kaufmann (1998)
21. Zhou, A., Jin, Y., Zhang, Q., Sendhoff, B., Tsang, E.: Combining model-based and genetics-based offspring generation for multi-objective optimization using a convergence criterion. In: Proceedings of the Congress on Evolutionary Computation. CEC, pp. 3234–3241. IEEE Press (2006)
22. Oja, H.: Nonparametric statistics with applications to science and engineering by Paul H. Kvam, Brani Vidakovic. Int. Stat. Rev. **76**(1), 150–151 (2008)
23. Feo, T., Resende, M.: Greedy randomized adaptive search procedures. J. Global Optim. **6**(2), 109–133 (1995)
24. Resende, M.: Greedy randomized adaptive search procedures greedy randomized adaptive search procedures. In: Floudas, C.A., Pardalos, P.M. (eds.) Encyclopedia of Optimization, pp. 1460–1469, Springer, US (2009)

Parallel Meta-heuristic Approaches to the Course Timetabling Problem

A. Jorge Soria-Alcaraz, Martin Carpio, Hector Puga, Jerry Swan,
Patricia Melin, Hugo Terashima and A. Marco Sotelo-Figueroa

Abstract The Course Timetabling problem is one of the most difficult combinatorial problems that arises with a University. The main objective of this problem is to obtain a timetable that minimises student conflicts between assigned activities. This is a discrete combinatorial problem that can be extremely difficult to solve for a human expert so computational heuristics are usually implemented in order to find good solutions within a reasonable time. With the advent of multi-core and hyper-threading technologies, parallel heuristics can speed up the solution process and with a proper parallel design these heuristics can improve the quality of solutions with the same number of Fitness evaluations than sequential algorithms. This paper explores the implementation of a parallel set of heuristic algorithms based on genetic algorithms, Scatter Search and discrete PSO for CTTP problem. Our experiments used as benchmark set instances from ITC2007 Track 2. Also the algorithms described in this paper make use of a layer of independence called methodology of design in order to be easily adaptable to new instances. Every parallel algorithm is compared against its sequential counterpart through speed-up metrics like Weak speed-up proposed by Alba et al.

A.J. Soria-Alcaraz (✉) · M. Carpio · H. Puga · A.M. Sotelo-Figueroa
Division de Estudios de Posgrado e Investigacion, Instituto Tecnologico de Leon, Leon
Guanajuato, Mexico
e-mail: Soajorgea@gmail.com

M. Carpio
e-mail: jmcarpio61@hotmail.com

H. Puga
e-mail: pugahector@yahoo.com

J. Swan
Department of Computer Science and Mathematics, University of Stirling, Stirling, UK

P. Melin
Tijuana Institute of Technology, Tijuana, B.C., Mexico

H. Terashima
Instituto Tecnologico y de Estudios Superiores de Monterrey,ITESM,
Monterrey, N.L., Mexico

1 Introduction

The timetabling problem is a common and recurring problem in many organizations. This paper focuses on the Course timetable problem (CTTP). The CTTP, commonly seen at every university, represents the assignment of a fixed number of subjects into a number of timeslots. The main objective of this problem is to obtain a timetable that minimizes the number of conflicts for a student [1]. There exist many possible conflict types in CTTP, but the principal conflicts are usually time-related i.e. one student with two or more subjects assigned to the same time-period.

Like most timetabling problems, the CTTP is known to be NP-Complete [2, 3]. Due to this complexity and the fact that course timetables are still often constructed by hand, it is necessary to automate timetable construction to improve upon the solutions reached by the human expert [4].

Unfortunately, automation of timetable construction is not an easy task, since it requires a deep knowledge of the problem itself as well as the particular characteristics of the instance to be solved. Such knowledge is in most cases unavailable to the typical researcher/end-user. This is the principal reason for the high popularity of metaheuristic solvers for the CTTP. However not all of the popular metaheuristics can provide a good solution in a reasonable time [5] which potentially requires a researcher to experiment with different heuristic approaches in order to build a successful timetabling solving system.

The interested researcher needs to find the most suitable metaheuristic for his/her own problem instance. Whilst there exists a large variety of possible metaheuristics that may be applied to the CTTP, not all of them offer a quality solution within a reasonable time. In some cases, the amount of time invested in obtaining solutions becomes another important restriction. In this Scenario the researcher needs to find a fast and robust algorithm to solve the problem.

Modern desktop PCs have several cores or processors, so it is desirable that to exploit this in order to obtain a parallel algorithm with corresponding speedup [6]. This paper explores adapts several parallel heuristic metaheuristics to the CTTP problem: Cellular Genetic Algorithms, Parallel scatter Search, Parallel Simulated Annealing, Parallel PSO and multi-start methods for Parallel Iterated Local Search. The anticipation is that timetabling researchers can use these algorithms in related problems. The main contribution of this paper is to inform use of parallel algorithms in combinatorial problems similar to the CTTP.

All of the parallel algorithms proposed in this paper utilize a generic representation named methodology of design [7]. This representation factors out commonality the solution of different CTTP instances via the identification of generic structure. This methodology is flexible in the sense that changes in the restrictions or policies of a specific instance (University) do not require the re-implementation of the entire solver. Rather, it suffices to frame this new instance in terms of the generic structure, allowing the same algorithm to be re-used in this new context. The objective of this methodology is to construct a single generic CTTP solver and then use it to solve different CTTP problem formulations.

The paper is organized as follows. Section 2 presents the problem definition and some important concepts, Sect. 3 presents the solution approach and its justification. Section 4 contains the experimental set-up, results, their analysis and discussion and gives the conclusions and describes some future work.

2 Important Concepts

In this Section, the generic representation (methodology of design) is briefly explained. We also describe each of the proposed parallel algorithms. The corresponding adaptations of these parallel algorithms to the generic methodology are also detailed in this section.

2.1 Problem Definition

The CTTP can be considered as a Constraint Satisfaction Problem (CSP) where the variables are events and the most common type of constraints are time-related. A clear and concise definition of the CTTP is given by Conant-Pablos et al. [8]: A set of events (courses or subjects) $E = \{e_1, e_2, \dots, e_n\}$ are the basic element of a CTTP. Also there are a set of time-periods $T = \{t_1, t_2, \dots, t_s\}$, a set of places (classrooms) $P = \{p_1, p_2, \dots, p_m\}$, and a set of agents (students registered in the courses) $A = \{a_1, a_2, \dots, a_o\}$. Each member $e \in E$ is a unique event that requires the assignment of a period of time $t \in T$, a place $p \in P$ and a set of students $S \subseteq A$, so that an assignment is a quadruple (e, t, p, S) . A timetabling solution is a complete set of n assignments, one for each event, which satisfies the set of hard constraints defined by each university or college. This problem is known to be at least NP-complete [2, 3].

The CTTP has been studied intensively, from early solution approaches based on graph heuristics [9], linear programming [10] and logic programming [11, 12] to metaheuristic schemes such as Tabu-list [13], Genetic Algorithms [14, 15], Ant Colony [16, 17], Variable Neighbourhood Search [18], Bee Algorithms [19], Simulated Annealing [20] and so on. In the same way, a great number of CSP solution schemes have been used to solve the CTTP problem [21, 22]. Also, in recent years Hyper-heuristic selection frameworks has been applied [23, 24] with encouraging results. Similar success has also been reported for constructive hyper-heuristic approaches [25]. Almost all of this research used two important benchmarks ITC2002 [26] and ITC2007 [27] in order to gather data about efficiency and solution quality. This paper takes parallel metaheuristics from the state of the art and applies them to ITC2007 instances. The methodology of design approach adopted [7] responds to the necessity of generic solution schemes [4], applicable to real world instances [28, 29]. The main objective of this effort is to yield state of the art metaheuristics that can be directly applicable to a wide range of real CTTP instances without losing time and resources in building instance-specific algorithms.

2.2 Methodology of Design for the Course Timetabling Problem

In the literature it can be seen that there exists a problem with the diversity of course timetabling instances due different university policies. This situation directly impacts in the reproducibility and comparison between course timetabling algorithms [2]. The state of art indicates some strategies to avoid this problem. For example, a more formal problem formulation [2] as well as the construction of benchmark instances [10]. These schemes are useful for a deeper understanding of the university timetabling complexity, but the portability and the reproducibility of a timetabling solver in another educational institution is still in discussion [30]. In this sense, we use a context-independent layer for the course timetabling resolution process. This new layer integrates timetabling constraints into three basic structures *MMA matrix*, *LPH list* and *LPA list*.

MMA: Matrix contains the number of students in conflict between subjects i.e. the number of conflicts if two subjects are assigned in the same timeslots. An example of this matrix can be seen in the Fig. 1.

LPH List: These structures have in its rows the subjects offered. In its columns have the offered timeslots, so this list gives information about the allowed timeslots per subject. one example of this list can be seen on Table 1.

LPA list: This list shows in its rows events and the classrooms available to be assigned to it without conflict. An example of this table can be seen on Table 2.

Fig. 1 MMA matrix

	ACM0403	SCB0421	SCE0418	ACH0408	ACM0401	ACM0404
ACM0403	4	1	1			
SCM0414	1	10	3	3		6
SCB0421	1	3	3	2		2
SCE0418		3	2	3		3
ACH0408						
ACM0401					4	1

Table 1 LPH list

	Day 1	Day 2
e_1	$\langle t_3 \rangle$	$\langle t_2 \rangle$
e_2	$\langle t_2 \rangle$	$\langle t_2 \text{ or } t_1 \rangle$

Table 2 LPA list

Events	Classrooms
e_1	$\langle P_1, P_2 \rangle$
e_2	$\langle Lab_1, Lab_3 \rangle$
e_3	$\langle P_3, Lab_2 \rangle$
e_4	$\langle Lab_4 \rangle$
\vdots	\vdots
e_n	$\langle P_2 \rangle$

These structures are constructed with the natural/original inputs of the CTTP problem. This process ensures *by design* the non-existence of violations in the selection of any values shown in LPH and LPA. So the main problem is how to deal with students conflicts. A proposal is to work with these conflicts by means of the next minimization function:

$$\text{Min}(FA) = \sum_{i=1}^k FA_{V_i} \quad (1)$$

$$FA_{V_j} = \sum_{s=1}^{(M_{V_j})-1} \sum_{l=1}^{M_{V_i}-s} (A_{j,s} \wedge A_{j,s+l}) \quad (2)$$

where FA = Student conflicts of current timetabling. V_i = Student conflicts from “Vector” i of the current Timetabling. $A_{j,s} \wedge A_{j,s+l}$ = students that simultaneously demand subjects s and $s + 1$ inside the “Vector” j . A = student that demands subject s in a timetabling j .

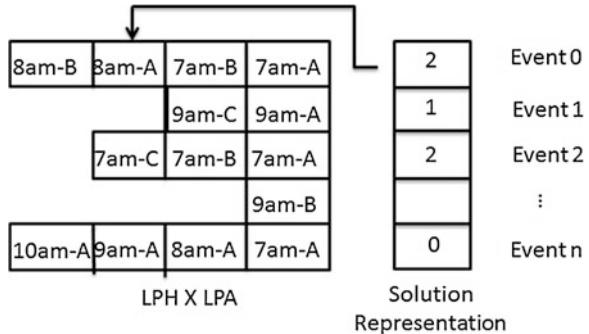
The concept of vector is the most important element in the design methodology. This vector is a binary representation of an event [31, 30]. It can be constructed as seen on Table 3 where each v_i is a vector that represents event e_i .

The vectors can be easily added and subtracted to construct sets. the symbols used for these sets of vectors are $V1, V2 \dots V_i$. One characteristic is that the number of vectors sets is related with the number of timeslots offered by the current timetabling. The main idea about vectors is to have a space where the events can be move without assigned them to a fixed timeslot. This independent layer of context generalizes in the solution process of the CTTP problem.

Having obtained these structures from original inputs of our CTTP problem, we ensure *by design* the non-existence of violations by the selection of any values shown in LPH and LPA, i.e. an important characteristic of this methodology is that each element of their Cartesian product represent a feasible classroom-timeslot pair (i.e. that selection can be applied to the reality). The solution representation used by a metaheuristic can therefore be an integer array with length equal to the number of variables (events/subjects) with elements indexing each pair (LPH_i, LPA_i) , where LPH_i is a valid temporal value for the event i and LPA_i is a valid space value for the event i . This array constructs a complete timetable assignment. The objective is then to search in this Cartesian product space $LPH_i \times LPA_i$, looking for minimum student conflicts by Eqs. (1) and (2) and MMA matrix. One example of this representation can be seen on Fig. 2.

Table 3 Cell codification

	e_1	e_2	...	e_l
Ind ₁	V_B	V_D	...	V_B
Ind ₂	V_A	V_B	...	V_C
⋮	⋮	⋮	⋮	⋮
Ind _n	V_D	V_A	...	V_C

Fig. 2 Representation used

2.3 Parallel Computing and Cellular Genetic Algorithms

The main objective of parallel computing is to execute code at the same time on different processors i.e. in the simplest sense, parallel computing is the simultaneous use of multiple compute resources to solve a computational problem for example: to be run using multiple CPUs, to solve a problem broken into discrete parts that can be executed at the same time and instructions from an algorithm executed simultaneously on different CPUs [32]. Concurrency is another form of parallelism where two or more tasks can start, run, and complete in overlapping time periods. It doesn't necessarily mean they'll ever both be running at the same instant. Eg. multitasking on a single-threaded machine. It is suggested to run the algorithms described in this paper in a true parallel machine (one task-one processor approach) but multi-threading technology is also a valid option.

One of the advantages of a true parallel approach is the Super linear Speedup concept. Speedup of a parallel computation is defined as $Sp = T/T_p$ [33], where T is the sequential time of a problem and T_p is the parallel time to solve the same problem using p processors. T_p was argued to be no greater than P in [34]. However, in practice, people observed “super linear speedup” i.e. the speedup with P processors is greater than P . Two main reasons for superlinear speedup are shown in [6]. The first reason is that in search problems, the termination time can be reduced when several searches are executed at the same time. And the second one is because of more efficient utilization of resources by multiprocessors.

As a metaheuristic, we use Genetic Algorithms. A Genetic Algorithm (GA) is a search heuristic that mimics the process of natural evolution [35]. This heuristic is routinely used to generate useful solutions to optimization and search problems. Genetic algorithms belong to the larger class of evolutionary algorithms (EA), which generate solutions to optimization problems using techniques inspired by natural evolution, such as inheritance, mutation, selection, and crossover [35].

In this article, we discuss our implementation of a Cellular Genetic Algorithm (cGA). The cGA was initially designed for massively parallel machines, composed of many Single-Instruction Multiple-Data (SIMD) processors, i.e. executing simultaneously the same instructions on different data [6]. The first cGA model

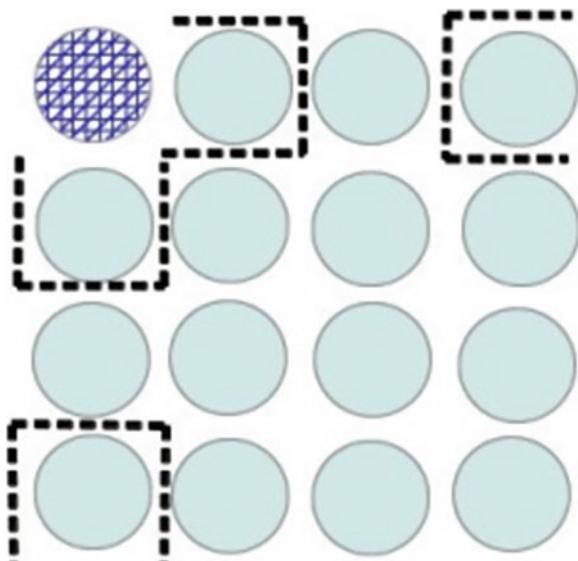
know was proposed by Robertson in [36] and implemented on a CM1 computer. In a cGA, all steps of the GA algorithm (selection, replacement, recombination and mutation) are executed in parallel [6]. This approach has shown great execution speed as well as better fitness performance when compared to sequential or canonical GA. The term fitness performance is used to refer a better fitness value obtained by the parallel algorithm against its sequential counterpart after the end of a fixed number of fitness evaluations.

Although cGA is still associated with massively parallel machines, modern technologies such as Java Threads or CUDA cores mean that a massively parallel computers are no longer required to run a parallel algorithm like cGA.

The cGA model simulates the natural evolution from the point of view of the individual. The essential idea of this model is to provide the population of a spatial structure defined as a connected graph, where each vertex is a common GA individual or Cell that may only communicate with its nearest neighbours. The most common connection topology is a toroidal mesh [6]. An example of this type of interaction can be seen on Fig. 3.

The neighborhood of a specific individual on the cellular grid overlaps that of its neighbors. This ensures that good traits and characteristics may travel throughout the grid. In cGAs the reproductive cycle is executed inside the neighborhood of each individual and generally consists of selecting from amongst its neighbors with a certain criterion (e.g. tournament or roulette-wheel selection) a parent with which the cell can apply any recombination operators and finally update its own genetic material.

Fig. 3 Simple toroidal interaction



Mutation is performed by the simple random selection of a cell and making some minimum change to its genetic material. The key aspect of cGA is that this reproduction cycle can be executed in parallel, executing each cell in a different Java thread or CUDA core. Pseudo-code for the canonical cGA (as per Alba and Dorronsoro [6]) can be seen in Listing 1.

Algorithm 1 Pseudo-code of a canonical cGA

```

1: procEvolve(cga)
2: GenerateInitialPopulation(cga.pop)
3: Evaluation(cga.pop)
4: while !StopCondition() do
5:   for individual  $\leftarrow$  1 to cga.popsizedo
6:     neighbors  $\leftarrow$  CalculateNeighborhood(cga,position(individual));
7:     parents  $\leftarrow$  Selection(neighbors);
8:     offspring  $\leftarrow$  Recombination(cga.Pc,parents);
9:     offspring  $\leftarrow$  Mutation(PM);
10:    Replacement(position(individual),auxiliaryPop,offspring);
11:   end for
12:   cga.pop  $\leftarrow$  auxiliaryPop;
13: end while
14: return Best(cga.pop)
  
```

From Listing 1, it can be seen that the cGA algorithm does not differ greatly from a sequential GA. The essential differences of the cGA approach are the Grid/Toroidal interaction and the attendant parallelism.

2.4 Parallel Scatter Search

Scatter Search (SS) [37] is an evolutionary algorithm that works with several sets of solutions during its execution. Unlike GAs, the SS algorithm performs a guided search via a combination of simple rules [6]. In order to carry out this strategy, a moderately-sized reference set (*refset*) is selected from a larger population of solutions. This set is generated and iteratively improved, always attempting to maintain a fair balance between intensification and diversification phases.

After combining solutions in the reference set, a local search procedure is applied to improve the resulting solution, and the *refset* is updated to incorporate good and diverse solutions. This process is repeated until the stop criterion is met. This method provides not only a single heuristic solution, but also a reduced set of diverse high quality solutions [31].

The SS algorithm was proposed in 1970 as an extension of the process of combining decision rules and problem constraints. The initial SS approaches utilized characteristics observed in the search space to construct new candidate solutions. Nowadays this concept is performed by the addition of a Tabu search technique inside the SS algorithm [30]. SS also uses an implicit form of memory, since this algorithm keeps track of the best and most diverse solutions. This implicit memory is improved by the adaptive memory principles of Tabu Search, and the hybrid approach SS-Tabu search has thus been successfully applied to a wide range of combinatorial problems [38–40].

The basic Sequential Scatter Search (sSS) can be seen in Listing 2. The Scatter Search generic algorithm consists of five main processes: *Diversification Generation Method*, which generates diverse solutions; *Improvement Method*, which enhances each solution; *Reference Set Update Method*, which builds and updates the reference set of N good solutions, *Subset Generation Methods*, which builds several subsets with different solutions and finally *Solution Combination Method*, which combines the solutions in the produced subsets.

In execution, this metaheuristic starts by generating a large set of diverse solutions *Pop* (obtained from *Generation Method*). An *Improvement Method* is applied to each solution, reaching a better solution, which is added to *Pop*. The *RefSet* is built from a selection of the solutions in *Pop*. It is important to note that selecting ‘good’ solutions does not necessarily mean the selection of solutions with higher fitness value, it is necessary to implement a selection method that also chooses diverse solutions. Several subsets of solutions are then selected by the *Subset Generation Method*. The *Combination Method* combines solutions with high fitness and high diversity in order to improve upon the best solution so far. Finally, *Reference Set Update Method* uses the obtained solution to update the *RefSet*.

Parallelising Scatter Search can be achieved in several ways. For example, a multi-core routine for generating solutions in the first phase is one of the most elementary options. Calculation of the fitness function (where CPU usage is typically intensive) may also be possible, depending about the problem that need to be solved (In our current CTTP case this can be done, since it is possible to find concurrently the number of conflicts per event). Moreover, parallel implementations allow also solving problems or finding improved solutions due to the partitioning of the search space, since several CPU cores can be assigned to different solution subsets and each one can be focused in a specific area of the search landscape.

Algorithm 2 Sequential Scatter Search (sSS)

```

1: Pop = Create Population()
2: Refset= Generate Reference Set(Pop)
3: while !StopCriteria1() do
4:   while !StopCriteria2() do
5:     Subset Generation Method(Refset)
6:     Solution Combination Method(Refset)
7:     Improving Method(Refset)
8:   end while
9:   Reference Set Update
10: end while
11: return Best(Pop)

```

Parallelism thus allows not only a reduction in the running time of local search methodologies but also an improvement their effectiveness and precision. In this paper, we implement two state of the art parallelizations of Scatter Search gather data about their performance on the CTTP.

2.4.1 Replicated Parallel Scatter Search

The Replicated Parallel Scatter Search (RPSS) [6] is a multistart search in which the local searches are replaced by SS methods with multiple distinct populations running on parallel processors. This method can be seen as a parallel hybridization of SS and multi-start Search. RPSS is described in detail in Listing 3.

Algorithm 3 Replicated Parallel Sequential Scatter Search

```

1: For processor  $r = 1, \dots, n_p$  do in parallel
2: Pop = Create Population()
3: Refset= Generate Reference Set(Pop)
4: while !StopCriteria1() do
5:   while !StopCriteria2() do
6:     Subset Generation Method(Refset)
7:     Solution Combination Method(Refset)
8:     Improving Method(Refset)
9:   end while
10:  Reference Set Update
11: end while
12: Endfor
13: return Best(Pop)

```

2.4.2 Multiple Combinations Scatter Search

This parallel SS metaheuristic utilizes multiple processors to execute a variety of combinations methods. These specific methods depend on the problem itself and the representation used. In this paper, we use 3 different types of combinations: *Single Random Point*, where for each pair of selected solutions a index is selected uniformly at random. This index point is used later to interchange the values of the parents producing the offspring, *Chess Combination*, where only the data in an odd position inside the representation is interchanged and *Arithmetic Combination*, in which the offspring of a pair of solutions holds the arithmetic mean value of the parents. The pseudocode for this approach is given in Listing 4.

Algorithm 4 Multiple Combinations Scatter Search

```

1: Pop = Create Population()
2: Refset= Generate Reference Set(Pop)
3: while !StopCriteria1() do
4:   while !StopCriteria2() do
5:     Subset Generation Method(Refset)
6:     For processor  $r = 1, \dots, n_p$  do in parallel
7:       Solution Combination Method(Refset)
8:       Improving Method(Refset)
9:     Endfor
10:   end while
11:   Reference Set Update
12: end while
13: return Best(Pop)
```

2.5 Parallel Particle Swarm Optimization

Particle Swarm Optimization is inspired by the behavior of social organisms and was developed by Kennedy and Eberhart [41]. Since then, PSO has been applied to almost every problem domain in optimization, computational intelligence and design/scheduling applications [42]. In this metaheuristic, each solution candidate is called a particle and moves through a multidimensional space represents the social

space or search space. This algorithm searches solution space by adjusting the trajectory of each of a population of particles (which in our implementation represent the time values of each variable/event) in a quasi-stochastic manner. The dimension of space is given by the number of variables used to represent the problem. Each particle is attracted toward the position of the current global best g^* and the best position x_i^* in its history, whilst being also subject to small random perturbations.

When a particle i finds a position that is better than any previously encountered, this position is taken as the new current best for particle i . There is a current best for all n particles at any time t during iterations. The aim is to find the global among all the current best solutions until some stopping criterion is met (e.g. the objective no longer improves or after a certain number of iterations).

Each particle also maintains a velocity vector. At each time step, the position of each particle is updated by using its current location and its velocity vector. Equation 3 calculates the velocity vector and Eq. 4 updates the position of a particle. Listing 5 gives the pseudocode for the Sequential PSO metaheuristic.

$$vi = \theta vi + \phi_1 * (xi - B_{Global}) + \phi_2 * (xi - B_{Local}) \quad (3)$$

$$xi = xi + vi \quad (4)$$

where:

- v_i is the velocity of the i -th particle.
- x_i is the position of the i -th particle.
- B_{Global} is the best position found so far by all particles. $-B_{Local}$ is the best position found by the i -th particle.
- ϕ_1 determines the magnitude of the random forces in the direction of neighbourhood best B_{Global} . In the simplest case this is a constant (usually between 0 and 1).
- ϕ_2 determines the magnitude of the random forces in the direction of personal best B_{Local} . In the simplest case this is a constant (usually between 0 and 1).
- θ represents the inertial weighting for the particle i , typically between 0.5 and 0.9. This is equivalent to introducing a virtual mass to stabilize the motion of the particles.

Algorithm 5 Sequential Particle Swarm Optimization Algorithm

Require: θ Inertia Function, ϕ_1 neighbourhood memory coefficient, ϕ_2 personal memory coefficient, n swarm size.

1: Start the swarm particles.

2: Start the v_i for each particle in the swarm.

3: **while** stopping criterion not met **do**

4: **for** $i=1$ to n **do**

5: If the i -th particle's fitness is better than the B_{Local} then replace the B_{Local} with the i -th particle.

6: If the i -th particle's fitness is better than the B_{Global} then replace the B_{Global} with the i -th particle.

7: Update the v_i by eq. 3.

8: Update the x_i by eq. 4.

9: **end for**

10: **end while**

11: **return**

B_{Global}

The parallel PSO can be easily conceptualized in the simplest form if each particle is assigned to a specific core, so each CPU core manages the position of a particle and its interaction with other particles. This is termed Asynchronous PSO [43], in which each particle is completely independent and only makes calls to another particle when it needs new information. As can be seen, each particle uses a messaging protocol to update or enquire the position of some arbitrary other particle. In order to have this information available for each particle, a list Pos is maintained, which keeps the record of the last known position for each particle i . This list needs to be reside in a common memory space from which any particle can read or write at any time. The pseudo code of the asynchronous PSO used in this paper can be seen in Listing 6.

One of the main advantages of this method is the parallelism in the calculation of the fitness function, since each processor is responsible for all the operations performed by a particle. A disadvantage is the exhaustive use of common memory space, which either requires atomic read/write or locking protocols (or else it can happen that a particle reads a position that has been updated meanwhile the read process is done, in which cases the particle will be using outdated information). The number of the particles used depends on the number of cpu-cores, so in order to maximize the efficiency of this approach, 8 or more cores (real or simulated) are really required.

Algorithm 6 Asynchronous Particle Swarm Optimization Algorithm

Require: θ Inertia Function ϕ_1 neighbourhood memory coefficient, ϕ_2 personal memory coefficient, n swarm size.

- 1: Start n_{proc} swarm particles.
 - 2: Start the v_i for each particle in the swarm.
 - 3: Assign each particle $i = 0, \dots, n_{proc}$ to a cpu-core.
 - 4: Create Pos_i and write the first know position for each particle i .
 - 5: **while** stopping criterion not met
 - 6: **do** 6: **For** particle $r = 1, \dots, n_p$ **do in parallel**
 - 7: Calculate Particle i fitness.
 - 8: If the i-particle's fitness is better than the B_{Local} then replace the B_{Local} with the i-particle.
 - 9: If the i-particle's fitness is better than the B_{Global} then replace the B_{Global} with the i-particle.
 - 10:Update the v_i by eq. 3 reading Pos_i list for each particle $j \neq i$.
 - 11: Update the x_i by eq. 4.
 - 12:Update Pos_i value with the new position.
 - 13: **Endfor**
 - 14: **endwhile**
 - 15: **return** B_{Global}
-

3 Solution Approach

3.1 Combining Methodology of Design with Parallel Metaheuristics for the CTTP

As seen in Sect. 2, each parallel metaheuristic detailed so far utilizes similar operators and parameters to their sequential counterparts. In this section we define the codification, operators and parameters used, as well as several details of each metaheuristic configuration.

We use the Methodology of Design approach shown in Sect. 2.2 in order to generalize the implementation of our metaheuristic over different CTTP instances. For each instance 3 lists (MMA, LPH and LPA) are built. The detailed construction process for each of these lists is beyond the scope and purpose of this article, but interested readers are referred to [7, 44]. In the same manner as discussed above, these lists ensure by design that every 3-tuple (e, t, p) represent a feasible selection in terms of time-space constraints. The main optimization exercise is to minimize

the student conflict by means of the permutation of the events into timeslots (i.e. integer values in our representation Fig. 2) reported in the MMA matrix. The function to be minimized has been described in Sect. 2.2 Eqs. 1 and 2.

The codification of each solution candidate or individual is an array of integer values where each integer represent an ID of a set of Vector.

The codification of each solution candidate or individual is an array of integer values where each integer represent an ID of a set of Vector. From Table 3 after the decoding of the integer representation we can read *event 1* is assigned into *Vector-set B* (8am-classroom A), *event 2* is assigned into *Vector-set D* (9am-classrom A)... and so on.

3.2 Operators Used in GA and cGA

For GA and cGA, the selection method used is the roulette-wheel, so every individual has the chance to reproduce with a neighbouring cell in proportion to their fitness. Given the ‘vector-of-integers’ representation, the recombination operator is the single point crossover. The mutation operator is a simple random operator that is done at the final of each generation, where the GA/cGA randomly selects a individual/cell where a single integer is changed at random. Elitism is used in the GA where a percentage of the best solutions are directly passed to the next generation. Elitism for cGA is implemented in form that in each execution the best cell from each grid is not changed i.e. the best cell can interact and update genetic information of its neighbours, but no other cell can modify its own genetic material.

The Parameters used in the GA/cGA are: for stop criteria, a fixed number of function points (a function point means a single decoding of the information in the individual/cell and the single execution of a fitness function), for the recombination and mutation probability we employ a percentage value fixed by the user.

In the cGA, the grid configuration divides the population into several islands. At the beginning of each iteration, all the islands send the cells of their first/last column/row to their neighbouring islands. After receiving the individuals from the neighbours, a sequential cGA is executed in each island/subpopulation (Fig. 4). This approach has been documented by Alba and Dorronsoro [6] with good results. Finally the neighbourhood model used by each cell and island is the NEWS model (North, West, East, South) similar to Fig. 4. Each island is executed in a Java thread and synchronously waits for all the other islands to complete in order to interchange individuals to continue the execution.

3.3 Operators Used in sSS, RPSS and MCSS

For the sequential Scatter Search (sSS) in the *Diversification Generation* method phase we use simple random generation, In this phase several solutions are created

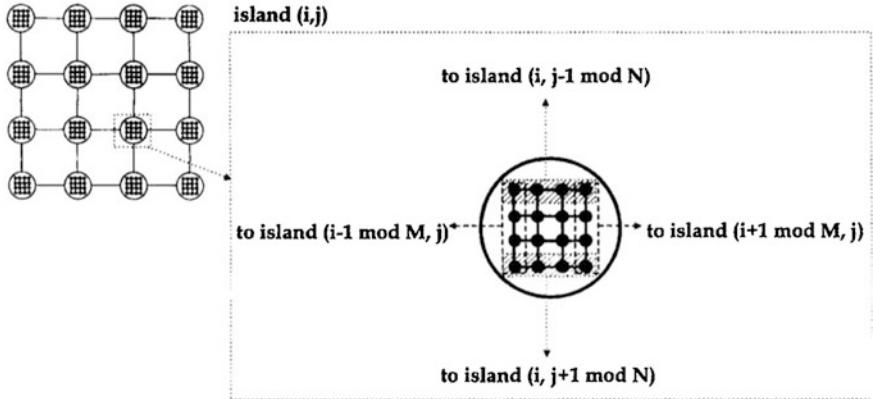


Fig. 4 Grid configuration model used. From Alba and Dorronsoro [6]

uniformly at random by always selecting feasible values inside the cartesian product $LPH_i \times LPA_i$ for each event i inside the current instance. For *Improvement* method, we apply a hill-climbing approach using simple random perturbation. The *Reference Set Update* method picks the best n solutions to update the *Refset* structure. The *Subset Generation* method generates two different subsets: the first one keeps the best l solutions according the fitness function (Sect. 2.2 Eqs. (1) and (2)), and the second one keeps the most diverse m solutions according to the *hamming distance*. Finally, the *Combination* method used is *Single Random Point*, as described above. A Tabu List is also used, which keeps the worse solution in terms of fitness function of the last w SS iterations, so new offsprings are rejected if they are less good than the current Tabu List recorded solutions.

For the RPSS algorithm, each of these combinations are also used in the context of the MCSS, two more combination methods are used along the *single random point* combinator: *Chess Combination* and *Arithmetic Combination*, as described above. In all the cases (SS, RPSS and MCSS) the same stop criterion (i.e. function points) is used.

3.4 Adaptations to PSO and APSO

The PSO and APSO algorithms are designed for real-value problems, however it is possible to adapt these algorithms to discrete scheduling problems such as the CTTP [45–48]. Since in traditional, PSO each particle is described by an array of real values, we need to adapt this representation to use only integer values, as in Fig. 2. It is possible to define the vector operations of PSO between vectors of integer values, so the PSO algorithm is still applicable. However, in order to keep the feasibility of solutions that the new values represent a valid choice in the

$LPH_i \times LPA_i$ space. Values out of range can be clamped to the range of permissible values for each variable.

In the case of APSO, the exact same considerations may be implemented. Finally, as stop criterion, a fixed number of function points is used.

4 Experiments and Results

In this section, several experiments are performed in order to find the speedup and performance difference between sequential and parallel approaches to CTTP. We describe each experiment together with the characteristics of the benchmarks adopted.

4.1 Test Instances

The methodology of design allows the solution of diverse problem formulations it is merely necessary that each instance be expressed in terms of the generic structures (MMA, LPH and LPA). A well-known CTTP benchmark from the second International Timetabling Competition, PATAT ITC2007 Track 2 [27], is used for comparison between sequential and parallel approaches. This benchmark has 24 instances with main characteristics as follows:

Patat 2007 These instances consist in:

- A set of n events that are to be schedule into 45 timeslots.
- A set of r rooms, each which has a specific seating capacity.
- A set *features* that are satisfied by rooms and required by events.
- A set of s students who attend various different combination of events.

The hard constraints are:

- No student should be required to attend more than one event at the same time
- Each case the room should be big enough for all the attending students
- Only one event is put into each room in any timeslot.
- Events should only be assigned to timeslots that are pre-defined as available
- Where specified, events should be scheduled to occur in the correct order.

The Soft constraints are:

- Students should not attend an event in the last timeslot of a day.
- Students should not have to attend three or more events in successive timeslots.
- Student should not be required to attend only one event in particular day.

4.2 Experimental Design

According to Alba and Dorronsoro [6], the fairest comparison between sequential and parallel algorithms uses function points as the stop criterion. We employ the *weak speedup* metric proposed by Alba and Dorronsoro [6] because it affords an effective comparison method between sequential and parallel algorithms. For each experiment, 100 independent tests were performed for each ITC2007 instance using Sequential (GA, sSS, PSO) and parallel (cGA, RPSS, MCSS and APSO) algorithms. Each algorithm executes exactly 1000 function points before stop. In each experiment a sequential algorithm (GA, sSS and PSO) is compared to its parallel counterpart.

4.2.1 GA Versus cGA

The parameters values used for the GA include a population of 256 individuals with a crossover probability of 0.92 %, Mutation of 0.13 % and a Elitism percentage between generations of 0.0625 %. These values were obtained by preliminary experiments. For the cGA, we implement 16 islands on a 4×4 grid, each island has a inner grid of 16 Cells in a 2D array of 4×4 cells (as depicted in Fig. 4), giving a total of 256 individuals/cells. The crossover, selection and mutation operators are the as described above for the GA. Elitism is implemented by selecting the best individual from each 4×4 cell grid. the results of the application of GA and CGA on ITC2007 can be seen on Table 4.

Table 4 shown the results of tests between cGA and GA for some ITC2007 instances. We reported only most important instances for space. The reported values for the first two columns were obtained by the fitness value of the best individual at the end of the execution, i.e. the number of student conflicts in the timetabling solution, with a lesser value implying a better solution to the CTTP. It should be emphasised that these evaluations considering only the ITC2007 hard constraints described above. The column *Best fit* shows the best fitness reached by the metaheuristic (sGA or cGA) in our experiments. *Avg fit* shows the average fitness from our algorithms over 100 independent tests and *std devf* shows the corresponding standard deviation. *Avg. time* shows the average time in seconds to run a single test, with *std devt* the corresponding standard deviation. Finally the *speedup* column shows the weak speed-up metric proposed by Alba and Dorronsoro [6].

4.2.2 sSS Versus RPSS Versus MCSS

The parameters values used for the Sequential SS (sSS) include a population size of 256 individuals and a simple hillclimbing improving method, with two solution subsets, the first one with the 50 best solutions (according to the fitness function)

Table 4 Results experiments ITC2007

Instance		Best fit	Avg fit	Std devt	Avg. time	Std devt	Speedup
ITC2007-1	sGA	1253	1362.4	55	11.1	3	
	cGA	881	975.3	45.96	3.92	1.05	2.83
ITC2007-2	sGA	1251	1388.5	56.87	11.21	3.06	
	cGA	892	999	42.78	4.57	0.37	2.45
ITC2007-3	sGA	434	556.8	57.8	4.79	1.11	
	cGA	215	286.75	27	2.66	0.28	1.8
ITC2007-7	sGA	265	334.9	28.96	4.5	1.27	
	cGA	154	187.8	15.5	2.46	0.39	1.82
ITC2007-8	sGA	287	375	36.14	5.75	0.93	
	cGA	147	196.3	18.72	2.5	0.42	2.3
ITC2007-9	sGA	1190	1403.1	69.77	10.52	3.26	
	cGA	860	979.23	52.67	4.29	0.7	2.45
ITC2007-10	sGA	1256	1400.3	53.58	11.05	3.41	
	cGA	899	1011.6	47.38	3.83	0.45	2.88
ITC2007-16	sGA	237	326.85	34.47	4.59	0.96	
	cGA	113	147.21	16.82	1.41	0.06	3.255
ITC2007-17	sGA	286	432.15	69.81	3.23	0.21	
	cGA	46	160.15	35.05	1.1	0.05	2.93
ITC2007-18	sGA	865	1000.59	56.14	4.47	0.95	
	cGA	512	635.17	38.87	1.47	0.04	3.04
ITC2007-19	sGA	686	814.96	58.1	7.14	2.24	
	cGA	426	482.5	32.52	1.88	0.05	3.79
ITC2007-20	sGA	738	879.07	59.34	10.53	2.86	
	cGA	430	497.52	38.96	2.38	0.05	4.42
ITC2007-21	sGA	761	847.17	28.99	12.97	4.34	
	cGA	516	614.27	26.82	3.03	0.07	4.28
ITC2007-22	sGA	1400	1556.15	59.93	16.87	4.98	
	cGA	1084	1185.74	75.48	3.84	0.08	4.39
ITC2007-23	sGA	2595	2882.6	114.88	7.93	2.42	
	cGA	1902	2137.85	88.12	2.36	0.07	3.36
ITC2007-24	sGA	757	886.48	52.31	8.78	2.61	
	cGA	453	527.9	35.12	2.41	0.06	3.64

and the second one with the 30 most diverse solutions according to the Hamming distance, since it measures the minimum number of substitutions required to change one integer string into the other. In this paper a higher hamming distance is used as a dispersion metric. A *Single Random Point* crossover is used to combine solutions from both subsets until 256 valid offspring are obtained to update the original population. Finally a tabu list keeps a record of the worse solutions of the last 50 iterations to prohibit solutions with lower fitness value from entry into the new

population. The RPSS algorithm was divided into 8 smaller SS algorithms with a population 32 individuals, each one implemented in a Java thread. Parameters used in these smaller SS are the same that the used in sSS. MCSS algorithm uses 3 combination methods each of them assigned to a CPU-core, these combination methods are executed in parallel. The Methods are: *Single Random Point*, *Chess Combination* and *Arithmetic Combination*. The results of the application of sSS, RPSS and MCSS over ITC2007 instances can be seen on Table 5.

Table 5 gives the results for our SS algorithms on ITC2007 instances. The reported values for the first two columns were obtained by the evaluation of the best individual at the end of the execution with the fitness function, so this is the number of conflicts (student conflicts) on the timetabling solution, a less value of fitness means a better solution for the CTTP. We need to say that these evaluations were made just only considering hard constraints for ITC2007, as the same as the GA experiment. The column *Best fit* shows the best fitness reached by the metaheuristic (sSS versus RPSS versus MCSS) in our experiments. *Avg fit* shows the average fitness from our algorithms over 100 independent tests and *std_devf* shows the corresponding standard deviation. *Avg. time* shows the average time in seconds to run a single test, with *std devt* the corresponding standard deviation. Finally the *speedup* column shows the weak speed-up metric proposed by Alba and Dorronsoro [6].

4.2.3 PSO Versus APSO

The parameters values used for the PSO (sSS) include a population size of 256 individuals, $\phi_1 = 0.6$, $\phi_2 = 0.4$, the inertial weighting was taken to 0.3. These values were obtained via preliminary experiment. For the APSO the same values are used. the results of the application of PSO and APSO over ITC2007 instances can be seen on Table 6.

Table 6 shown the results of our PSO algorithms over ITC2007 instances. Our fitness function evaluates the number of conflicts (student conflicts) on the timetabling built by the algorithms, so a less value of fitness means a better solution for the CTTP as the same as the GA experiment. The column *Best fit* shows the best fitness reached by the metaheuristic (PSO or APSO) in our experiments. *Avg fit* shows the average fitness from our algorithms over 100 independent tests and *std_devf* shows the corresponding standard deviation. *Avg. time* shows the average time in seconds to run a single test, with *std devt* the corresponding standard deviation. Finally the *speedup* column shows the weak speed-up metric proposed by Alba and Dorronsoro [6].

Table 5 Results experiment ITC2007 sSS versus RPSS versus MCSS

Instance		Best fit	Avg fit	Std devt	Avg. time	Std devt	Speedup
ITC2007-1	sSS	1500	1753.8	60.5	15.5	5	
	RPSS	970	1230.2	35.46	6.32	1.05	2.83
	MCSS	890	909.5	29.5	7.41	1.05	2.83
ITC2007-2	sSS	1356	1569.4	45.2	20.5	5.6	
	RPSS	1150	1386.1	32.2	9.5	3.2	2.15
	MCSS	930	980.5	14.25	12.5	3.7	1.64
ITC2007-3	sSS	450	750.5	28.5	10.5	3.8	
	RPSS	320	430.8	22.4	5.6	2.15	1.87
	MCSS	250	359.4	15.9	6.8	3	1.54
ITC2007-7	sSS	270	330.4	30.4	6.5	2.3	
	RPSS	260	315.7	18.45	3.56	1.5	1.82
	MCSS	193	250.48	15.5	4.12	1.5	1.57
ITC2007-8	sSS	240	376.78	26.5	6.32	1.12	
	RPSS	210	254.12	21.15	2.21	0.48	2.85
	MCSS	160	230.4	18.56	3.45	0.78	1.83
ITC2007-9	sSS	1217	1538.7	44.05	12.23	2.23	
	RPSS	970	1127.4	22.15	7.45	1.56	1.64
	MCSS	850	1089.1	19.56	9.45	1.88	1.29
ITC2007-10	sSS	1335	1568.2	45.5	12.4	2.12	
	RPSS	942	1245.5	32.12	5.26	1.45	2.35
	MCSS	913	1120.2	23.2	6.26	1.56	1.98
ITC2007-17	sSS	295	335.23	39.18	3.26	1.23	
	RPSS	148	259.45	27.94	2.49	0.84	1.3
	MCSS	78	168.45	30.12	2.81	0.92	1.16
ITC2007-21	sSS	722	848.26	34.33	12.11	2.1	
	RPSS	638	681.5	18.56	6.15	1.46	1.96
	MCSS	507	588.46	16.45	7.74	1.84	1.56
ITC2007-23	sSS	2665	3054.5	156.45	8.46	2.88	
	RPSS	2156	2566.15	76.55	6.65	1.64	1.27
	MCSS	1969	2332.6	54.81	7.92	1.92	1.06
ITC2007-24	sSS	780	1001.25	34.54	9.06	3.01	
	RPSS	656	848.56	28.48	4.56	1.56	1.99
	MCSS	470	636.23	23.26	6.65	1.88	1.36

Table 6 Results experiment ITC2007 PSO versus APSO

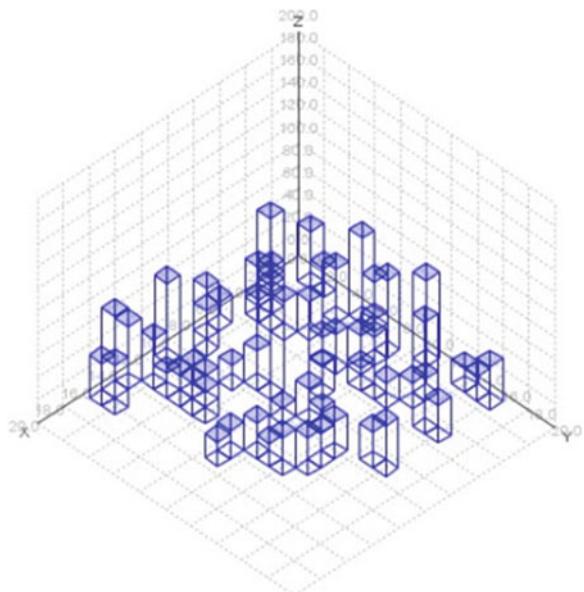
Instance		Best fit	Avg fit	Std devf	Avg. time	Std devt	Speedup
ITC2007-1	PSO	1332	1526.32	62.56	8.2	2.46	
	APSO	1268	1596.5	32.56	4.56	1.16	1.79
ITC2007-2	PSO	1270	1432.1	45.56	9.62	1.23	
	APSO	1112	1402.2	32.23	3.32	1.55	2.89
ITC2007-3	PSO	460	672.12	62.3	4.36	0.98	
	APSO	330	599.23	30.12	2.32	0.36	1.87
ITC2007-7	PSO	288	350.26	33.65	5.6	0.68	
	APSO	268	344.23	31.85	2.1	0.84	2.66
ITC2007-8	PSO	312	398.45	33.23	4.16	0.78	
	APSO	298	378.62	27.45	2.12	0.69	1.96
ITC2007-9	PSO	1256	1584.68	59.56	7.46	1.23	
	APSO	1156	1472.88	56.12	3.36	1.04	2.22
ITC2007-10	PSO	1312	1567.23	26.12	8.45	1.75	
	APSO	1287	1552.48	32.15	4.26	1.28	1.98
ITC2007-13	PSO	780	870.45	41.22	7.45	1.01	
	APSO	764	856.66	40.18	2.45	0.78	3.04
ITC2007-14	PSO	750	866.12	32.15	6.55	1.25	
	APSO	732	832.55	31.23	2.72	1.09	2.4
ITC2007-15	PSO	272	499.18	32.44	2.99	1.16	
	APSO	254	447.75	26.48	1.34	0.99	2.23
ITC2007-16	PSO	278	350.2	36.26	3.99	1.12	
	APSO	256	332.15	30.11	1.88	0.83	2.12
ITC2007-17	PSO	312	465.88	29.93	3.01	0.87	
	APSO	296	443.57	28.24	1.55	0.77	1.94
ITC2007-18	PSO	878	1156.2	45.26	3.57	0.99	
	APSO	776	1087.7	44.21	1.15	0.98	3.1
ITC2007-19	PSO	690	820.15	36.26	6.28	1.04	
	APSO	512	807.47	33.64	3.03	0.76	2.07
ITC2007-20	PSO	756	890.15	53.33	8.43	2.05	
	APSO	737	886.52	50.15	4.51	1.73	1.86
ITC2007-21	PSO	782	879.56	45.56	10.52	1.75	
	APSO	765	866.57	44.86	4.48	1.1	2.34
ITC2007-22	PSO	1465	1532.1	45.56	12.42	1.23	
	APSO	1405	1502.2	32.23	4.82	0.55	2.57
ITC2007-23	PSO	2678	3177.64	61.83	15.26	1.25	
	APSO	2456	3025.83	59.47	7.25	1.65	2.1
ITC2007-24	PSO	766	912.56	54.32	7.88	1.85	
	APSO	712	877.56	57.85	3.48	0.88	2.26

4.3 Analysis of Results

As it can be seen on Table 4 we achieve a faster execution times in the cGA than the sGA. This is unsurprising, since we utilize Java threads so that each island or sub population is working in parallel. However we obtained not only a faster execution time but a good performance as well, since both sequential and parallel algorithms were allowed to perform the exact same number of fitness evaluation before stop, as seen on the results table, our cGA presents a lower fitness value and a lower standard deviation value for both the time and the objective function. This result can be explained by the manner in which elitism acts in the cellular model. This approach conserves several cell/individual with different genetic value, then in the phase of interchange this genetic material travels over the grid. In the sGA, cells preserved by the elitism have similar genetic material. An snapshot of the Cellular Grid as well as its performance graph can be seen in Figs. 5 and 6. The weak speedup metric indicates the number of times that the parallel algorithms complete the same function points faster than their sequential counterpart.

For the SS experiments, Table 5 shows the results over 100 independent executions with each one having 1000 function points as stop criterion. MCSS gives the best results so far in this experiment but this approach does not achieve the best weak speedup. Since MCSS has more operators than sSS and RPSS, this is why the results are better than approaches with only one combination method. This also explains why this approach is not the fastest, since it needs to execute different operators for a specific pair of individuals. Despite MCSS being slower than RPSS,

Fig. 5 Snapshot of cGA grid after a test



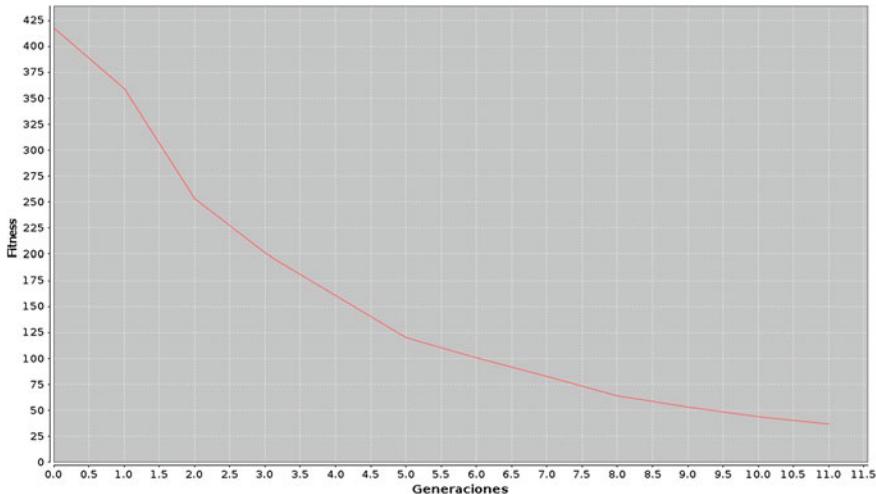


Fig. 6 Performance of cGA over a test

this approach presents a good speedup against sSS, since with the same number of fitness points MCSS achieves better results in term of student conflicts.

PSO is well-documented as being a fast and flexible strategy that can be applied to a wide range of problems. As expected, the APSO algorithm improves in all cases on the execution time of its sequential counterpart, but since no important changes were implemented in the APSO code, the performance is similar in terms of solution quality. The PSO and APSO approaches can be seen to offer similar behaviour over the ITC 2007 Track 2 instances.

If we compare the best parallel algorithms cGA, MCSS and APSO, then the APSO algorithm is easily the fastest algorithm with the highest speedup value against its sequential counterpart. However, its performance in term of solution quality is also the worst. The MCSS algorithm implements a set of combination methods, with operator diversity being its main advantage against sSS, since MCSS can perform different operations over a pair of solutions. Finally, the cGA algorithm achieves the best results in term of solution quality, which can be explained if one considers the action of migration operators in addition to the grid interrelationships. The grid approach allows each individual or cell to be improved unless the cell is already the best solution on the grid. This situation does not always happen in a regular GA, where a selection operator can leave some solutions without offspring. Also the migration operator guarantees the flow and mixing of the information between solutions.

5 Conclusions and Future Work

This paper presents several comparisons between sequential and parallel computing models for GA, SS and PSO metaheuristics in the context of the Course Timetabling Problem. The parallel approach aims to yield good solutions in a short time. Our best algorithm, the cGA, utilizes a toroidal grid configuration that ensures the interchange of good genetic material over the whole population.

We utilize the ‘Methodology of Design’ approach to generalize the process of CTTP solution by means of generic structures. This model ensures that if any other type of CTTP can be represented by the same structures, then the metaheuristics detailed in this paper can work with these new instances without any reimplementation. This paper has also explored the use of parallel computing for ITC2007 Track 2, a well-known set of benchmarks instances adopted by state of the art CTTP approaches.

The cGA algorithm gives encouraging results in terms of speedup and solution quality when compared to sequential GA with similar parameters. More exhaustive experiments with ITC 2007 are a topic for future work. The application of cGA to other instances such as ITC2002 and UNITIME is proposed. Significantly, more research is needed in the interaction of new interaction models and migration processes. Finally, the concurrency mechanism for our model is currently Java threads, and we propose to move to Nvidia CUDA in order to increase the number of available cores.

Acknowledgment Authors thanks the support received from the Consejo Nacional de Ciencia y Tecnología (CONACYT) México and University of Stirling UK.

References

1. Soria-Alcaraz, J.A.: Diseño de horarios con respecto al alumno mediante técnicas de cómputo evolutivo. Master’s thesis, Instituto Tecnológico de León (2010)
2. Cooper, T.B., Kingston, J.H.: The complexity of timetable construction problems. Ph.D. thesis, The University of Sydney (1995)
3. Willemen, R.J.: School timetable construction: Algorithms and complexity. Ph.D. thesis, Institute for Programming research and Algorithms (2002)
4. Lewis, R.: Metaheuristics for university course timetabling. Ph.D. thesis, University of Nottingham, Aug 2006
5. Wolpert, H., Macready, G.: No free lunch theorems for search. Technical report The Santa Fe Institute, vol. 1 (1996)
6. Alba, E., Dorronsoro, B.: The state of the art in cellular evolutionary algorithms. In: Cellular Genetic Algorithms, vol. 1. Springer, LLC, Berlin (2008)
7. Soria-Alcaraz Jorge, A., Carpio, M., Puga, H., Sotelo-Figueroa, M.: Comparison of Metaheuristic Algorithms with a Methodology of Design for the Evaluation of Hard Constraints over the Course Timetabling Problem, Studies in Computational Intelligence, vol. 451. Springer, Berlin (2013)

8. Conant-Pablos, S.E., Magaa-Lozano, D.J., Terashima-Marin, H.: Pipelining memetic algorithms, constraint satisfaction, and local search for course timetabling. In: MICAI Mexican international conference on artificial intelligence, vol. 1, pp. 408–419 (2009)
9. de Werra, D.: An introduction to timetabling. *Eur. J. Oper. Res.* **19**(2), 151–162 (1985)
10. Carter, M.: A survey of practical applications of examination timetabling algorithms. *Oper. Res.* **34**, 193–202 (1986)
11. Lajos, G.: Complete university modular timetabling using constraint logic programming. In: Burke, E. Ross, R. (eds.) Practice and Theory of Automated Timetabling (PATAT) I, vol. 1153, pp. 146–161 (1996)
12. Boizumault, P., Delon, Y., Peridy, L.: Logic programming for examination timetabling. *Logic Program* **26**, 217–233 (1996)
13. Lu, Z., Hao, J.K.: Adaptive tabu search for course timetabling. *Eur. J. Oper. Res.* **200**(1), 235–244 (2010)
14. Colom, A., Dorigo, M., Maniezzo, V.: Metaheuristics for high-school timetabling. *Comput. Optim. Appl.* **9**, 277–298 (1997)
15. Yu, E., Sung, K.S.: A genetic algorithm for a university weekly courses timetabling problem. *Trans. Oper. Res.* **9**, 703–717 (2002)
16. Mayer, A., Nothegger, C., Chwatal, A., Raidl, G.: Solving the post enrolment course timetabling problem by ant colony optimization. In: International Timetabling Competition 2007 (2008)
17. Socha, K., Knowles, J., Samples, M.: A max-min ant system for the university course timetabling problem. In: Dorigo, M., Caro, G.D., Samples, M. (eds.) Proceedings of Ants 2002 —3rd International Workshop on ant Algorithms. Lecture Notes in Computer Science, pp. 1–13. Springer, Berlin (2002)
18. Burke, E., Eckersley, A., McCollum, B., Petrovic, S., Qu, R.: Hybrid variable neighbourhood approaches to university exam timetabling. *Eur. J. Oper. Res.* **206**(1), 46–53 (2010)
19. Sabar, N.R., Ayob, M., Kendall, G., Qu, R.: A honey-bee mating optimization algorithm for educational timetabling problems. *Eur. J. Oper. Res.* **216**(3), 533–543 (2012)
20. Thompson, J.M., Dowsland, K.A.: A robust simulated annealing based examination timetabling system. *Comput. Oper. Res.* **25**, 637–648 (1998)
21. Rudova, H., Muller, T., Murray, K.: Complex university course timetabling. *J. Sched.* **14**, 187–207 (2011). URL <http://dx.doi.org/10.1007/s10951-010-0171-3>. doi:[10.1007/s10951-010-0171-3](http://dx.doi.org/10.1007/s10951-010-0171-3)
22. Cambazard, H., Hebrard, E., O'Sullivan, B., Papadopoulos, A.: Local search and constraint programming for the post enrolment-based course timetabling problem. *Ann. Oper. Res.* **194**, 111–135 (2012). doi:[10.1007/s10479-010-0737-7](http://dx.doi.org/10.1007/s10479-010-0737-7)
23. Burke, E.K., McCollum, B., Meisels, A., Petrovic, S., Qu, R.: A graph-based hyperheuristic for educational timetabling problems. *Eur. J. Oper. Res.* **176**(1), 177–192 (2007)
24. Soria-Alcaraz, J.A., Terashima-Marin, H., Carpio, M.: Academic timetabling design using hyper-heuristics. *Adv. Soft Comput.* **1**, 158–164 (2010)
25. Qu, R., Burke, E.K., McCollum, B.: Adaptive automated construction of hybrid heuristics for exam timetabling and graph colouring problems. *Eur. J. Oper. Res.* **198**(2), 392–404 (2009)
26. URL <http://www.idsia.ch/Files/ttcomp2002/>
27. URL <http://www.cs.qub.ac.uk/itc2007/>
28. Causmaecker, P.D., Demeester, P., Berghe, G.V.: A decomposed metaheuristic approach for a real-world university timetabling problem. *Eur. J. Oper. Res.* **195**(1), 307–318 (2009)
29. Kahar, M., Kendall, G.: The examination timetabling problem at universiti Malaysia Pahang: comparison of a constructive heuristic with an existing software solution. *Eur. J. Oper. Res.* **207**(2), 557–565 (2010)
30. Glover, F.: Tabu search for nonlinear and parametric optimization (with links to genetic algorithms). *Discrete Appl. Math.* **49**, 231–255 (1994)
31. Glover, F.: A template for scatter search and path relinking. Selected papers from the 3rd European conference on artificial evolution, AE '97, pp. 3–54. Springer, London, UK (1998)

32. Barney, B.: Introduction to parallel computing. URL <https://computing.llnl.gov/tutorials/parallelcomp/>
33. Patterson, D., Hennessy, J.: Computer Architecture a Quantitative Approach. Morgan Kaufmann, Massachusetts (1996)
34. Faber, V., Lubeck, O., White, A.: Superlinear speedup of an efficient sequential algorithms is not possible. *Parallel Comput.* **3**, 259–260 (1986)
35. Holland, J.: Adaptation in natural and artificial systems. The University of Michigan Press, Ann Harbor (1975)
36. Robertson, G.: Parallel implementation of genetic algorithms in a classifier system. In: Proceedings of the 2nd International Conference on Genetic Algorithms (ICGA), pp. 140–147 (1987)
37. Laguna, M., R. M.: Scatter Search: Methodology and Implementations. Kluwer, The Netherlands (2003)
38. Battiti, R., Brunato, M., Mascia, F.: Reactive Search and Intelligent Optimization, Operations Research/Computer Science Interfaces Series, vol. 45. Springer, Berlin (2009)
39. Wang, Y., L, Z., Glover, F., Hao, J.K.: Probabilistic grasp-tabu search algorithms for the UBQP problem. *Comput. Oper. Res.* **40**, 3100–3107 (2012)
40. Xu, Y., Qu, R.: A hybrid scatter search meta-heuristic for delay-constrained multicast routing problems. *Appl. Intell.* **36**, 229–241 (2012)
41. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of IEEE International Conference on Neural Networks, vol. 1, pp. 1942–1948 (1995)
42. Yang, X.S.: Nature-Inspired metaheuristics algorithms, 2nd edn. Luniver Press, pp. 63–69 (2010)
43. Koh, B.I., George, A.D., Haftka, R.T., Fregly, B.J.: Parallel asynchronous particle swarm optimization. *Int. J. Numer. Meth. Eng.* **67**(4), 578–595 (2006)
44. Soria-Alcaraz Jorge, A., Carpio, M., Puga, H., Sotelo-Figueroa, M.: Methodology of design: a novel generic approach applied to the course timetabling problem. In: Melin, P., Castillo, O. (eds.) Soft Computing Applications in Optimization, Control, and Recognition, Studies in Fuzziness and Soft Computing, vol. 294, pp. 287–319. Springer, Berlin (2013)
45. Jarboui, B., Damak, N., Siarry, P., Rebai, A.: A combinatorial particle swarm optimization for solving multi-mode resource-constrained project scheduling problems. *Appl. Math. Comput.* **195**(1), 299–308 (2008)
46. Liao, C.J., Tseng, C.T., Luarn, P.: A discrete version of particle swarm optimization for flowshop scheduling problems. *Comput. Oper. Res.* **34**(10), 3099–3111 (2007)
47. Tian, Y., Liu, D., Yuan, D., Wang, K.: A discrete PSO for two-stage assembly scheduling problem. *Int. J. Adv. Manuf. Technol.* 1–19 (2012)
48. Tseng, C.T., Liao, C.J.: A discrete particle swarm optimization for lot-streaming flow shop scheduling problem. *Eur. J. Oper. Res.* **191**(2), 360–373 (2008)

Simplification of Decision Rules for Recommendation of Projects in a Public Project Portfolio

**Laura Cruz-Reyes, César Medina Trejo,
Fernando López Irrarragorri and Claudia G. Gómez Santillan**

Abstract In this paper, we propose the use of decision rules to aid in the recommendation of a portfolio of public projects. A decision table indicates what decisions should be made when the condition attributes are satisfied. Projects can be modeled as decision tables, where the characteristics of the projects are condition attributes and the qualification of each project is the decision attribute. Reducing the decision rules, we can give a simple explanation of why a certain project has its qualification; this simplification is a useful procedure because most decision problems can be formulated in a decision table. Public portfolio problem, due to its nature, has been approached by multi-criteria algorithms, which generate a set of solutions in the Pareto frontier. The selection of a portfolio depends on the decision maker, so the simplified decision rules can help him/her to analyze why a project have been added to a certain portfolio and justify the final selection.

1 Introduction

In this paper we propose the application of decision rules to aid in the justification of the recommendation for the selection of the public project portfolio with the use of rough sets. The selection of the portfolio is a nontrivial problem because it

L. Cruz-Reyes (✉) · C.M. Trejo · C.G. Gómez Santillan
Tecnológico Nacional de México, Instituto Tecnológico de Ciudad Madero,
México. 1o. de Mayo y Sor Juana I. de la Cruz S/N C.P. 89440 Cd., Madero,
TAMPS, Mexico
e-mail: lauracruzreyes@itcm.edu.mx

C.M. Trejo
e-mail: cesarmediatetrejo@gmail.com

C.G. Gómez Santillan
e-mail: cggs71@hotmail.com

F.L. Irrarragorri
Universidad Autónoma de Nuevo León, León, Mexico
e-mail: flopez65@gmail.com

involves a high dimensionality in the optimization process and has several attributes in conflict. That is why multi-criteria algorithms are used to solve this optimization problem; however they don't offer one solution but a set of solutions.

The selection of a portfolio among efficient portfolios depends on an entity called the *decision maker* (DM), which may be a person in charge of making the decision or a group of deciders. Each decision maker has its base of preferences and beliefs, different DMs prefer different portfolios. Here is the problem of submitting a portfolio where the DM is convinced that this suits his/her preference model.

Providing explanations and justifications with the recommended portfolio is an important feature of the tools to support the decision [1]. The DM must be convinced that the proposed portfolio is the best for him/her, to accept it more safely. There is a demand of the recommendations shown to the DM in justifying and explaining the set of solutions on decision support systems [2], which aim to aid a user in making decisions.

The Decision tables are an organized way of presenting information as decision rules; each row of the table can be viewed as a decision rule. A rule having some condition attributes produces a consequence called a decision attribute, the rules can be expressed as “if..then..else” rules. Decision rules can assist people in making or understanding decisions.

The set of decision rules can be very large, so the simplification of the rules helps the cognitive capacity of the decision maker in finding the relevant information quickly. In this work, for each portfolio presented, a set of decision rules to summarize why certain projects are or not in the portfolio is derived.

2 Public Portfolio Problem

In organizations or institutions they usually have many projects and there are no resources to support them all, the issue is to choose the project portfolio which provides the most benefit. In public projects this problem is even greater, because the public projects seek to provide a benefit to society, poor decisions can have a major impact in the social sphere.

The public projects have special characteristics hence the projects need a special treatment; Fernandez et al. gives a description of these [3, 4]:

- The way in which the projects are rated is usually performed by multiple conflicting criteria.
- The requirements of a project to be fund are not known for certain.
- The effects of these projects in the society sometimes are only reflected in the long term and indirectly; therefore, they are difficult to measure. It should be considered in equal terms the society and its impact on individuals, and take into account the economic benefits of the projects in society to have a general social landscape.

The problem of the selection of public project portfolio has many conflicting attributes hence the nature of the problem is multi-criteria. Because of this, the problem is usually addressed through multi-criteria algorithms. The majorities of these algorithms do not generate a solution, but a set of solutions that are on the Pareto frontier; good decision must be made on the set of solutions shown to favor society. The DM chooses the final solution based on his preferences, and a justification for the solution is needed to verify that those preferences are fulfilled within the solution.

3 Rough Set Theory

Rough set is a mathematical tool to approach imperfect and vague knowledge [5], and share common features with the Dempster-Shafer theory of evidence [6]. Rough sets are complements of fuzzy sets; they address imprecision in different ways.

In the rough set theory there is no need of additional information for the original data, is based on the supposition that in the data there are implicit additional knowledge about a universe of discourse. The indiscernibility relation is the basis of the rough set theory; it is generated by the characterization of the objects, if they have the same attributes with the available information, they are indiscernible objects.

The knowledge about the universe is formed from elementary sets; each set is constructed with the indiscernible objects.

In the rough sets, there are objects that can't be classified accurately, they possibly belong or not belong to the concept of interest. Under this approach, the vagueness of the concepts is replaced with two precise concepts: the lower and upper approximation. The former being all the objects that certainly belongs to the concept, and the latter the objects that possibly belong to the concept. Unlike fuzzy sets, the rough sets don't approach the vagueness with a membership function; instead, it uses a boundary region, which is the difference between the approximations.

The prime applications of the rough set theory are as follow [7]:

- The reduction of attributes,
- Generation of rules,
- Prediction.

3.1 Indiscernibility Relation

The indiscernibility relation expresses the fact that we cannot discern certain objects employing the available information because of the absence of information.

From a simpler view, it is better to define the basics of the theory of rough sets in terms of data [8].

It is assumed that the data about the problem is in the form of an *information system*, which is a data table that contains objects of interest in their rows and columns with attributes, in this particular case of project portfolio, each row are the projects to consider and each column the attributes describing them.

Let the pair $I = (U, A)$ an information system where U is the universe of finite objects and A is the finite set of attributes, such that function $U \rightarrow V_a$ where V_a is the set of attribute values of a . The entries of the table are pairs (x, a) where $x \in U$ and $a \in A$.

For a given information system (U, A) , for any subset B of A there is an equivalence relation associated, called the indiscernibility relation, represented by [8]:

$$\text{“}xI(B)y \text{ if and only if } a(x) = a(y) \text{ for every } a \in B,\text{”}$$

where $a(x)$ is the value of the entry (x, a) on the information system.

If the pair $(x, y) \in I(B)$ then x and y are indiscernible by attributes of B .

3.2 Approximations

The partition built from B , denoted by U/B , is the family of all equivalence classes of $I(B)$, and the block of the partition that contains x is represented by $B(x)$.

In many cases, with the available information is not possible to classify certain objects with certainty, in such cases two operations are defined [8],

$$B^*(X) = \{x \in U : B(x) \cap X \neq \emptyset\} \quad (1)$$

$$B_*(X) = \{x \in U : B(x) \subseteq X\} \quad (2)$$

the upper and lower approximations of X , respectively, the lower approximation being all the objects that certainly belongs to the concept of interest, and the upper approximation are all the objects that possibly belongs to the concept of interest.

It is from these upper and lower approximations that the rough set is defined, the difference between the approximations is called the boundary region of X , expressed by:

$$BN_B(X) = B^*(X) - B_*(X) \quad (3)$$

A set is called rough (inexact) regarding B if $BN_B(X) \neq \emptyset$ and in the other case when $BN_B(X) = \emptyset$ the set is called crisp (exact).

3.3 Decision Rules

The decision rules summarize the relationship between the properties, it is one of the techniques commonly used to present the knowledge gained from an information system, in the information system the set of attributes A can be divided in two subsets, C and D , being the condition and decision attributes respectively, $C, D \subseteq A$.

The decision rules are commonly expressed as logical expressions [9] of the form IF (condition attributes) THEN (decision attribute); such rules are used in machine learning. Decision rules are one of the most used techniques in knowledge representation because of their easy understanding that adapts to the cognitive capacity of humans.

Commonly on the information system exists some attributes that are dispensable, removing them preserves the basic properties of the original table. The subset of attributes B with the same classification power as the original data being that if $B \subseteq A$ and $I(B) = I(A)$ is called a *reduct*.

With a reduction of the attributes, the decision rules can be easier to understand. Due to a reduced number of clauses “IF” in each rule, it’s possible that there are more than one reduct in the information system. The problem of obtaining the minimum reduct is not an easy task, the *minimal reduct* is the one that have a minimal cardinality, finding this reduct minimize further the rules.

4 Proposed Approach

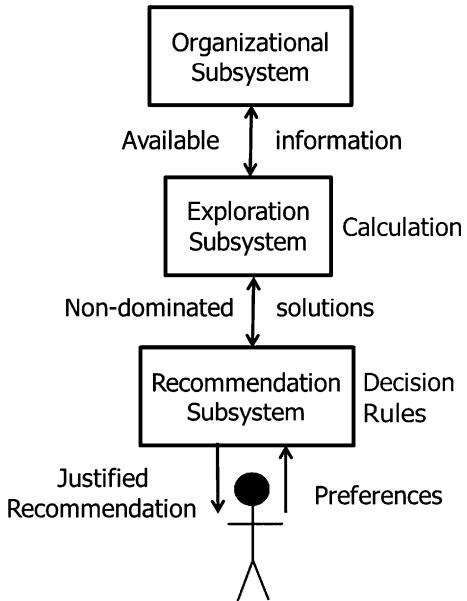
When a portfolio is recommended through an optimization process by multi-criteria algorithms it is important to know how it fits with the policies of the organization. It has a greater impact when more than one portfolio is presented as a recommendation to the DM [10]. It is important to have a basis for explaining why certain projects are or are not in the portfolio; the use of decision rules is proposed to describe the funding policy used in the construction of the portfolio.

Figure 1 shows an overview of a framework proposed to support the decision with decision rules, shows connectivity between the subsystems involved in the referral process, where there is a feedback between the DM and the system.

In the organizational subsystem the necessary information is obtained from the DM, which are the preferences on certain characteristics of the projects and the parameters of the evaluation model. All the collected information serves as input to the exploration subsystem, where based on this information the appropriate algorithm of multi-criteria decision analysis will be selected from a set of algorithms for running instances of problems; the obtained solutions for recommendation will be the input for the recommendation subsystem.

A thorough interaction with the DM is made to help him/her select the solution that best satisfies his preferences between the range of solutions; the recommendation should be justified, using decision rules summarize how the portfolio was

Fig. 1 General scheme of a multicriteria decision analysis process with interaction



constructed. It could be that in the light of the information of how the portfolios are built, the DM decides to change his preferences and then the process restart. This interactivity helps in a self-learning of the DM's preferences.

4.1 Reduction of Attributes

To simplify the decision rules, we calculate the reducts using a genetic algorithm, which is based on the principle of the natural selection and evolution of the species. Besides, we use the concept of distinction table and the fitness function as used by Wroblewski [11]. The distinction table is a binary matrix of dimensions $\frac{(m^2-m)}{2} \times N + 1$, where the first dimension is the size of all the possible combinations of pairs of objects, and the second dimension is the size of the condition attributes plus one decision attribute. An entry of the distinction table is of the form $b(i(j, k))$, correspond to the attribute a_i and a pair of objects to compare (x_j, x_k) :

for $i \in \{1 \dots N\}$:

$$\begin{aligned}
 b((j, k), i) &= \begin{cases} 1, & \text{if } a_i(x_j) \neq a_i(x_k) \\ 0, & \text{if } a_i(x_j) = a_i(x_k) \end{cases} \\
 b((j, k), N + 1) &= \begin{cases} 1, & \text{if } a_i(x_j) = a_i(x_k) \\ 0, & \text{if } a_i(x_j) \neq a_i(x_k) \end{cases}
 \end{aligned} \tag{4}$$

The representation that is chosen for the chromosome is a binary representation, constructed from Eq. (4), where the 1 s numbers identify the attributes that are part of the reduct and 0 s attributes that will be removed, the initial population is chosen randomly on the distinction table.

We calculate the fitness function of the current population:

$$F(r) = \frac{N - L_r}{N} + \frac{C_r}{K} \quad (5)$$

where N is the number of attributes, L_r is the number of 1 s on the candidate, C_r are the covered rows of the table and K the number of all possible combinations of pair of objects; The first part of Eq. (5) gives the candidate credit for containing fewer attributes (fewer 1's) and the second part of the determines the degree to which the candidate can distinguish between objects.

The population is ordered in descending order. For the next population, the first father is the first in the list of the current generation and the second parent is randomly selected for a crossover using one crossing point. The offspring chromosome may mutate with bit flip with a probability of 0.05; if the fitness function of their children is better than the parents, they are replaced. For the next offspring, the first parent is the second on the list and the second is a random parent, and so on. The chromosome with the best fitness at the end of the generations will provide the reduct.

Experimentation to Measure the Reduction Efficiency To ensure the quality of the reducts, we consider experiments with decision tables taken from the UCI Repository of Machine Learning Databases [12]. The results show the accuracy of classifiers based only in the attributes of the reducts against using all the attributes from the databases.

The results are shown on Table 1, where originals mean all the attributes from the databases and reduced are only the attributes of the reducts, size is the number of attributes, we can see from the results than in some cases the accuracy of the reduct is higher than the original, due to some noise on the data, that the rough set methodology can suppress, on other cases there was a reduction in size and a compromise on the accuracy, depends on the DM if this compromise is reasonable.

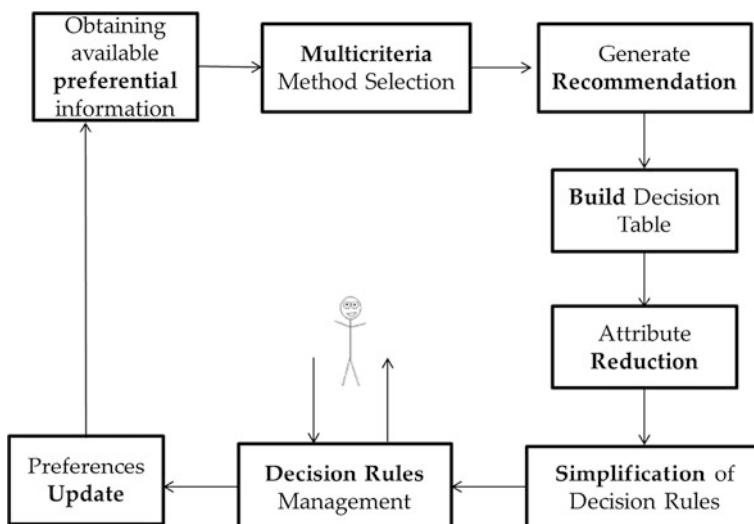
4.2 Proposed Framework

A more specific scheme is shown in Fig. 2, specifying in more detail the obtaining of decision rules in the recommendation subsystem for public project portfolio selection using rough set theory.

First we have to obtain the preferential information available from the DM, this is necessary for the selection of the appropriate multicriteria method. The selected

Table 1 Accuracy of the generated results

Instance	Attributes			
	Originals		Reduced	
	Accuracy	Size	Accuracy	Size
Balance	0.7674	4	0.7674	4
Balloons (a + s)	1.00	4	1.00	2
Balloons (y - s + a - s)	0.625	4	0.562	3
Breast cancer	0.7552	9	0.6678	4
Car	0.9236	6	0.9322	5
Flags	0.7113	26	0.6443	5
Hayes-roth	0.8030	4	0.8030	3
Lenses	0.8333	4	0.875	3
Lymphography	0.7702	18	0.7635	4
Monks1	0.9652	6	1.00	3
Monks2	0.6712	6	0.6712	5
Monks3	1.00	6	1.00	3
Shuttle-landing	0.40	6	0.40	2
Soybean-small	0.9787	35	0.9574	3
Spect	0.8314	22	0.8277	16
Tic-tac-toe	0.8455	9	0.7995	6
Vote	0.9632	16	0.9494	7

**Fig. 2** Proposed framework for public project portfolio with decision rules

method will explore the possible solutions, on the Pareto front, that are more adequate with the DM preferences to provide us with a recommended solution. Here is where the decision rules are going to be used, in a justification between the DM and the framework to explain why the solution satisfies the preferences of the DM.

Throughout the interaction, with the information of the explanation there is a possibility for the DM to update his preferences and the process need to be restarted in order to obtain the new recommendation.

4.3 Decision Rules to Summarize Information

The decision rules are used to summarize the information on the construction of the portfolio to the DM, they are not used to create knowledge; For example, Litvinchev et al. [10] uses an instance of 25,000 projects to generate 61 rules on the construction of one portfolio, some of the rules are presented in Table 2, it uses four condition attributes: m for the minimal funding of the project, M for the maximal funding, A for the knowledge area and w is the evaluation of the project.

Based on the rules obtained, one can summarize the following respecting the portfolio:

- (a) Without considering their impact, the inexpensive projects are supported.
- (b) Projects with a low impact level that are expensive or more are rejected.
- (c) The preference on the order is sustained upon project evaluation and project areas.
- (d) There is a tendency to support the projects that have a high impact, regardless if those are very expensive.

With this information is possible to explain how the construction of the portfolio is made and summarize the decision policy, to see if these construction satisfy the DM beliefs.

Table 2 Example of decision rules from Litvinchev experimentation of 25,000 projects

m	M	Area	w	Decision
Very expensive	Very expensive	Any	Low or worst	Reject projects with these attributes
Expensive	Very expensive	Any	Worst	Reject most projects with these attributes, exceptionally support some
Standard	Expensive	2, 3, 4	Worst	Preferably reject projects with these attributes, but it is acceptable to support some
Expensive	Expensive	Any	Worst	Preferably reject projects with these attributes, but it is acceptable to support some
Expensive	Very expensive	1, 2	At least low	Support all projects with these attributes with maximum

5 Conclusions and Future Work

It is proposed mainly to address the problem of generating recommendations to the DM, such that allows him to choose a portfolio satisfactory under conditions of uncertainty, incompleteness and subjectivity among others.

Under these conditions, it is required a model of effective explanations that fits the nature of the problem and the users. We believe that simple decision rules can aid in the explanations/justifications for the portfolio selection.

Current work in progress is the calculation of reducts by rows; the decision rules can be further reduced by means of a simultaneous reduction of rows and columns.

Future work is to merge of the rough set theory (simple decision rules) with the argumentation theory, which is a formal way of model interactivity with dialogue, between a system and a user.

Future work also requires refining the proposed framework with new methods: (1) a lattice of multi-criteria evaluation methods with its communication protocols; (2) a recommendation method based on argumentation theory. The main motivation behind our work is to present to the DM a final recommendation with justifications for the public project portfolio selection.

Acknowledgments This work was partially financed by CONACYT, COTACYT, DGEST, TECNM, and ITCM.

References

1. Labreuche, C., Maudet, N., Ouerdane, W.: Minimal and complete explanations for critical multi-attribute decisions. In: Algorithmic Decision Theory, pp. 121–134. Springer, Berlin, Heidelberg (2011)
2. Ouerdane, W., Dimopoulos, Y., Liapis, K., Moraitis, P.: Towards automating decision aiding through argumentation. *J. Multi Criteria Decis.* (2011)
3. Fernandez, E., Navarro, J.: A genetic search for exploiting a fuzzy preference model of portfolio problems with public projects. *Ann. Oper. Res.* **117**(1–4), 191–213 (2002)
4. Fernández-González, E., Vega-Lopez, I., Navarro-Castillo, J.: Public portfolio selection combining genetic algorithms and mathematical decision analysis. In: Bio-Inspired Computational Algorithms and Their Applications, pp. 139–160 (2012)
5. Pawlak, Z.: Rough sets present state and further prospects. *Intell. Autom. Soft Comput.* **2**(2), 95–101 (1996)
6. Skowron, A., Grzymal, J.: From rough set theory to evidence theory. In: Yager, R.R., Kacprzyk, J., Fedrizzi, M. (eds.) *Advances in the Dempster-Shafer theory of evidence*, pp. 193–236. Wiley, New York (1994)
7. Düntsch, I., Gediga, G.: Rough set data analysis. *Encycl. Comput. Sci. Technol.* **43**(28), 281–301 (2000)
8. Pawlak, Z., Skowron, A.: Rudiments of rough sets. *Info. Sci.* **177**(1), 3–27 (2007)
9. Stefanowski, J.: Changing representation of learning examples while inducing classifiers based on decision rules. *Artificial Intelligence Methods, AI-METH* (2003)

10. Litvinchев, I.S., López, F., Alvarez, A., Fernández, E.: Large-scale public R&D portfolio selection by maximizing a biobjective impact measure. *Syst. Man Cybern. Part A Syst. Hum. IEEE Trans.* **40**(3), 572–582 (2010)
11. Wroblewski, J.: Finding minimal reducts using genetic algorithms. In: Proceedings of the 2nd annual join conference on information science, pp. 186–189, Sept, 1995
12. Newman, D.J., Hettich, S., Blake, C.L., Merz, C.J.: UCI repository of machine learning databases. University of California, Irvine, Department of information and computer sciences (1998), <http://www.ics.uci.edu/~mlearn/MLRepository.html>

A Survey of Grey Systems Applied to Multi-objective Problem

Fausto Balderas, Eduardo Fernandez, Claudia Gómez and Laura Cruz-Reyes

Abstract Decision-making processes involve a series of steps: identifying the problems, constructing the preferences, evaluating the alternatives, and determining the best alternatives (Simon, The new science of management decision. In: Proceedings of the 33rd Conference of the Operational Research Society of New Zealand, Citeseer, 1960). Using a multi-criteria decision making approach, the grey system theory has been used to capture the complexity inherent in selection process. The grey system theory proposed by Deng (J. Grey Syst. 1(1), 1–24 1989) is based on the assumption that a system is uncertain and that the information regarding the system is insufficient to build a relational analysis or to construct a model to characterize the system. The aim of this chapter is to provide a short review of the three mostly seen research methods employed for the investigation of uncertain systems: probability and statistics, fuzzy mathematics and grey systems theory. This chapter provides a short review of the general framework, current research trends and future research topics on grey systems applied to decision problems.

1 Introduction

The fundamental characteristic of uncertain systems is the in-completeness and inadequacy in their information. Due to the dynamics of system evolutions, the biological limitations of the human sensing organs, and the constraints of relevant

F. Balderas · C. Gómez · L. Cruz-Reyes (✉)

Tecnológico Nacional de México, Instituto Tecnológico de Ciudad Madero, Madero,
TAMPS, Mexico

e-mail: cruzreyeslaura@gmail.com

F. Balderas

e-mail: fabj21@gmail.com

C. Gómez

e-mail: cggs71@hotmail.com

E. Fernandez

Universidad Autónoma de Sinaloa, Sinaloa, Mexico

e-mail: eddyf@uas.edu.mx

economic conditions and technological availabilities, uncertain systems exist commonly [1].

The grey systems theory, established by Julong Deng in 1982, is a new methodology that focuses on the study of problems involving small samples and poor information (Deng 1982). It deals with uncertain systems with partially known information through generating, excavating and extracting useful information from what is available. So, systems' operational behaviors and their laws of evolution can be correctly described and effectively monitored. In the natural world, uncertain systems with small samples and poor information exist commonly. That fact determines the wide range of applicability of grey systems theory [1].

2 Definitions and Background Concepts

This section presents the basic concepts in multi-objective optimization and we give some basics definitions of the grey system, grey set and grey number in grey theory, also about project portfolio problem.

2.1 Multi-objective Optimization

Real-world optimization problems are extremely complex with many attributes to evaluate and multiple objectives to optimize [2, 3]. The attributes are expressed in terms of decision variables, they described the problem and correspond to quantitative values. The objectives can be to maximize or minimize they are the direction for improvement the attributes.

In many cases, due to the conflicting nature of attributes is not possible to obtain a single solution and therefore the ideal solution for a multi-objective problem (MOP) cannot be achieved because there is no one solution the problem. Typically, solving a MOP has a set of solutions that reached an aspiration level expected by the DM [2].

Definition 1 Multi-objective optimization problem

Given a vector function and its feasible solution space, the MOP consists in find a vector that optimizes the vector function.

Definition 2 Pareto dominance

A vector $f(\vec{x})$ dominates $f(\vec{y})$ (denoted by $f(\vec{x}) \prec f(\vec{y})$) if the following conditions apply:

1. The solution “ $f(\vec{x})$ is better than $f(\vec{y})$ ” in at least one of the objectives of the criteria:

$$f_j(\vec{x}) > f_j(\vec{y}), \quad \exists j \in [1, 2, \dots, k] \quad (1)$$

2. The solution “ $f(\vec{x})$ ” is no worse than “ $f(\vec{y})$ ” in other objectives:

$$f_i(\vec{x}) \geq f_i(\vec{y}), \quad \forall i \in [1, 2, \dots, k] \quad y \quad (2)$$

Definition 3 Pareto optimal

A vector \vec{x}^* is Pareto optimal if not exists a vector $\vec{y} \in \Omega$ such that $f(\vec{y}) \prec f(\vec{x}^*)$.

Definition 4 Pareto optimal set

Given a MOP, the Pareto optimal set is defined as $P^* = \{\vec{x}^* \in \Omega\}$.

Definition 5 Pareto front

Given a MOP and its Pareto optimal set P^* , the Pareto front is defined as $PF^* = \left\{ \vec{f}(\vec{x}^*) \mid \vec{x} \in P^* \right\}$.

2.2 Grey System Theory

Definition 1 A grey system is defined as a system containing uncertain information presented by a grey number and grey variables.

Definition 2 Such a number instead of its range whose exact value is unknown is referred to as a grey number. In applications, a grey number in fact stands for an indeterminate number that take its possible value within an interval or a general set of numbers. This grey number is generally written using the symbol “ \otimes ”. There are several types of grey numbers.

Definition 2.1 *Grey numbers with only a lower bound:* This kind of grey number \otimes is written as $\otimes \in [\underline{a}, \infty)$ or $\otimes (\underline{a})$, where \underline{a} stands for the definite, known lower bound of the grey number \otimes . The interval $[\underline{a}, \infty)$ is referred to as the field of \otimes .

Definition 2.2 *Grey numbers with only an upper bound:* This kind of grey number \otimes is written as $\otimes \in (\infty, \bar{a}]$ or $\otimes (\bar{a})$, where a stands for the definite, known upper bound of \otimes .

Definition 2.3 *Interval grey numbers:* This kind of grey number \otimes has both a lower a and an upper bound \bar{a} , written $\otimes \in [\underline{a}, \bar{a}]$.

Definition 2.4 *Continuous and discrete grey numbers:* A grey number that only takes a finite number or a countable number of potential values is known as discrete. If a grey number can potentially take any value within an interval, then it is known as continuous.

Definition 2.5 *Black and white numbers:* When $\otimes \in (-\infty, +\infty)$, that is, when \otimes has neither any upper nor lower bound, then \otimes is known as a black number. When $\otimes \in [\underline{a}, \bar{a}]$ and $\underline{a} = \bar{a}$, \otimes is known as a white number.

Definition 2.6 Essential and non-essential grey numbers: The former stands for such a grey number that temporarily cannot be represented by a white number; and the latter such grey numbers each of which can be represented by a white number obtained either through experience or certain method. The definite white number is referred to as the whitenization (value) of the grey number, denoted $\tilde{\otimes}$. Also, we use \otimes (a) to represent the grey number(s) with a as its whitenization.

Definition 3 Grey number operation is an operation defined on sets of intervals, rather than real numbers. The modern development of interval operation began with R.E. Moore's dissertation [4]. We cite literatures [5, 6] to define the basic operation laws of grey numbers $\otimes a_1 = [\underline{a}_1, \bar{a}_1]$ and $\otimes a_2 = [\underline{a}_2, \bar{a}_2]$, on intervals where the four basic grey number operations on the interval are the exact range of the corresponding real operation.

$$\begin{aligned}\otimes a_1 + \otimes a_2 &= [\underline{a}_1 + \underline{a}_2, \bar{a}_1 + \bar{a}_2] \\ \otimes a_1 - \otimes a_2 &= [\underline{a}_1 - \bar{a}_2, \bar{a}_1 - \underline{a}_2] \\ \otimes a_1 \times \otimes a_2 &= [\min(\underline{a}_1 \underline{a}_2, \underline{a}_1 \bar{a}_2, \bar{a}_1 \underline{a}_2, \bar{a}_1 \bar{a}_2), \max(\underline{a}_1 \underline{a}_2, \underline{a}_1 \bar{a}_2, \bar{a}_1 \underline{a}_2, \bar{a}_1 \bar{a}_2)] \\ \otimes a_1 \div \otimes a_2 &= [\underline{a}_1, \bar{a}_1] \times \left[\frac{1}{\underline{a}_2}, \frac{1}{\bar{a}_2} \right]\end{aligned}$$

Definition 4 The length of grey number $\otimes G$ is defined as

$$L(\otimes a) = [\bar{a}_1, \underline{a}_2]$$

Definition 5 Comparison of grey numbers

Shi et al. [7] proposed a degree of grey possibility to compare the ranking of grey numbers.

For two grey numbers $\otimes a_1 = [\underline{a}_1, \bar{a}_1]$ and $\otimes a_2 = [\underline{a}_2, \bar{a}_2]$, the possibility degree of $\otimes a_1 \leq \otimes a_2$ can be expressed as follows:

$$P(\otimes a_1 \leq \otimes a_2) = \frac{\max(0, L^* - \max(0, \bar{a}_1 - \underline{a}_2))}{L^*}$$

For the position relationship between $\otimes a_1$ and $\otimes a_2$, there exist four possible cases on the real number axis. The relationship between $\otimes a_1$ and $\otimes a_2$ is determined as follows:

1. If $\underline{a}_1 = \underline{a}_2$ and $\bar{a}_1 = \bar{a}_2$, we say that $\otimes a_1$ is equal to $\otimes a_2$, denoted as $\otimes a_1 = \otimes a_2$. Then $P(\otimes a_1 \leq \otimes a_2) = 0.5$.
2. If $\underline{a}_2 > \bar{a}_1$, we say that $\otimes a_2$ is larger than $\otimes a_1$, denoted as $\otimes a_2 > \otimes a_1$. Then $P(\otimes a_1 \leq \otimes a_2) = 1$.
3. If $\bar{a}_2 < \underline{a}_1$, we say that $\otimes a_2$ is smaller than $\otimes a_1$, denoted as $\otimes a_2 < \otimes a_1$. Then $P(\otimes a_1 \leq \otimes a_2) = 0$.

4. If there is an intercrossing part in them, when $P(\otimes a_1 \leq \otimes a_2) > 0.5$, we say that $\otimes a_2$ is larger than $\otimes a_1$, denoted as $\otimes a_2 > \otimes a_1$. When $P(\otimes a_1 \leq \otimes a_2) < 0.5$, we say that $\otimes a_2$ is smaller than $\otimes a_1$, denoted as $\otimes a_2 < \otimes a_1$.

2.3 Portfolio Problem

A project is a temporary, unique and unrepeatable process which pursues a specific set of objectives [8]. In this work, it is not considered that the projects can be broken down into smaller units such as tasks or activities. In other words, a project cannot be divided to run only a part, however, different versions of the same project can be proposed, each version may vary in amount of activity, time required and requested resources.

A portfolio consists of a set of projects that can be performed in the same period of time [8]. For this reason, projects in the same portfolio share available resources in the organization. They can complement each other, which is called synergy. Thus, it is not sufficient to compare the projects individually, but must compare groups of projects to identify what portfolio makes the greatest contribution to the organization objectives.

The proper selection of projects to integrate the portfolio, which will receive the organization's resources, is one of the most important decision problems for both public and private institutions [9, 10]. The main economic and mathematical models to the portfolio problem assume that there is a defined set of n projects, each project well characterized with costs and revenues, of which the distribution over time is known. The Decision Maker (DM) is responsible for selecting the portfolio that the company will implement [11].

3 Solution Techniques

Probability and statistics, fuzzy mathematics, and grey systems theory are three mostly seen research methods employed for the investigation of uncertain systems. Their research objects all contain certain kinds of uncertainty, which represents their commonality. It is exactly the differences among the uncertainties in the research objects that these three theories of uncertainty are different from each other with their respective characteristics.

3.1 Probability and Statistics

Study the phenomena of stochastic uncertainty with emphasis placed on revealing the historical statistical laws. They investigate the chance for each possible outcome of the stochastic uncertain phenomenon to occur. Their starting point is the availability of large samples that are required to satisfy a certain typical form of distribution.

3.2 Fuzzy Mathematics

Fuzzy mathematics emphasizes on the investigation of problems with cognitive uncertainty, where the research objects possess the characteristic of clear intension and unclear extension. For instance, “little money” is a fuzzy concept, because each person knows the intension of “little money.” However, if you are going to determine the exact range within which is little money and outside which is not little money, then you will find yourself in a great difficulty. That is because the concept of little man does not have a clear extension. For this kind of problem of cognitive uncertainty with clear intension and unclear extension, the situation is dealt with in fuzzy mathematics by making use of experience and the so-called membership function.

3.3 Grey System Theory

The focus of grey systems theory is on the uncertainty problems of small samples and poor information that are difficult for probability and fuzzy mathematics to handle. It explores and uncovers the realistic laws of evolution and motion of events and materials through information coverage and through the works of sequence operators. One of its characteristics is construct models with small amounts of data.

4 Current Research on Grey System Theory

This section reviews the current research trends on grey systems theory.

As far as we investigated, we found few studies that mention.

4.1 Developing a Grey-Based Decision-Making Model for Supplier Selection [12]

Supplier selection is widely considered to be one of the most important responsibilities of management. If we have different criteria including conflicting criteria such as quality and price it creates a problem more difficult to the decision maker.

4.2 Application of Gray Systems and Fuzzy Sets in Combination with Real Options Theory in Project Portfolio Management [13]

They consider all possible uncertain conditions that a project can encounter and express their methods for each of these situations. They used two applicable approaches in different situations of uncertainty.

Their proposed methods in terms of computational time are fast, yet also by using genetic algorithms or data development analysis the performance of portfolios that satisfy the restrictions could be improved.

4.3 Ranking Grey Numbers Based on Dominance Grey Degrees [14]

The concept of dominance grey degree is defined, and then a method of ranking interval grey numbers based on the dominance grey degree is proposed.

The method proposed can more adequately describe the relative importance of the target objects, providing an effective total-order technology for multiple-attribute ranking decision making with a wide range of application prospects in the research of decision making with uncertain information.

4.4 On the Optimization Concept of Grey Systems [15]

They define the concept of gray optimization, through theoretical analysis and numerical examples, it is demonstrated that the proposed concept is reasonable and meaningful, but they only consider the single-objective problem.

5 Future Research on Grey System Theory

This section presents the authors perspective about what is still need to be researched in Grey Systems Applied to Multi-objective Problem.

An interval solution for decision making is more appropriate, because instead of a certain number, a range of numbers is given.

For this kind of problems where exists different criteria, the decision maker should determine, along with some uncertainty the inputs and decisions.

Despite the methodologies and techniques, they are still subject to limitations and are not able to fully capture the complexity of the decision process.

What would be of interest, it would take fully carried out for all the inaccuracies that exist, that means modeling imprecision portfolio problem with gray philosophy. To model the imprecision of the project portfolio selection problem with grey systems theory.

To focus on fully working on all kinds of uncertainty presented, by example, uncertainty in the results of the projects, (Was used probability distributions. It is the most common.), uncertainty in the resources that the projects needs (Similar to Partial Support. Projects do not need exactly the resources they require. An attempt has been with fuzzy sets.), and uncertainty in the time of the projects (as far as we have looked, we could not find any work related to this).

References

1. Liu, S., Lin, Y., Forrest, J.Y.L.: Grey Systems: Theory and Applications, vol. 68. Springer, Berlin (2010)
2. Coello Coello, C.A., Lamont, G.B., Van Veldhuizen, D.A.: Evolutionary Algorithms for Solving Multi-objective Problems. Genetic and Evolutionary Computation, 2nd edn. Springer, Berlin (2007)
3. Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs. Springer Series Artificial Intelligence, 3rd edn. Springer, New York (1996)
4. Moore, R.E.: Interval Analysis. Prentice-Hall, Englewood Cliffs (1966)
5. Wang, Q., Wu, H.: The concept of grey number and its property. In: Proceedings of NAFIPS, USA, pp. 45–49 (1998)
6. Wu, Q., Zhou, W., Li, S., Wu, X.: Application of grey numerical model to groundwater resource evaluation. Environ. Geol. **47**, 991–999 (2005)
7. Shi, J.R., Liu, S.Y., Xiong, W.T.: A new solution for interval number linear programming. J. Syst. Eng. Theor. Pract. **2**, 101–106 (2005). (in Chinese)
8. Carazo, A.F., Gómez, T., Molina, J., Hernández-Díaz, A.G., Guerrero, F.M., Caballero, R.: Solving a comprehensive model for multiobjective project portfolio selection. Comput. Oper. Res. **37**(4), 630–639 (2010)
9. Castro, M.: Development and implementation of a framework for I&D in public organizations. Master's thesis, Universidad Autónoma de Nuevo León (2007)
10. García, R.: Hyper-Heuristic for solving social portfolio problem. Master's thesis, Instituto Tecnológico de Cd. Madero (2010)

11. Fernández, E., López, E., Bernal, S., Coello Coello, C.A., and Navarro, J.: Evolutionary multiobjective optimization using an outranking—based dominance generalization. *Comput. Oper. Res.* **37**(2), 390–395 (2010)
12. Golmohammadi, D., Mellat-Parast, M.: Developing a grey-based decision-making model for supplier selection. *Int. J. Prod. Econ.* **137**(2), 191–200 (2012)
13. Arasteh, A., Aliahmadi, A., Omran, M.M.: Application of gray systems and fuzzy sets in combination with real options theory in project portfolio management. *Arab. J. Sci. Eng.* **39**(8), 6489–6506 (2014)
14. Liu, Y., Forrest, J., Xie, N.: Ranking grey numbers based on dominance grey degrees. *Syst. Eng. Electron. J.* **25**(4), 618–626 (2014)
15. Deb, K.: *Multi-Objective Optimization using Evolutionary Algorithms*. Wiley, Chichester (2001)
16. Deng, J.L.: The introduction of grey system. *J. Grey Syst.* **1** (1), 1–24 (1989)
17. Simon, S.M.D.: The new science of management decision. In: Proceedings of the 33rd Conference of the Operational Research Society of New Zealand, Citeseer, (1960)

An Efficient Representation Scheme of Candidate Solutions for the Master Bay Planning Problem

**Paula Hernández Hernández, Laura Cruz-Reyes, Patricia Melin,
Julio Mar-Ortiz, Héctor Joaquín Fraire Huacuja,
Héctor José Puga Soberanes and Juan Javier González Barbosa**

Abstract The master bay planning problem (MBPP) arises in the context of maritime transportation. In particular, MBPP consists of determining an efficient plan to stowage the containers into the containership such that the total loading time is minimized. This problem is classified as NP-hard due to the large number of possible solutions generated by the combination of assigning containers to locations in the containership. These solutions are both feasible and infeasible, which increases even more the hardness of MBPP. To deal with this problem, there are several exact and heuristic approaches that are well documented in the literature. One of the most important exact methods is in the form of an integer linear programming (ILP) formulation. However, the number of variables and constraints generated by this ILP model is very large. In this chapter, we propose a new exact algorithm based on a branch and bound (B&B) approach. The main feature is the

P.H. Hernández (✉) · L. Cruz-Reyes · H.J.F. Huacuja · J.J.G. Barbosa
Tecnológico Nacional de México, Instituto Tecnológico de Ciudad Madero,
Ciudad Madero, T.A.M.P.S., Mexico
e-mail: paulahdz314@hotmail.com

L. Cruz-Reyes
e-mail: lauracruzreyes@itcm.edu.mx

H.J.F. Huacuja
e-mail: automatas2002@yahoo.com.mx

J.J.G. Barbosa
e-mail: jjgonzalezbarbosa@hotmail.com

P. Melin
Tecnológico Nacional de México, Instituto Tecnológico de Tijuana,
Tijuana, B.C., Mexico
e-mail: pmelin@tectijuana.edu.mx

J. Mar-Ortiz
Universidad Autónoma de Tamaulipas, Tampico, T.A.M.P.S., Mexico
e-mail: jmar@uat.edu.mx

H.J.P. Soberanes
Tecnológico Nacional de México, Instituto Tecnológico de León, Leon, Mexico
e-mail: pugahector@yahoo.com

usage of an efficient representation structure of candidate solutions. We test the proposed B&B on a set of small-sized instances. Experimental results demonstrate that, within this set of instances, our B&B is competitive with respect to the ILP model from the literature.

1 Introduction

The Master Bay Planning Problem (MBPP) is one of the relevant problems involved in the efficient operation of ports.

MBPP is an NP-hard combinatorial optimization whose goal is to find optimal plans for stowing containers into a containership with low operational costs, subject to a set of structural and operational constraints [1, 2].

There are several exact and heuristics approaches to tackle this problem documented in the literature [3–5]. One of the most important exact methods is in the form of an integer linear programming (ILP) formulation [1], based in the binary representation scheme. However, the number of variables and constraints generated by this ILP model is very large.

In this chapter, we describe two types of scheme to represent a candidate solution for the master bay planning problem. The first one is the binary representation whereas the second scheme is based on the so-called *partial permutations*.

This chapter is organized into six sections. Section 2 describes MBPP. Section 3 describes two types of scheme to represent a candidate solution for MBPP. Section 4 presents our proposed Branch-and-Bound based on *partial permutation* representation scheme. Section 5 shows the experimental results. Finally, Sect. 6 shows the conclusions.

2 Description of Master Bay Plan Problem

MBPP is an NP-hard combinatorial optimization problem. In Avriel et al. [2] the demonstration of the complexity is presented. This problem consists of stowing a set of containers into a containership and can be defined as follows [6]:

Given a set C of n containers of different types to be loaded on the ship and a set S of m available locations on the containership, we have to determine the assignment of each container to a location of the ship, in such a way, to satisfy all the given structural and operational constraints related to ship and containers, and to minimize the total stowage time.

In MBPP, each container $c \in C$ must be stowed in a location $l \in S$ of the ship. The l -th location is actually addressed by the indices i, j, k representing, respectively: the bay (i), the row (j), and the tier (k). We denote by I, J and K , respectively, the set of bays, rows and tiers of the ship, and by b, r and s their corresponding cardinality.

The objective function is expressed in terms of the sum of the time t_{lc} required for loading a container c , $\forall c \in C$, in location l , $\forall l \in S$, such that $L = \sum_{lc} t_{lc}$. However, when two or more quay cranes are used for the loading operations the objective function is given by the maximum over the minimum loading time (L_q) for handling all containers in the corresponding ship partition by each quay crane q , that is, $L = \max_{QC} \{L_q\}$, where QC , is the set of available quay cranes.

The main constraints that must be considered for the stowage planning process for an individual port are related to the structure of the ship and focused on the size, type, weight, destination and distribution of the containers to be loaded. The description of each of them will be found below.

Size of containers. Usually, set C of containers is considered as the union of two subsets, T and F , consisting, respectively, of 20 and 40 ft containers, such that $T \cap F = \emptyset$ and $T \cup F = C$. Containers of 40 ft require two contiguous locations of 20 ft. Note that according to a practice adopted by the majority of maritime companies, bays with even number are used for stowing 40' containers and correspond to two contiguous odd bays that are used for the stowage of 20' containers. Consequently, if a 40' container is stowed in an even bay (for instance bay 02) the locations of the same row and tier corresponding to two contiguous odd bays (e.g. bay 01 and bay 03) are not anymore available for stowing containers of 20'.

Type of containers. Different types of containers can usually be stowed in a containernesship, such as standard, carriageable, reefer, out of gauge and hazardous. The location of reefer containers is defined by the ship coordinator (who has a global vision of the trip), so that we know their exact position.

Weight of containers. The standard weight of an empty container ranges from 2 to 3.5 tons, while the maximum weight of a full container to be stowed in a containernesship ranges from 20 to 32 and 30 to 48 tons for 20' and 40' containers, respectively. The weight constraints force the weight of a stack of containers to be less than or equal to a given tolerance value. In particular, the weight of a stack of 3 containers of 20' and 40' cannot exceed an a priori established value, say MT and MF , respectively. Moreover, the weight of a container located in a tier cannot be greater than the weight of the container located in a lower tier having the same row and bay and, as it is always required for security reasons; both 20' and 40' containers cannot be located over empty locations.

Destination of containers. A good general stowing rule suggests to load first (i.e., in the lower tiers) those containers having as destination the final stop of the ship and load last those containers that have to be unloaded first.

Distribution of containers. Such constraints, also denoted stability constraints, are related to a proper weight distribution in the ship. In particular, we have to verify different kinds of equilibrium, namely:

- Horizontal equilibrium: the weight on the right side of the ship, including the odd rows of the hold and upper deck, must be equal (within a given tolerance, say Q_1) to the weight on the left side of the ship.
- Cross equilibrium: the weight on the stern must be equal (within a given tolerance, say Q_2) to the weight on the bow.

- Vertical equilibrium: the weight on each tier must be greater or equal than the weight on the tier immediately over it. Let us denote by L and R , respectively, the set of rows belonging to the left/right side of the ship and by A and P , respectively, the sets of anterior and posterior bays of the ship.

In the next section, the 0/1 Linear Programming Model proposed by Ambrosino et al. [1] is presented. They assume that all containers are ready to be loaded on the quay without considering their stock position in the yard.

2.1 Basic Model of MBPP

In the literature, Ambrosino et al. proposed a binary linear programming formulation [1, 2]. It uses one single type of decision variable, x_{lc} , with $l = 1, \dots, m$ and $c = 1, \dots, n$. Specification of this variable is the following:

$$x_{lc} = \begin{cases} 1 & \text{if a container } c \text{ is stowed in location } l \\ 0 & \text{otherwise} \end{cases},$$

where location l is addressed by its bay (i), row (j) and tier (k), while c identifies the number of the stowed container. This means that variable $x_{lc} = x_{ijkc}$, directly gives the location where container c is stowed, when it is set to 1. Therefore, at the optimal solution we have the exact position of each container in the ship.

The definition of variable x_{lc} , $\forall l \in S$, $c \in C$, enables an easy formulation of the underlying model for MBPP, reported below.

$$\min L = \sum_l \sum_c t_{lc} x_{lc} \quad (1)$$

$$\sum_l \sum_c x_{lc} = n \quad (2)$$

$$\sum_l x_{lc} \leq 1 \quad \forall c \quad (3)$$

$$\sum_c x_{lc} \leq 1 \quad \forall l \quad (4)$$

$$\sum_l \sum_c w_c x_{lc} \leq Q \quad (5)$$

$$\sum_{c \in T} x_{ijkc} = 0 \quad \forall i \in E, j \in J, k \in K \quad (6)$$

$$\sum_{c \in F} x_{ijkc} = 0 \quad \forall i \in O, j \in J, k \in K \quad (7)$$

$$\sum_{c \in T} x_{i+1jkc} + \sum_{c \in F} x_{ijkc} \leq 1 \quad \forall i \in E, j \in J, k \in K \quad (8)$$

$$\sum_{c \in T} x_{i-1jkc} + \sum_{c \in F} x_{ijkc} \leq 1 \quad \forall i \in E, j \in J, k \in K \quad (9)$$

$$\sum_{c \in T} x_{i+1jk+1c} + \sum_{c \in F} x_{ijkc} \leq 1 \quad \forall i \in E, j \in J, k = 1, \dots, |K| - 1 \quad (10)$$

$$\sum_{c \in T} x_{i-1jk+1c} + \sum_{c \in F} x_{ijkc} \leq 1 \quad \forall i \in E, j \in J, k = 1, \dots, |K| - 1 \quad (11)$$

$$\begin{aligned} \sum_{c \in T} w_c x_{ijkc} + \sum_{c \in T} w_c x_{ijk+1c} + \sum_{c \in T} w_c x_{ijk+2c} &\leq MT \\ \forall i \in I, j \in J, k = 1, \dots, |K| - 2 \end{aligned} \quad (12)$$

$$\begin{aligned} \sum_{c \in F} w_c x_{ijkc} + \sum_{c \in F} w_c x_{ijk+1c} + \sum_{c \in F} w_c x_{ijk+2c} &\leq MF \\ \forall i \in I, j \in J, k = 1, \dots, |K| - 2 \end{aligned} \quad (13)$$

$$\sum_c w_c x_{ijkc} - \sum_c w_c x_{ijk+1c} \geq 0 \quad \forall i \in I, j \in J, k = 1, \dots, |K| - 1 \quad (14)$$

$$\sum_c d_c x_{ijkc} - \sum_c d_c x_{ijk+1c} \geq 0 \quad \forall i \in I, j \in J, k = 1, \dots, |K| - 1 \quad (15)$$

$$-Q_2 \leq \sum_{i \in A, j, k} \sum_c w_c x_{ijkc} - \sum_{i \in P, j, k} \sum_c w_c x_{ijkc} \leq Q_2 \quad (16)$$

$$-Q_1 \leq \sum_{i, j \in L, k} \sum_c w_c x_{ijkc} - \sum_{i, j \in R, k} \sum_c w_c x_{ijkc} \leq Q_1 \quad (17)$$

$$x_{lc} \in \{0, 1\} \quad \forall l, c \quad (18)$$

Equation (1) is the objective function that minimizes the total stowage time L , expressed in terms of the sum of time t_{lc} required for loading a container c , $\forall c \in C$, in location l , $\forall l \in S$. Note that loading times t_{lc} are expressed in 1/100 of minute and depend on the row and tier address in which the container will be loaded.

Constraint (2) establishes that the number of assigned locations must be equal to the number of given containers. Constraints (3) and (4) are the well-known

assignment constraints forcing each container to be stowed only in one ship location and each location to have at most one container.

The capacity constraint (5) establishes that the total weight of all containers cannot exceed the maximum weight capacity Q of the containership.

Constraints (6)–(11) are the size conditions. In particular, (6) and (7) force, respectively, 40' containers to be stowed in even bays and 20' container to be stowed in odd bays, while (8) and (9) make unfeasible the stowage of 20' containers in those odd bays that are contiguous to even locations already chosen for stowing 40' containers, and inversely; (10) and (11) prevent 20' containers being positioned over 40' ones.

Weight constraints (12) and (13) say that a stack of at most three containers of either 20' or 40' cannot exceed values MT and MF , respectively. They usually correspond to $MT = 45$ and $MF = 66$ tons. Note that such constraints verify the corresponding tolerance value in all occupied tiers in the same row and bay for any possible stack of three containers, as it is required by the weight constraints.

Constraints (14) force heavier containers not to be put over lighter ones. It is worth noting that constraints (14) also avoid the stowing of both 20' and 40' containers over empty locations.

Destination constraints (15) avoid positioning containers that have to be unloaded first below those containers that have a later destination port.

Constraints (16) and (17) are the horizontal and cross equilibrium conditions, stating that the difference in weight between the anterior and the posterior bays and between the left and right side must be at most Q_2 and Q_1 tons respectively. Finally, in (18) the binary decision variables of the problem are defined.

In this formulation is assumed that the ship starts its journey in one port, and successively visits a given number of other ports where only unloading operations are allowed.

Moreover, it is assumed that the number of containers to load on board is not greater than the number of available locations; this means that we are not concerned with the problem of selecting some containers to be loaded among all and that the capacity constraint (5) is only related to the maximum weight available for all containers.

3 Analysis of Representation Schemes of Candidate Solutions

In this section, we describe two types of schemes to represent a candidate solution for the master bay planning problem. The first one is the binary representation, which is commonly used to design mathematical formulations. The second scheme is based on the so-called *partial permutations*.

3.1 Binary Representation Scheme

As stated in Sect. 2.1, there is a binary formulation for MBPP documented in the literature. In this model, each container $c \in C$ is placed into an available particular location $l \in S$. If this assignment occurs, binary variable x_{lc} takes the value of 1, otherwise $x_{lc} = 0$.

The total number of these variables is $|S| \times |C| = m \times n$. Thus, the number of all possible different combinations of zeros and ones is $2^{m \times n}$, which is the size of the solution space SS . This implies that we have to evaluate $2^{m \times n}$ configurations to know the optimal solution for a given instance.

Let us consider a hypothetical instance with $m = 4$ locations and $n = 3$ containers. Under this scheme, we must evaluate $2^{4 \times 3} = 2^{12} = 4096$ different solutions to find the best stowage plan.

On the positive side, this scheme allows designing binary mathematical formulations, as the one presented in the previous section. These formulations can be solved by using a commercial optimization engine. However, they are highly dependent on the strategies incorporated in the solver.

It is important to point out that this scheme might not be suitable for ad hoc algorithms. In consequence, a different representation scheme should be addressed. In the following section we discuss another scheme whose size of solution space is smaller.

3.2 Partial Permutation Representation Scheme

This scheme is designed to reduce the size of the solution space. This might lead to improve the efficiency of a search algorithm. Specifically, we select a subset of n available locations to store the entire set of containers. This is done by using a one dimensional array of size n , whose elements represent locations and its positions containers. Thus, the i -th element denotes the location where the i -th container will be placed. Once a container is assigned to a location, it cannot be reassigned to another location and vice versa.

Since we have m locations, the number of different ways in which n locations can be selected is given by $m!/(m - n)!$, which is the size of the solution space. We called this scheme *partial permutation* because of this property [7].

Let us consider again the previous example with $m = 4$ and $n = 3$. Under this scheme we only have to evaluate $4!/(4 - 3)! = 24$ solutions. This is a considerable reduction with respect to the binary scheme. The reason is that we do not consider solutions that we know a priori to be unfeasible. For example, under the binary scheme, configuration with all variables set to 0 is considered as a solution to be

evaluated. Nevertheless, it obviously violates the assignment constraint [see constraint (2)]. On the other hand, in the partial permutation scheme this configuration is not even admissible.

In order to compare both representation schemes, we have designed a Branch-and-Bound algorithm that uses the *partial permutation* scheme. This algorithm is described in the next section.

4 Branch-and-Bound Algorithm

A Branch-and-Bound (B&B) is an exact algorithm. It models the solution space as a search tree. This tree is modeled in the following way. Each level represents a container $c \in C$ to be allocated. Thus, the i -th level corresponds to the i -th container. Moreover, the number of main subtrees is the number m of available locations. It is easy to see that the search tree is symmetrical when $m = n$ and asymmetrical otherwise.

Visiting a node of the search tree represents the assignation of the corresponding container (given by the level) to the location represented by the current node. For example, visiting node 5 at level 3 means that container 3 is put into location 5. Recall that the location is addressed by indices i, j and k . When B&B visits a leaf node, a complete solution is found. This solution is encoded as a *partial permutation*. However, obtaining a complete solution is not guarantee of feasibility.

We now describe the B&B algorithm, which is shown in Algorithm 1. A Branch-and-Bound requires two bounds to delimit the solution space, namely, the *Upper* and *Lower* bounds. They are progressively tightened on the objective value. At the beginning, the *Upper Bound* (UB) starts with a very large value, i.e., $UB = \infty$ while the *Lower Bound* (LB) with 0.

UB is only updated when (i) B&B is visiting a leaf node, (ii) the objective value of the current solution (branch) is better than UB and (iii) the solution is feasible (lines 3–7). On the other hand, LB is updated iteratively at each visited level. Specifically, we compute LB as the loading times required to stowage the current number of assigned containers (line 9). Thus, when a new location is assigned we only must add its corresponding loading time to the current sum of loading times.

Once LB is computed, we compare it with the upper bound. If $LB < UB$ and the current partial solution is feasible, we perform a branching process (lines 2–18). It consists of selecting available locations S to be added to list of locations to be evaluated U (line 18).

This process is repeated iteratively until all the nodes in the search tree are explored. It is important to point out that we use a Last In First Out (LIFO) stack instead of the recursive process.

Algorithm 1. *B&B Algorithm (x_k , $Node_k$, UB)*

1. Let x_k be the solution up to $Node_k$ and k be the last level assigned. Indeed, if $k < n$ then the solution is not complete.
2. Let S and U be the sets of unassigned and assigned locations, respectively.
3. If ($Node_k$ is a leaf) /*Complete solution*/
 4. Calculate the total objective value $L(x)$
 5. If ($L(x) < UB$)
 6. If x is feasible
 7. $UB = L(x)$
 8. Else
 9. Calculate the lower bound $LB = L(x_k)$
 10. If $LB < UB$
 11. If x_k is feasible
 12. $k = k + 1$
 13. While ($U \neq \{\emptyset\}$)
 14. Select a location $l \in S$ in nondecreasing order
 15. $S = S \setminus \{l\}$
 16. $U = U \cup \{l\}$
 17. $Node_k = l$
 18. $B\&B (x_k, Node_k, UB)$

5 Experimental Results

In this section, the performance of the B&B algorithm is tested. In the following subsections, we describe the test instances, experimental environment and the performance analysis.

5.1 Test Cases

The test cases are composed by four characteristic sets:

1. Container set
2. Containership characteristics
3. Tolerances
4. Loading times

In order to validate our exact approach, we generate a set of containers. It was constructed according to the format and conditions defined in [8].

Table 1 reports the characteristics of the set of containers previously mentioned, which show the total number of containers expressed in TEU and absolute number (n); the numbers of containers whose type is 20' (T) and 40' (F), respectively; the number of containers for three classes of weight (L : low, M : medium, H : high); and the partition of containers by destination. Three classes of weight are considered, namely low (from 5 to 15 tons), medium (from 16 to 25 tons) and high containers (from 26 to 32 tons).

These instances concern to a hypothetical containership of small size with maximum capacity of 24 TEU. It is composed of 4 odd bays, 2 even bays, 2 rows and 3 tiers. Table 2 shows the loading times for this small containership.

Table 1 Containers for the set of small-sized instances

Instance	TEU	n	Size (n)		Weight (n)			Destination (n)		
			T	F	L	M	H	1	2	3
1	5	5	5	–	–	5	–	3	2	–
2	5	5	5	–	–	5	–	2	2	1
3	6	5	4	1	–	5	–	3	2	–
4	6	5	4	1	–	5	–	2	2	1
5	7	5	3	2	–	5	–	3	2	–
6	7	5	3	2	–	5	–	2	2	1
7	9	5	1	4	–	5	–	3	2	–
8	9	5	1	4	–	5	–	2	2	1
9	5	5	5	–	2	2	1	3	2	–
10	5	5	5	–	2	2	1	2	2	1
11	6	5	4	1	2	2	1	3	2	–
12	6	5	4	1	2	2	1	2	2	1
13	7	5	3	2	2	2	1	3	2	–
14	7	5	3	2	2	2	1	2	2	1
15	9	5	1	4	2	2	1	3	2	–
16	9	5	1	4	2	2	1	2	2	1
17	5	5	5	–	1	2	2	3	2	–
18	5	5	5	–	1	2	2	2	2	1
19	6	5	4	1	1	2	2	3	2	–
20	6	5	4	1	1	2	2	2	2	1
21	7	5	3	2	1	2	2	3	2	–
22	7	5	3	2	1	2	2	2	2	1
23	9	5	1	4	1	2	2	3	2	–
24	9	5	1	4	1	2	2	2	2	1

Table 2 Loading times for the set of small-sized instances, the times are expressed in 1/100 of minute

	Level 02	Level 04	Level 82
Row 02	150	144	138
Row 01	144	138	132

The tolerances for the set of instances are the following. Maximum horizontal weight tolerance (Q_1) was fixed to 20 tons and maximum cross weight tolerance (Q_2) to 40 tons. Tolerances MT and MF were fixed to 45 and 66 tons, respectively.

5.2 Infrastructure

The following configuration corresponds to the experimental conditions:

- *Software*: Operating system Microsoft Windows 7 Home Premium; Java programming language, Java Platform, JDK 1.6; and integrated development, NetBeans 7.2. Solver Gurobi 5.0.1.
- *Hardware*: Computer equipment with processor Intel (R) Core (TM) i5 CPU M430 2.27 GHz and RAM memory of 4 GB.

5.3 Performance Analysis

Table 3 shows the comparison of the results obtained by two solution methods for the set of small-sized instances.

Table 3 is basically formed by three main headings. The first one shows the identifier number of instances and their optimal value (L). The second column shows the performance of our proposed B&B, used to validate the *partial permutation* scheme. In particular, the running time (CPU), expressed in seconds, the total number of explored nodes (EN) and exploration speed (ES) are displayed. Finally, the last heading is concerned to the performance of the basic model of the master bay planning problem (BMMB) solved by the commercial optimization engine Gurobi. The same attributes from the B&B are also used to measure the performance of the BMMB.

This BMMB was solved by Gurobi with all default settings parameters on. Comparing the number of Explored Nodes, we can see that our B&B explores a larger number. It is important to mention that Gurobi has several strategies incorporated for handling constraints. It is expected that these strategies improves the performance of the solver. However, we can see that our approach has a similar performance than the one obtained by the solver in terms of running time. B&B

Table 3 Performance of algorithms based in different representation schemes

Instance	L	B&B			BMMB (Gurobi)		
		CPU	EN	ES	CPU	EN	ES
1	720	0.365	1,030,088	2,822,158.90	0.216	302	1398.08
2	720	0.174	1,030,088	5,920,045.98	0.139	144	1035.91
3	720	0.130	804,272	6,186,707.69	0.092	39	423.89
4	720	0.127	804,272	6,332,850.39	0.083	28	337.33
5	720	0.067	441,815	6,594,253.73	0.060	8	133.33
6	720	0.068	441,815	6,497,279.41	0.068	22	323.51
7	720	0.018	128,848	7,158,222.22	0.088	94	1068.12
8	720	0.019	128,848	6,781,473.68	0.095	78	821.01
9	714	0.131	890,648	6,798,839.69	0.183	258	1409.77
10	714	0.131	890,648	6,798,839.69	0.250	235	939.94
11	714	0.094	598,792	6,370,127.66	0.102	49	480.36
12	714	0.101	598,792	5,928,633.66	0.070	30	428.56
13	708	0.036	239,629	6,656,361.11	0.058	11	189.65
14	708	0.037	239,629	6,476,459.46	0.061	8	131.14
15	714	0.017	113,967	6,703,941.18	0.087	28	321.83
16	714	0.016	113,967	7,122,937.50	0.077	123	1597.29
17	720	0.145	891,400	6,147,586.21	0.085	89	1047.00
18	720	0.145	891,400	6,147,586.21	0.094	145	1542.46
19	726	0.138	846,600	6,134,782.61	0.118	220	1864.29
20	726	0.147	889,864	6,053,496.60	0.109	171	1568.71
21	720	0.066	424,279	6,428,469.70	0.084	31	369.03
22	720	0.066	424,279	6,428,469.70	0.077	47	610.38
23	720	0.018	116,496	6,472,000.00	0.079	15	189.86
24	720	0.018	116,496	6,472,000.00	0.089	19	213.47
Average	718	0.09,475	545,705.5	5759,424.80	0.1027	91.417	890.37

consumes 7.7 % less amount of CPU time with respect to BMMB, for set of evaluated instances.

In addition, it is important to observe that B&B explores a much larger number of Explored Nodes in a relative the same amount of time. In particular, B&B explores around 545,705.5 nodes in 0.09475 s, which implies a speed of exploration about 5'759,424.8 nodes per second. On the other hand, BMMB explores 91.417 nodes in 0.1027. Its speed of exploration is 890.37 nodes per second. Consequently, our proposed B&B, which is based on the *partial permutation* scheme, is 6468.5 times faster than the solution obtained by Gurobi for BMMB, which uses the *binary* representation scheme.

6 Conclusions

In this work, we present one different form of representation scheme for candidate solutions for the Master Bay Planning Problem. This scheme is named *partial permutation*, and its solution space is smaller than the binary representation scheme documented in the literature.

We solved the basic model for the master bay planning problem (BMMB) available in the literature to test the efficiency of the *binary* scheme. We used the commercial optimization engine Gurobi to solve it. Additionally, we proposed a Branch-and-Bound based on the *partial permutation* scheme.

According to the experimental results, we can conclude the following. Our B&B outperforms BMMB in running time and speed of exploration by 7.7 % and 6468.5 times, respectively. These advantages will be used to improve the strategies (e.g. handling hard constraints) of our proposed algorithm in the future works.

Acknowledgments This work was partially financed by CONACYT, DGEST and ITCM. We also thank Gurobi for allowing us to use their optimization engine.

References

1. Ambrosino, D., Sciomachen, A., Tanfani, E.: Stowing a containership: the master bay plan problem. *Transp. Res. Part A Policy Pract.* **38**, 81–99 (2004)
2. Avriel, M., Penn, M., Shpirer, N.: Container ship stowage problem: complexity and connection to the coloring of circle graphs. *Discrete Appl. Math.* **103**(1), 271–279 (2000)
3. Cruz-Reyes, L., Hernández, P.H., Melin, P., Huacuja, H.J.F., Mar-Ortiz, J., Soberanes, H.J.P., Barbosa, J.J.G.: A loading procedure for the containership stowage problem. In: Recent Advances on Hybrid Approaches for Designing Intelligent Systems, pp. 543–554. Springer, Berlin (2014)
4. Hernández, P.H., Cruz-Reyes, L., Melin, P., Mar-Ortiz, J., Huacuja, H.J.F., Soberanes, H.J.P., Barbosa, J.J.G.: An ant colony algorithm for improving ship stability in the containership stowage problem. *Advances in Soft Computing and Its Applications. Lecture Notes in Computer Science*, vol. 8266, pp. 93–104 (2013)
5. Ambrosino, D., Anghinolfi, D., Paolucci, M., Sciomachen, A.: A new three-stepheuristic for the master bay plan problem. *Marit. Econ. Logistics* **11**, 98–120 (2009)
6. Ambrosino, D., Sciomachen, A., Tanfani, E.: A decomposition heuristics for the container ship stowage problem. *J. Heuristics* **12**, 211–233 (2006)
7. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: *Introduction to Algorithms*. MIT Press, Cambridge (2003)
8. Cruz-Reyes, L., Hernández, P., Melin, P., Huacuja, H.J.F., Mar-Ortiz, J.: Constructive algorithm for a benchmark in ship stowage planning. In: Recent Advances on Hybrid Intelligent Systems, pp. 393–408. Springer, Berlin (2013)

Verifying the Effectiveness of an Evolutionary Approach in Solving Many-Objective Optimization Problems

**Laura Cruz-Reyes, Eduardo Fernandez, Claudia Gomez,
Patricia Sanchez, Guadalupe Castilla and Daniel Martinez**

Abstract Most approaches in the evolutionary multi-objective optimization were found to be vulnerable in solving many-objective optimization problems (four or more objectives). This is mainly due to the fact that these algorithms lack from ability to handle more than three objectives adequately. For this reason, researchers have been focusing in developing algorithms capable of addressing many-objective optimization problems. Recently, authors of the Non-dominated Sorting Genetic Algorithm-II (NSGA-II) have proposed an extension of this approach, called NSGA-III. This new approach has opened new directions for research and development to solve many-objective optimization problems. In this algorithm the maintenance of diversity among population members is aided by supplying a number of well-spread reference points. In this work, a comparative study of the performance of NSGA-II and NSGA-III was carried out. Our aim is to verify the effectiveness of NSGA-III to deal with many-objectives problems and extend the range of problems that this approach can solve. For this, the comparison was made addressing the project portfolio problem, using instances with three and nine objectives.

L. Cruz-Reyes (✉) · C. Gomez · P. Sanchez · G. Castilla · D. Martinez
Tecnologico Nacional de Mexico, Instituto Tecnologico de Ciudad Madero,
Ciudad Madero, Tamaulipas, Mexico
e-mail: lauracruzreyes@itcm.edu.mx

C. Gomez
e-mail: cggs71@hotmail.com

P. Sanchez
e-mail: jpatricka.sanchez@gmail.com

G. Castilla
e-mail: gpe_cas@yahoo.com.mx

D. Martinez
e-mail: danielmartinezvega@hotmail.com

E. Fernandez
Universidad Autonoma de Sinaloa, Culiacan, Sinaloa, Mexico
e-mail: eddyf@uas.edu.mx

1 Introduction

In the real world, most problems involve multiple criteria to be considered simultaneously, and often these are in conflict with each other [1, 2]. Given the conflicting nature of the criteria, it is not possible to obtain a single optimum and consequently the ideal solution of Multi-objective Optimization Problems (MOPs) cannot be reached.

The optimal solution for a MOP is not a single solution as for mono-objective optimization problems, but a set of solutions determined as Pareto optimal solutions. This set represents the compromise solutions between the different conflicting objectives. Solving this class of problems implies obtaining a set of Pareto optimal solutions that fulfill the requirements of convergence to the true Pareto front and exhibit uniform diversity [3].

Multi-Objective Evolutionary Algorithms (MOEAs) are particularly attractive to solve MOPs because they deal simultaneously with a set of possible solutions which allows them to obtain an approximation of the Pareto frontier in a single algorithm's run.

However, these algorithms lack of ability to handle more than three objectives adequately [4]. Many-objective problems are defined as those having four or more objectives.

Currently, many-objective optimization has attracted increasing attention in Evolutionary Multi-objective Optimization (EMO) community, mainly from two aspects [4]. First, the optimization problems involving a high number of objectives appear in many real-world applications. Second, the MOEAs have encountered great difficulties in many-objective optimization, because almost all the solutions in each population become non-dominated when the number of objectives is increased, which would lead to the severe loss of Pareto-based selection pressure towards the Pareto front (PF).

To overcome these drawbacks, some efforts have been reported in the literature such as the adoption of new diversity promotion mechanisms. This is due to that Pareto-dominance could not provide sufficient selective pressure to improve a given population, so the diversity criterion plays a key role in such cases [4].

In this work, we carry out a comparative study between the well-known NSGA-II and a recently version of this approach called NSGA-III. This new algorithm developed by Deb and Jain [4] is based on a set of supplied reference points which is used to deal with many-objective problems. The purpose of the study is to assess the capability of NSGA-III in solving many-objectives problems. The comparison was done in the project portfolio problem by using a number of instances with three and nine objectives.

2 Background

In this section, some basic definitions about multi-objective optimization and quality indicators are first given. Then, the original NSGA-II and NSGA-III will be briefly described.

2.1 Pareto Optimality

In Multi-objective Optimization Problems (MOP); where the objectives are in conflict with each other, a set of compromise solutions are usually sought rather than a single solution. Therefore, the idea of *optimal* most commonly adopted in these cases is given by Pareto [5]. This notion is known as the *Pareto optimality*. It is said that a vector of decision variables $\vec{x}^* \in F$ (where F is the feasible zone) is *Pareto optimal* if for each $\vec{x} \in F$ (assuming maximization):

$$f_i(\vec{x}^*) \geq f_i(\vec{x}), \quad \forall i \in [1, 2, \dots, k] \text{ and} \quad (1)$$

$$f_j(\vec{x}^*) > f_j(\vec{x}), \quad \exists j \in [1, 2, \dots, k] \quad (2)$$

That is to say, a vector \vec{x}^* is Pareto optimal if there is no other feasible vector \vec{x} that improves the performances on a criterion without decreasing the quality of at least another criterion.

However, this concept almost always gives not one but several solutions called as the *Pareto optimal set*. The image of this set in the objective space is denoted as the *Pareto front*. The set of vectors corresponding to a solution contained in the Pareto optimal set are called *non-dominated*.

The concept of *Pareto dominance* can be defined as follows: a vector \vec{u} dominates to other vector \vec{v} if and only if fulfills with the Eqs. 1 and 2. When two solutions are compared, only can be three possible results, A dominates B, A is dominated by B or A and B are non-dominated.

2.2 Hypervolume

By measuring the quality of multi-objective algorithms two aspects are generally taken into account: minimizing the distance from the Pareto front obtained to the exact Pareto front of the problem and maximize the distribution of solutions on the front so that it is as uniform as possible. The Hypervolume (HV) is a metric designed to measure both aspects: convergence and diversity in a given front.

It is a metric originally proposed by Zitzler and Thiele in [6], is also known as the *S* metric [7] or the Lebesgue measure [8]. This metric reflects the volume of

objective space covered by the individuals of a set of non-dominated solutions. HV captures in one scalar both the nearness of the solutions to the optimal set and their extent across objective space [9]. The algorithms that achieve higher HV values are better. That is, if a set S has a greater HV value than a set S' ; S is taken to be a better set of solutions than S' .

The HV of a set of solutions is measured in relation to a reference point, usually called anti-optimal point. This point can be found by constructing a vector with the worst values of the objective functions.

The HV has also been used in multi-objective evolutionary algorithms [10], either as a diversity mechanism, or as part of the selection process, or as part of an archiving mechanism.

The major problem with the HV is that its exact calculation is very expensive. For this reason, some algorithms have been proposed to compute a HV approximation.

2.3 Non-dominated Sorting Genetic Algorithm II (NSGA-II)

NSGA-II developed by Deb et al. [11, 12], is one of the most popular Multi-Objective Evolutionary Algorithms (MOEAs). The basic framework of the NSGA-II is as follows. First, a parent population P_0 is randomly created and is sorted based on the non-domination. Then, genetic operations are carried out by creating an offspring population Q_0 . Both populations are of size N . The procedure is different from the first generation onward.

Let us describe the t -th generation. Suppose that there are a parent population P_t and an offspring population Q_t . Both populations are combined to form a population $R_t = P_t \cup Q_t$ (of size $2N$).

The population R_t is sorted according to non-domination levels or fronts (F_1, F_2 and so on). Then, a new parent population P_{t+1} is constructed with individuals of the best non-dominated fronts, one at a time, starting from F_1 , until the size of P_{t+1} is equal or exceeds to N . Let us consider that the last front included is the l -th front. Thus, all solutions from front $(l + 1)$ onwards are rejected. Often, the number of solutions in all fronts included, from F_1 to F_l , could be larger than the population size. Therefore, the solutions of the last accepted front are accepted partially to construct P_{t+1} .

In such a case, instead of discarding arbitrarily some elements from the last front, only those solutions that will maximize the diversity are chosen.

To choose exactly N population members, NSGA-II uses a niche-preservation operator, called the crowded-comparison operator, to compute the crowding distance for every solution of the last front.

Subsequently, these solutions are sorted in descending order using the crowded-comparison operator and those having larger crowding distance values are chosen. Then, the new parent population P_{t+1} is created.

The new population P_{t+1} is now used for selection, crossover, and mutation to create a new offspring population, but the selection criterion is now based on the niche-preservation operator.

2.4 Many Objective NSGA-II or NSGA-III

NSGA-III is a recent evolutionary many-objective optimization algorithm proposed by Deb and Jain [4]. This new approach remains similar to NSGA-II; the difference between these algorithms lies in its selection mechanism to maintain of diversity among population members.

NSGA-III starts with the definition of a number of well-spread reference points. At the t -th generation, suppose the current parent population is P_t and the offspring population Q_t is created from P_t by using randomly selection, recombination and mutation operators. The size of P_t and Q_t are both N . The populations P_t and Q_t are combined to form a new population $R_t = P_t \cup Q_t$ (of size $2N$). To choose the best N members from R_t and construct with these the next parent population P_{t+1} , R_t is sorted into different non-domination levels (F_1 , F_2 , and so on).

Then, each non-domination level is selected individually to create a new population S_t , starting from F_1 , until the size of S_t is equal to N or for the first time becomes greater than N . The last level included is the l -th level (F_l), hence all solutions from level ($l + 1$) onwards are rejected.

In most situations, F_l is only accepted partially. In such case, members in $S_t \setminus F_l$ are chosen for the next parent population P_{t+1} , and the remaining solutions are selected from F_l such that a desired diversity is maintained in the population.

In the original NSGA-II, the members in F_l with the largest crowding distance values are chosen. Nevertheless, the crowding distance measure does not work well for many-objective problems [13].

Thus, the selection mechanism in NSGA-III is modified by using the supplied reference points to select the remaining members.

To achieve this, objective values and reference points are first normalized so that they have a single range. After normalization, orthogonal distance between every member in S_t and each of the reference lines (joining the ideal point and a reference point) is computed. Each member in S_t is then associated with a reference point having the smallest orthogonal distance.

Next, the niche count ρ_j for the j -th reference point, defined as the number of members in $S_t \setminus F_l$ that are associated with the j -th reference point, is computed. Moreover, a niche-preservation operation is performed to select members from F_l in order to ensure a diverse set of solutions and it works as follows.

First, the reference point set $J_{\min} = \{j: \arg \min_j \rho_j\}$ having the minimum ρ_j value is identified. In case of multiple reference points, one $\bar{j} \in J_{\min}$ is randomly chosen.

If $\rho_{\bar{j}} = 0$, can exist one or more members in front F_l that are associated with the reference point \bar{j} . In this case, the one having the shortest orthogonal distance to the

j -th reference line is added to P_{t+1} . The count of $\rho_{\bar{j}}$ is then incremented by one. In the case of $\rho_j \geq 1$, a randomly chosen member from front F_l that is associated with the reference point \bar{j} is added to P_{t+1} . The count of $\rho_{\bar{j}}$ also needs increasing by one. In the case where the front F_l does not have any member associated with the reference point \bar{j} , this point is excluded from further consideration for the current generation.

After niche counts are updated, the above procedure is repeated for a total of $K = N - |S_t \setminus F_l|$ times to fill up population P_{t+1} . Then, the new parent population P_{t+1} is used to create offspring population Q_{t+1} by using randomly selection, crossover and mutation.

3 Experimentation and Results

In this section, we present the case of study and the results of the experimentation carried out.

3.1 Case of Study: The Project Portfolio Problem

A portfolio is a subset of projects that can be done on the same time period and by other hand, a project is a process which pursues a set of aims [14]. Projects that belong to the same portfolio share the organization's available resources. Proper selection of the projects that will integrate the portfolio to receive resources of the organization is one of the most important decision issues for any institution [15, 16]. The decision maker (DM) is responsible for the selection of the portfolio that the institution will implement [17].

To formalize these concepts, consider the following: a total budget denoted by B , a set of N projects, a portfolio modeled as a binary vector $x = \langle x_1, x_2, \dots, x_N \rangle$ where $x_i = 1$ if the i -th project will be supported and $x_i = 0$ otherwise. Each project i has a cost denoted by c_i and corresponds to an area (health, education, etc.) denoted by a_i . Each area j has lower and upper budgetary limits denoted by L_j and U_j respectively. Likewise, each project i corresponds to a geographical region and also has lower and upper budgetary limits. Each project i is represented as a p -dimensional vector $f(i) = \langle f_1(i), f_2(i), f_3(i), \dots, f_p(i) \rangle$, where each $f_j(i)$ represents the benefit of project i to objective j into a problem of p objectives.

Based on this, a portfolio is feasible when it satisfies the constraint of the total budget (Eq. 3) and the constraint for each area j (Eq. 4).

$$\left(\sum_{i=1}^N x_i c_i \right) \leq B \quad (3)$$

$$L_j \leq \sum_{i=1}^N x_i g_i(j) c_i \leq U_j \quad (4)$$

where g may be defined as:

$$g_i(j) = \begin{cases} 1 & \text{if } a_i = j, \\ 0 & \text{otherwise} \end{cases} \quad (5)$$

The union of the benefits of each of the projects that compose to a portfolio determine its quality. This may be expressed as:

$$z(x) = \langle z_1(x), z_2(x), z_3(x), \dots, z_p(x) \rangle \quad (6)$$

where $z_j(x)$ in its simplest form, is determined by:

$$z_j(x) = \sum_{i=1}^N x_i f_j(i) \quad (7)$$

Therefore, considering R_F as the feasible region, the problem of project portfolio is to determine one or more portfolios that solve:

$$\max_{x \in R_F} \{z(x)\} \quad (8)$$

3.2 Results

In this section, we provide the experimental results. We compared NSGA-II with NSGA-III to investigate its performance. We experimented with two instances addressing the project portfolio problem.

The parameters of the evolutionary search in both algorithms were as follows, crossover probability = 1; mutation probability = 0.05. Parameters about instances and experimentation are shown in Table 1.

We use the concept of Pareto dominance and the Hypervolume as performance metrics. Each instance was run five times by every algorithm. First, we compared the performance using the Pareto dominance. The results are illustrated in Tables 2 and 3.

Table 1 Information of instances and experimentation parameters

Instance	No. of objectives	No. of projects	Population size	No. ref. points	No. of iterations
1	3	100	92	91	500
2	9	100	166	165	500

Table 2 Comparative results between NSGA-II and NSGA-III in three objectives

Exp	Algorithm	Size of the solution set	Solutions that remain non-dominated in $A \cup B$	Percent of solutions that remain non-dominated (%)
1	NSGA-II	92	51	55.4
	NSGA-III	92	89	96.7
2	NSGA-II	88	47	53.4
	NSGA-III	91	75	82.4
3	NSGA-II	91	60	65.9
	NSGA-III	92	75	81.5
4	NSGA-II	93	61	65.6
	NSGA-III	92	86	93.5
5	NSGA-II	93	39	41.9
	NSGA-III	94	90	95.7

Note: A is the set of solutions obtained by NSGA-II; B is the set obtained by NSGA-III

Table 3 Comparative results between NSGA-II and NSGA-III in nine objectives

Exp	Algorithm	Size of the solution set	Solutions that remain non-dominated in $A \cup B$	Percent of solutions that remain non-dominated (%)
1	NSGA-II	200	131	65.5
	NSGA-III	193	189	97.9
2	NSGA-II	191	134	70.2
	NSGA-III	182	175	96.2
3	NSGA-II	197	124	62.9
	NSGA-III	190	190	100.0
4	NSGA-II	199	112	56.3
	NSGA-III	199	198	99.5
5	NSGA-II	201	130	64.7
	NSGA-III	193	191	99.0

Note: A is the set of solutions obtained by NSGA-II; B is the set obtained by NSGA-III

Subsequently, was computed the hypervolume value for each set of solutions. In Table 4 are summarized the worst and best hypervolume values obtained by each algorithm.

We can see that in both instances, NSGA-III obtained high percentages (over 80 %) of solutions that remain non-dominated. By other hand, NSGA-II obtains at most 70 % of solutions non-dominated.

Similarly, NSGA-III obtained the highest values of hypervolume in both instances.

Table 4 Worst and best hypervolume values

No. of objectives	NSGA-II	NSGA-III
3	0.51	0.56
	0.55	0.57
9	0.062	0.079
	0.089	0.097

4 Conclusions and Future Work

We have presented a comparative study between NSGA-II and its successor, the recent many-objective algorithm NSGA-III.

This new algorithm arises because, in the literature has been shown that NSGA-II does not work well for many-objectives problems.

For this reason, this study aims to verify the effectiveness of NSGA-III to deal with many-objectives problems. Besides extend the range of problems that this new approach can solve. For this, the comparison was made addressing the project portfolio problem.

For instances with three and nine objectives, NSGA-III obtains a better performance than its antecessor, the NSGA-II. More experimentation is required to reach definitive conclusions.

As immediate work we are going to explore a new and adaptive relocation strategy for reference points, proposed by the authors of NSGA-III. This aims to implement in our research these new strategies as a mechanism for diversity.

Acknowledgments This work was partially financed by CONACYT, COTACYT, DGEST, TECNM and ITCM.

References

1. Deb, K.: Multi-Objective Optimization using Evolutionary Algorithms, vol. 16. Wiley, New York (2001)
2. Coello, C.C., Lamont, G.B., Van Veldhuizen, D.A.: Evolutionary Algorithms for Solving Multi-Objective Problems, 2nd edn. Springer, New York (2007)
3. Talbi, E.G.: Metaheuristics: from Design to Implementation, vol. 74. Wiley, New York (2009)
4. Deb, K., Jain, H.: An evolutionary many-objective optimization algorithm using reference-point based nondominated sorting approach, part i: solving problems with box constraints. In: Proceedings of IEEE Transactions on Evolutionary Computation (2013)
5. Pareto, V.: Politique, Cours D' economie. Rouge, Lausanne, Switzerland (1896)
6. Zitzler, E., Thiele, L.: Multiobjective optimization using evolutionary algorithms: a comparative case study. In: Eiben, A.E., Bäck, T., Schoenauer, M., Schwefel, H.-P. (eds.) PPSN. Lecture Notes in Computer Science, vol. 1498, pp. 292–301. Springer, Berlin (1998)
7. Zitzler, E.: Evolutionary algorithms for multiobjective optimization: methods and applications. Ph.D. dissertation, Swiss Federal Inst. Technology (ETH) Zurich, Switzerland (1999)
8. Laumanns, M., Zitzler, E., Thiele, L.: A unified model for multi-objective evolutionary algorithms with elitism. In: Proceedings of the 2000 Congress on Evolutionary Computation, 2000, vol. 1, pp. 46–53. IEEE (2000)
9. While, L., Hingston, P., Barone, L., Huband, S.: A faster algorithm for calculating hypervolume. Evol. Comput. IEEE Trans. **10**(1), 29–38 (2006)
10. While, L., Bradstreet, L., Barone, L.: A fast way of calculating exact hypervolumes. Evol. Comput. IEEE Trans. **16**(1), 86–95 (2012)
11. Deb, K., Agrawal, S., Pratap, A., Meyarivan, T.: A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. Lect. Notes Comput. Sci. **1917**, 849–858 (2000)

12. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. *Evol. Comput. IEEE Trans.* **6**(2), 182–197 (2002)
13. Kukkonen, S., Deb, K.: Improved pruning of non-dominated solutions based on crowding distance for bi-objective optimization problems. In: Proceedings of the World Congress on Computational Intelligence (WCCI-2006), pp. 1179–1186. IEEE Press, Vancouver (2006)
14. Carazo, A., Gomez, T., Molina, J., Hernandez-Diaz, A., Guerrero, F., Caballero, R.: Solving a comprehensive model for multiobjective project portfolio selection. *Comput. Oper. Res.* **37**(4), 630–639 (2010)
15. Castro, M.: Development and implementation of a framework for I&D in public organizations. Master's thesis, Universidad Autónoma de Nuevo León, Mexico (2007)
16. Garcia, R.: Hyper-Heuristic for solving social portfolio problem. Master's Thesis, Instituto Tecnológico de Cd. Madero (2010)
17. Fernandez, E., Navarro, J.: A genetic search for exploiting a fuzzy preference model of portfolio problems with public projects. *Annals OR* **117**, 191–213 (2002)

Comparative Study on Constructive Heuristics for the Vertex Separation Problem

Norberto Castillo-García, Héctor Joaquín Fraire Huacuja,
José Antonio Martínez Flores, Rodolfo A. Pazos Rangel,
Juan Javier González Barbosa and Juan Martín Carpio Valadez

Abstract The vertex separation problem (VSP) consists of finding a linear ordering of the vertices of an input graph that minimizes the maximum number of vertex separators at each cut-point induced by the ordering. VSP is an NP-hard problem whose efficient solution is relevant in fields such as very large scale integration design, computer language compiler design, graph drawing and bioinformatics. In the literature reviewed, we found several exact algorithms and two metaheuristics based on the variable neighborhood search approach. These metaheuristics are currently the best stochastic algorithms for solving VSP. One of the key points of their efficiency is the usage of heuristics to construct a high-quality initial solution that considerably improves the algorithm performance. In this chapter we augment the literature on VSP by proposing a new set of heuristics. The proposed constructive heuristics are compared with the best ones found in the state-of-the-art and with random solution generator (*Rnd*). Experimental results demonstrate the importance of constructive algorithms. The best constructive improves *Rnd* by 89.96 % in solution quality.

N. Castillo-García · H.J.F. Huacuja (✉) · J.A.M. Flores · R.A. Pazos Rangel · J.J.G. Barbosa
Tecnológico Nacional de México, Instituto Tecnológico de Ciudad Madero,
Ciudad Madero, Mexico
e-mail: automatas2002@yahoo.com.mx

N. Castillo-García
e-mail: norberto_castillo15@hotmail.com

J.A.M. Flores
e-mail: jose.mtz@itcm.edu.mx

J.J.G. Barbosa
e-mail: jjgonzalezbarbosa@hotmail.com

J.M.C. Valadez
Tecnológico Nacional de México, Instituto Tecnológico de León, León, Mexico
e-mail: jmcarpio61@hotmail.com

1 Introduction

The Vertex Separation Problem (VSP) belongs to a family of graph layout problems in which the goal is to find the best separator of vertices or edges in a generic graph [1]. Particularly, it consists of finding a linear ordering of the vertices such that the maximum number of vertex separators at each cut-point induced by the ordering is minimized [2].

Unfortunately, finding such ordering is really hard to achieve in practice since VSP is NP-hard for both generic graphs [3] and some structured graphs [4–7]. Additionally, the number of possible solutions that satisfies the problem grows exponentially with respect to the number of vertices of the graph.

The solution of VSP is important due to its applications in several domains, which include very large scale integration (VLSI) design [8], computer language compiler design [9], natural language processing [10], processing of manufactured products [11] and bioinformatics [12].

Despite its practical applications, there are only two metaheuristic algorithms to find good solutions for big-sized instances in short computing times. They are based on a variable neighborhood search (VNS) framework [1, 13]. One key point of their efficiency is the usage of a constructive heuristic at the beginning, which allows them to start the search in a promising area of the solution space.

A constructive heuristic is an algorithm that creates a solution element by element, that is, it selects one element at the time. In the context of VSP, an element of the solution is a vertex of the graph. More precisely, a constructive heuristic iteratively selects one vertex by using a greedy function that measures the suitability of choosing the candidate vertex. Once a vertex is selected, it cannot be selected again due to feasibility conditions, that is, one vertex cannot be placed at two or more positions in the same solution.

In [1], the authors proposed two constructive heuristics and tested them to determine the most suitable one for their purposes. Additionally, in [13] the authors proposed an improvement of the best constructive reported in [1]. Thus, there are only three constructive heuristics for VSP available in the current state-of-the-art. Considering this lack of constructive heuristics for VSP, in this chapter we propose two constructive algorithms. Each heuristic can be configured in four different ways, totalizing a number of eight new heuristics.

We carried out an experiment to show the importance of constructive heuristics in the context of VSP. In particular, we solved 248 instances from the standard benchmark by using all the constructive algorithms and compared the results against an algorithm to construct a random solution. Experimental results confirmed that it is strongly recommended the usage of constructive heuristics rather than a random solution. Specifically, the solutions generated by the best constructive improves the solution quality of the random solution algorithm by 89.96 % in average. Additionally, the best constructive found 136 out of 248 best solutions whereas the random solution algorithm found only 1.

The remaining of this chapter is organized as follows. Section 2 presents the formal definition of the approached problem. In Sect. 3 we describe both the constructive heuristics available in the literature and our proposed constructive algorithms. Section 4 describes the numerical experiments carried out to test all the constructives. Finally, in Sect. 5 we discuss the experimental findings of this study.

2 Formal Definition of VSP

Let $G = (V, E)$ be a connected undirected graph in which V stands for the set of vertices and E for the set of edges. The numbers of vertices and edges are $n = |V|$ and $m = |E|$, respectively. A *linear ordering* is a bijection $\varphi : V \rightarrow \{1, 2, \dots, n\}$ which assigns an integer to every vertex in G . This is also called a *labeling* and represents a solution for VSP. It is easy to see that the total number of different solutions is $n!$. The position of some vertex $u \in V$ in ordering φ is denoted by $\varphi(u)$.

Let $\Phi(G)$ be the set of all possible linear orderings, $\varphi \in \Phi(G)$ be a solution and $p \in \{1, 2, \dots, n\}$ be an integer representing a cut-point (position) in the ordering. We define $L(p, \varphi, G) = \{u \in V : p \leq \varphi(u)\}$ as the set of *left* vertices with respect to cut-point p . Similarly, $R(p, \varphi, G) = \{u \in V : p > \varphi(u)\}$ is defined as the set of *right* vertices with respect to p . The set of *vertex separators* at cut-point p contains all the vertices in $L(p, \varphi, G)$ with one or more adjacent vertices in $R(p, \varphi, G)$, that is,

$$\delta(p, \varphi, G) = \{u \in L(p, \varphi, G) : \exists v \in R(p, \varphi, G) : (u, v) \in E\}. \quad (1)$$

The number of vertex separators at cut-point p , i.e., $|\delta(p, \varphi, G)|$, defines a value known as *cut-value*. There are n cut-values for each solution since there are n cut-points for a graph with n vertices. Thus, the objective value $vs(\varphi, G)$ associated to a solution φ is computed as the largest cut-value. In mathematical terms,

$$vs(\varphi, G) = \max_{p=1, \dots, n} \{|\delta(p, \varphi, G)|\}.$$

Finally, the goal of VSP is to find the linear ordering $\varphi^\star \in \Phi(G)$ with the *minimum* objective value. Therefore, φ^\star must satisfy:

$$vs(\varphi^\star, G) = \min_{\varphi \in \Phi(G)} \{vs(\varphi, G)\}.$$

3 Heuristic Approaches

As mentioned previously, a constructive heuristic creates a solution element by element. In the case of VSP, the elements are the vertices. In this section we discuss several constructive algorithms to obtain an initial solution.

We start by describing two constructive heuristics taken from the state-of-the-art. They play an important role to improve the efficiency of one of the best metaheuristic algorithms for the approached problem. We then explain our proposed set of constructive heuristics. They use a different greedy function to select the elements.

3.1 Constructive Heuristics from the Literature

In 2012, Duarte et al proposed a variable neighborhood search algorithm (BVNS) to tackle VSP [1]. The first step of BVNS consists of constructing an initial solution with objective value as low as possible. To do that, the authors proposed two constructive algorithms, C_1 and C_2 .

The first constructive, C_1 , starts by creating the sets of unlabeled (U) and labeled (L) vertices. At the beginning, $U = V$ and $L = \emptyset$. The vertex $u \in V$ with the minimum degree is labeled with 1. In other words, u occupies the first position in the solution. Then, unlabeled and labeled sets are updated, i.e., $U = U \setminus \{u\}$ and $L = L \cup \{u\}$. For the rest of the vertices, C_1 evaluates the following greedy function:

$$g(v) = |N_L(v)| - |N_U(v)| \quad \forall v \in U,$$

where $N_L(v)$ contains those vertices adjacent to v that are already labeled, and $N_U(v)$ represents the set of vertices adjacent to v that remain unlabeled. The vertex with the maximum g -value is selected and hence receives the next label. This process ends when all the vertices of the graph have been labeled, i.e., $U = \emptyset$.

The second constructive, C_2 , is based on the creation of level structures. This means that the set of vertices V is partitioned into the sets $L_1, L_2, \dots, L_\lambda$, which are called levels. The first level L_1 contains only one vertex. The remaining levels L_i with $i = 2, 3, \dots, \lambda$ contain all the vertices adjacent to some vertex in L_{i-1} that are not contained in any L_j with $1 \leq j < i$. The total number of levels λ entirely depends on the graph and the vertex in the first level L_1 .

Thus, C_2 constructs n different level structures, each with a different vertex placed in L_1 . This constructive keeps the level structure with the largest λ . In order to obtain a linear ordering, C_2 explores the selected level structure in breadth and assigns labels to visited vertices incrementally.

3.2 Proposed Constructive Heuristics

We propose two constructive heuristics. Each one can be configured in four different ways, which results in eight different constructive procedures. Both proposed heuristics use the cut-value to select the most suitable vertex to be placed at each

position. The main difference between them lies in the number of candidate vertices to be considered.

Our first constructive heuristic, *HA*, considers all the unlabeled vertices as candidates whereas the second one, *HN*, considers only the unlabeled vertices that are adjacent to some vertex already labeled. Let U and L be the sets of unlabeled and labeled vertices, respectively. Initially, $U = V$ and $L = \emptyset$. *HA* begins by selecting one vertex $u \in V$ with either the largest or the lowest degree. This implies two different possibilities to choose the first vertex. Then, sets U and L are updated, i.e., $U = U \setminus \{u\}$ and $L = L \cup \{u\}$.

In order to label the next vertex, we iteratively place each unlabeled vertex $v \in U$ at the next pending position $k = |L| + 1$ and compute the cut-value. Of course, we need a complete solution to compute it. With this aim, we consider $L \cup \{v\}$ as the set of left vertices at position k , i.e., $L(k, \varphi, G)$ and the remaining unlabeled vertices $U \setminus \{v\}$ as the set of right vertices $R(k, \varphi, G)$. To obtain the complete solution φ , we incrementally label the vertices in $L(k, \varphi, G)$ with integers $\{1, \dots, k\}$ and the vertices in $R(k, \varphi, G)$ with $\{k + 1, \dots, n\}$. Once obtained φ , we evaluate it with the following greedy function:

$$f(v) = |\delta(k, \varphi, G)| \quad \forall v \in U,$$

which represents the number of vertex separators at cut-point k [see Eq. (1)]. This constructive selects the vertex v whose f -value is either maximum or minimum. In this case there are another two possibilities to configure *HA*. The method ends when all the vertices in G have been labeled, i.e., $U = \emptyset$. Figure 1 shows the pseudocode of *HA*.

As mentioned before, we have two options to select the first vertex and another two options to select the remaining vertices. Consequently, we have four possible different configurations for *HA*.

Pseudocode of *HA*.

1. Let U and L be the sets of unlabeled and labeled vertices, respectively.
2. $U = V$ and $L = \emptyset$.
3. Select vertex $u \in V$ with the lowest (or largest) adjacency degree.
4. Assign label $l = 1$ to u .
5. $U = U \setminus \{u\}$ and $L = L \cup \{u\}$.
6. **while** $U \neq \emptyset$ **do**
7. $l = l + 1$.
8. $v = \text{argmin}_{w \in U} \{f(w)\}$ **or** $v = \text{argmax}_{w \in U} \{f(w)\}$, according to the chosen criterion.
9. Assign label l to selected vertex v .
10. $U = U \setminus \{v\}$ and $L = L \cup \{v\}$.
11. **end while.**

Fig. 1 Pseudocode of our first proposed constructive heuristic *HA*

Pseudocode of *HN*.

1. Let U and L be the sets of unlabeled and labeled vertices, respectively.
2. Let N be the set of unlabeled vertices adjacent to some vertex already labeled.
3. $U = V$ and $L = \emptyset$.
4. Select vertex $u \in V$ with the lowest (or largest) adjacency degree.
5. Assign label $l = 1$ to u .
6. $U = U \setminus \{u\}$ and $L = L \cup \{u\}$.
7. **while** $U \neq \emptyset$ **do**
8. $l = l + 1$.
9. $N = \{x \in U \mid \exists y \in L : (x, y) \in E\}$.
10. $v = \operatorname{argmin}_{w \in N} \{f(w)\}$ or $v = \operatorname{argmax}_{w \in N} \{f(w)\}$, according to the chosen criterion.
11. Assign label l to selected vertex v .
12. $U = U \setminus \{v\}$ and $L = L \cup \{v\}$.
13. **end while.**

Fig. 2 Pseudocode of our second proposed constructive heuristic *HN*

Our second constructive, *HN*, considers only those unlabeled vertices adjacent to some vertex already labeled. The rest of the method is similar to our first constructive *HA*. Figure 2 shows the pseudocode of *HN*.

Table 1 summarizes all possible configurations for our constructive heuristics. The first heading shows the name of the variant. Headings two and three show the criteria to select the first vertex, either with the lowest or the largest adjacency degree. Finally, headings four and five show the criteria to select the remaining vertices, either with the lowest or the largest f -value. The first four rows represent the variants for *HA* and the last four rows the variants for *HN*.

Table 1 All possible configurations for our proposed constructive heuristics *HA* and *HN*

Constructive	Adjacency degree		f -value	
	The lowest	The largest	The lowest	The largest
<i>HA</i> ₁	✓		✓	
<i>HA</i> ₂	✓			✓
<i>HA</i> ₃		✓	✓	
<i>HA</i> ₄		✓		✓
<i>HN</i> ₁	✓		✓	
<i>HN</i> ₂	✓			✓
<i>HN</i> ₃		✓	✓	
<i>HN</i> ₄		✓		✓

4 Computational Experiment

In this section we describe the numerical experiment carried out to test the performance of our constructive approaches. We compare our results with some of the best constructive algorithms available in the literature and with an algorithm to generate a random solution.

The experiment was conducted on a standard silicon-based computer with 2.4 GHz of clock-rate and 4 GB of RAM. The algorithms were implemented in Java JRE 1.6.0_65 under a Mac OS X (10.6.8) operating system. In this experiment we used 164 instances from the standard benchmark [14] and 84 *Small* instances, totalizing 248 different graphs. Of course, all the algorithms solved all the instances. The description of the sets of instances used in this experiment is given in the following paragraphs.

- **Grid.** This dataset consists of 52 instances of two dimensional meshes, with an optimal objective value ranging from 3 to 54. The number of vertices and edges ranges from 9 to 2916 and from 12 to 5724, respectively [1].
- **Small.** This set of instances consists of 84 graphs, whose optimal value is not known. In this dataset the number of vertices and edges ranges from 16 to 24 and from 18 to 49, respectively [15].
- **Tree.** This dataset includes 50 different trees: 15 with optimal objective value of 3, 15 with optimal value of 4 and 20 with optimal value of 5. The number of vertices and edges varies from 22 to 202 and from 21 to 201, respectively [1].
- **Harwell Boeing (HB).** This dataset consists of 62 instances whose optimal values are unknown. For these instances the number of vertices and edges ranges from 24 to 960 and from 34 to 3721, respectively [1].

We measure both the solution quality and the execution time of each constructive. We did not fix any kind of limitation for an early termination. In order to quantify the importance of the constructive algorithms, we also generate a random solution (*Rnd*) by iteratively picking a different vertex with a uniform probability.

Tables 2 and 3 show the solution quality of the heuristics. Particularly, they show the average objective values and the number of best solutions found by all the algorithms, respectively. Additionally, Table 4 presents the execution time spent by each heuristic. Tables 2, 3 and 4 have the same structure. The columns show each group of instances and the rows present the tested algorithms. More precisely, the first column indicates the name of the algorithm. Columns 2–5 present the value of the corresponding measured attribute for each set of instances. In particular, the average objective value for Table 2, the total number of best solutions for Table 3 and the average execution time for Table 4. The last column of Tables 2 and 4 averages the objective values and running times for all the instances, respectively. The last column of Table 3 show the total number of best solutions found by each algorithm out of 248 instances. The boldfaced quantities denote the best values of the corresponding attributes.

Table 2 Average objective values obtained by the constructive algorithms

Heuristic	Grid (52)	Small (84)	Tree (50)	HB (62)	Average
<i>Rnd</i>	553.077	9.262	43.600	229.032	185.153
<i>C</i> ₁	29.500	6.179	4.660	40.097	19.242
<i>C</i> ₂	28.500	5.012	11.080	34.694	18.581
<i>HA</i> ₁	38.192	4.321	5.240	53.645	23.940
<i>HA</i> ₂	684.346	12.214	61.700	294.581	233.714
<i>HA</i> ₃	38.500	4.262	5.220	53.081	23.839
<i>HA</i> ₄	688.000	11.679	61.680	294.145	234.185
<i>HN</i> ₁	28.500	4.048	7.660	49.226	21.198
<i>HN</i> ₂	687.942	6.762	12.860	235.452	207.992
<i>HN</i> ₃	30.442	4.143	7.520	51.274	22.121
<i>HN</i> ₄	697.962	6.512	13.360	233.516	209.625

Table 3 Number of best solutions found by the constructive algorithms

Heuristic	Grid (52)	Small (84)	Tree (50)	HB (62)	Total
<i>Rnd</i>	0	0	0	1	1
<i>C</i> ₁	0	3	46	22	71
<i>C</i> ₂	52	15	6	28	101
<i>HA</i> ₁	0	37	19	5	61
<i>HA</i> ₂	0	0	0	1	1
<i>HA</i> ₃	0	42	20	7	69
<i>HA</i> ₄	0	0	0	1	1
<i>HN</i> ₁	52	54	15	15	136
<i>HN</i> ₂	0	2	1	1	4
<i>HN</i> ₃	1	46	15	7	69
<i>HN</i> ₄	1	2	0	1	4

Table 4 Average running time spent by the constructive algorithms

Heuristic	Grid (52)	Small (84)	Tree (50)	HB (62)	Average
<i>Rnd</i>	0.001	0.001	0.001	0.001	0.001
<i>C</i> ₁	5.539	0.001	0.003	0.405	1.263
<i>C</i> ₂	0.198	0.001	0.002	0.029	0.049
<i>HA</i> ₁	1.887	0.001	0.002	0.075	0.415
<i>HA</i> ₂	2.166	0.001	0.002	0.086	0.476
<i>HA</i> ₃	1.904	0.001	0.002	0.077	0.419
<i>HA</i> ₄	1.882	0.001	0.002	0.084	0.416
<i>HN</i> ₁	0.009	0.001	0.001	0.004	0.003
<i>HN</i> ₂	0.035	0.001	0.001	0.006	0.009
<i>HN</i> ₃	0.006	0.001	0.001	0.006	0.003
<i>HN</i> ₄	0.045	0.001	0.001	0.006	0.011

From the experimental results we can observe the following facts. There is no only one heuristic that outperforms the others for all the instances. For example, C_2 and HN_1 obtained the best average objective values and the largest number of best solutions found for datasets *Grid* and *HB*, and *Grid* and *Small*, respectively. Although C_2 possesses the best average solution quality (see Table 2), HN_1 reached the largest number of best solutions found. This is because C_2 performs better for dataset *HB*, which contains some of the biggest instances. Thus, the number of different objective values is larger since they range from 1 to $n - 1$. Therefore, the difference in solution quality between HN_1 and C_2 for dataset *Small* is lower than the one for dataset *HB*, which is reflected in the overall average.

Besides, C_1 was the best heuristic for dataset *Tree* in both solution quality and number of best solutions found. It is easy to see that some heuristics are more suitable than others for a specific dataset. Nevertheless, C_2 reached the best solution quality in average. In fact, the gap between C_2 and *Rnd* is quite large, i.e., 166.5 units, which means that C_2 is 9.9 times more effective than *Rnd* for the set of evaluated instances.

We can also observe that constructives HA_2 , HA_4 , HN_2 and HN_4 are even worse than *Rnd* in overall average solution quality. However, HN_2 and HN_4 outperform *Rnd* for datasets *Small* and *Tree*. The bad performance of these four constructives exhibits the importance of knowing the problem structure. In particular, the aforementioned constructives have something in common. They choose the vertices to be placed at positions $p = 2, \dots, n$ whose greedy value is maximum. Due to VSP is a minimization problem, this criterion does not favor this objective. In fact, the probability of constructing a high-quality solution is highly reduced when the vertex with the largest greedy value is chosen at each cut-point.

Regarding the execution time, *Rnd* is the fastest algorithm since it does not have to follow any criterion to choose the vertices. HN_1 , HN_2 , HN_3 , and HN_4 are the fastest constructives due to the reduced set of candidates vertices at each cut-point. The slowest heuristic is C_1 , spending 1.26 s per instance in average. This might be caused by the set of operations involved in the greedy function and the number of candidates to be evaluated.

5 Conclusions

In this chapter we approached the vertex separation problem. In particular, we proposed eight constructive heuristics for VSP. Each heuristic obtain an initial solution in a short time. We tested all our constructives with a set of 248 instances and compared the results with two of the best constructive heuristics available in the literature.

We additionally compared the resulting solutions from the constructives with a random solution. This allowed us to measure the impact of generating an initial solution by using ad-hoc heuristics instead of generating a solution randomly.

Experimental results showed that constructive C_2 achieves the best average solutions quality for the set of evaluated instances, i.e., 18.58 units against the 185.15 units obtained by Rnd . Thus, C_2 achieves an improvement in solution quality of 89.96 % with respect to Rnd .

Additionally, constructive HA_1 reaches 136 best solutions out of 248. This means that HA_1 has an effectiveness of 54.8 %, which is the best record. Rnd only found 1 best solution, which gives an effectiveness of 0.4 %. These results clearly show the importance of using a constructive heuristic instead of a random solution in the context of VSP.

Regardless of all criteria to select the vertices, all the algorithms are really fast. Especially HA_1 and HA_3 , which generate a solution in 0.003 s in average.

The solutions constructed by all the algorithms studied here can be used either as a good starting point in the solution space or as a good approximate solution for the vertex separation problem.

References

1. Duarte, A., Escudero, L., Martí, R., Mladenovic, N., Pantrigo, J., Sánchez-Oro, J.: Variable neighborhood search for the vertex separation problem. *Comput. Oper. Res.* **39**(12), 3247–3255 (2012)
2. Díaz, J., Petit, J., Serna, M.: A survey of graph layout problems. *ACM Comput. Surv.* **34**(3), 313–356 (2002)
3. Lengauer, T.: Black-white pebbles and graph separation. *Acta Informatica* **16**, 465–475 (1981)
4. Díaz, J., Penrose, M.D., Petit, J., Serna, M.: Approximating layout problems on random geometric graphs. *J. Algorithms* **39**(1), 78–116 (2001)
5. Goldberg, P.W., Golumbic, M.C., Kaplan, H., Shamir, R.: Four strikes against physical mapping of DNA. *J. Comput. Biol.* **2**(1), 139–152 (1995)
6. Gustedt, J.: On the path width of chordal graphs. *Discrete Appl. Math.* **45**(3), 233–248 (1993)
7. Monien, B., Sudborough, I.H.: Min cut is np-complete for edge weighted trees. *Theor. Comput. Sci.* **58**(1–3), 209–229 (1988)
8. Leiserson, C.: Area-efficient graph layouts (for VLSI). In: Proceedings of IEEE Symposium on Foundations of Computer Science, pp. 270–281 (1980)
9. Bodlaender, H., Gustedt, J., Telle, J.: Linear time register allocation for a fixed number of registers. In: Proceedings of the Symposium on Discrete Algorithms (1998)
10. Kornai, A.: Narrowness, path-width, and their application in natural language processing. *Discrete Appl. Math.* **36**, 87–92 (1997). (Elsevier Science Publishers B. V. (1992))
11. Lopes, I., de Carvalho, J.: Minimization of open orders using interval graphs. *IAENG Int. J. Appl. Math.* **40**(4), 297–306 (2010)
12. Luque, G., Alba, E.: Metaheuristics for the DNA fragment assembly problem. *Int. J. Comput. Intell. Res.* **1**(2), 98–108 (2005)
13. Sánchez-Oro, J., Pantrigo, J., Duarte, A.: Combining intensification and diversification strategies in VNS. An application to the Vertex separation problem. *Comput. Oper. Res.* **52** (part B), 209–219 (2013)
14. VSPLIB 2012. Home page: <http://www.optsicom.es/vsp/>
15. Pantrigo, J.J., Martí, R., Duarte, A., Pardo, E.G.: Scatter search for the cutwidth minimization problem. *Ann. Oper. Res.* **199**(1), 285–304 (2012)

Part VII
Fuzzy Logic Applications

A New Approach for Intelligent Control of Nonlinear Dynamic Plants Using a Benchmark Problem

Leticia Cervantes and Oscar Castillo

Abstract In this paper an approach for control is illustrated, this method is applied in complex control problems, and this is the main objective to use an approach to solve complex control problems and obtain better results. In this paper simulation results are presented when the proposed method is applied using a benchmark problem. The first part in this paper shows the proposed method for intelligent control combining the outputs of the modules, then the approach is illustrated using a problem with 2 DC motors in a simulation plant, the proposed method shows how we can achieve the total control of the case of study.

1 Introduction

This paper is based on fuzzy logic, granular computing and control area [3–8, 10], when two or more different areas work together to solve any problem, results obtained can be better [12–16]. We explain and illustrate the proposed approach with a benchmark problem, this case of control [17, 18, 21–23] is an example of a simulation plant with two DC Motors. Each motor has a different behavior and both are connected in the same simulation plant [24–26]. In this case of study we illustrate the complete total control of the problem. The main contribution in this paper is applied a granular approach that improves complex control problems. The rest of the paper is organized as follows: In Sect. 2 we present some basic concepts to understand this work, in Sect. 3 we define the proposed method, Sect. 4 describes automatic design of DC motor and simulation results, and finally conclusions are presented in Sect. 5.

L. Cervantes · O. Castillo (✉)
Tijuana Institute of Technology, Tijuana, Mexico
e-mail: ocastillo@tectijuana.mx

L. Cervantes
e-mail: letty2685@hotmail.com

2 Background and Basic Concepts

In this section we provide some concepts to understand this work.

2.1 Granular Computing

Granular computing is based on fuzzy logic. There are many misconceptions about fuzzy logic. To begin with, fuzzy logic is not fuzzy. Basically, fuzzy logic is a precise logic of imprecision. Fuzzy logic is inspired by two remarkable human capabilities. First, the capability to reason and make decisions in an environment of imprecision, uncertainty, incompleteness of information, and partiality of truth. And second, the capability to perform a wide variety of physical and mental tasks based on perceptions, without any measurements and any computations. The basic concepts of graduation and granulation form the core of fuzzy logic, and are the main distinguishing features of fuzzy logic. More specifically, in fuzzy logic everything is or is allowed to be graduated, i.e., be a matter of degree or, equivalently, fuzzy. Furthermore, in fuzzy logic everything is or is allowed to be granulated, with a granule being a clump of attribute values drawn together by indistinguishability, similarity, proximity, or functionality. The concept of a generalized constraint serves to treat a granule as an object of computation. Graduated granulation, or equivalently fuzzy granulation, is a unique feature of fuzzy logic. Graduated granulation is inspired by the way in which humans deal with complexity and imprecision. The concepts of graduation, granulation, and graduated granulation play key roles in granular computing. Graduated granulation underlies the concept of a linguistic variable, i.e., a variable whose values are words rather than numbers. In retrospect, this concept, in combination with the associated concept of a fuzzy if-then rule, may be viewed as a first step toward granular computing [1, 2, 9, 20]. Granular Computing (GrC) is a general computation theory for effectively using granules such as subsets, neighborhoods, ordered subsets, relations (subsets of products), fuzzy sets (membership functions), variables (measurable functions), Turing machines (algorithms), and intervals to build an efficient computational model for complex with huge amounts of data, information and knowledge [29].

2.2 DC Motor

A brushed DC motor is the simplest of all motor types, and typically consists of the following parts; Stator is the stationary part of the motor in which the rotor revolves. The stator typically takes the form of a metal can, open at one end, with two or more curved magnets mounted inside it. The stator housing often doubles up as the housing for the motor as a whole. Rotor is the rotating part of the motor,

mounted axially in the centre of the stator housing. The motor windings are wound on the rotor. Brushes are a series of electromechanical contacts which enable current to flow to the rotating motor windings in the correct sequence and direction. The stationary brushes make electrical contact with part of the rotor known as the commutator. This arrangement creates the correct sequence of current through the motor coils as the rotor rotates. The commutator is usually in the form of a cylinder, consisting of segments of conducting material interspersed with, and insulated from one another by, an insulating material. Current flow is in through the left hand brush and out through the right hand brush, and is indicated by the red arrows. Unlike other electric motor types (i.e., brushless DC, AC induction), BDC motors do not require a controller to switch current in the motor windings. Instead, the commutation of the windings of a BDC motor is done mechanically. A segmented copper sleeve, called a commutator, resides on the axle of a BDC motor. As the motor turns, carbon brushes slide over the commutator, coming in contact with different segments of the commutator. The segments are attached to different rotor windings, therefore, a dynamic magnetic field is generated inside the motor when a voltage is applied across the brushes of the motor. It is important to note that the brushes and commutator are the parts of a BDC motor that are most prone to wear because they are sliding past each other [11].

3 Proposed Method

The main objective in this paper is to test the proposed method in complex control problems and evaluate the behavior of the case of study. The scheme of this approach is shown in Fig. 1.

The use of the Type-1 granular model is a contribution of this paper to improve the solution of control problem that is going to be considered. The problem is divided in sub-problems to control each controller individually and then when the individual control is obtained the results of each controller are going to the

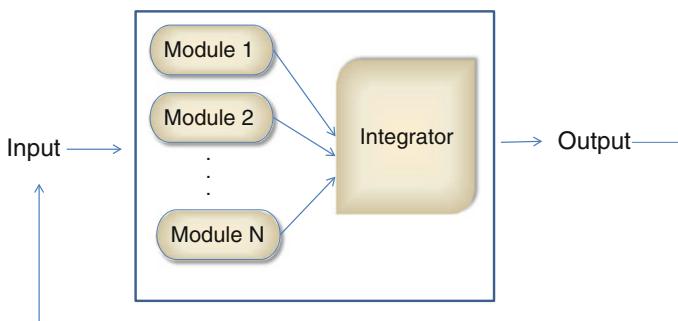


Fig. 1 Proposed approach for control

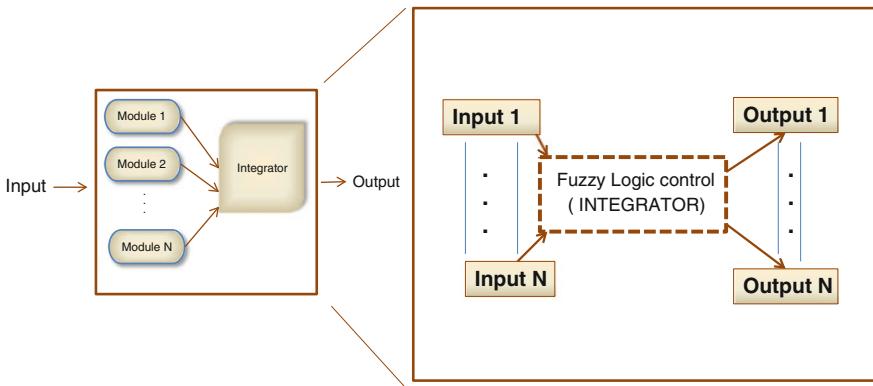


Fig. 2 Proposed granular fuzzy logic control

integrator (all the behavior of each individual controllers going to the integrator). We can use this architecture in many cases to develop each controller separately. In Fig. 2 we show an example that how can we improve this method in a complex problem with more than one controller.

4 Automatic Design of DC Motor Control

In this section we consider the case of study when in a complete control problem we have two or more individual controllers, in this case of study we have a simulation plant where is necessary to control two DC motors, but each motor has a different behavior. First we show the scheme to control it, and this is illustrated in Fig. 3.

In Fig. 3 we present the control problem, but in this case two DC motors need to be controlled, to apply the proposed method first we need to divide the total problem in sub-problems, and this is illustrated in Fig. 4.

Having the total problem divided in sub-problems a fuzzy system was designed for each DC motor, both of the motors have 2 inputs and one output. The inputs are (error and change), output is the control of the motor [19, 27, 28, 30]. The first fuzzy systems for the first DC motor are shown in Figs. 5, 6, 7 and 8.

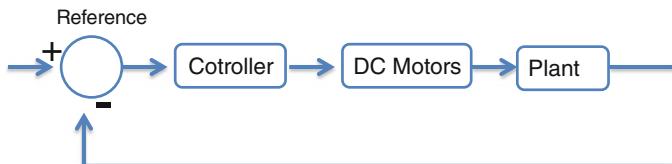


Fig. 3 Control scheme of the problem

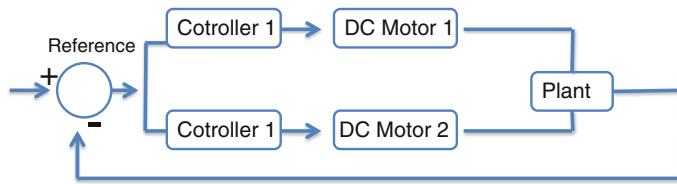


Fig. 4 Control scheme of the problem with sub-problem

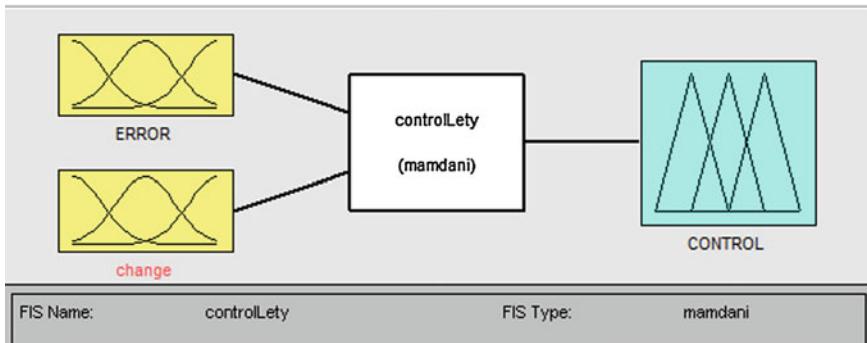


Fig. 5 Fuzzy system for DC controller 1

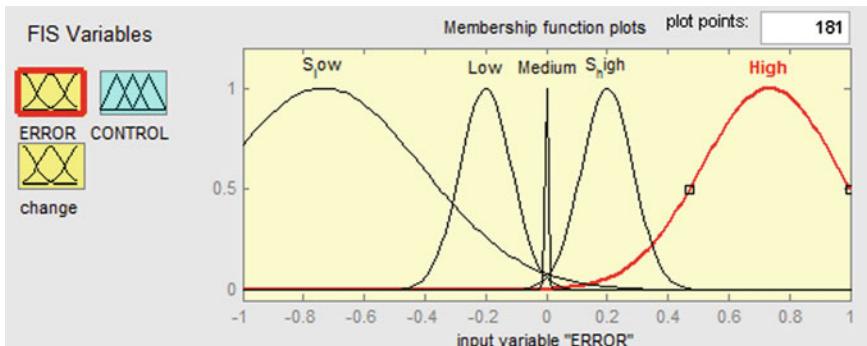


Fig. 6 Input for DC controller 1

Then the Fuzzy systems for the second controllers are illustrated in Figs. 9, 10 and 11.

The rules used in both controllers are different because of the different behavior of the DC motors, rules of the controllers are presented in Figs. 12 and 13.

Is important to mention that the simulation was performed using different types of membership functions and when we use different types of membership function. After using the fuzzy systems as individual controllers we proceed to use the

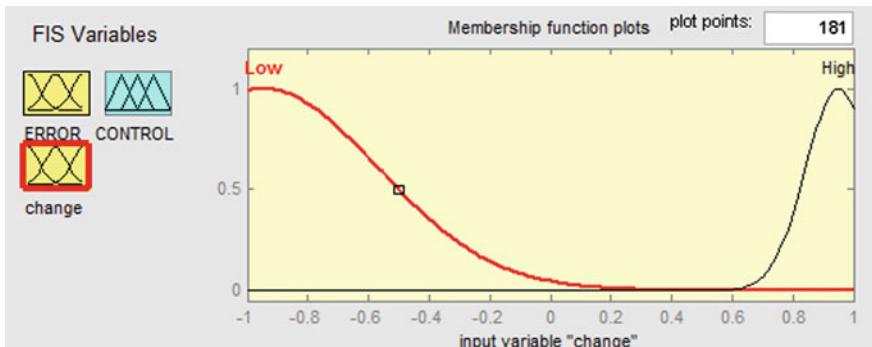


Fig. 7 Input for DC controller 1

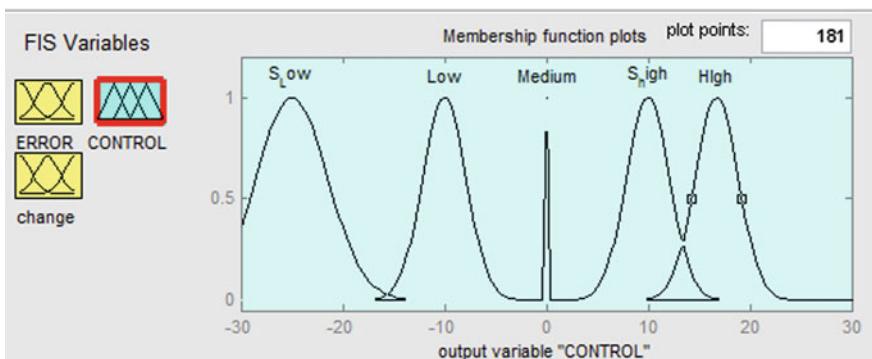


Fig. 8 Output for DC controller 1

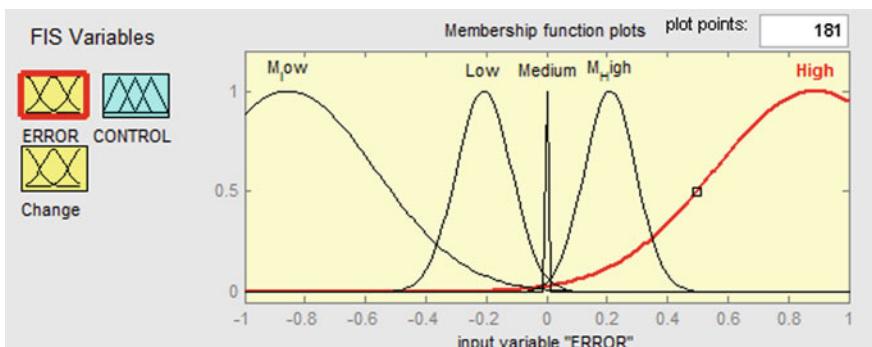


Fig. 9 Input of DC controller 2

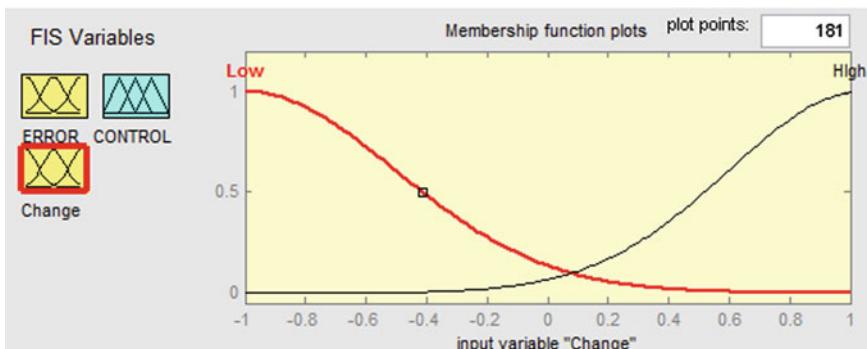


Fig. 10 Input of DC controller 2

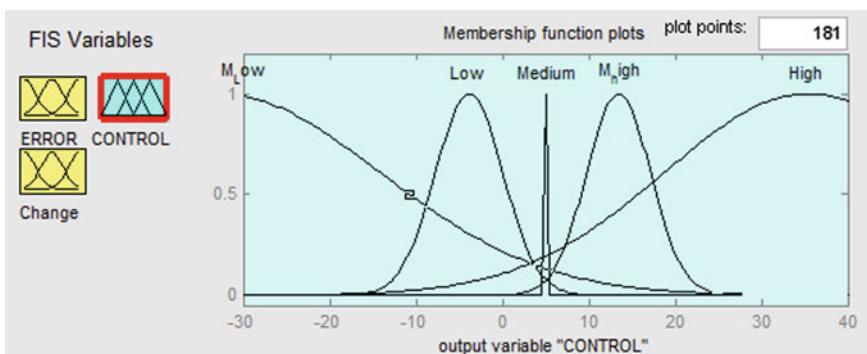


Fig. 11 Output of DC controller 2

1. If (ERROR is High) then (CONTROL is High) (1)
2. If (ERROR is S_low) then (CONTROL is S_Low) (1)
3. If (ERROR is Medium) and (CAMBIO is Low) then (CONTROL is Low) (1)
4. If (ERROR is Medium) and (CAMBIO is High) then (CONTROL is S_high) (1)
5. If (ERROR is Low) then (CONTROL is Low) (1)
6. If (ERROR is S_high) then (CONTROL is S_high) (1)
7. If (ERROR is Medium) then (CONTROL is Medium) (1)

Fig. 12 Rules for the DC motor 1

approach where we took the values of the outputs of each individual controller and this outputs going to the integrator as an inputs to obtain a new outputs and improve the control. The fuzzy integrator is shown in Figs. 14, 15, 16, 17 and 18.

The rules of the integrator for the controller are presented in Fig. 19.

Simulation results using different types of membership functions and results when we applied the integrator are shown in Table 1.

1. If (ERROR is High) then (CONTROL is High) (1)
2. If (ERROR is M_low) then (CONTROL is M_Low) (1)
3. If (ERROR is M_High) then (CONTROL is M_high) (1)
4. If (ERROR is Low) then (CONTROL is Low) (1)
5. If (ERROR is Medium) and (CAMBIO is Low) then (CONTROL is High) (1)
6. If (ERROR is Medium) and (CAMBIO is Hlgh) then (CONTROL is M_Low) (1)
7. If (ERROR is Medium) then (CONTROL is Medium) (1)

Fig. 13 Rules for the DC motor 2

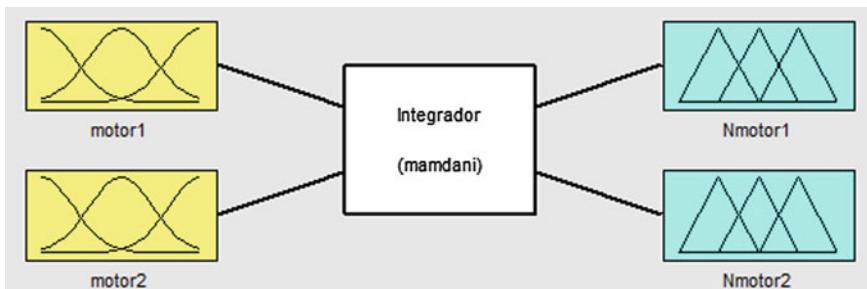


Fig. 14 Integrator for the controllers

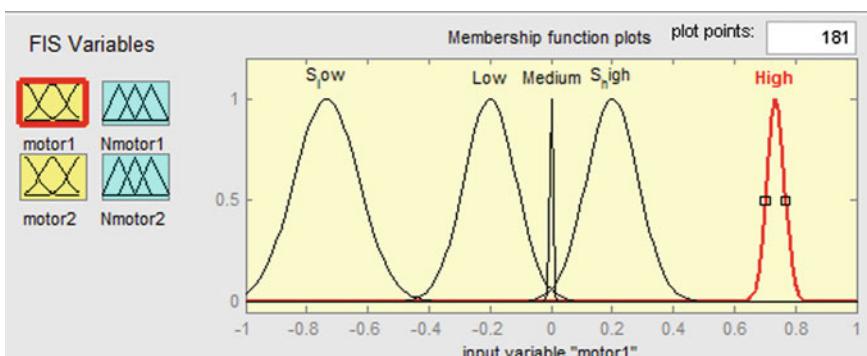


Fig. 15 Input 1 of the integrator

In Table 1 results are presented using the individual controllers with different types of membership functions and we can appreciate that Gaussian membership functions has better behavior individually but when the integrator is applied the error of both controllers decreases.

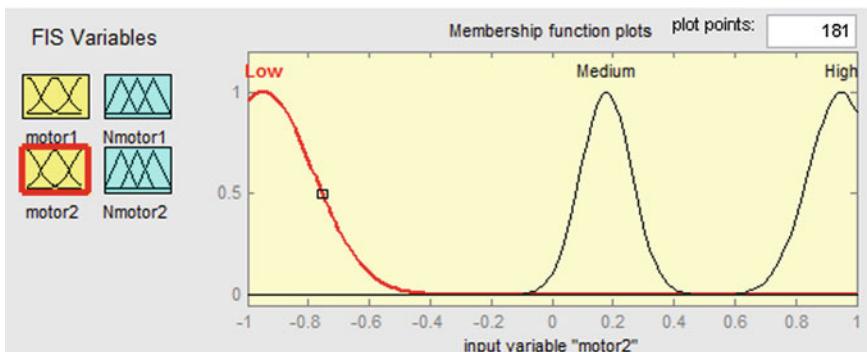


Fig. 16 Input 2 of the integrator

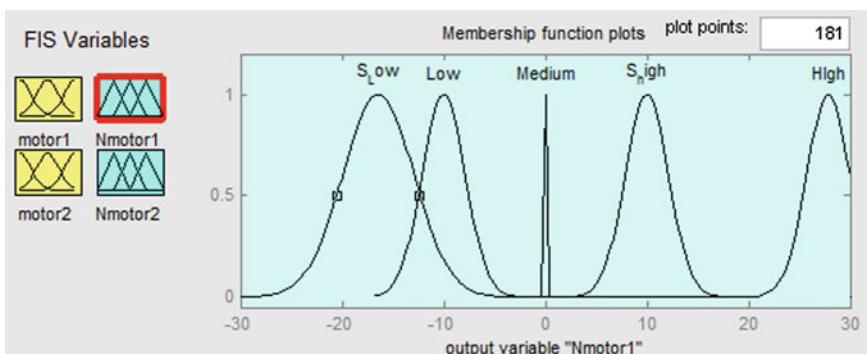


Fig. 17 Output 1 of the integrator

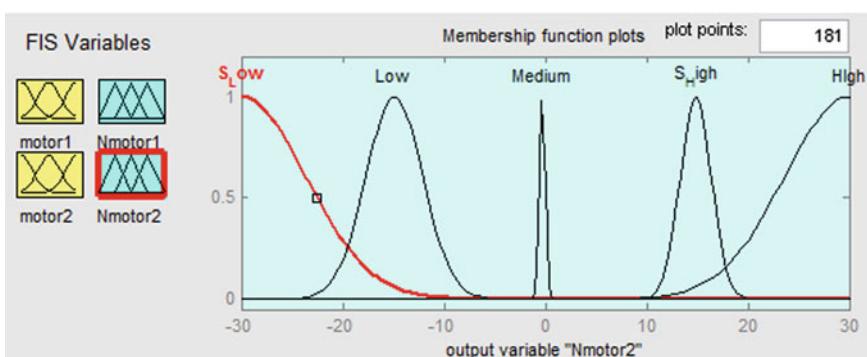


Fig. 18 Output 2 of the integrator

1. If (motor1 is High) and (motor2 is High) then (Nmotor1 is High)(Nmotor2 is S_High) (1)
2. If (motor1 is Medium) and (motor2 is Medium) then (Nmotor1 is Medium)(Nmotor2 is Medium) (1)
3. If (motor1 is S_low) and (motor2 is Low) then (Nmotor1 is S_Low)(Nmotor2 is S_Low) (1)
4. If (motor1 is Low) and (motor2 is Low) then (Nmotor1 is Low)(Nmotor2 is Low) (1)
5. If (motor1 is S_high) and (motor2 is High) then (Nmotor1 is S_high)(Nmotor2 is S_High) (1)
6. If (motor1 is S_high) and (motor2 is High) then (Nmotor1 is S_high)(Nmotor2 is High) (1)
7. If (motor1 is High) and (motor2 is Medium) then (Nmotor1 is Medium)(Nmotor2 is High) (1)
8. If (motor1 is Medium) and (motor2 is High) then (Nmotor1 is Medium)(Nmotor2 is S_High) (1)
- 9. If (motor1 is High) and (motor2 is High) then (Nmotor1 is S_high)(Nmotor2 is S_High) (1)**

Fig. 19 Rules for the integrator**Table 1** Results for the simulation plant

	Motor 1	Motor 2
Gaussian	0.6359	0.6136
Triangular	0.7043	0.6410
Trapezoidal	0.7092	0.6102
Bell	0.6486	0.6151
Integrator	0.0065	0.0059

5 Conclusions

In this paper we could appreciate that when we use fuzzy systems to control in this case two motors we can obtain good results but to decrease the error we needed to implement the proposed method where the outputs of each motor were going to an integrator to make an input, once having the inputs in the integrator we obtained new outputs to achieve the control. In this case we applied the granularity when we divide the problems in sub-problems and we granulate the individual fuzzy controller in an integrator.

References

1. Bargiela, A., Wu, P.: Granular Computing: An Introduction. Kluwer Academic Publish, Dordercht (2003)
2. Bargiela, A., Wu, P.: The roots of granular computing. In: GrC, pp. 806–809. IEEE, New York (2006)
3. Blakelock, J.: Automatic Control of Aircraft and Missiles. Prentice-Hall, New Jersey (1965)
4. Cervantes, L., Castillo, O.: Design of a fuzzy system for the longitudinal control of an F-14 airplane. In: Soft Computing for Intelligent Control and Mobile Robotics, vol. 318, pp. 213–224. Springer, Berlin (2011)
5. Cervantes, L., Castillo, O.: Intelligent control of nonlinear dynamic plants using a hierarchical modular approach and type-2 fuzzy logic. In: Lecture Notes in Computer Science, vol. 7095, pp. 1–12. Springer, Berlin (2011)
6. Cervantes, L., Castillo, O.: Hierarchical genetic algorithms for optimal type-2 fuzzy system design. In: Annual Meeting of the North American Fuzzy Information Processing Society, pp. 324–329 (2011)

7. Cervantes, L., Castillo, O.: Automatic design of fuzzy systems for control of aircraft dynamic systems with genetic optimization. In: World Congress and AFSS International Conference, pp. OS-413-1–OS-413-7 (2011)
8. Cervantes, L., Castillo, O.: Comparative Study of Type-1 and Type-2 Fuzzy Systems for the Three Tank Water Control Problem, LNAI 7630, pp. 362–373. Springer, Berlin (2013)
9. Computer Society: The 2011 IEEE Internaional Conference on Granular Computing Sapporo, Japan, IEEE GrC (2011)
10. Dorf, R.: Modern Control Systems. Addison-Wesley Pub Co, Reading (1997)
11. Hill, C.: An Introduction to Low Voltage DC Motors. Koninklijke Philips Electronics N.V (2004)
12. Jamshidi, M., Vadiee, N., Ross, T.: Fuzzy Logic and Control: Software and Hardware Applications, vol. 2. Prentice-Hall, New Jersey (1993)
13. Kadmiry, B., Driankov, D.: A fuzzy flight controller combining linguistic and model based fuzzy control. *Fuzzy Sets Syst.* **J.** **146**(3), 16, pp. 313–347 (2004)
14. Karnik, N., Mendel, J.: Centroid of a type-2 fuzzy set. *Inf. Sci.* **132**, 195–220 (2001)
15. Keviczky, T., Balas, G.: Receding horizon control of an F-16 aircraft: a comparative study. *Control Eng. Pract.* **J.** **14**(9), 1023–1033 (2006)
16. Liu, M., Naadimuthu, G., Lee, E.S.: Trayectory tracking in aircraft landing operations management using the adaptive neural fuzzy inference system. *Comput. Math. Appl.* **J.** **56**(5), 1322–1327 (2008)
17. Melin, P. Castillo, O.: Intelligent control of aircraft dynamic systems with a new hybrid neuro-fuzzy–fractal approach. *J. Inf. Sci.* **142**(1), 161 (2002)
18. Melin, P., Castillo, O.: Adaptive intelligent control of aircraft systems with a hybrid approach combining neural networks, fuzzy logic and fractal theory. *J. Appl. Soft Comput.* **3**(4), 353 (2003)
19. Mendel, J.: Uncertain Rule-Based Fuzzy Logic Systems: Introduction and New Directions. Prentice-Hall, New Jersey (2001)
20. Pedrycz, W., Skowron, A. et al.: Handbook Granular Computing. Wiley Interscience, New York (2008)
21. Rachman, E., Jaam, J., Hasnah, A.: Non-linear simulation of controller for longitudinal control augmentation system of F-16 using numerical approach. *Inf. Sci. J.* **164**(1–4), 47–60 (2004)
22. Reiner, J., Balas, G., Garrard, W.: Flight control design using robust dynamic inversion and time scale separation. *Autom. J.* **32**(11), 1493–1504 (1996)
23. Sanchez, E., Becerra, H., Velez, C.: Combining fuzzy, PID and regulation control for an autonomous mini-helicopter. *J. Inform. Sci.* **177**(10), 1999–2022 (2007)
24. Sefer, K., Omer, C., Okyay, K.: Adaptive neuro-fuzzy inference system based autonomous flight control of unmanned air vehicles. *Expert Syst. Appl.* **J.** **37**(2), 1229–1234 (2010)
25. Song, Y., Wang, H.: Design of flight control system for a small unmanned tilt rotor aircraft. *Chin. J. Aeronaut.* **22**(3), 250–256 (2009)
26. Walker, D.J.: Multivariable control of the longitudinal and lateral dynamics of a fly by wire helicopter. *Control Eng. Pract.* **11**(7), 781–795 (2003)
27. Wu, D.: A Brief Tutorial on Interval Type-2 Fuzzy Sets and Systems (2010)
28. Wu, D., Jerry, Mendel: On the continuity of type-1 and interval type-2 fuzzy logic systems. *IEEE Trans. Fuzzy Syst.* **19**(1), 179–192 (2011)
29. Zadeh, L.A.: Some reflections on soft computing, granular computing and their roles in the conception, design and utilization of information/intelligent systems. *Soft. Comput.* **2**, 23–25 (1998)
30. Zadeh, L.A.: Outline of a new approach to the analysis of complex systems and decision processes. *IEEE Trans. Syst. Man Cybern. SMC-3* 28–44 (1973)

Fuzzy Pre-condition Rules for Activity Sequencing in Intelligent Learning Environments

Francisco Arce and Mario García-Valdez

Abstract In a learning environment a participant interacts, uses their skill, creates or uses tools and artifacts to obtain and interpret information to construct their learning. Intelligent learning environments are those that use computational systems and devices to enhance the learning process. To establish an order in the activities to be performed in these environments initiatives have been used as the simple sequenced which establishes a learning sequence which guides the learner through the learning activity. As each learner or user learns differently these environments it's hard to supply them appropriate activities. In this paper we propose the modification on the inputs of the precondition rules of the IMS Simple Sequencing Specification using fuzzy logic to provide appropriate resources for each activity in the sequence.

1 Introduction

Learning environments (LE) are places where learning is created; these can be classrooms, museums, environment etc. [17]. According to Philips [17] a learning environment is a place where resources, time and reasons are available for a group of people to nurture, support and value the learning of a limited set of information. The LE are social places even when only one person is found there. One of the challenges facing the design of learning environments is human complexity, because each person thinks and assimilate information in different ways making it difficult to identify which resources are adequate for everyone. Intelligent learning

F. Arce · M. García-Valdez (✉)
Tijuana Institute of Technology, Tijuana, Mexico
e-mail: mario@tectijuana.edu.mx

F. Arce
e-mail: francisco.j.arce@gmail.com

environments (ILE) are a new type of intelligent educational system, which combines characteristics of traditional intelligent tutoring systems (ITS) [11] and learning environments. According to Self [14] ITS are learning systems based on computers that try to adapt to the needs of the learner. ITS research is the only one of the IT's in general which has as its scientific objective create forms of psychological and social educational knowledge. In some ILEs to motivate the learner, he is placed in realistic contexts and authentic representations of tasks in the real world. Resources are given to the user which will help him do these tasks. The resources in the LE are represented in 2 ways, the first is physically (books, printed images, toys etc.), the other is digitally where devices are used to display images, texts, videos, audios helping to contextualize the activity or the environment itself. Currently there are other devices besides computers that can be used in the LE as Smartphone's, Tablet's, projectors, video game consoles etc. The challenge for LE (as we mentioned before) is to select appropriate resources but must also be a learning sequence that satisfies the user needs and make sense of what is being learned. There are many techniques for establishing learning sequences as [1, 6, 8, 10]. In this case we focus on the simple sequence [9] which has the task of generating a sequence of activities of the subject to be learned by the user. In each of these activities there are precondition rules [9] which serve to make decisions about what to do with activities like hide, show, skip or repeat the activity. The rules require certain values to make these decisions for example when there is an activity that displays information that the user may already know and do not need to see it this activity is skipped or if an activity requires the user to have some age to view it. Fuzzy logic has been previously used in learning environments work as [4, 15] the advantage offered by fuzzy logic is that it can work with linguistic variables which take values such as "you are young" and turn it into something measurable as 18 years. We propose to use fuzzy logic on the precondition rules. This paper is organized as follows: In the Sect. 2 basic concepts about learning environments, ILE, ITS and IMS Simple Sequencing. The proposal is presented in Sect. 3. Section 4 contains the experiment performed. Section 5 contains the experimental results and in Sect. 6 the conclusions of this work are presented.

2 Concepts

Duarte [5] defines the learning environment as a derivative of the interaction between man and the natural environment that surrounds it as classrooms. However, these learning environments are given not only in educational institutions, whereas Duarte [5] thinks that "human beings need to continuously learn in and out of school", e.g. home, the street, the malls among others. Learning environments inside and outside the classroom are derived from the idea that man,

individually and organized in a social group, is developed in multiple activities, specific scenarios such as sunlight, soil, air and others, like scenarios social as friends, family, school, highlighting the interaction or form of emotional communication that occurs. This shows the learning environments as a specific context within the school and other spaces. There are learning systems that use learning environments such as Intelligent Tutoring Systems (ITS), this systems cover any computer program that has some intelligence and can be used in learning [3, 13]. The traditional ITS model contains four components: the domain model, student model, teaching model, and a learning environment or user interface. Brusilovsky [2] says that The Intelligent Learning Environments (ILE) is based in learning environments where students and teachers can create knowledge. In other words, the environment represents a cognitive space for a learning community. The ILE seeks to provide adaptive navigation and adaptive sequencing as is commented on [7, 15]. The adaptive navigation presents the content of a course in optimized order, where the optimization criteria takes into consideration the learner's backgrounds and performance, whereas adaptive sequencing is defined as the process for selection of learning objects from a digital repository and sequencing them in a way which is appropriated for the targeted learning community or individuals [12, 16]. The **IMS Simple Sequencing** [10] specification defines the required behaviors and functionality that conforming systems must implement. It incorporates rules that describe the branching or flow of instruction through content according to the results of the interactions of the learner with the content. The main components of this standard are the Learning Activities and Tree Activities. A Learning Activity is denied as a pedagogically neutral unit of instruction, knowledge, evaluation, etc.

3 Proposal

When a learning sequence is generated learning activities are established and they not change, what changes is the action as if you skip, performed, jump, hide, repeat or show an activity, this decision is taken by the rules of precondition. In this paper we seek to generate diversity of resources displayed in the activities adapting the resources of the activity to the user, for example we have an introductory activity of a particular subject and 2 users of different ages perform the activity, the first user of 25 years will be shown more detailed and textual information, while the second user of 6 years of age is in a very early stage of learning resources that show you are easier to understand and have more audio-visual content. With fuzzy logic generate inputs for use in the precondition rules for example: fuzzy rule *if user.age is low and user.academic_level is low then user.level is begginer*. On the precondition rule we will use the output of the fuzzy rule will use the output of that fuzzy rule as follows *if user = begginer then pre_condition = show*. Thus a set of resources is obtained for a novice user and these are shown on the right devices for each resource through a web browser running in this device as we can see on the Fig. 1.

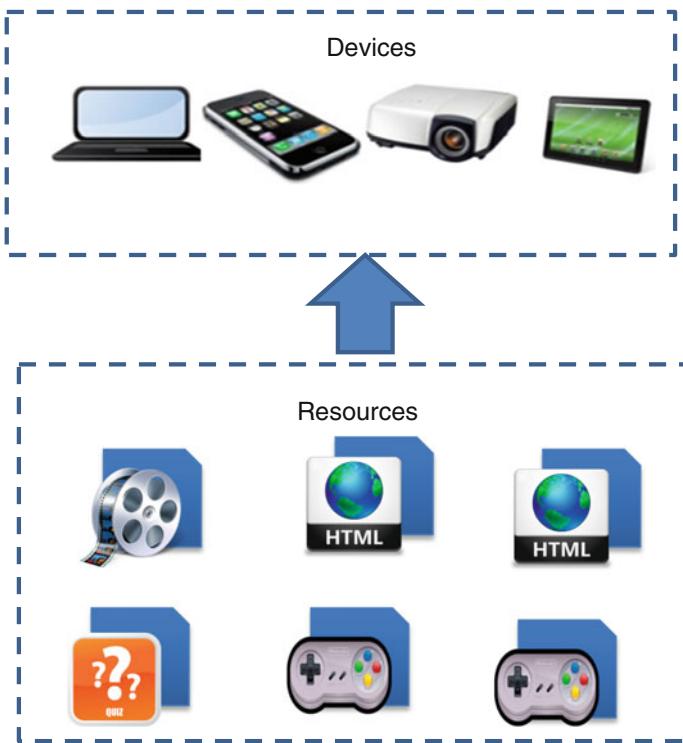


Fig. 1 Set of resources and their devices

4 Experiment

We performed an experiment with a group of 10 users with different profiles (Table 1) with the prototype environment to see the impression that each user had at performing a learning activity and get feedback from their experience. This environment is planned for use in large areas but in this case a small space was used for the test. The devices used in the test were 1 computer, a 7-inch tablet as a master device, headphones and monitors with different resolutions are used to emulate a space (Fig. 2) where each monitor represented a wall.

The learning activity used in the experiment was the Mexican national anthem, which features 10 activities. At the beginning of the test, the user was instructed with few instructions about how to use environment and what would happen in each activity, in each activity the content displayed in each device was changed until they reached the last activity. Every activity contained an audio explaining the contents of the activity such as the activity “lyric” the lyric was displayed on the central monitor and on the background national anthem music was played. Here the user had control of the navigation on learning sequence in other words the user decided when moving to the next activity.

Table 1 User profiles: for this case only age and academic level which ranges from 1 to 21 where 1 is a first grader and 21 have PhD was used

User	Age	Academic level
1	6	1
2	48	19
3	23	17
4	23	18
5	23	18
6	24	17
7	31	19
8	35	20
9	45	21
10	29	19



Fig. 2 Devices used in the test environment

5 Results

When users ended up using the environment were conducted a small survey, this survey consisted of 5 questions about your experience using ambient questions as if it is difficult for the user to use it or if he liked the resources that was shown in the activities, in order to obtain user feedback for future enhancements and changes the results can be seen in Tables 2 and 3.

The results of the survey were positive about acceptance and the enjoy of the environment, the environment had a score of 9 in an 1 to 10 scale on how easy it

Table 2 Survey part 1

Questions	Mean
From 1 to 10 how easy it was use the environment?	9
From 1 to 10, how was your satisfaction about the information displayed in the environment?	9.3

Table 3 Survey part 2

Questions	Answer	Frequency
Did you like the idea of using various visual elements?	Yes	8
	No	2
Was necessary technical assistance to use the environment?	Yes	0
	No	10
Would you recommend the use of this environment to others?	Yes	10
	No	0

was to use the environment and none of the users need technical assistance, we can say that the environment does not require special training to learn to use the environment and can be mastered with a little instruction. 8 out of 10 users would have liked the visual elements of the activities, it was not a low rating but this means we need to work a little more on the quality of the visual content. And the 10 users agree that they would recommend using this intelligent learning environment.

6 Conclusions

After observing that user reactions were positive at the use environment with resources tailored to each user we conclude that the test provided good feedback on the use and content of the environment. For future work, we plan to use other learning activities with different content (also improve the quality thereof) to use in a large room (a classroom or a room) in which will improve the user experience by interacting with other devices besides the master device. Also we will use a more diverse group of users unlike this test.

References

1. Beckett, D. (ed): Resource Description Framework (RDF): W3C Recommendation (2004)
2. Brusilovsky, P.: Student model centered architecture for intelligent learning environments. In: International Center for Scientific and Technical Information, Kuusinen str. 21b, Moscow 125252 (1994) (Russia plb@plb.icsti.su)

3. Burton, R.R.: The environment module of intelligent tutoring systems. In: Polson, M.C., Richardson, J.J. (eds.) *Foundations of Intelligent Tutoring Systems*. Lawrence Erlbaum Associates, Hillsdale (1988)
4. Dattolo, A., Vicenzo, L.: A fuzzy approach to adaptive hypermedia. In: Proceedings 4th International Conference on Knowledge Based Intelligent Engineering Systems and Allied Technologies, vol. 2, pp. 700–703 (2000)
5. Duarte, J.: Learning environments. A conceptual approach. *Estudios pedagógicos* (Valdivia) (29), 97–113. Recuperado en 15 de noviembre de 2014, de http://www.scielo.cl/scielo.php?script=sci_arttext&pid=S0718-07052003000100007&lng=es&tlang=es. doi:10.4067/S0718-07052003000100007
6. Fischer, S.: Course and exercise sequencing using metadata in adaptive hypermedia learning systems. *ACM J. Educ. Resour. Comput.* **1** (2001)
7. Hannafin, M., Land, S., Oliver, K.: Open learning environments: foundations, methods, and models. In: Reigeluth, C. (ed.) *Instructional Design Theories and Models*, pp. 115–140. Lawrence Erlbaum Associates, Mahwah, NJ (1999)
8. Herder, E.: Modeling user navigation. In: UM 2003, LNAI 2702, pp. 417–419 (2003)
9. IMS Global Learning Consortium, I: IMS simple sequencing best practice and implementation guide version 1.0 final specification. Rep. tec. IMS Global Learn. Consort. Inc reviewed in Mar 2012 (2003)
10. Intelligence in Education: **10**, 350–364 (1999)
11. J. Psotka, Sharon A. Mutter: *Intelligent Tutoring Systems: Lessons Learned*. Lawrence Erlbaum Associates (1988) (ISBN 0-8058-0192-8)
12. Kolmogorov, G.: Decision support models for composing and navigating through e-learning objects. In: Proceedings of the 36th Hawaii International Conference on System Sciences (HICSS03) (2003)
13. Rieber, L.: Designing learning environments that exit serious plays. In : Annual meeting of the Australasian Society in Learning in Tertiary Education (ASCLITE), Melbourne, Australia (2001)
14. Self, J.: The defining characteristics of intelligent tutoring systems, research: its care, precisely. *Int. J. Artif. Intell. Educ.* **10**, 350–364 (1999)
15. Valdez, M., Sandoval, G., Castillo, O., Garza, A.: *A Fuzzy Approach for the Sequencing of Didactic Resources in Educational Adaptive Hypermedia Systems*. Springer, Berlin (2007)
16. Van der Veer, G.C.: Human-computer interaction: learning, individual differences, and design recommendations. Albllasserdam: Haveka (1990)
17. Phillips, R.: Towards a generalised conceptual framework for learning: the learning environment, learning processes and learning outcomes (LEPO) framework. Educational Development Unit, Murdoch University, Western Australia, Australia (2010)

A Pre-filtering Based Context-Aware Recommender System using Fuzzy Rules

Xochilt Ramirez-Garcia and Mario Garcia-Valdez

Abstract Most of recommendation systems work with traditional recommender algorithms such as collaborative filtering. Recently context has been incorporated in these algorithms as a fundamental factor to improve the quality of recommendations provided in the context of the user. Context is used as a filter for selecting items suitable for the current situation of the user. This work contributes to the improvement of contextual recommendations through the proposed method that uses a pre-filtering approach, traditional collaborative filtering and fuzzy rules. The Movie lens dataset was used for testing the method and the experiments conducted show promising results.

1 Introduction

Context-Aware Recommender Systems (CARS) can improve recommendations by taking into account the current situation of the user. K. Dey defines context as: “any information used to characterize the situation of an entity. An entity is a person, place or object considered relevant to the interaction user-application, including user and applications themselves” [1]. CARS have the challenge of selecting the most important features from a very large search space of not only all possible situations but also the changes from one to another. Sometimes, the physical features in the context may be important; while in another may be completely irrelevant. CARS uses context to model the user situation to get contextual recommendations. In previous work pre-filtering approach was used for a restaurant domain, but the results are not satisfactory because of the sparsity problem in the

X. Ramirez-Garcia
Tijuana Institute of Technology, Tijuana, Mexico
e-mail: xochilt.ramirez@gmail.com

M. Garcia-Valdez (✉)
Instituto Tecnológico de Tijuana, Tijuana, Mexico
e-mail: mario@tectijuana.edu.mx

ratings matrix. Then we propose the use of pre-filtering approach with a robust dataset, however, in this dataset we found problems of sparsity and null context. Despite these problems, pre-filtering approach has advantages such as: (1) they can be adapted in current recommender system and, (2) gets contextual recommendations using a traditional collaborative filtering. For this work, the dataset have contextual information almost null about the users situations, therefore the context was defined with the “time” when the ratings were done for each item, which means the benchmark for data segmentation was the date.

Contextual information is normally described using linguistic terms, i.e., using measuring terms as “a lot”, “little”, “sometimes”, etc. The linguistic variables are used in a Fuzzy Inference System (FIS) for obtain a weight value that defines the significance level of the recommendation for the user; these variables are: (1) *popularity* of items for a particular context and, (2) *user participation* in the same context (see Sect. 3 for an explanation more detailed).

The contents of this work are presented as follows: in Sect. 2, related works is presented; in Sect. 3, the proposed method is explained; Sect. 4, the setup for the experiments is explained; in Sect. 5, the experimental results are described; in Sect. 6, conclusions about this work are presented; and finally, in Sect. 7 extension to current work is discussed.

2 Related Work

The time in RS represents a feature about the user tastes in the context, the time influence in the choice of one or another item can change through a period, the user behavior related with the way of rated an item is different when the time (as context) is, for example, in a different season. In a day, time is changing, it can be segmented in several contexts as “morning”, “afternoon”, “evening”, “midnight”, “noon”, etc., that means that for instance, for the user each context is a different situation when he/she is going to watch a movie in the morning or in the evening. Then, time as contextual factor in CARS has influence when the user put a rating.

The CARS proposed in this work uses post-filtering and time segmentation using the ratings date for define the contexts, another researches performed apply context with different techniques and are briefly described below. Contextual factor “time” in Recommender Systems (RS) was used in works reviewed [2–5], these works use different methods and evaluation metrics that are focused to their goals. In [4] the pre-filter approach was used, time was divided in time intervals and the size of time intervals is directly proportional to the distance from initiating the historical information to the current user context, in this algorithm the authors introduced baseline parameters to model the time dependency of the ratings. In [1] a tracking model of user behavior over the lifetime of data is proposed, in order to exploit the relevant elements of all data instances. In [2] propose a novel pre-filtering technique for context-aware called item splitting. In this approach, the

ratings of certain items are split, according to the value of an item-dependent contextual condition. Each split item generates two fictitious items that are used in the prediction algorithm instead of the original one. In [3] a new approach called micro-profiling is proposed, the user profile is split into several sub-profiles, each one representing the user in a particular context. The predictions are done using these micro-profiles instead of a single user model. In [6], the recommendations for places of interest (POI) are based in the popularity into the user community considering contextual factors as “weather”, “time day”, “season” and “weekday” when the user rates the POI, ratings inside the context are considered for to recommend a POI for the user. For instance, whether the user has a current context where the weather is *sunny* and the season is *summer*, just items that have ratings under these conditions are used for making predictions. In [5], the author proposes a Time-context-Based Collaborative Filtering algorithm (TBCF) to improve the performance for traditional collaborative filtering, using time intervals for improve accuracy and recall ratio of standard collaborative filtering.

3 Proposed Method

In this work we propose pre-filtering approach where the time segmentation was done prior of collaborative filtering, in order to provide contextual recommendations and recent user preferences without discarding tastes in the past (historical information).

The first step in pre-filtering is defining the segmentation of 3 contexts based on the date of user current context, every context represent a period where ratings are classified for to contextualize ratings; the amount of contexts is tested manually, subsequently it can be automate. This process is depicted in Fig. 1.

In next step the current user context is obtained, from this information the algorithm defines 3 contexts that represents time segments of 3 months each one, in total the algorithm considers all the ratings done during 9 months prior the current context. Subsequently, contextual ratings are classified by contexts; size of matrix depends of users’ participation during the last 9 months.

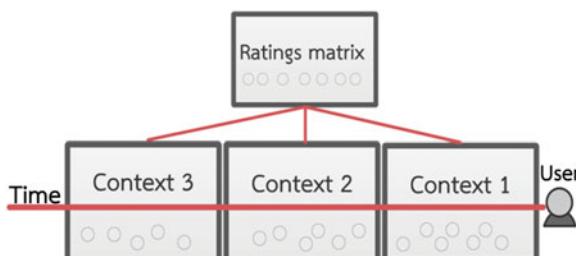


Fig. 1 The figure depicts the 3 time segmentations for contexts based on current user context

RS uses collaborative filtering in order to find relevant items for the user [7]. User's profiles are used for determine the similarity inside user community calculated with Pearson correlation. The similarity could provide valuable information as long as the user participation has been enough (at least 10 ratings).

The next step is to obtain the recommendations list (Top-N), 3 contextual lists are the outputs of collaborative filtering and contains predictions for recommended items.

Prediction by *popularity* is a simplest strategy for exploit popularity statistics. There are 2 types of popularity: item and user. Item popularity is the tendency of certain item to be liked/purchased by most users. Similarly, user popularity is the phenomenon of certain users who like/purchase more items than others [8].

Recommendations by popularity are obtained with FIS, used in a similar way as in [7] for no-contextualized recommendations. In this case FIS (Fig. 2) considers other variables that imply the context: (1) rating average (popularity) of the ratings that users assign for items in a specific context and (2) user's participation in the same context, which means how an item is preferred by users in the context.

The memberships are Gaussian type for inputs and output these are depicted in Fig. 3.

FIS recommendation has a high value when the rating matrix is sparse because is based in the popularity of items for predictions. The rules used in FIS are:

1. If **popularity** is **low** and **userParticipation** is **low** then **recommendation** is **low**.
2. If **popularity** is **low** and **userParticipation** is **high** then **recommendation** is **low**.
3. If **popularity** is **medium** and **userParticipation** is **low** then **recommendation** is **low**.
4. If **popularity** is **medium** and **userParticipation** is **high** then **recommendation** is **high**.

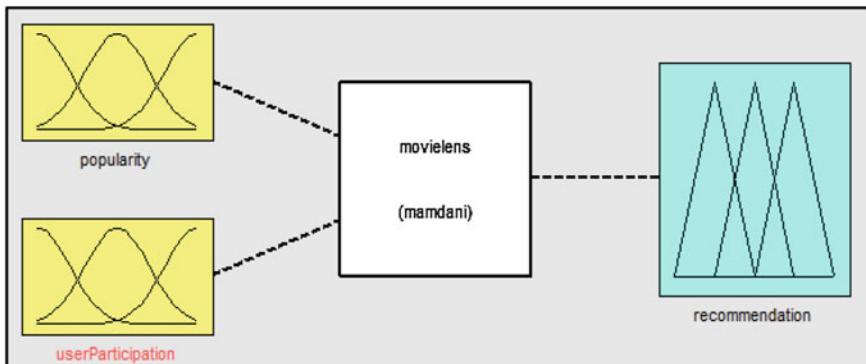


Fig. 2 Fuzzy inference system (FIS) used for recommendations. *Popularity* and *userParticipation* are the inputs used to assign a weight value for *recommendation* variable

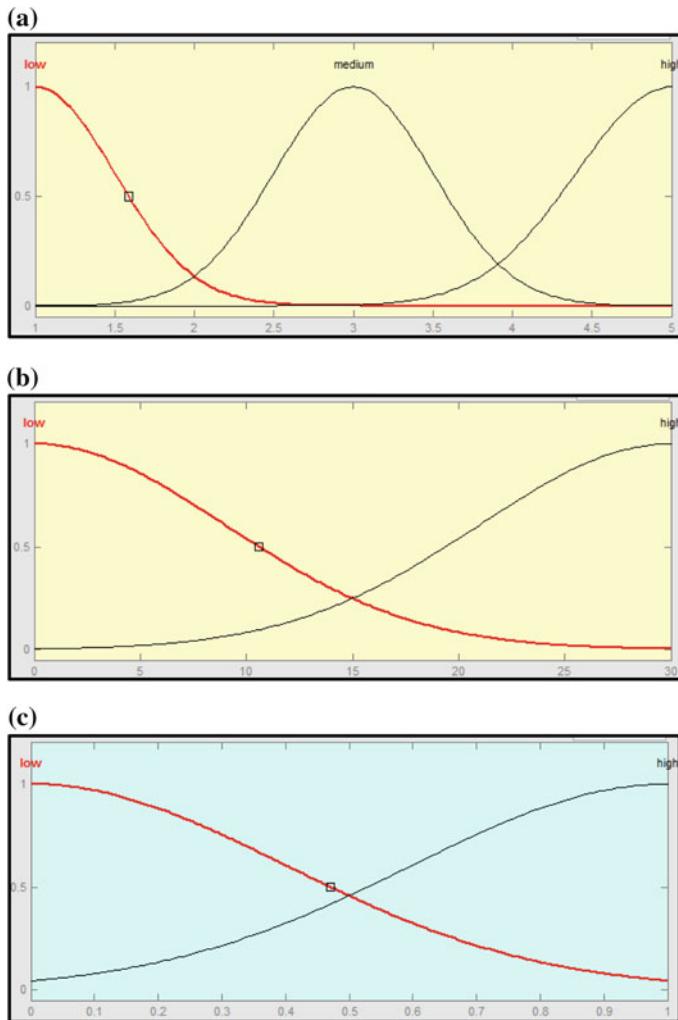


Fig. 3 **a** The variable *popularity* in range 1–5 for ratings average has values “low”, “medium” and “high”, **b** the second one variable is *userParticipation* in range from 0 to 20 for user participation in the specific context, **c** the third one variable is *recommendation* for a weight value used in the prediction

5. If **popularity** is **high** and **userParticipation** is **low** then **recommendation** is **high**.
6. If **popularity** is **high** and **userParticipation** is **high** then **recommendation** is **high**.

Finally, the RS gets recommendation lists for the user in the contexts, these lists contain predictions that are used for calculate weighted average for each item

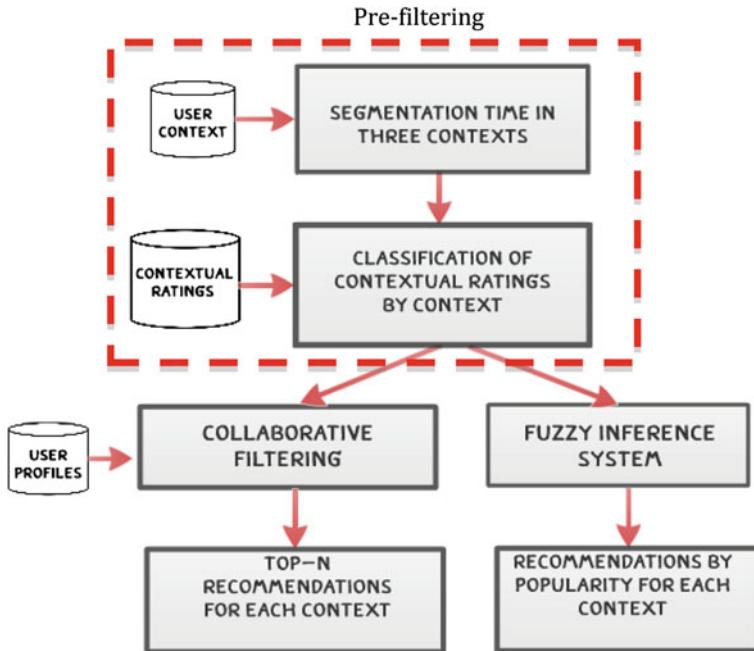


Fig. 4 The figure shows the flowchart of pre-filtering and recommendation process

recommended. Next figure (Fig. 4) depicts pre-filtering process into red box where contextual matrix is prepared, then, collaborative filtering and FIS are making predictions for items recommended.

4 Setup

4.1 Dataset

In this work, Movie lens dataset doesn't contain explicit contextual information, dates as context are used, however, the result shows that it is not a good choice, the accuracy does not improve when tries to combine a traditional method and the context when the context is inappropriate. Movielens dataset doesn't have contextual information; it was collected with a RS using traditional collaborative filtering with no specific context. Movielens dataset contains 100,000 ratings (1–5 scales) rated by 943 users on 1682 movies, and each user at least rated 20 movies, the sparsity is given as: $1 - \frac{100,000}{(943 \times 1682)} = 93.69\%$.

4.2 Metrics

RS is widely used in different domains, therefore the goals can be different for each application, and metrics are related with these goals. In some applications the evaluation of accuracy is essential; in others the item relevance is fundamental. In this work the measurement of accuracy is the main goal, there are several measurements for accuracy in the literature: Root Mean Square Error (RMSE) [9, 10], Mean Absolute Error (MAE) [9], and Precision and Recall [9, 10] are examples. For this particular case, MAE was chosen to measure the accuracy, results has shown in next section.

5 Experimental Results

The experiments were performed with Movielens dataset; Fig. 5 shows an example of data distribution where the dataset was segmented in 3 contexts for the user. The similarity metric was used for find the neighbors of current user, then the recommendations were made only for users who had neighbors into the context.

In the experiment the dataset was tested using 3 contexts with different size into “time”. The first test was 3 months defined manually, subsequently the months was increased and decreased but it doesn’t found a significant difference. The results are in Table 1.

According to previous work, in [7] the evaluation of the method without context was made with RMSE, the error in the accuracy was 0.82. Although the error was measured with different metrics, the goal was to improve the accuracy.

Fig. 5 The ratings matrix distribution by contexts for the user is depicted

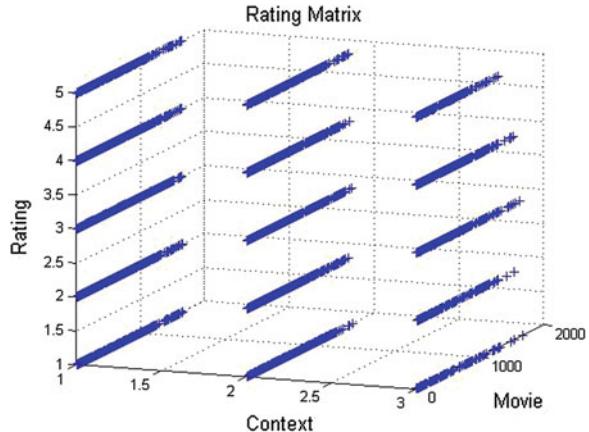


Table 1 Table of MAE results by size of contexts

Months	MAE
1	1.00
2	1.02
3	1.07
4	1.07
5	1.09

6 Conclusions

The pre-filtering is suitable for this domain considering context “time”, a query for the whole dataset prior the predictions improves the recommender process, this is not feasible for a post-filtering when the process starts with a sparse matrix, this way easily leads to bad predictions for the user.

According the previous work where we use different approach, was necessary to test the method with a larger dataset, we choose Movie lens because of amount of ratings. In Movie lens, we observe that the sparsity is not related with the density of matrix, furthermore the amount of ratings is not proportional with the number of users, and neither the data distribution is well done. So far, the results show the effects of the problems in dataset.

We are going to do experiments with different datasets with real contextual information for a possible better result.

The proposed method can be used in other domains, as long as it has sufficient information to enable optimal performance. We have this experiment as benchmark for apply the context studied in another domain as Learning Objects.

7 Future Work

We considered the next proposals for future work:

- Test the system with other contextual datasets [12], some changes can be done for explore another choices, the amount of segments could be automated, only we need to divide the history period into different length intervals and the length could be a direct ratio with the distance from then until now, in order to explore we can add another fuzzy variables or increase the granularity level for each one.
- Add other contextual factors for identify more details the current situation of the user, such as the weather, user companion or location, more knowledge about context means better recommendations. This information helps to improve the accuracy and user satisfaction, the dataset does not have these parameters, we need another dataset with contextual information to apply another parameter.
- Evaluating recommender systems from different facets is essential, since it not only makes the system more diverse, specific and comprehensive, but also meets

users' needs. We need to apply another metric for an evaluation criteria based on recommender algorithms such as: coverage, novelty, serendipity and, diversity; or with an evaluation criteria depend on recommender algorithms such as: confidence, scalability, adaptability, privacy and, trust [11].

- Apply the method in a different domain to figure out the performance.

References

1. Dey, K.A., Abowd, G.D.: Towards a better understanding of context and context-awareness. Graphics, Visualization and Usability Center and College of Computing. Georgia Institute of Technology, Atlanta, GA (1999)
2. Baltrunas, L., Ricci, F.: Context-based splitting of item ratings in collaborative filtering. The ACM Conference Series on Recommender Systems. New York, USA (2009)
3. Baltrunas, L., Amatriain, J.: Towards time-dependant recommendation based on implicit feedback. CARS (2009)
4. Koren, Y.: Collaborative filtering with temporal dynamics. Yahoo! Research, Haifa, Israel. ACM (2009)
5. He, L., Wu, F.: A time-context-based collaborative filtering algorithm. Department of computer Science and Technology. IEEE, Shanghai, China (2009)
6. Baltrunas, L., Ludwig, B., Peer, S., Ricci, F.: Context-aware places of interest recommendations and explanations. University of Bolzano, Italy. (2011)
7. Ramirez-Garcia, X.; Garcia-Valdez, M.: Restaurant recommendations based on a domain model and fuzzy rules. International Seminary in Computer Intelligence (ISCI). Tijuana Institute of Technology. Tijuana B.C., Mexico (2012)
8. Schein Andrew, I., Popescul, A., Ungar Lyle, H.: CROC: a new evaluation criterion for recommender systems, pp. 51–74. Electronic Commerce Research, Springer, Berlin (2005)
9. Campochiaro, E., Cassata, R., Cremonesi, P., Turrin, R.: Do metrics make recommender algorithms? In: International Conference on Advanced Information Networking and Applications Workshops. Milano, Italy (2009)
10. Caraciolo, M.: Artificial intelligence in motion (2013). URL: <http://aimotion.blogspot.mx/2011/05/evaluating-recommender-systems.html>
11. Wu, W., He, L., Yang, J.: Evaluating recommender systems. In: IEEE. Department of Computer Science and Technology. East China Normal University. Shanghai, China (2012)
12. Zheng Y., Burke, R., Bamshad, M.: Differential context relaxation for context-Aware Travel Recommendation. DePaul University, Chicago Illinois (2013)

A Fitness Estimation Strategy for Web Based Interactive Evolutionary Applications Considering User Preferences and Activities Using Fuzzy Logic

J.C. Romero and M. García-Valdez

Abstract Interactive evolutionary computation applications are applications where users are involved in the evolution process replacing the fitness function. In such applications the user generally evaluates subjectively information of the population in large quantities. One of the main problems for these applications is the user fatigue problems, these problems can be caused by different reasons. For example not having friendly interfaces for the evaluation, evaluating large amounts of individuals without any user interest. Have individuals not representing the interest of the user, could be because the interactive algorithm is not generating good individuals in the population. Which leads us to believe that the fitness function strategy may be the key to generating better individuals that capture the interest of users in these applications. In this paper we present a strategy for a fitness expression using fuzzy logic considering the preferences that users have over the individuals of the population, as well as the activities to be performing to interact with the application.

1 Introduction

Evolutionary interactive Web-based applications are applications that are supported by interactive evolutionary computation (IEC). IEC is a branch of evolutionary computing where users are part of the evolutionary process by replacing the fitness function. In these applications the users typically evaluate individuals in a population by using their personal preferences [1]. This evaluation is subjective because the user uses his own perception of the environment, using his senses, interests, etc., in order to evaluate. Usually in these applications users lose interest in using them.

J.C. Romero · M. García-Valdez (✉)
Tijuana Institute of Technology, Tijuana, Mexico
e-mail: mario@tectijuana.mx

J.C. Romero
e-mail: jcromerohdz@gmail.com

This may be due to different causes. For example having a very specialized user interface, evaluating large amounts of individuals, where these individuals do not attract the attention of users. This takes the user to the point of boredom, which is known as the fatigue problem [1] in interactive evolutionary computation and consequently the users lose interest in participating. One way to attract the user's interest is presenting individuals that cause an impact in relation to their senses, likes, interests, etc. The question is how to produce such individuals in the population? We know that the user participates in the evolutionary process by replacing the fitness function. Then we could design a fitness function expression that takes all the elements that the user performs in order to evaluate individuals. Thus the evolutionary interactive algorithm will produce individuals that attract the user interest in some degree.

There are some works that use the strategy of designing fitness expressions. For example Ohsaki et al. [2] in 1998, developed a method using discrete fitness value in order to reduce the psychological stress that is generated in the process of evaluating individuals.

Yuan and Gong [3] suggest an interactive evolutionary algorithm that divides the population into several clusters. The way they assigns the fitness to the individuals is by the evaluation the user performs only for central individual in each cluster, the others are estimated based on this evaluation.

Gong and Yuan [4] and Gong et al. [5] also propose to adopt a fuzzy number to express the fitness function of the individual to reflect the user cognition and use a stochastic variable with a normal distribution to represent the stochastic behavior in the fitness based on the error theory.

Our approach is focuses on using a fuzzy system [6] based on rules to infer a value that is obtained based on the preference rate that the user has on an individual as well as the level of experience that a user acquires when is using the application. Once we have the defuzzified value from the fuzzy system, we proceed to assign to our fitness expression, which we will explain in detail in Sect. 3. The application we use to test the our fitness function expression, is developed on the EvoSpace_I framework [7]. This application is inspired on digital art which creates animations from squares, circles, lines, colors, overlaps, backgrounds, etc. The animations represent individuals in the population. This paper is organized as follows: In Sect. 2 we present the related work, in Sect. 3 we present our strategy of fitness function expression using fuzzy logic, in Sect. 4 we present our experimental results, and finally in Sect. 5 we present our conclusions.

2 Related Work

In this section we presented the work related to different approaches to evaluate fitness function and the Evospace-interactive framework.

2.1 Approaches to Evaluate Fitness Function

In 1998, a new input method for human operators of an interactive genetic algorithm to reduce the psychological weight is proposed. This method uses a discrete fitness values to reduce the psychological stress involved in the input procedure. They perform simulations to investigate the influence of the resulting quantization noise from the use of discrete values of fitness in convergence. Showing that the quantization noise does not significantly worsen the convergence. In this method they evaluated using two subjective tests involving the task of drawing faces. The subjective test results shows that this method significantly reduce the level of psychological stress of human interactive genetic algorithms operators [2]. Another approach, proposed to use novel method evaluation, where the user only evaluates a satisfactory or unsatisfactory individual. These approaches consider the level of sensibility of the different users to their perception of the beautiful and the ugly, and fitness is automatically calculated based on user evaluations and time. They also propose effective strategies for comparing different individuals of the same generation in uncertain fitness conditions of an individual. Where they obtains the probability of an individual dominance by use of the probability of the interval domain, and translate to a fuzzy number in a range based on α -cut set [4]. They determine the dominant individual in tournament selection with size being two on base given by the probability of a particular domain. This approach was applied to an interactive evolutionary system for fashion design. In Fig. 1 we can see different user interfaces they used. Based on this approach, another work was derived. Where the approach adopt a fuzzy number described with a Gaussian membership function to express an individual's fitness. In order to compare the different individuals, they generated a fitness interval based on a cut set, and obtain the probability of interactive genetic algorithms with individual's fuzzy fitness. The contributions in this approach can improve the performance of existing income generating activities in alleviating user fatigue and finding optimal solutions to an optimization problem, so it is beneficial for solving complicated problems with implicit or fuzzy indices [5].



Fig. 1 Different user interfaces interactive evolutionary system for fashion design [5]

2.2 EvoSpace-I

EvoSpace-Interactive is an open source framework focused on Web environments for collaborative interactive evolutionary applications. This framework defines three main components for each application, which are:

- Individual.
- Processing Script.
- Worker Script.

The individual is represented internally as a data dictionary stored in Redis [6]; the individual contains three main attributes: id, chromosome, mom, dad, and views. This attributes represents the key information of the individual as the individual offspring, the number of times that the individual has been selected, etcetera, as we can see in Fig. 2.

As we can observe on Fig. 3, this work uses database management systems to implement collaborative interactive evolutionary applications. One of the reasons that this framework is using Redis [6] is because it provides a hash-based implementation of sets and queues, which are natural data structures for the EvoSpace model [8]. On the other hand this framework uses a relational database to save basic information about the user extracted from the social platform (Facebook) through open graph API and OAuth2 authentication.

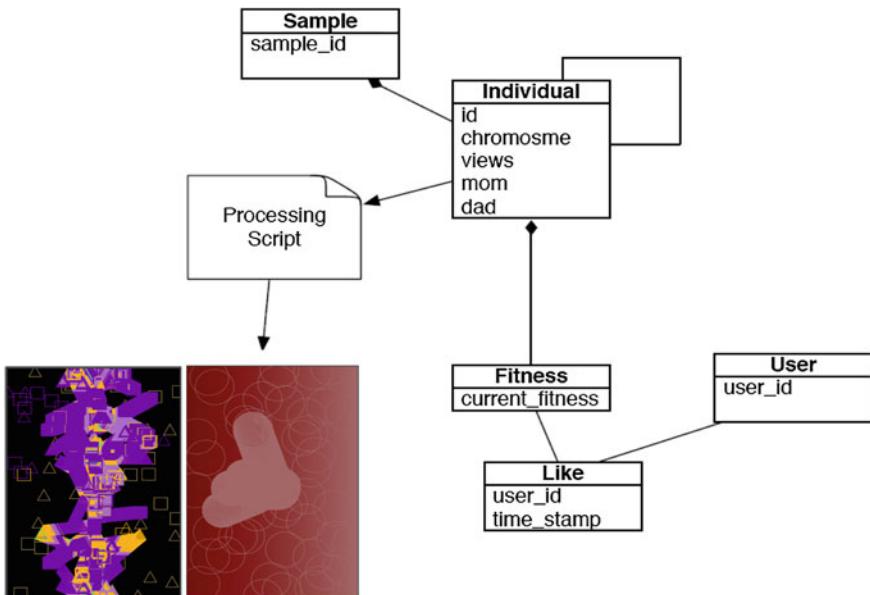


Fig. 2 Individual representation [7]

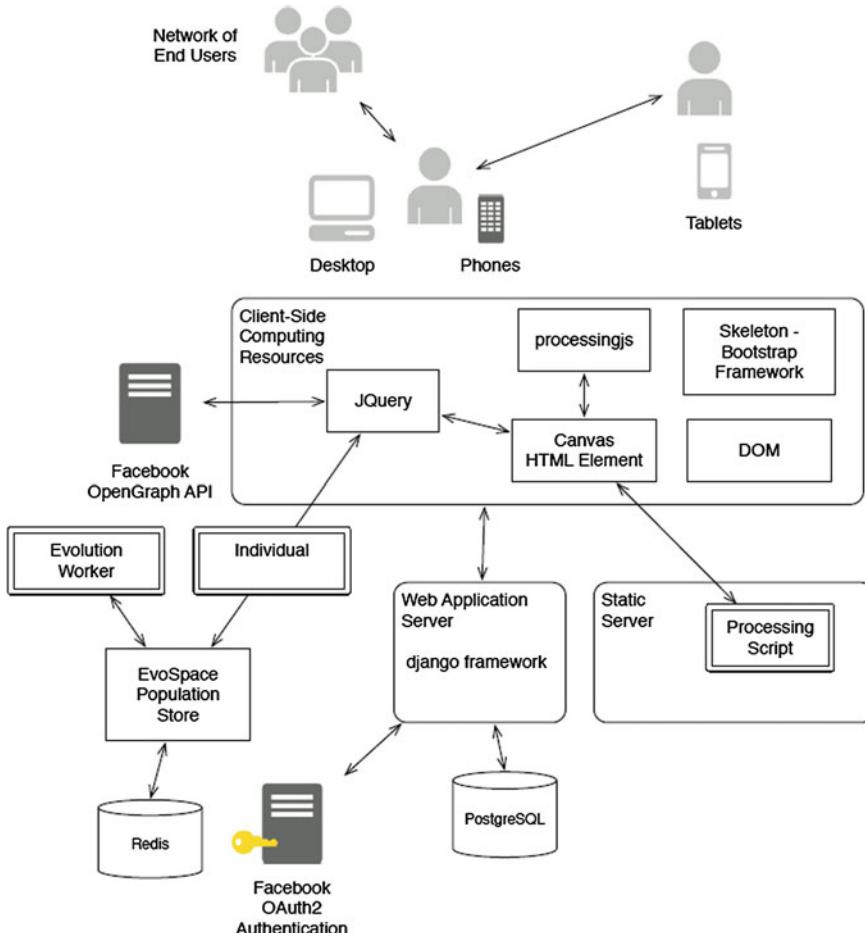


Fig. 3 EvoSpace-I framework [7]

3 Fitness Estimation Strategies for Web Based Interactive Evolutionary Applications Considering User Preferences and Activities Using Fuzzy Logic

This section explains the proposed strategy to calculate the fitness of the individual in a web-based interactive evolutionary computation applications using fuzzy logic. Before showing our strategy, it is necessary to explain how the individual evaluation is made in the shapes animation application. The Fig. 4 shows the user interface where the user interacts with the application. The main goal of this interaction is the individuals evaluation, the first action of the user, in order to evaluate, is to login through a Facebook [9] account in order to have different

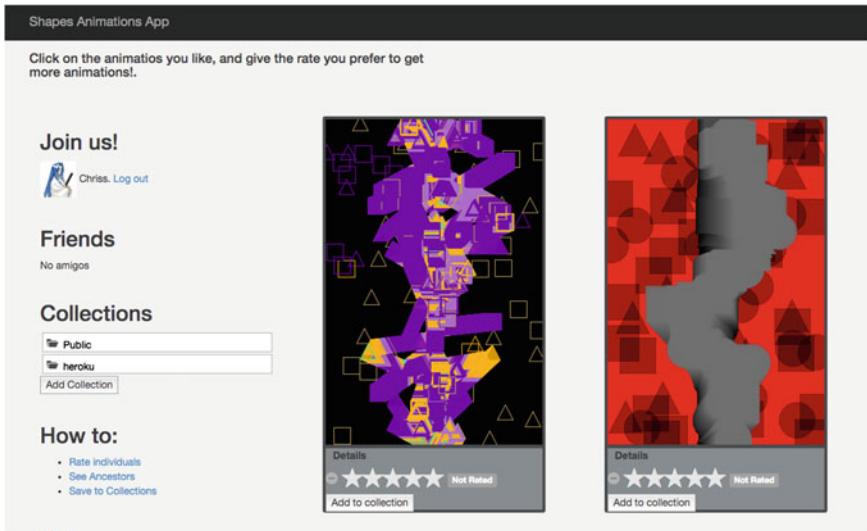


Fig. 4 Shapes animation application user interface

futures as a collection creation, store individual in a collection, etc. The evaluation takes place through a five-star rate selection by the user; this rate represents the degree of user preference for an individual. The application keeps the record of every user activity using the activities stream standard [10]. In these particular case the activities represents the user experience.

In this paper we propose the using of fuzzy logic [6] in order to obtain a defuzzify value to be used to calculate the individual fitness trough a fitness function expression. It is used by modeling a fuzzy Mamdani type inference system [11, 12] as we can see in Fig. 5. This model was designed empirically. The model consists of two input variables, which are the preference and the experience of the user as well as an output that we called fuzzy rate. The first one has 3 linguistic variables, which are low, medium and high, representing the preference with

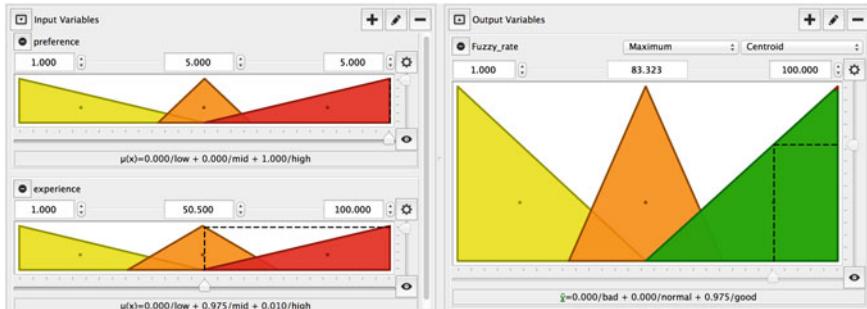


Fig. 5 Fuzzy system of Mamdani type [2, 11, 12]

triangular membership functions over a range of 1–5. The second one also has three linguistic variables, which are low, medium and high representing the experience with triangular membership functions over a range of 1–100. Finally we have the output consisting of three linguistic variables bad, normal and good representing the fuzzy rate with triangular membership functions in a range of 1–100.

Below we show the rules IF-THEN of the fuzzy system:

1. *If preference is low and experience is low then fuzzy-rate is bad*
2. *If preference is low and experience is mid then fuzzy-rate is bad*
3. *If preference is low and experience is high then fuzzy-rate is normal*
4. *If preference is mid and experience is low then fuzzy-rate is normal*
5. *If preference is mid and experience is mid then fuzzy-rate is normal*
6. *If preference is mid and experience is high then fuzzy-rate is good*
7. *If preference is high and experience is low then fuzzy-rate is normal*
8. *If preference is high and experience is mid then fuzzy-rate is good*
9. *If preference is high and experience is high then fuzzy-rate is good*

These rules will give us a fuzzy rate value result, this is the value need it to be defuzzify by the centroid method in order to be used in our fitness expression, given by Eq. 1. This expression is responsible to represent the individual fitness.

$$F = \frac{\sum_{i=0}^n x_i y_i}{\sum_{i=0}^n y_i} \quad (1)$$

where n represents the number of users that have given an evaluation of the individual, x is the rate of preference for the individual given by the user, y is a function that calls the fuzzy system in order to have the fuzzy rate. This function needs x and the user experience level. The user experience level is given by the total activities that user have at the moment. In each activity we assign the score, for example if the user log in (join) to the application we assign 5 points, if the user evaluates (likes) an individual we give 3 points, etc.; in Fig. 6 we can show the flow for assigning fitness to the individual.

4 Results

To view the results given by the application, we need to know how different individuals were produced by interactive evolutionary algorithm. One way to know how different a set is, we use the information entropy measure, which is defined in Eq. 2. This measure allows us to understand how different the resulting individuals are from each other. If all individuals were the same then the entropy tends to be 0. The more diversity we have in the resulting individuals, then the information entropy tends to be higher.

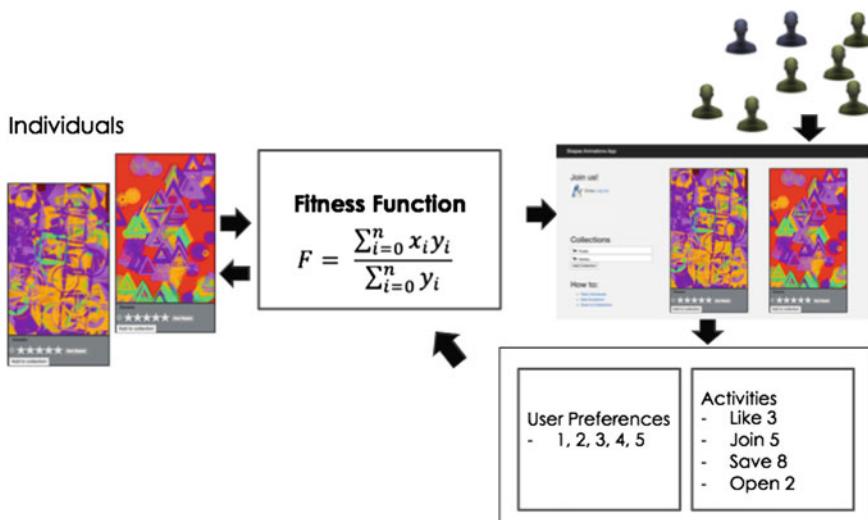


Fig. 6 Assign fitness flow

$$H(X) = \sum_{i=1}^n p(x_i) \log_2 \left(\frac{1}{p(x_i)} \right) = - \sum_{i=1}^n p(x_i) \log_2 p(x_i) \quad (2)$$

The experiment consisted of 10 users used the application to evaluate individuals for a period of 3 min. We initialize the population with 80 individuals.

Table 1 shows the results of this experiment using the accumulated entropy and compares with historical data that had been produced using the same individuals chromosome structure with a precise fitness assignment strategy. Figure 7 shows this structure.

This structure is represented by real numbers, which is a data array of 15 positions. Each index of the array represents a behavior of the individual in the animation. For example the index 0 represents the number of figures that were used in the animation, the index 1 represents the size of the figures in the animation, the index 2 represents whether or not a triangle, etc.

In Fig. 8 shows the best individuals that occurred during the experiment.

Table 2 shows the identifier of the individual as well as the number of evaluations received from users, including their current fitness.

Table 1 Experiment results

	Fitness strategy using fuzzy logic	Precise strategy
Initial population	80	100
All time generated	264	762
Accumulated ted entropy	1.66	1.89

Fig. 7 Individual chromosome structure

Chromosome: [60, 23, 1, 1, 0, 0, 1, 0, 0, 3, 2, 0, 0, 1, 0]
 [60] #. Of figures
 [23] Figures Size
 [1] Triangle
 [1] Square
 [0] Circle
 [0] Overlap
 [1] Color
 [0] Orienting Towards Another
 [0] Restricted to canvas
 [3] Spinning Figure
 [2] Initial position
 [0] Outline or Fill
 [0] Line
 [1] Line type
 [0] Background



Fig. 8 Best individuals at the time

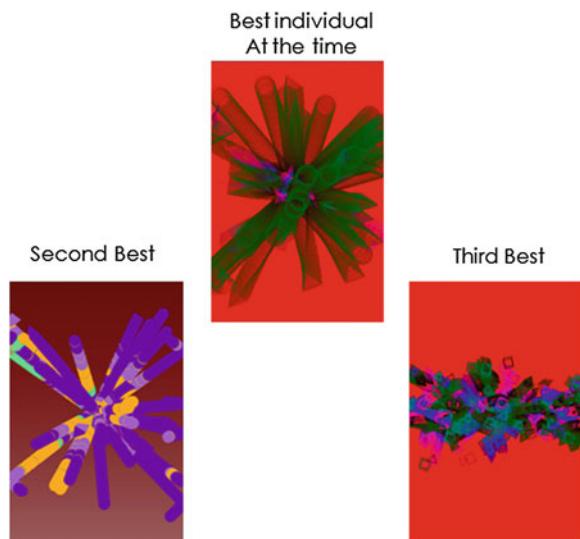


Table 2 Experiment results

Individual identifier	Evaluation by users	Current fitness
Pop:individual:8	9	2.3979
Pop:individual:39	8	1.9390
Pop:individual:35	7	1.7622

5 Conclusions

The results show that the proposed fitness expression is working as we expect. However the number of users that was used in the experiment was very small. It is necessary to expand the number of users in order to know whether or not this fitness expression is working properly, in other words if it is attracting the interest of users.

References

1. Takagi, H.: Interactive evolutionary computation: fusion of the capabilities of EC optimization and human evaluation. IEEE (2001)
2. Ohsaki, M., Takagi, H., Ohya, K.: An input method using discrete fitness values for interactive GA. Intell. Fuzzy Syst. **6**(1), 131–145 (1998)
3. Yuan, J., Gong, D.W.: Large population size IGAs with individuals' fitness not assigned by user. In: Proceedings of the 4th International Conference on Intelligent Computing, LNAI, vol. 5227, pp. 267–274 (2008)
4. Gong, D.W., Yuan, J.: Impact of individuals fitness expressions on interactive genetic algorithms performance. IEEE (2008)
5. Gong, D.W., Yuan, J., Sun, X.-Y.: Interactive genetic algorithms with individual fuzzy fitness. Comput. Human Beh. **27**, 1482 (2011)
6. Zadeh, L.A.: Fuzzy sets. Inf. Control **8**, 338–352 (1965)
7. García-Valdez, M., Trujillo, L., Fernández-de-Vega, F., Merelo-Guervós, J.J., Olague, G.: EvoSpace-interactive: a framework to develop distributed collaborative-interactive evolutionary algorithms for artistic design. In: Proceedings of the 2nd International Conference on Evolutionary and Biologically Inspired Music, Sound, Art and Design, EvoMUSART (2013)
8. García-Valdez, M., Trujillo, L., Fernández de Vega, F., Merelo-Guervós J.J., Olague G.: EvoSpace: a distributed evolutionary platform based on the tuple space model. In: 16th European Conference, EvoApplications Proceedings. 2013, pp. 499–508. Vienna, Austria 3–5 April (2013)
9. “Facebook”: <http://facebook.com/>
10. “Activity Stream”: <http://activitystrea.ms/>
11. Mamdani, E.H., Baaklini, N.: Prescriptive method for deriving control policy in fuzzy-logic controller. Electr. Lett. **11**, 625 (1975)
12. Mamdani, E.H.: Application of fuzzy algorithms for the control of a dynamic plant. Proc. IEEE **121**(12), 1585 (1974)

Design of a Fuzzy System for Diagnosis of Hypertension

Juan Carlos Guzmán, Patricia Melin and German Prado-Arechiga

Abstract One of the most dangerous diseases for humans is the Arterial Hypertension, which this kind of disease that often leads to fatal outcomes, such as heart attack, stroke and renal failure. The hypertension seriously threats the health of the people worldwide. One of the dangerous aspects of the hypertension is that you may not know that you have it. In fact, nearly one-third of people who have high blood pressure don't know it. The only way to know if the blood pressure is high is through the regular checkups. The evaluation of a patient with Hypertension should (1) confirm the diagnosis of hypertension, (2) detect causes of secondary hypertension, (3) assess cardio vascular risk and organ damage. Therefore, is very important a correct measurement of the blood pressure (BP). Traditionally, office BP measurement has been performed using a sphygmomanometer and stethoscope. Recently, automated office and home BP measurements has been proposed as an alternative to traditional measurement. It has several advantages over manual BP, especially in routine clinical practice. Therefore, we have developed a Fuzzy System for the diagnosis of the Hypertension. Firstly, the input parameters include Systolic Blood Pressure and Diastolic Blood Pressure. Secondly, we have as an output parameter: Blood Pressure Levels (BPL). The input linguistic value includes Low, Low Normal, Normal, High Normal, High, Very High, Too High and Isolated Systolic Hypertension. Finally, we have 14 fuzzy rules to determine the diagnosis output.

1 Introduction

Nowadays different techniques of artificial intelligence, such as fuzzy systems are largely used in medical areas. As we know, the control of the hypertension is considered when the systolic blood pressure >140 mmHg and the diastolic blood

J.C. Guzmán · P. Melin (✉)
Tijuana Institute of Technology, Tijuana, B.C., México
e-mail: pmelin@tectijuana.mx

G. Prado-Arechiga
Cardio-Diagnóstico Tijuana, Tijuana, B.C., México

pressure >90 mmHg. Thus, the use of an expert system that provides information to the user about the factors and dangers of high blood pressure is very important.

Fuzzy logic is used to model nonlinear systems, which are difficult to model mathematically. It is a logic system that is based on fuzzy set theory and continuous variables. Conclusions that are based on vague, imprecise, missing input information are simply provided by fuzzy logic (FL). Fuzzy logic uses different concepts, i.e. fuzzification, defuzzification, membership function, linguistic variables, domain, rules etc. In Boolean algebra or Boolean logic crisp sets are used, which have only two values 0 and 1, but in fuzzy logic, sets have an infinite number of values between 0 and 1. In Boolean logic an element is completely inclusive or exclusive membership is used, but in a FL completely inclusive, exclusive or between these two memberships is used. Also the fuzzy system is a system in which fuzzy rules are used with membership functions (MF) to find the conclusion or result. Fuzzy logic has been applied to many areas or fields of application, for example fuzzy logic has played an important role in the field of medicine [5, 6, 8]. They are used in control, automobiles, household appliances and decision making systems.

Hypertension or high blood pressure, sometimes called arterial hypertension, is a chronic medical condition in which the blood pressure in the arteries is elevated [14]. The optimal level for blood pressure is below 120/80, where 120 represent the systolic measurement (peak pressure in the arteries) and 80 represents the diastolic measurement (minimum pressure in the arteries). Systolic BP between 120 and 129 and/or Diastolic BP between 80 and 84 is called Normal BP. And Systolic BP between 130 and 139 and/or Diastolic BP 85 and 89 is called High Normal. And a blood pressure of 140/90 or above is considered hypertension in three different grades and other like the Isolated systolic hypertension that is Systolic BP over 140 and Diastolic BP under 90 mmHg ESH/ESC guidelines [10].

Hypertension may be classified as essential or secondary. Essential hypertension is the term for high blood pressure with an unknown cause [15]. It accounts for about 95 % of cases. Secondary hypertension is the term for high blood pressure with a known direct cause, such as kidney disease, tumors or others.

The paper is organized as follows: in Sect. 2 a methodology of hypertension is presented, in Sect. 3 simulation and results of the prediction of the data that will be the input to the fuzzy system are presented, in Sect. 4 the design and development of the fuzzy logic system is described, and in Sect. 5 the conclusion obtained after tests the fuzzy system of diagnosis of hypertension.

2 Methodology

2.1 Type of Blood Pressure Diseases

Hypertension is the most common disease and it markedly increases both morbidity and mortality from cardiovascular and many other diseases [12]. Different types of

Table 1 Definitions and classification of the blood pressure levels (mmHg)

Category	Systolic		Diastolic
Hypotension	<90	and/or	<60
Optimal	<120	and	<80
Normal	120–129	and/or	80–84
High normal	130–139	and/or	85–89
Grade 1 hypertension	140–159	and/or	90–99
Grade 2 hypertension	160–179	and/or	100–109
Grade 3 hypertension	≥180	and/or	≥110
Isolated systolic hypertension	≥140	and	<90

hypertension are observed when the disease is sub-categorized. These types are shown in Table 1.

In Table 1 the blood pressure (BP) category is defined by the highest level of BP, whether systolic or diastolic. And should be graded 1, 2 or 3 according to the systolic or diastolic BP value. Isolated systolic hypertension it is according to the systolic BP value in the ranges indicated.

2.2 Risk Factors

Some of the primary risk factors for essential hypertension include the following [1]:

- Obesity
- Lack of exercise
- Smoking
- Consumption of salt
- Consumption of alcohol
- Stress level
- Age
- Sex
- Genetic factors.

2.3 Fuzzy Logic and Hypertension

Nowadays we cannot be comfortable with the traditional medical analysis because the complexity of medical practices makes traditional quantitative approaches of analysis inappropriate [13]. Every trust worthy expert knows that his/her medical knowledge and resulting diagnosis are pervaded by uncertainty with imprecise formulations. Medical processes can be so complex and unpredictable that physicians sometimes must make decisions based on their experience or intuition or sometimes

they might require a specialization like Cardiology, Internal Medicine, etc. Computers are capable of making calculations at high and constant speed and of recalling large amounts of data and can, therefore, be used to manage decision networks of high complexity [11]. Fuzzy logic developed by Zadeh [16] makes it possible to define these inexact medical entities as fuzzy sets. Fuzzy logic together with the appropriate rules of inference provides a power framework for managing uncertainties pervaded in medical diagnosis [3, 4, 7, 9]. Fuzzy logic technology is adopted in this paper for the diagnosis of hypertension. This is because, fuzzy logic can adequately address the issue of uncertainty and lexical imprecision of knowledge [2], but fuzzy systems still require human expert to discover rules about data relationship.

By applying fuzzy logic, a fuzzy rule base system for the diagnostic of hypertension was developed with the help of the domain expert.

3 Simulation and Results

The following graphic interface in Fig. 1 shows the information to be simulated and used for prediction of blood pressure following the result is selected.

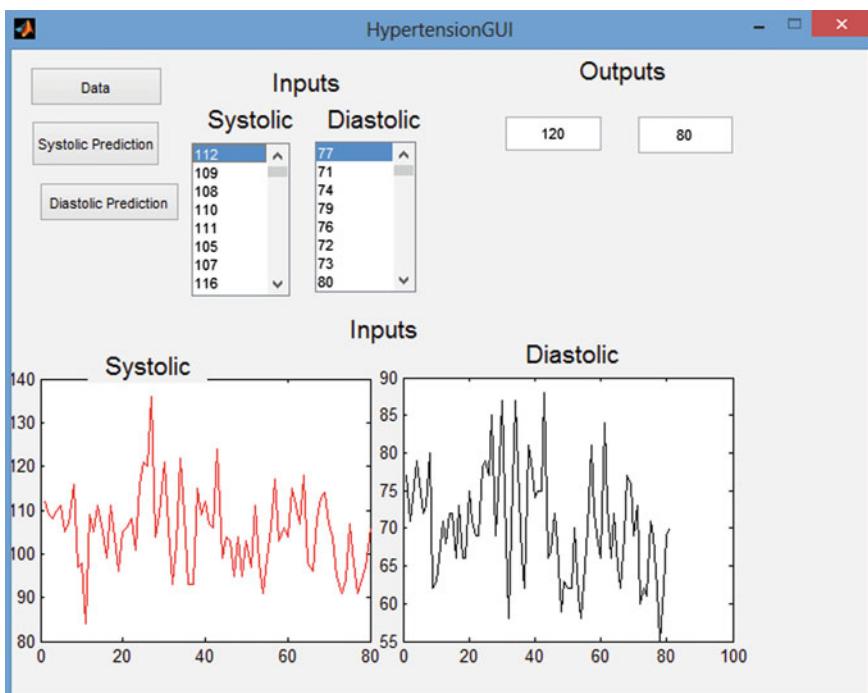


Fig. 1 Graphic interface and select file window

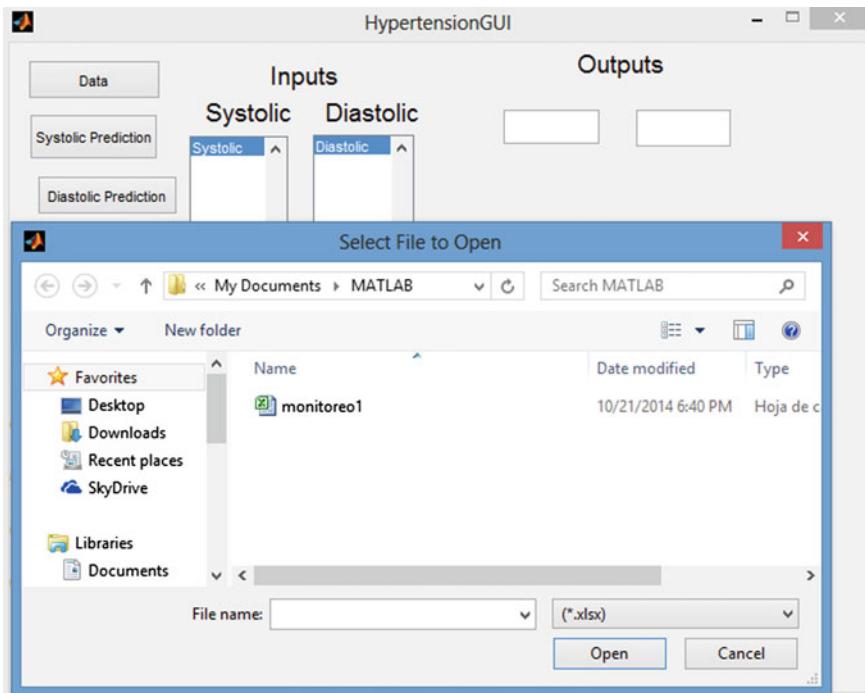


Fig. 2 Simulation and results of the graphic interface

The following graphic interface in Fig. 2 simulates the monitoring of blood pressure of a patient, which is based on the information given. A prediction of its next blood pressure is performed, and the result of the prediction is systolic and diastolic and this information is the input to the fuzzy system.

4 Design and Development of the Fuzzy Logic System

A fuzzy logic system is a collection of membership functions and fuzzy rules that are used to determine the diagnosis. This design has been divided into several steps. The steps are fuzzification, rule evaluation and finally defuzzification. To design the system, the FIS tool in MATLAB R2013a is used.

In this study, we propose a fuzzy system for the diagnosis of the hypertension. The fuzzy system has 2 inputs including the Systolic and Diastolic, and 1 output BP_level, in the inputs we have eight membership functions, such as Low, low normal, normal, high normal, high, very high, too high and isolated systolic Hypertension (ISH) and in the output there are eight member functions such as Hypotension, Optimal, Normal, High normal, Grade 1, Grade 2, Grade 3 and

Isolated Systolic Hypertension (ISH) and Mamdani inference engine and centroid defuzzification.

The analysis focused on how to design a fuzzy logic system for diagnosis hypertension. This is performed by using a range of systolic and diastolic blood pressure. First, the linguistic values and corresponding membership functions have been determined in the next figures: Fig. 3 shows the fuzzy system of diagnosis of hypertension, Fig. 4 shows linguistic variable and membership function of "Systolic", Fig. 5 shows linguistic variable and membership function of the input "Diastolic", Fig. 6 shows the linguistic variable and membership function of the output "BP_level".

Now we will show in the following more details of the fuzzy system: Fig. 7 shows the rules of our fuzzy system of diagnosis of hypertension, Fig. 8 shows the result of rules of the fuzzy system of diagnosis of hypertension and finally Fig. 9 shows the surface view of the fuzzy system of diagnosis of hypertension.

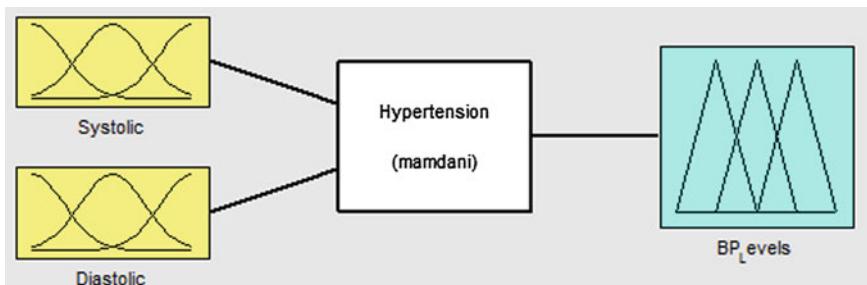


Fig. 3 Fuzzy system for diagnosis of hypertension

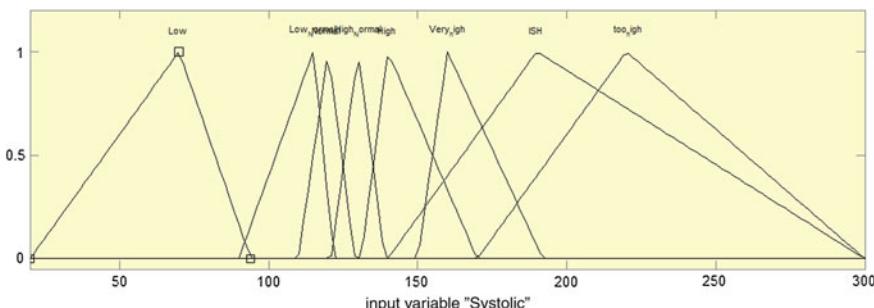


Fig. 4 Linguistic variable and membership functions of "Systolic"

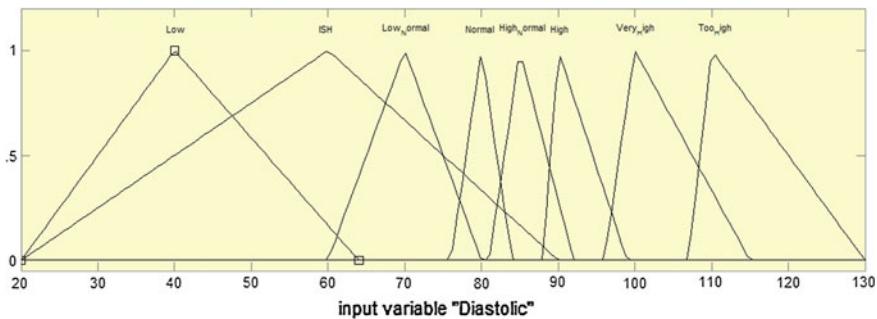


Fig. 5 Linguistic variable and membership functions of the input “Diastolic”

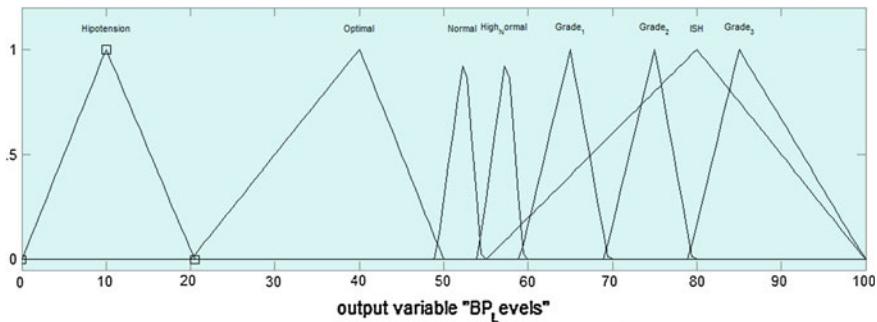


Fig. 6 Linguistic variable and membership functions of the output “BP_level”

1. If (Systolic is Low) and (Diastolic is Low) then (BP_Levels is Hipotension) (1)
2. If (Systolic is Low_Normal) and (Diastolic is Low_Normal) then (BP_Levels is Optimal) (1)
3. If (Systolic is Normal) and (Diastolic is Normal) then (BP_Levels is Normal) (1)
4. If (Systolic is High_Normal) and (Diastolic is High_Normal) then (BP_Levels is High_Normal) (1)
5. If (Systolic is High) and (Diastolic is High) then (BP_Levels is Grade_1) (1)
6. If (Systolic is Very_high) and (Diastolic is Very_High) then (BP_Levels is Grade_2) (1)
7. If (Systolic is too_high) and (Diastolic is Too_High) then (BP_Levels is Grade_3) (1)
8. If (Systolic is ISH) and (Diastolic is ISH) then (BP_Levels is ISH) (1)
9. If (Systolic is Very_high) and (Diastolic is High) then (BP_Levels is Grade_2) (1)
10. If (Systolic is too_high) and (Diastolic is Very_High) then (BP_Levels is Grade_3) (1)
11. If (Systolic is too_high) and (Diastolic is High) then (BP_Levels is Grade_3) (1)
12. If (Systolic is High) and (Diastolic is Very_High) then (BP_Levels is Grade_2) (1)
13. If (Systolic is High) and (Diastolic is Too_High) then (BP_Levels is Grade_3) (1)
14. If (Systolic is Very_high) and (Diastolic is Too_High) then (BP_Levels is Grade_3) (1)

Fig. 7 Fuzzy rules of the fuzzy system for diagnosis of hypertension

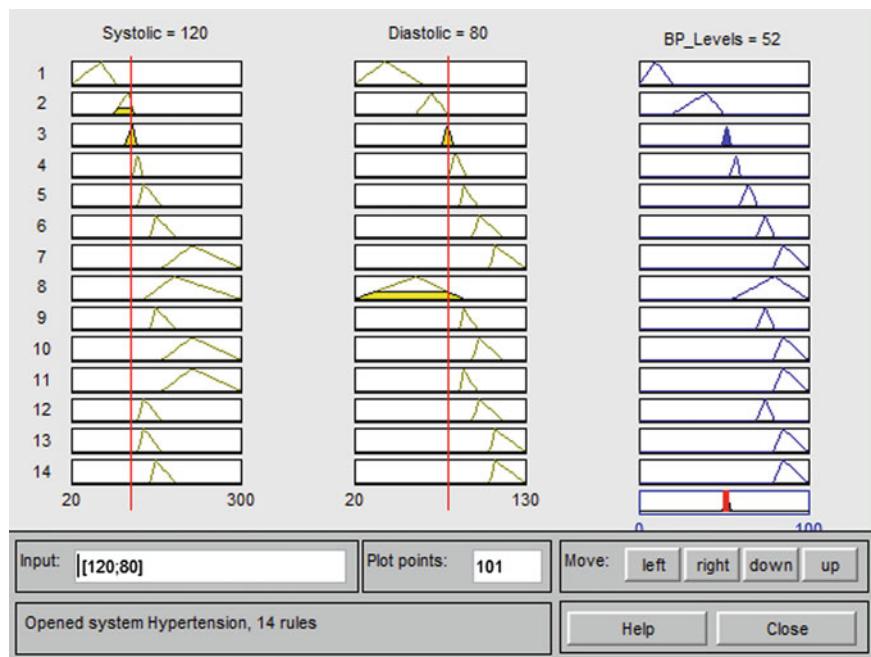


Fig. 8 The inference with the rules of the fuzzy system for diagnosis of hypertension

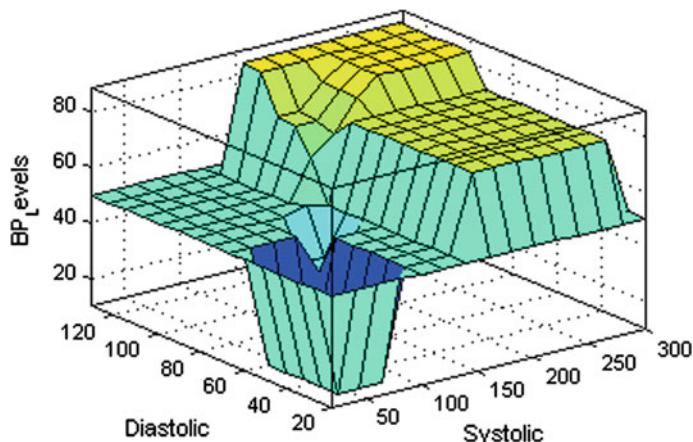


Fig. 9 Surface view of the fuzzy system for diagnosis of hypertension

5 Conclusions

This type of fuzzy systems actually implements the human intelligence and reasoning. Using a set of decision rules, provide different suggestions for diagnosing diseases, in this case hypertension. This is a very efficient, less time consuming and more accurate method to diagnose the risk and the grade of hypertension. Finally we can see that is a very effective method for an early and accurate diagnostic of hypertension, which can help a physician to get a better medical treatment when giving a diagnosis to the patient.

Acknowledgments We would like to express our gratitude to the CONACYT and Tijuana Institute of Technology for the facilities and resources granted for the development of this research.

References

1. Abrishami, Z., Tabatabaei, H.: Design of a fuzzy expert system and a multi-layer neural network system for diagnosis of hypertension. *MAGNT Res. Rep.* **2**(5), 913–926 (2014). ISSN: 1444-8939
2. Akinyokun, O.C., Adeniji, O.A.: Experimental study of intelligent computer aided diagnostic and therapy. *AMSE J. Model. Simul. Control* **27**(3), 9–20 (1991)
3. Abdullah, A.A., Zulkarnay, Z., Mohammad, N.F.: Design and development of fuzzy expert system for diagnosis of hypertension. In: International Conference on Intelligent Systems, Modelling and Simulation. IEEE (2011)
4. Das, S., Ghosh, P.K., Kar, S.: Hypertension diagnosis: a comparative study using fuzzy expert system and neuro fuzzy system. In: IEEE International Conference on Fuzzy Systems. IEEE (2013)
5. Djam, X.Y., Kimbi, Y.H.: Fuzzy expert system for the management of hypertension. *Pac. J. Sci. Technol.* **12**(1) (2011). (Spring)
6. Fuller, R., Giove, S.: A neuro-fuzzy approach to FMOLP problems. In: Proceedings of CIFT'94, pp. 97–101. Trento, Italy (1994)
7. Kaur, A., Bhardwaj, A.: Genetic neuro fuzzy system for hypertension diagnosis. *Int. J. Comput. Sci. Inf. Technol.* **5**(4), 4986–4989 (2014)
8. Kaur, R., Kaur, A.: Hypertension diagnosis using fuzzy expert system. *Int. J. Eng. Res. Appl. (IJERA)*. ISSN 2248-9622. National Conference on Advances in Engineering and Technology, AET 29th March 2014
9. Ludmila, I.K., Steimann, F.: Fuzzy Medical Diagnosis. School of Mathematics, University of Wales, Bangor (2008)
10. Mancia, G., Fagard, R., Narkiewicz, K., Redon, J.: 2013 ESH/ESC guidelines for the management of arterial hypertension. *J. Hypertens.* **31**, 1281–1357 (2013)
11. Merouani, M., Guignard, B., Vincent, F., Borron, S.W., Karoubi, P., Fosse, J.P., Cohen, Y., Clec'h, C., Vicaut, E., Marbeuf-Gueye, C., Lapostolle, F., Adnet, F.: Can fuzzy logic make things more clear? *Crit. Care* **13**, 116 (2009)
12. O'Brien, E., Parati, G., Stergiou, G.: European society of hypertension position paper on ambulatory blood pressure monitoring. *J. Hypertens.* **31**, 1731–1768 (2013)
13. Rahim, F., Deshpande, A., Hosseini, A.: Fuzzy expert system for fluid management in general anesthesia. *J. Clin. Diagn. Res.* **4**, 256–267 (2007)

14. Srivastava, P.: A note on hypertension classification scheme and soft computing decision making system. ISRN Biomathematics, Volume 2013 (2013), Article ID 342970, <http://dx.doi.org/10.1155/2013/342970>
15. Sumathi, B., Santhakumaran, A.: Pre-diagnosis of hypertension using artificial neural network. *Glob. J. Comput. Sci. Technol.* **11**(2) Version 1.0 February 2011
16. Zadeh, L.A.: Fuzzy sets and systems. In: Fox, J. (ed.) *Proceedings Symposium on System Theory*, pp. 29–37. Polytechnic Institute of Brooklyn, New York, April 1965

Trajectory Metaheuristics for the Internet Shopping Optimization Problem

Mario C. López-Locés, Kavita Rege, Johnatan E. Pecero,
Pascal Bouvry and Héctor J. Fraire Huacuja

Abstract In this chapter we propose two trajectory-based heuristics, particularly Simulated Annealing and Tabu Search, to solve instances of the Internet Shopping Optimisation Problem (ISOP). Since these metaheuristics are relatively less costly in terms of computational resources such as CPU time and memory, compared to population-based metaheuristics, but with a better performance than simpler heuristics, its use in the context of internet shopping is relevant. In such context the user of the service expects good quality solutions, while the provider wants to maintain the use of resources at a low level.

1 Introduction

Internet has changed the perspective of shopping; it has revolutionized the way we do our shopping with just a click of a button. Unlike the conventional way of shopping, the people purchasing their products online saves not only money, but also time and energy as there is no need to personally go to a store and stand in long queues at the billing counter to accomplish the same thing. In the traditional way,

M.C. López-Locés · H.J. Fraire Huacuja (✉)
Tecnológico Nacional de México, Instituto Tecnológico de Ciudad Madero,
Ciudad Madero, Mexico
e-mail: automatas2002@yahoo.com.mx

M.C. López-Locés
e-mail: mariocesar@lopezloc.es

M.C. López-Locés · K. Rege · J.E. Pecero · P. Bouvry
Computer Science and Communications Research Unit, University of Luxembourg,
Luxembourg City, Luxembourg
e-mail: kavita.rege.001@student.uni.lu

J.E. Pecero
e-mail: johnatan.pecero@uni.lu

P. Bouvry
e-mail: pascal.bouvry@uni.lu

people usually are restricted to choose from the limited stock of products in the store, while in online shopping one can choose the required product from a vast range of stores. This, however also creates a new problematics, as the large amount of available online stores increases the difficulty to find the best deal available, and even more when we have to acquire multiple products at once. This problem was formally defined by Blazewicz et al. [1] as the Internet Shopping Optimisation Problem with delivery costs.

This chapter is organized as follows. In Sect. 2 we discuss the state of art in the field of the Internet Shopping Optimisation Problem. In Sect. 3 we describe the formal definition proposed by [1] in detail. Section 4 deals with the Tabu Search method approach, its parameters and different strategies applied, while Sect. 5 deals with the same matters using the Simulated Annealing metaheuristics. In Sect. 6 we describe the computational experiments performed to evaluate de correctness and performance of the proposed solution methods, and in Sect. 7 we present the results obtained. Finally, in Sect. 8 we present the conclusions based on the results obtained and propose future works.

2 Related Work

Some of the previously proposed works related to the Internet Shopping Optimisation problem were presented by Tolle and Chen in 2000 [2]. In their survey they proposed intelligent software agents, which were widely used in price comparison sites. These agents collect offers from online shops and rank them based on customer needs. Kwon and Sadeh in 2004 [3] proposed a context aware comparative shopping. But these approaches targeted only single product buying, while multiple product purchase was not taken into account.

Blazewicz et al. in 2010 [1] proposed the current definition of the Internet Shopping Optimisation Problem (ISOP). In that article they investigated the problem of purchasing multiple products available on multiple shops, where they proposed a formal definition of this optimization problem along with proof that the problem is strongly NP-hard. The authors also proposed two algorithms SHOP-ENUM and PRODUCT-ENUM. The SHOP-ENUM algorithm enumerates all possible selections of shops that contain all required products. The time complexity of the SHOP-ENUM algorithm is $O(n^2m)$, which is polynomial if number m of shops is constant. The second algorithm PRODUCT-ENUM explores all possible shop choices for each product. The time complexity of the algorithm PRODUCT-ENUM is $O(nm^n)$, which is polynomial if the number n of products is constant [1].

3 Problem Definition

In [1], Blazewicz et al. give the following formal definition for the Internet Shopping Optimization Problem with Delivery Costs:

Given a buyer who wants to acquire the products in a shopping list $N = \{1, \dots, n\}$, where n denotes the number of products, on a list of stores $l = \{1, \dots, m\}$, where m denotes the number of stores, with a multiset N_l which represents a subset of products from N that are available at a store l . Each product from N_l has an associated cost c_{jl} and each store l has an associated delivery cost d_l [1].

The objective in the Internet Shopping Optimization Problem is to find a disjoint selection of products purchased from different stores $X = (X_1, \dots, X_m)$ that minimizes the total cost (products cost + delivery cost) subject to the following restrictions: all the products selected from a given store are available in that store, $X_l \subseteq N_l$, all the products from the shopping list are bought, $\cup_{l=1}^m X_l = N$, and the total cost of the shopping list is $F(X) = \sum_{l=1}^m (\delta(|X_l|)d_l + \sum_{j \in X_l} c_{jl})$, where $|X_l|$ is the cardinality of the multiset X_l and the delivery cost is included once per store, independently of the number of products acquired in that store, $\delta(x) = 1$ if $x > 0$ and $\delta(x) = 0$ if $x = 0$ [1].

4 Tabu Search

Tabu Search is a metaheuristic method originally proposed by Glover [4]. It was named after the word “tabu” that comes from Tongan, a language of Polynesia, used by the aborigines of Tonga to indicate things that cannot be touched because are sacred.

The Tabu Search method is used to solve combinatorial optimization problems. Some of the current applications of Tabu Search are financial analysis, scheduling, energy distribution, molecular engineering, waste management, etc. [4]. Tabu Search can be considered as a local search method with a memory structure called tabu list. This memory structure is maintained to avoid getting stuck into local optima. The tabu list holds recently visited elements, thus avoiding cycling by preventing moves to already visited elements.

The Tabu Search algorithm structure is shown in Algorithm 1. This metaheuristics starts with the generation of an initial solution x . Then, based on the initial solution x , the algorithm generates a neighborhood $N(x)$, where N is a neighborhood structure. From this neighborhood, we select those elements in the solution, which are not in the tabu list T . This reduced neighborhood set is denoted by N^* . For each neighbor solution in N^* , the algorithm evaluates its fitness and assigns the best solution found to y . If $fitness(y)$ is better than $fitness(x)$ then we assign y to x . Next, the tabu list is updated by adding y at the end. If the termination condition is not satisfied the algorithm repeats the process, otherwise the best solution found is returned.

Algorithm 1 Tabu Search

Generic structure of the Tabu Search metaheuristics.

Input: Instance of an optimisation problem.

Output: A candidate solution for an optimisation problem.

{*generateInitialSolution(x)*} Generates a candidate solution x for an optimisation problem.

{*generateNeighbourhood*} Generates candidate solutions that are immediate neighbours of an input solution, given a neighbour structure.

{*generateReducedNeighborhood(N,T)*} Generates a list of candidate solutions from the solutions in the neighbor list N that are not in the tabu list, denoted by T

generateInitialSolution(x)

while (\neg *terminationCondition()*) **do**

$N = \text{generateNeighborhood}(x)$

$N^* = \text{generateReducedNeighborhood}(N, T)$

$y = \text{getBest}(\text{fitness}(N^*))$

if *fitness(y)* is better than *fitness(x)* **then**

$x = y$

end if

end while

In our implementation of Tabu Search for finding the optimal solution of ISOP, we start with the generation of an initial solution where we will select those shops where the total cost (product cost + delivery cost) of a product is minimal.

The neighborhood structure was constructed by randomly selecting one product from the current shopping list and purchasing it in all the other stores in the store list where the product is available.

For the Small dataset, the length of the tabu list was set to 3, while for larger instances it was set to 7. The stopping criteria selected was 1000 iterations for the Tabu Search algorithm. In our problem of ISOP we consider a fixed-length short-term memory tabu list.

The construction of the initial solution for this particular implementation of a Tabu Search algorithm applied to ISOP was generated by iteratively selecting the shops where the total cost (product cost + delivery cost) is minimal.

5 Simulated Annealing

Simulated annealing is a probabilistic method proposed by Kirkpatrick et al. [5]. The idea behind Simulated Annealing is taken from the process of annealing in metal work. Annealing is the process of heating and cooling a material to alter its

physical properties due to the change in the internal structure. As the metal cools, a new structure is formed and the metal acquires a new set of properties [5]. This process of controlled cooling is implemented in the Simulated Annealing algorithm, transforming an unordered problem solution to a highly optimized solution.

The structure of the Simulated Annealing algorithm is shown in Algorithm 2. This metaheuristic method starts with an initial solution, a temperature parameter, set to a conveniently chosen high value, and a coefficient of temperature reduction. Then the algorithm enters a cycle in which it selects a neighbor solution from the current one, and using a threshold function with a threshold function, it decides if the new solution is accepted as the current solution or not. The threshold function considers the energies of both solutions (objective values), the temperature and a random value (which involves the Boltzman function) for performing the decision.

Then, the temperature is reduced according to the coefficient of temperature. This makes more difficult that new solutions with worse energy states are accepted in future iterations. The algorithm finishes once the termination condition is reached and the Simulated Annealing method returns the current solution.

Algorithm 2 Simulated Annealing

Generic structure of the Simulated Annealing algorithm

Input: Instance of an optimisation problem

Output: A candidate solution for an optimisation problem

{*selectNeighbor*} Generates a neighbor solution given a neighbor structure and an origin solution.

{*fitness(y)*} Calculates the energy (objective value) of a candidate solution x .

{*acceptance*} Determines if a new solution is accepted to become the current solution, based on the Boltzman function.

{*reduceTemperature*} Decreases the temperature according to a reduction coefficient.

generateInitialSolution(x)

reductionCoefficient

t_k /Initial temperature

repeat

$y = \text{selectNeighbor}(N, x)$

$\Delta E = \text{fitness}(x) - \text{fitness}(y)$

$\text{threshold} = \text{random}(0...1)$

if *acceptance(fitness(x), fitness(y), t_k , threshold)* **then**

$x = y$

end if

$t_k = \text{reduceTemperature}(t_k, \text{reductionCoefficient})$

until *terminationCondition()*

return x

Trajectory Metaheuristics for the Internet Shopping Optimization Problem.

The initial solution for the proposed implementation of a Simulated Annealing algorithm applied to the ISOP was generated by selecting those shops where the total cost (product cost + delivery cost) of a product is minimal.

The initial temperature was initially set to 1000, while the reduction coefficient was set to 0.98. Those values were established empirically.

The neighborhood solution was generated by randomly selecting one product from the current shopping list and purchasing it in another shop where that product is available.

The acceptance condition was calculated using the Boltzman function, with a random probability acceptance value between 0 and 1; and finally, the termination condition was to stop when the temperature was less than 0.1.

6 Computational Experimentation

To test the performance and correctness of the proposed metaheuristic algorithms, we performed a series of computational experiments.

First, we generated several sets of instances with the model proposed in [1], which simulates the real conditions of online shopping book stores, considering price and price dispersion on internet sites like Amazon, Barnes and Noble, Borders.com, Buy.com, Books a million, empik.com and merlin.pl.

This model assumes that each store has all the products in the shopping list of the client. The product prices are randomly generated as explained next. Starting with an initial reference price (ref) for a given product $j : ref_j \in \{2, 4, \dots, 100\}$, the price is randomly chosen with the probabilities of occurrence of 40 % from 0 to 20, 16 % from 22 to 30, 12 % from 32 to 40, 16 % from 42 to 60, and 16 % from 62 to 100. The price for the product j from store i , is $p_{ij} \in [a_{ij}, b_{ij}]$, where $a_{ij} \geq 0.75ref_j$, $b_{ij} \leq 1.36ref_j$ and the intervals between $[a_{ij}, b_{ij}]$ are the following [1]:

Prob. occurrence (%)	$[a_{ij}]$	$[b_{ij}]$
8	$minimum$	$minimum + \frac{ref - minimum}{4}$
3	$minimum + \frac{ref - minimum}{4}$	$minimum + \frac{ref - minimum}{2}$
9	$minimum + \frac{ref - minimum}{2}$	$minimum + \frac{ref - minimum}{1.25}$
21	$minimum + \frac{ref - minimum}{1.25}$	ref
24	ref	$ref + \frac{maximum - ref}{4}$
9	$ref + \frac{maximum - ref}{4}$	$ref + \frac{maximum - ref}{2}$
10	$ref + \frac{maximum - ref}{2}$	$ref + \frac{maximum - ref}{1.25}$
16	$ref + \frac{maximum - ref}{1.25}$	$maximum$

where the percentage indicates the probability for the price of a product to occur.

The delivery costs for each store were taken arbitrarily from 0 to 20 [1].

With this model we created two sets of instances: Small ($10m20n$ with 10 instances, $20m20n$ with 20 instances) with known optimal values, and Large ($40m40n$ with 30 instances).

To test the correctness of both algorithms, we solved the Small set to verify that the proposed metaheuristics did not obtain better solutions than the optimal values, and that the solutions were close to optimality.

To verify the consistency of the solutions obtained by both of the proposed methods, we performed a second computational experiment, in which the algorithms solved the entire Small set 30 times, and then, the results obtained were evaluated with the Friedman test.

The objective to perform the Friedman test was to determine if there were not statistically significant differences in the solutions calculated for each instance during all the repetitions of the experiment.

Finally, once the consistency of both algorithms was determined, we performed a performance test, in which both methods solved the instances in the Large set. The results obtained by both methods were evaluated with the Wilcoxon signed-ranks test to determine the existence of statistically significant differences in the quality of the solutions calculated by the proposed metaheuristics and the heuristics from the state of the art.

The implementation of both metaheuristics was made in the C++ programming language and built with the open source compiler suite, GCC, version 4.2.1. The statistical tests were performed using the statistical computing programming language R, version 3.0.2. The heuristics from the state of the art were implemented in the PHP programming language.

7 Results

The results of the first test to determine if the solutions calculated by the proposed metaheuristics were correct are shown in Fig. 1, where we can observe that there are no anomalous results for any of the instances of the Small set. Also, we can observe a lesser amount of dispersion in the case of the Tabu Search algorithm, which also attains the largest amount of instances for which the optimal value was obtained. In the case of Simulated Annealing, we observed a higher degree of dispersion with few optimal solutions reached.

In the case of the Tabu Search algorithm, this method was able to reach near optimal values on the Small set, with consistency in the quality of the results, as demonstrated with the Friedman Test with multiple samples. Taking a significance value $\alpha = 0.01$, the conclusion of the test showed a p -value of 0.2914, which cannot disprove the null hypothesis H_0 , that all the fitness values calculated by the heuristics are statistically the same over all the repeated measures.

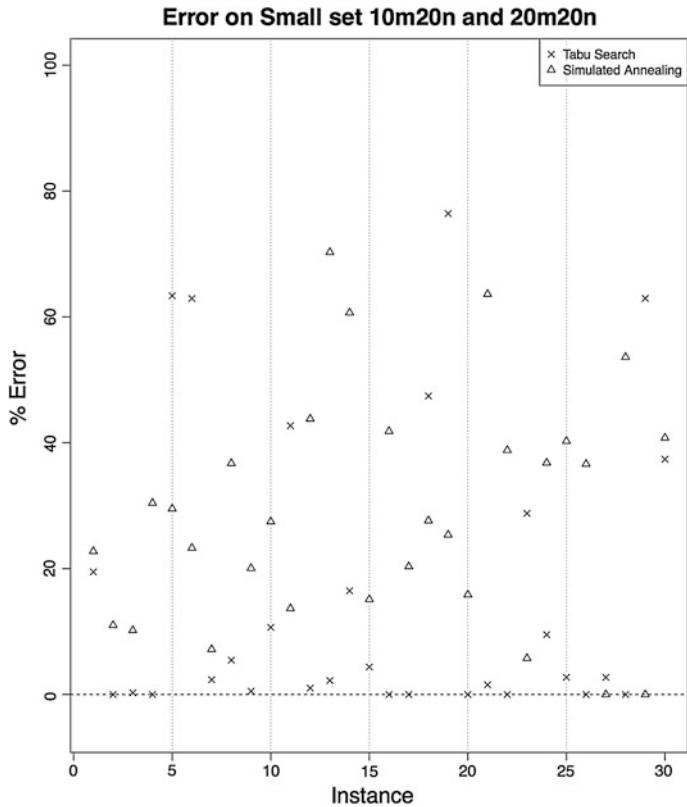


Fig. 1 Percentage of deviation from optimum value for Tabu Search and simulated annealing

On the other hand, the Simulated Annealing algorithm was not consistent in the quality of the results that obtains for the instances of ISOP. When the Friedman Test was applied to the 30 repetitions of the experiment on the Small set, the p -value calculated was $3.092\text{e-}06$, which given a significance value $\alpha = 0.01$, rejects the null hypothesis H_0 , that all the fitness values calculated by the heuristic are statistically the same over all the repeated measures.

For this reason, we used the Tabu Search algorithm to compare its performance versus the heuristics from the state of the art (SHOP-ENUM and PRODUCT-ENUM). Since these heuristics do not include any random stage in their processes, it was not necessary to perform the Friedman Test to the results calculated with these methods. For the Tabu Search algorithm, we performed once again this statistical test on the Large set, and the p -value calculated was 0.03575, which indicates a larger variation than that of the Small set, but it is statistically similar on all the measures given a significance value $\alpha = 0.01$.

To compare the performance of Tabu Search versus SHOP-ENUM and Tabu Search versus PRODUCT-ENUM we solved the Medium set with these methods

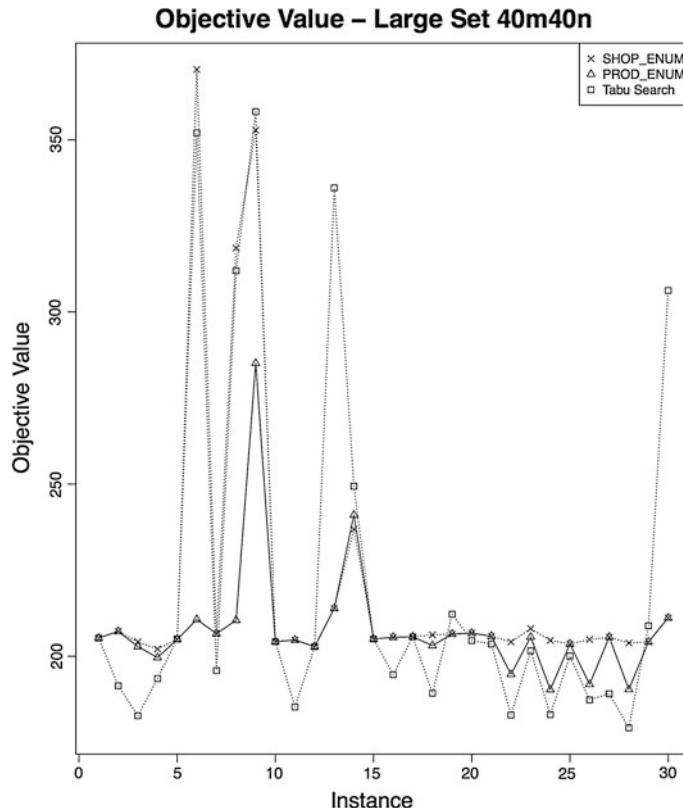


Fig. 2 Fitness values obtained by the methods SHOP-ENUM, PRODUCT-ENUM and Tabu Search on the large set

and applied the Wilcoxon signed-ranks test to the results obtained. In both cases, with calculated p -values of 0.0293 and 0.5905, and a significance value $\alpha = 0.01$, we cannot disprove the null hypothesis that there is no statistical difference between Tabu Search and the state of the art heuristics in terms of performance.

Figure 2 shows the behavior of the three methods, where Tabu Search obtains better solutions for the instances in most cases, but in some instances, the quality of the solution is worse than those computed by the heuristics from the state of art.

8 Conclusions

We proposed two metaheuristics applied to the Internet Shopping Optimization Problem: Tabu Search and Simulated Annealing. The selection of these two heuristics allows dealing with the circumstances of the problem, in which the time to obtain a solution is a primordial factor to consider.

The computational results did not show any significant difference in terms of quality of solution. However, given the nature of the heuristics and metaheuristics, the latter still have a higher margin for improvement with a different adjustment of parameters and the opportunity to include additional intensification and diversification techniques in their processes.

As a future work, we plan to improve the performance of both metaheuristics, while retaining a low consumption in computational time, with special attention on improving the consistency of the objective values found.

Acknowledgments This research project has been partly financed by CONACyT, COTACyT, the Tecnológico Nacional de México campus Instituto Tecnológico de Ciudad Madero, and the University of Luxembourg.

References

1. Blazewicz, J., Kovalyov, M., Musial, J., Urbanski, A., Wojciechowski, A.: Internet shopping optimization problem. *Int. J. Appl. Math. Comput. Sci.* **20**(2), 385–390 (2010)
2. Tolle, K., Chen, H.: Intelligent software agents for electronic commerce. In: Shaw, M., Blanning, R., Strader, T., Whinston, A. (eds.) *Handbook on Electronic Commerce*, pp. 265–382. Springer, Berlin (2000)
3. Kwon, O.B., Sadeh, N.: Applying case-based reasoning and multi-agent intelligent system to context-aware comparative shopping. *Decis. Support Syst.* **37**(2), 199–213 (2004)
4. Glover, F., Laguna, M.: *Tabu search*, 1997. Kluwer Academic Publishers, Dordrecht (1997)
5. Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by simulated annealing. *Science* **220** (4598), 671–680 (1983)

Analysis of Some Database Schemas Used to Evaluate Natural Language Interfaces to Databases

**Rogelio Florencia-Juárez, Juan J. González B.,
Rodolfo A. Pazos R., José A. Martínez F.
and María L. Morales-Rodríguez**

Abstract Most research work about the development of Natural Language Interface to Databases (NLIDB) has been focused on the study of the interpretation and translation from natural language queries to SQL queries. For this purpose and in order to improve the performance in a NLIDB, researchers have addressed different issues related to natural language processing. In addition to this, we consider that the performance of a NLIDB also depends on its ability to adapt to a database schema. For this reason, we analyzed the Geobase, ATIS and Northwind database schemas, commonly used to evaluate NLIDBs. As a result of this analysis, we present in this paper some issues arising from the three database schemas analyzed, which they should be considered in the implementation of a NLIDB to improve its performance.

1 Introduction

A Natural Language Interface to Databases (NLIDB) is a system that allows the user to access information stored in a database by typing requests expressed in some natural language. The user enters a natural language query and the NLIDB trans-

R. Florencia-Juárez · J.J. González B. · R.A. Pazos R. (✉) · J.A. Martínez F. ·
M.L. Morales-Rodríguez

Instituto Tecnológico de Ciudad Madero, Tecnológico Nacional de México,
Madero, México
e-mail: r_pazos_r@yahoo.com.mx

R. Florencia-Juárez
e-mail: rogelio.florencia@live.com.mx

J.J. González B.
e-mail: jjgonzalezbarbosa@hotmail.com

J.A. Martínez F.
e-mail: jose.mtz@gmail.com

M.L. Morales-Rodríguez
e-mail: lmoralesrdz@gmail.com

forms it or translates it to a database query, such as SQL (Structured Query Language). This SQL query is used to extract the information requested by the user from the database. The objective is to facilitate to casual users, i.e. users that do not have knowledge in database query languages, the access to information stored in a database. Although good results have been reported in the literature, the NLIDBs have failed to achieve the performance expected by users, since its implementation has proven more challenging than expected.

To improve performance, researchers have focused on the interpretation and translation from Natural Language queries to SQL queries. To this end, they have been addressed different issues related to natural language processing (e.g. ellipsis and ambiguity problems). However, we consider that the performance also depends on the ability of an interface to adapt to a database and to handle situations present in the database schema.

In this chapter, we present some issues arising from a database schema. To this end, we analyzed the schema of the databases commonly used in the literature. The analyzed databases were Geobase, ATIS, and Northwind. The results obtained from the analyzed databases indicate that the Geobase and Northwind schemas have similar size and both present similar issues. Although the ATIS schema is the biggest and the most complex schema, it is small compared to databases of the real world.

In Sect. 2, we mention some databases that have been used to evaluate NLIDBs; in Sect. 3, we present the analysis of the Geobase, ATIS, and Northwind database schemas; in Sect. 4 we present a comparative numerical analysis of the characteristics of each analyzed database schema, and in Sect. 5, we present some conclusions and remarks.

2 Background

The NLIDBs research topic began to be developed around the 60s. The first NLIDBs were designed to be used in a particular database. One of the pioneer systems was BASEBALL [12]. It answered questions generating queries against the structured database containing the baseball data. The most well-remembered other early work is the LUNAR system [34], designed to enable a lunar geologist to conveniently access, compare and evaluate the chemical analysis data on lunar rock and soil composition that was accumulating as a result of the Apollo moon mission. Hendrix [14] implemented LADDER. It was designed as a natural language interface to a database of information about US Navy ships.

In the 80s, the researchers focused in achieving the portability of the NLIDBs. CHAT-80 [32] was implemented in Prolog. The database consists of facts (i.e. oceans, major seas, major rivers and major cities) about 150 of the countries world. PARLANCE [1] was tested in two different versions of a large Navy database. JANUS [3] used several knowledge bases maintained by the U.S. Navy. The knowledge bases contain, among other things, information about the deployment

schedules, locations, and readiness conditions of the ships in the Pacific Fleet. TEAM [13] used a database that comprises four files or relations of geographic data.

To date, a large number of NLIDBs have been designed, such as, spoken language understanding systems, NLIDBs that use techniques such as Learning Semantic Parsing (process of mapping a natural language sentence into a complete formal meaning representation or logical form), NLIDBs that map natural language words to database schema elements, and recently, interfaces oriented towards the Semantic Web, where one important aspect of research is the use of ontologies as a suitable representation of knowledge available on the Web [4]. Table 1 show the databases commonly used in evaluating some of these NLIDBs.

Based on the Table 1, it can be seen that the three databases commonly used to evaluate NLIDBs are Geobase, ATIS and Northwind, which were used by 14, 6 and 3 NLIDBs respectively. A numerical analysis of the characteristics of these schemas is shown in Table 2.

Table 1 Databases commonly used in evaluating NLIDBs

Interface	Date	Databases
CMU [31]	1990	ATIS
MIT [24]	1990	ATIS
SRI [21]	1991	ATIS
BBN Byblos [18]	1992	ATIS
AT&T [29]	1994	ATIS
CHILL [35]	1996	Geobase
ENGLISH WIZARD [27]	1997	Northwind
COCKTAIL [26]	2001	Geobase
ELF [8]	2002	Northwind
ENGLISH QUERY [19]	2002	Northwind
NLPQC [25]	2002	Virutal Library (Cindi)
WOLFIE [28]	2003	Geobase
PRECISE [23]	2004	Geobase, ATIS
GINSENG [2]	2005	Geobase
KRISP [15]	2006	Geobase
QUERIX [16]	2006	Geobase
SCISSOR [10]	2006	Geobase
WASP [33]	2006	Geobase
NLP-REDUCE [17]	2007	Geobase, Restaurants
ORAKEL [6]	2007	Geographical facts about Germany
PANTO [30]	2007	Geobase, Restaurants, Jobs
C-PHRASE [20]	2010	Geobase
FREyA [7]	2010	Geobase
GIORDANI [11]	2010	Geobase

Table 2 Characteristics of the three database schemas analyzed

Characteristic	Northwind	Geobase	ATIS
Tables	13	9	27
Columns	88	41	123
Relationships	12	8	29
Compound keys	0	0	1
Tables with multiple foreign keys	0	0	1
Multiple paths	NO	NO	YES
Length of the longest path	3	3	4

3 Databases Analysis

In this section we present some issues identified in the Geobase, ATIS and Northwind schemas. These issues may improve the performance of a NLIDB.

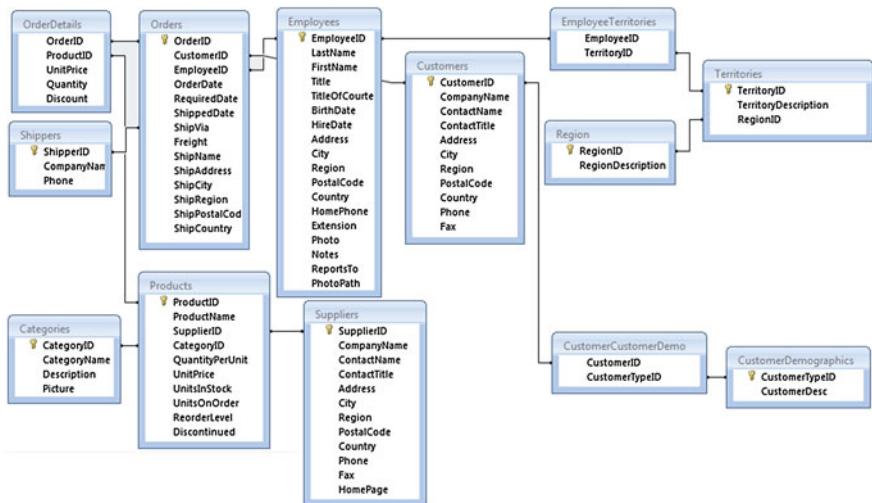
3.1 Northwind

Northwind is a sample database with information on a shipping company. The database comes as a demo with all distributed copies of Microsoft Access and as a sample database in SQL Server 7.0.

The Northwind database schema is composed of 13 tables, 88 columns and 12 relationships. The tables are joined through simple primary keys. A *simple primary key* is just a single attribute or column that uniquely identifies a row. This is an acyclic schema, that is, there is only one path between a table and another. The Northwind database schema is presented in Fig. 1.

We believe that the major challenge of Northwind to a NLIDB is the *ambiguity* which occurs under the following circumstances:

- *Columns with the same name but belonging to different tables.* As an example, we can mention the *CompanyName* column which is part of the *Suppliers*, *Shippers* and *Customers* tables. If a NLIDB receives the query “Show me the company name of the order 10259”, it would have to determine whether the user wants to see the *CompanyName* of the *Customers* table or of the *Shippers* table, since both tables are related to the *Orders* table, as can be seen in Fig. 1.
- *A same word being part of different column names or table names.* Generally, the table and column names are composed of more than one word where one of these words may be part of the name of other table names and/or other column names. As an example we can mention the *EmployeesTerritories* and *Employees* tables, which have in common the *Employee* word. Another example is the *OrderDate*, *ShippedDate* and *RequiredDate* columns of the *Orders* table, which have in common the *Date* word (see Fig. 1). If a NLIDB receives the query

**Fig. 1** Northwind database schema

“Show me the date of order 10528”, it would have to determine the column which is referred by the *date* word.

- A *same value stored in different columns*. For example, the *Ernst Handel* value is stored in both the *CompanyName* and *ShipName* columns of the *Customers* and *Orders* tables, respectively (see Fig. 2). If a NLIDB receives the query “*What can you tell me about Ernst Handel?*”, it would have to determine whether the user wants to see the general information about this customer or the orders related to this customer.

Other issues may arise from how the information was standardized in the schema of a database. For example, Northwind presents the following issues:

Customers			
CustomerID	CompanyName	ContactName	ShipName
DRACD	Drachenblut Delikatess	Sven	
DUMON	Du monde entier	Janir	
EASTC	Eastern Connection	Ann	
ERNSH	Ernst Handel	Roland	
FAMIA	Familia Arquibaldo	Aria	
FISSA	FISSA Fabrica Inter. Sal.	Diego	
FOLIG	Folies gourmandes	Mart	
FOLKO	Folk och fä HB	Mari	
FRANR	France restauration	Carin	
FRANS	Franchi S.p.A.	Paolo	
FRANK	Frankenversand	Pete	
FLRID	Furtado Pecalau e Frutor	Liane	

Orders			
OrderID	Custon	ShipName	ShipAd
10002	DU MON	Du monde ent	67, rue de
10890	DUMON	Du monde ent	67, rue de
10683	DUMON	Du monde ent	67, rue de
10532	EASTC	Eastern Conne	35 King G
11056	EASTC	Eastern Conne	35 King G
11047	FASTC	Eastern Conne	35 King G
10836	ERNSH	Ernst Handel	Kirchgass
11017	ERNSH	Ernst Handel	Kirchgass
10390	ERNSH	Ernst Handel	Kirchgass
10895	ERNSH	Ernst Handel	Kirchgass
10595	ERNSH	Ernst Handel	Kirchgass
10514	ERNSH	Frnc Handel	Kirchgaess

Fig. 2 The Ernst Handel value is stored in the customer and orders tables

Fig. 3 Information stored in the FirstName and LastName columns of the Employee table

FirstName	LastName
Brenda	Diaz
Brian	Goldstein
Brian	Lloyd
Brian	Welcker
Brian	LaMee
Britta	Simon
Bryan	Walton
Bryan	Baker
Candv	Snoon

- *Header and detail tables.* They are used to regularly store items in: service orders, invoices, quotes, etc. In Northwind, the information about customer orders is stored in the *Orders* and *OrderDetails* tables. The *Orders* table (header table) stores general information about orders and *OrderDetails* (detail table) stores the items specified in the orders. If a NLIDB receives the query “*Show me details of the order 10528*”, it would have to determine whether the *details* word refers to the header information or to the detail information.
- *Storage of the information.* The full name of the employees is stored in the *FirstName* and *LastName* columns in the Employee table. If a NLIDB receives the query “*What are our employee names?*” is desirable that it shows both columns. Show only the name (*FirstName*) could be non-functional if there are employees with the same name (see the left side of the Fig. 3). For this reason, an NLIDB should know that the name is composed of both columns (see the right side of the Fig. 3).

3.2 Geobase

Geobase is a system that was supplied as a sample application with Turbo Prolog 2.0 (Borland International 1988). This system provides a database already coded in Prolog. The database contains about 800 Prolog facts asserting relational tables for basic information about U.S. states, including: population, area, capital city, neighboring states, major rivers, major cities, and highest and lowest points along with their elevation.

This database has been migrated to different relational database schemas. For example, Chandra [5] generated a relational database schema composed of 10 tables, 37 columns and 9 relationships (see the left side of the Fig. 4) and ELF [9] generated a relational database schema composed of 9 tables, 8 relationships and approximately 41 columns (see the right side of the Fig. 4). In both schemas, the tables are related through simple primary keys and both schemas are acyclic schemas.

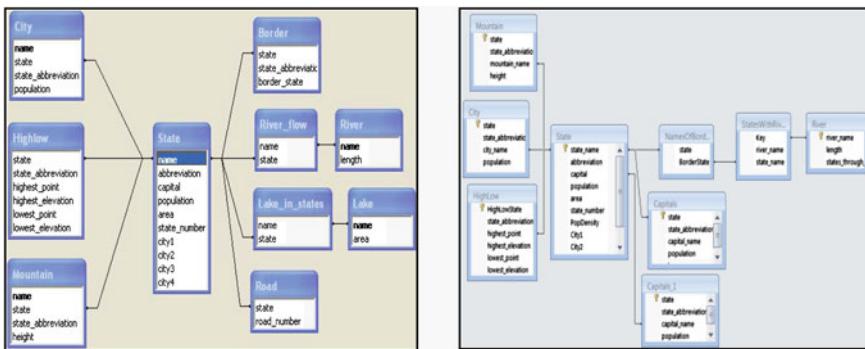


Fig. 4 Schemas used by Chandra [5] and ELF [9]

Both schemas store descriptive information for the user in primary key columns as shown to the left of Fig. 5. This situation allows an interface to show correct information to the user, since primary key columns and columns that store descriptive information for the user are the same. However, this situation is not common in databases, since primary key columns regularly store unintelligible information for the users as shown to the right of Fig. 5. Storing unintelligible information for the user in primary key columns could imply that an interface must identify which columns should be shown to the user.

In the schema used by Chandra [5] is stored the names of the states surrounding a state (e.g. Arizona is surrounded by California, Colorado, Nevada, New Mexico and Utah). This information is stored in the *Border* and *State* tables which are joined through the *State* column of the *Border* table which is a foreign key column to the *State* table. In the same way, we consider that the *Border_State* column of the *Border* table should also be a foreign key to the *State* table.

In the schema used by ELF [9] exist the *City* and *Capitals* tables. This might not be optimal, since a capital is also a city, so there may be redundant information.

Primary key	
Name	Abbreviation
alabama	al
alaska	ak
arizona	az
arkansas	ar
california	ca

Primary key	
StateId	Name
1	alabama
2	alaska
3	arizona
4	arkansas
5	california

Fig. 5 Primary key columns with descriptive information and primary key columns with unintelligible information to the users

3.3 ATIS

The ATIS database consists of data obtained from the Official Airline Guide [22], organized under a relational schema. It contains information about flights, fares, airlines, cities, airports, and ground services, and includes twenty-five supporting tables.

The schema is composed of 27 tables, 123 columns and 29 relationships. The tables are joined through 28 simple primary keys and 1 compound key. A *compound key* is a key that consists of two or more attributes that uniquely identify an entity occurrence. The *compound key* is used to join the *ground_service* and *airport_service* tables through the *airport_code* and *city_code* columns existing in both tables (see Fig. 6). This is a cyclic schema, since there is more than one path to join the *Flight* and *City* tables. The ATIS database schema is presented in Fig. 7.

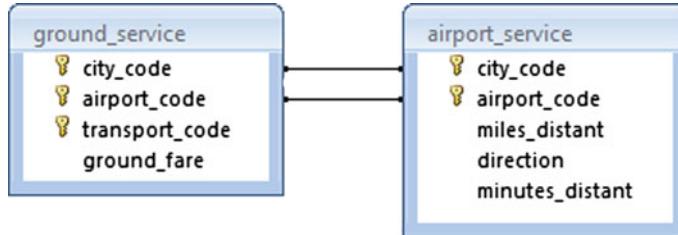
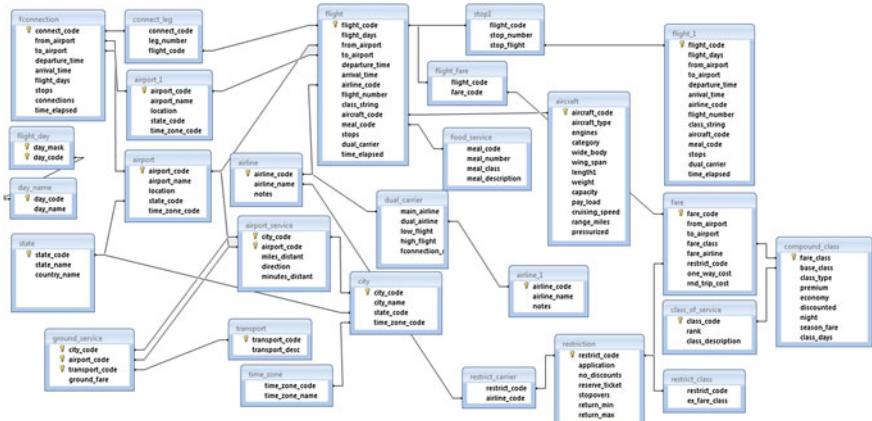


Fig. 6 Compound key



The ATIS database schema mainly presents two major problems:

- *Multiple foreign keys pointing to the same primary key of another table.* The challenge for domain-independent NLIDBs is to successfully perform table joins. For example, in the query “Show me Flights from Atlanta to Boston”, the interface must identify that the *Atlanta* and *Boston* values are stored in the *City* table and that they are referenced by different columns (*Atlanta* is referred by the *from_airport* column of the *Flight* table and *Boston* by the *to_airport* column). Subsequently, table joins must be performed considering that the *Atlanta* and *Boston* values are stored in the same table (*City*). For this, the tables needed to join the *Flight* table to the *City* table should be included twice in the FROM clause, but with different aliases. To illustrate this, we show a possible SQL query below.

```

SELECT *
FROM flight A,
     airport B1, airport_service B2, city B3,
     airport C1, airport_service C2, city C3
WHERE A.from_airport = B1.airport_code AND
      B1.airport_code = B2.airport_code AND
      B2.city_code = B3.city_code AND
      B3.city_name = 'ATLANTA' AND
      A.to_airport = C1.airport_code AND
      C1.airport_code = C2.airport_code AND
      C2.city_code = C3.city_code AND
      C3.city_name = 'BOSTON'

```

- *Cycles or multiple paths to join tables.* It occurs when there is more than one path to connect a pair of tables. Choose the correct path to perform the join of tables in the SQL query is not a trivial task. For example, there are two paths in ATIS to join the *Flight* and *City* tables, however only one is the correct path. One path is formed by joining the *Flight-Airport-State-City* tables and the other is formed by joining the *Flight-Airport-Airport_Service-City* tables. The choice of the path depends on the information stored in the tables in the path. To answer the query “Show me flights from Baltimore to Philadelphia” the first path is incorrect. This is because in the *Airport* table there are two airports related to the state of Pennsylvania (the Pittsburgh and Philadelphia airports) and when the *Airport* table is joined to the *Flight* table, a Cartesian product is performed between both tables considering both airports. This causes that flights to Pittsburgh, not required by the user, be also shown. This is not the case in the second path because *Airport_Service* table stores information about the cities that are served by an airport.

Fig. 8 Information stored in the Day_Name table

day_code	day_name
1	MONDAY
2	TUESDAY
3	WEDNESDAY
4	THURSDAY
5	FRIDAY
6	SATURDAY
7	SUNDAY

Besides these two major problems, there are other issues that could prevent a good performance to a NLIDB. For example, in the *flight_days* column of the *Flight* table is stored the weekdays when a flight is scheduled. The weekdays are represented with numbers from 1 to 7 as shown in Fig. 8.

For example, the value 123456—indicates that a flight is scheduled from Monday to Saturday. The value-6-indicates that a flight is only scheduled on Saturday. If a NLIDB receives the query “*Show me all the flights from Dallas to Atlanta leaving on a Saturday morning*”, the interface must identify that the Saturday is represented by the number 6 and subsequently, it should look for occurrences of this number in the string stored in the *flight_days* column.

Another similar problem occurs with time intervals in a day (*morning*, *afternoon*, *evening*, *night*, *day*, etc.). These time intervals or ranges are defined in the *time_interval* table (Fig. 9).

If a NLIDB receives the query “*Please show the cost of the morning flights from Dallas to Denver*”, the interface must identify that the morning word represents a range by identifying that morning word is a value stored in the *Period* column of the *time_interval* table. The *begin_time* and *end_time* columns of this table represent the initial and final value of the range respectively. To illustrate this, we show a possible SQL query below.

Fig. 9 Information stored in the Time_Interval table

period	begin_time	end_time
morning	0	1200
afternoon	1200	1800
evening	1800	2000
day	600	1800
night	1800	2400
night	0	600
early morning	0	800
mid-morning	800	1000
late morning	1000	1200
early afternoon	1200	1400
mid-afternoon	1400	1600
late afternoon	1600	1800

```

SELECT DISTINCT A.from_airport, A.to_airport
FROM flight A, time_interval B,
     airport C1, airport_service C2, city C3,
     airport D1, airport_service D2, city D3
WHERE A.departure_time
      BETWEEN B.begin_time AND B.end_time AND
      B.period = 'morning' AND
      A.from_airport = C1.airport_code AND
      C1.airport_code = C2.airport_code AND
      C2.city_code = C3.city_code AND
      C3.city_name = 'DALLAS' AND
      A.to_airport = D1.airport_code AND
      D1.airport_code = D2.airport_code AND
      D2.city_code = D3.city_code AND
      D3.city_name = 'DENVER'

```

4 Analysis of Results

In this section, we present a comparative numerical analysis of the characteristics of the three database schemas analyzed.

Based on the characteristics described in the Table 2, we believe that the three databases can be classified according to the size and complexity of their schemas.

For the classification by size, we consider the number of tables, columns, relationships and the length of the longest path. For the classification by complexity, we consider the number of compound keys, the number of multiple foreign keys and whether the schema has multiple paths.

Based on size, it can be said that Geobase and Northwind are similar because both have similar number of tables, relationships, and also, the length of the longest path is the same. Northwind has a larger number of columns, but this does not make it longer than Geobase. ATIS is the database that has the longer schema.

Based on complexity, it can be say that ATIS is the database that has the most complex schema, as it has multiple foreign keys, multiple paths and compound keys.

Looking into the literature we found the Adventureworks database. It models the common departments of an enterprise such as purchasing, sales, production, personnel, human resources, etc. Its schema is shown in Fig. 10.

We compared the ATIS and Adventureworks schemas. A comparative numerical analysis is shown in Table 3.

As can be seen, the Adventureworks database schema is longer and more complex than the ATIS database schema.

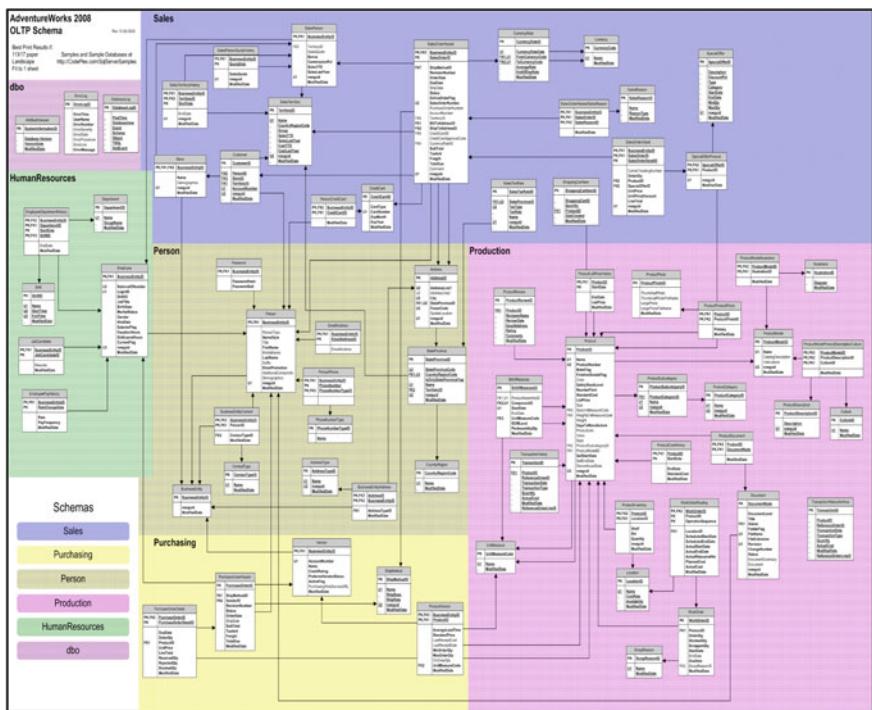


Fig. 10 Microsoft Adventureworks database schema

Table 3 Characteristics of the ATIS and Adventureworks schemas

Characteristic	ATIS	Adventureworks 2008
Tables	27	68
Columns	123	465
Relationships	29	91
Compound keys	1	5
Tables with multiple foreign keys	1	5
Multiple paths	YES	YES
Length of the longest path	4	8

5 Conclusions and Remarks

A NLIDB is a system that allows the user to access information stored in a database by typing requests expressed in some natural language. Although good results have been reported in the literature, the NLIDBs have failed to achieve the performance expected by users.

To improve performance, researchers have focused on the interpretation and translation from Natural Language queries to SQL queries. However, we consider that the performance of a NLIDB also depends on its ability to handle issues present in a database schema.

We analyzed the schema of the Geobase, ATIS and Northwind databases. We identified that each analyzed schema has different issues, so it is necessary a deeper characterization of issues underlying to database schemas. It was also identified that, although ATIS is the most complex schema, it is small compared to databases of real world as Adventureworks.

As a final reflection, we believe that an interface can provide more intelligent services and even, increase its performance whether issues related to database schemas are integrated into its knowledge. We also pretend to encourage new NLIDBs be evaluated using resources similar to those used in real world applications which exhibit a greater degree of complexity.

References

1. Bates, M.: Rapid porting of the parlance natural language interface. In: Proceedings of the workshop on Speech and Natural Language (pp. 83–88). Association for Computational Linguistics. (1989)
2. Bernstein, A., Kaufmann, E., Kaiser, C.: Querying the semantic web with Ginseng: a guided input natural language search engine. In: 15th workshop on information technologies and systems (WITS 2005), Las Vegas, NV. (2005)
3. Bobrow, R.J., Resnik, P., Weischedel, R.M.: Multiple underlying systems: translating user request into programs to produce answers. In: Proceedings of the 28th annual meeting of ACL, Pittsburgh, pp. 227–234. (1990)
4. Buraga, S.C., Cojocaru, L., Nichifor, O.C.: Survey on web ontology editing tools, transactions on automatic control and computer science, Romania, pp. 1–6. (2006)
5. Chandra, Y.: Natural language interfaces to databases. Doctoral dissertation, University of North Texas. (2006)
6. Cimiano, P., Haase, P., Heizmann, J.: Porting natural language interfaces between domains: an experimental user study with the ORAKEL system. In: Proceedings of the 12th international conference on intelligent user interfaces, Honolulu, Hawaii, pp. 180–190. (2007)
7. Damjanovic, D., Agatonovic, M. Cunningham, H.: Natural language interfaces to ontologies: combining syntactic analysis and ontology-based lookup through the user interaction. In: Proceedings of the 7th extended semantic web conference (ESWC 2010). Springer, Heraklion (2010)
8. ELF software (English Language Frontend), “Demo Gallery”, <http://www.elfsoft.com/help/acelf/overview.htm>. (2002)
9. ELF software (English Language Frontend), “GeoQuery Example”, <http://www.elfsoft.com/GeoQuery.htm>. (2012)
10. Ge, R., Mooney, R.J.: A statistical semantic parser that integrates syntax and semantics. In: Proceedings of the ninth conference on computational natural language learning, pp. 9–16. (2005)
11. Giordani, A., Moschitti, A.: Semantic mapping between natural language questions and SQL queries via syntactic pairing. In: Natural language processing and information systems, pp. 207–221. Springer, Berlin (2010)

12. Green, Jr., B.F., Wolf, A.K., Chomsky, C., Laughery, K.: Baseball: an automatic question-answerer. In: Papers presented at the May 9–11, 1961, western joint IRE-AIEE-ACM computer conference, pp. 219–224. ACM (1961)
13. Grosz, B.J., Appelt, D.E., Martin, P.A., Pereira, F.C.: TEAM: an experiment in the design of transportable natural-language interfaces. *Artif. Intell.* **32**(2), 173–243 (1987)
14. Hendrix, G.G., Sacerdoti, E.D., Sagalowicz, D., Slocum, J.: Developing a natural language interface to complex data. *ACM Trans. Database Syst. (TODS)* **3**(2), 105–147 (1978)
15. Kate, R.J., Mooney, R.J.: Using string-kernels for learning semantic parsers. In: Proceedings of the 21st international conference on computational linguistics and the 44th annual meeting of the association for computational linguistics, pp. 913–920. Association for Computational Linguistics (2006)
16. Kaufmann, E., Bernstein, A., Zumstein, R.: Querix: a natural language interface to query ontologies based on clarification dialogs. In: 5th international semantic web conference (ISWC 2006), pp. 980–981 (2006)
17. Kaufmann, E., Bernstein, A., Fischer, L.: NLP-Reduce: a naïve but Domain-independent natural language interface for querying ontologies. In: 4th European semantic web conference (ESWC 2007), Innsbruck, A (2007)
18. Kubala, F., Barry, C., Bates, M., Bobrow, R., Fung, P., Ingria, R., Stallard, D.: BBN Byblos and HARC February 1992 ATIS benchmark results. In: Proceedings of the workshop on speech and natural language, pp. 72–77. Association for Computational Linguistics (1992)
19. Microsoft TechNet., chapter 32-English Query Best Practices. www.microsoft.com/technet/prodtechnol/sql/2000/reskit/part9/c3261.mspx?mfr=true
20. Minock, M., Olofsson, P., Näslund, A.: Towards building robust natural language interfaces to databases. In: Natural language and information systems, pp. 187–198. Springer, Berlin (2008)
21. Murveit, H., et al.: SRI's speech and natural language evaluation (in this Proceedings)
22. Official Airline Guides, Official Airline Guide, North American Edition with Fares, Oakbrook, Illinois, Volume 16, No. 7, January 1, 1990
23. Popescu, A.: Modern natural language interfaces to databases: composing statistical parsing with semantic tractability. University of Washington (2004)
24. Seneff, S., et al.: Development and Preliminary Evaluation of the M1T ATIS System (in this Proceedings)
25. Stratica, N.: Using semantic templates for a natural language interface to the CINDI virtual library. *Data Knowledge Eng.* **55**, 4–19 (2005). doi:[10.1016/j.datak.2004.12.002](https://doi.org/10.1016/j.datak.2004.12.002). (Elsevier Science Publishers, The Netherlands)
26. Tang, L.R., Mooney, R.J.: Using multiple clause constructors in inductive logic programming for semantic parsing. In: Machine learning: ECML 2001, pp. 466–477. Springer, Berlin (2001)
27. Technology, Linguistic: English Wizard—Dictionary Administrator's Guide. Linguistic Technology Corp, Littleton (1997)
28. Thompson, C.A., Mooney, R.J.: Acquiring word-meaning mappings for natural language interfaces. *J. Artif. Intell. Res.* **18**(1), 1–44 (2003)
29. Tzoukermann, E.: The use of a commercial natural language interface in the ATIS task. In: HLT (1991)
30. Wang, C., Xiong, M., Zhou, Q.: PANTO: a portable natural language interface to ontologies. *Lecture notes in computer science*, vol. 4519/2007, pp. 473–487 (2007)
31. Ward, W.: Current status of the CMU ATIS System (in this Proceedings)
32. Warren, D.H.: Efficient processing of interactive relational data base queries expressed in logic. In: Proceedings of the seventh international conference on very large data bases, vol. 7, pp. 272–281. VLDB Endowment (1981)
33. Wong, Y.W., Mooney, R.J.: Learning for semantic parsing with statistical machine translation. In: Proceedings of human language technology conference/North American chapter of the association for computational linguistics annual meeting, pp. 439–446 (2006)

34. Woods, W., Kaplan, R., Webber, B.: The Lunar Sciences Natural Language Information System. Bolt Beranek and Newman Inc., Cambridge, Massachusetts Final Report. B. B. N. Report No 2378 (1972)
35. Zelle, J.M., Mooney, R.J.: Learning to parse database queries using inductive logic programming. In: Proceedings of the thirteenth national conference on artificial intelligence, pp. 1050–1055 (1996)

Part VIII

Fuzzy Logic and Metaheuristics

Cuckoo Search Algorithm via Lévy Flight with Dynamic Adaptation of Parameter Using Fuzzy Logic for Benchmark Mathematical Functions

Maribel Guerrero, Oscar Castillo and Mario García

Abstract The proposal described in this paper uses the Cuckoo Search (CS) Algorithm via Lévy flights as an optimization method and its enhancement using a fuzzy system to dynamically adapt its parameter. The original method is compared with the proposed method called Fuzzy Cuckoo Search (FCS). In this case we consider a fuzzy system to dynamically change the Pa variable. Simulation results on a set of mathematical functions with the FCS outperform the traditional CS.

1 Introduction

The Cuckoo Search Algorithm (CS) is a meta-heuristic optimization method proposed by Xin-She Yang and Suash Deb in 2009. CS is based on the brood parasitism of some cuckoo species. In addition, this algorithm is enhanced by the so-called Lévy flights [18].

CS was inspired by the obligate brood parasitism of some cuckoo species by laying their eggs in the nests of other host birds (of other species). The control parameters for the CS algorithm are: The mutation probability value (Pa) that replaces nests with no good solutions, the scale factor (β) and the step size, which should be related to the scales of the problem of interests (α).

Some host birds can engage in direct conflict with the intruding cuckoos. For example, if a host bird discovers the eggs are not their own, simply abandons its nest and builds a new nest elsewhere.

Fuzzy logic or multi-valued logic is based on fuzzy set theory proposed by Zadeh [19], which helps us in modeling knowledge, through the use of if-then fuzzy rules [8, 19].

M. Guerrero · O. Castillo (✉) · M. García
Tijuana Institute of Technology, Tijuana, B.C., Mexico
e-mail: ocastillo@tectijuana.mx

Fuzzy set theory provides a systematic calculus to deal with linguistic information, and that improves the numerical computation by using linguistic labels stipulated by membership functions [6, 8].

The use of fuzzy systems in evolutionary algorithms and bio-inspired optimization is becoming a common approach to improve the performance of these algorithms [3, 8, 14, 15].

In this paper, we propose the use fuzzy logic to dynamically adjust the Pa parameter in the Cuckoo Search algorithm. The aim is to improve the Cuckoo Search algorithm.

It is no exaggeration to say that optimization is everywhere, from engineering design to business planning and from the routing of the Internet to holiday planning. In almost all these activities, we are always trying to achieve certain objectives or to optimize something such as profit, quality and time. As resources, time and money are always limited in real-world applications, and we have to find solutions to optimally use these valuable resources under various constraints. Mathematical optimization or programming is the study of such planning and design problem using mathematical tools. Nowadays, computer simulations become an indispensable tool for solving such optimization problems with various efficient search algorithms [18].

In spite of being a virtually young algorithm, the CS algorithm has been implemented in different areas of application. To mention a few: Data clustering using Cuckoo Search Algorithm (CSA) in which they propose new approaches for using (CSA) to cluster data [7]. A Hybrid CSA has been proposed in [5]. Cuckoo Search based adaptive infinite impulse response (IIR) system identification scheme [9], object-oriented software system that implements a Cuckoo Search (CS) meta heuristic for unconstrained optimization problems. They developed the algorithm in JAVA programming language, which is faster and easier for maintenance since it is object-oriented [1]. Modified Cuckoo Search: A new gradient free optimization algorithm, which can be regarded as a modification of the recently developed Cuckoo Search is proposed in [16].

The contents of this paper are structured as follows: Sect. 2 shows the concept of the CS, Sect. 3 describes the proposed method, Sect. 4 presents the benchmark mathematical functions, in Sect. 5 the proposed Fuzzy system, is described Sect. 6 contains the experiments and methodology, Sect. 7 shows the simulation results and in Sect. 8 the conclusions are presented.

2 Cuckoo Search Algorithm

The Cuckoo is a fascinating bird, not only because of the beautiful sound it can make, but also because of their aggressive reproduction strategy. Some species such as the Ani and Guira cuckoos lay their eggs in communal nests, though they may remove others' eggs to increase the hatching probability of their own eggs. Quite a number of species engage the obligate brood parasitism by laying their eggs in the nests of other host birds (often other species) [17].

2.1 Rules of Cuckoo Search

The breeding behavior of cuckoos is described below in the basic steps of cuckoo search using the following three rules [11]:

1. Each cuckoo lays one egg at a time, and dumps its egg in randomly chosen nest.
2. The best nests with high quality of eggs will carry over to the next iterations.
3. The number of available host nests is fixed, and the egg laid by a cuckoo is discovered by the host bird with a probability Pa in $[0, 1]$. In this case, the host bird can either throw the egg away or abandon the nest, and build a completely new nest. For simplicity, this last assumption can be approximated by the Pa fraction of the n nests is replaced by new nests (with new random solutions).

2.2 Flowchart of Cuckoo Search Algorithm

The basic steps of the Cuckoo Search (CS) can be summarized as the flowchart of the conventional CS shown in Fig. 1.

We implemented the fuzzy system for the dynamic adaptation of the Pa parameter of worse nests and build new ones at new locations via Lévy flights.

2.3 Lévy Flights

On the other hand, various studies have shown that the flight behavior of many animals and insects have demonstrated the typical characteristics of Lévy flights [2, 5, 10, 12, 13].

A recent study by Reynolds and Frye shows that fruit flies or *Drosophila melanogaster*, explore their landscape using a series of straight flight paths punctuated by a sudden 90° turn, leading to a Lévy-flight-style intermittent scale free search pattern [5]. In Fig. 2, we observe the behavior of a Lévy flight.

2.4 Equations to Generate New Positions

When generating new solutions $x_i^{(t+1)}$, for say a cuckoo i , a Lévy flight is performed using Eq. 1:

$$x_i^{(t+1)} = x_i^t + \alpha \cdot \text{randn}(\text{size}(D)) \oplus \text{Lévy}(\beta) \oplus S_i \quad (1)$$

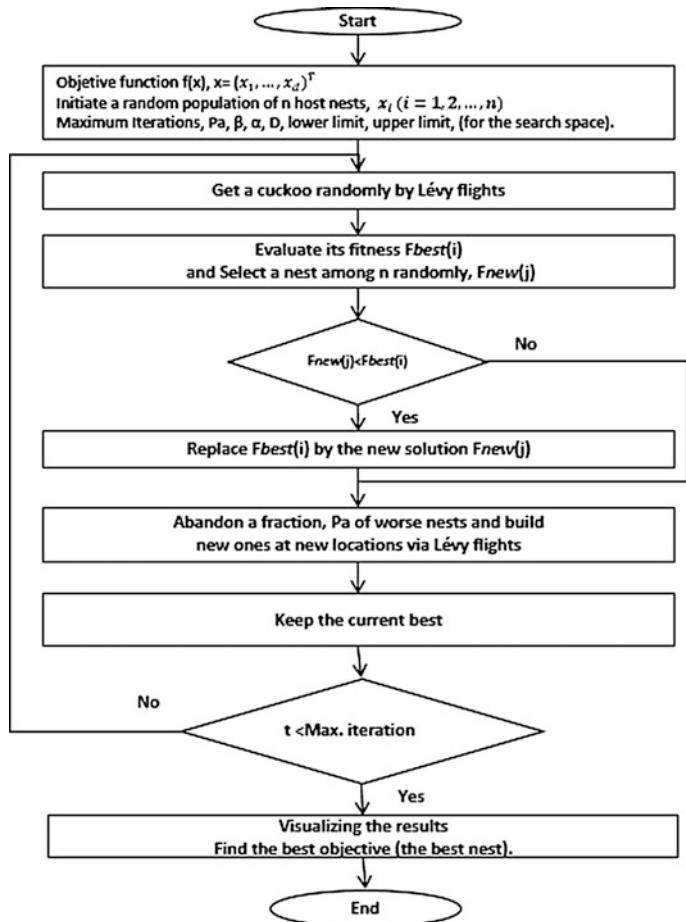
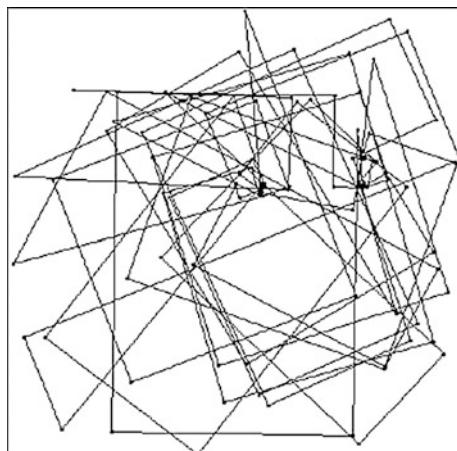


Fig. 1 Flowchart of the conventional Cuckoo search algorithm

Fig. 2 Possible Levy flight path



where

$x_i^{(t+1)}$: is the vector representing the new position.

x_i^t : is the vector representing current position.

α : is the step size, which should be related to the scales of the problem of interest, where $\alpha \geq 0$.

$randn(size(D))$: is a random scalar value.

The product \oplus means entry-wise multiplications.

Lévy(β): it is a Lévy flight.

In Eq. 2 we have that S_i is defined by the following difference:

$$S_i = (x_i^t - x_{best}) \quad (2)$$

where

x_i^t : is the position vector in the nest we are evaluating.

x_{best} : is a vector containing the best position found so far.

If $x_i^t = x_{best}$, the solution is maintained.

2.4.1 Equation of the Lévy Flight

The main equation of the Lévy flight is:

$$\text{Lévy}(\beta) = \frac{\sigma(\beta) \times \text{randn}}{|\text{randn}|^{\frac{1}{\beta}}} \quad (3)$$

where

β : is a constant ($1 < \beta \leq 2$)

The *randn* function generates a uniform integer in the interval [1 d].

The standard deviation sigma is calculated with the following equation:

$$\sigma(\beta) = \left\{ \frac{\Gamma(1 + \beta) \times \sin(\frac{\pi\beta}{2})}{\Gamma\left[\frac{(1+\beta)}{2}\right] \times \beta 2^{\left(\frac{\beta-1}{2}\right)}} \right\}^{\frac{1}{\beta}}, \quad \sigma(\beta) = 1 \quad (4)$$

The Gamma function is defined by the integral:

$$\Gamma(z) = \int_0^\infty t^{z-1} e^{-t} dt \quad (5)$$

The Lévy flight represents a random trajectory and it is the length of the step obtained by means of a Lévy distribution (Eq. 6):

$$\text{Lévy} \sim u = t^{-\beta}, \quad (1 < \beta < 2) \quad (6)$$

Which has an infinite variance with an infinite mean.

3 Proposed Method

In Fig. 3 we show the proposal where we have a FCS1 and FCS2 with a dynamically update of the Pa parameter, for a problem of minimization using fuzzy logic applied to mathematical functions.

The main goal of this paper is to improve the Cuckoo Search algorithm by using fuzzy logic.

4 Benchmark Mathematical Functions

The test functions set that are used in the experiments includes different types of test functions in order to find out the success of an optimization algorithm in searching the global optimum. The literature does not include any optimization algorithm that can solve all test functions of different types. In the assessment of the success of an optimization algorithm, it is rather important to find out which types of test functions the optimization algorithm solves more successfully. While interpreting the test results, it is necessary to have knowledge about the general features and structures of the test functions used in the tests [4].

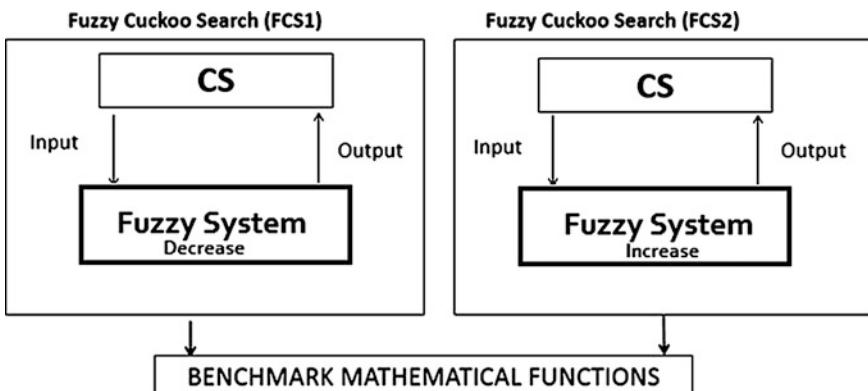


Fig. 3 Fuzzy system for the dynamic adaptation of the Pa parameter

Table 1 The benchmark functions used in tests

Function	Expression
Sphere	Sphere function is continuos, convex and unimodal $f(x) = \sum_{j=1}^{n_x} x_j^2 \quad (7)$ Whose global minimum $f(x^*) = 0$ occurs at $x^* = (0, 0, \dots, 0)$ Here n_x is the dimension
Rosenbrock	Rosenbrock function is a non-convex $f(x) = \sum_{j=1}^{n_x/2} [100(x_{2j} - x_{2j-1}^2)^2 + (1 - x_{2j-1})^2] \quad (8)$ With a global minimum $f(x^*) = 0$ at $x^* = (0, 0, \dots, 0)$ Where $j = 1, 2, d$
Rastrigin	Rastrigin function is based on Sphere function with the addition of cosine modulation to produce frequent local minima and highly multimodal $f(x) = 10n + \sum_{j=1}^{n_x} [x_j^2 - 10 \cos(2\pi x_j)] \quad (9)$ That has a unique global minimum $f(x^*) = 0$ occurs at $x^* = (0, 0, \dots, 0)$ Where $j = 1, 2, d$
Ackley	Ackley function is continuous, multimodal function obtained by modulating an exponential function with a cosine wave of moderate amplitude $f(x) = 20 + e - 20e^{-1/5} \sqrt{\frac{1}{n} \sum_{j=1}^n x_j^2} - e^{\frac{1}{n}} \sum_{j=1}^n \cos(2\pi x_j) \quad (10)$ With a global minimum $f(x^*) = 0$ at $x^* = (0, 0, \dots, 0)$ Where $j = 1, 2, d$
Griewank	Griewank function has many local minima $f(x) = \frac{1}{4000} + \sum_{n=1}^{n_x} x_n^2 - \prod_{n=1}^N \cos\left(\frac{x_n}{\sqrt{n}}\right) + 1 \quad (11)$ But a single global minimum $f(x^*) = 0$ at $x^* = (0, 0, \dots, 0)$ Where n_x is the dimension

To test the CS, FCS1 and FCS2 algorithms, in this paper we considered a 5 benchmark functions used to evaluate the performance of optimization algorithms were obtained, called Spherical, Rosenbrock, Ackley, Rastrigin and Griewank Functions.

The functions are evaluated with 8, 16, 32, 64 and 128 dimensions. The name of the functions, search range and formulations are shown in Table 1.

The search space ranges for the above benchmark functions considered in the experiments are listed in Table 2.

5 Fuzzy Cuckoo Search (FCS)

In Fig. 4 we can find the fuzzy system used for parameter adaptation of CS is Mamdani type, which has one input (Iteration) and one output (Pa), and has 3 fuzzy rules.

Table 2 Search space for each test functions

Function	Search space
Sphere	$-5 \leq x_i \leq 5$
Rosenbrock	$-5 \leq x_i \leq 10$
Rastrigin	$-5.12 \leq x_i \leq 5.12$
Ackley	$-15 \leq x_i \leq 30$
Griewank	$-600 \leq x_i \leq 600$

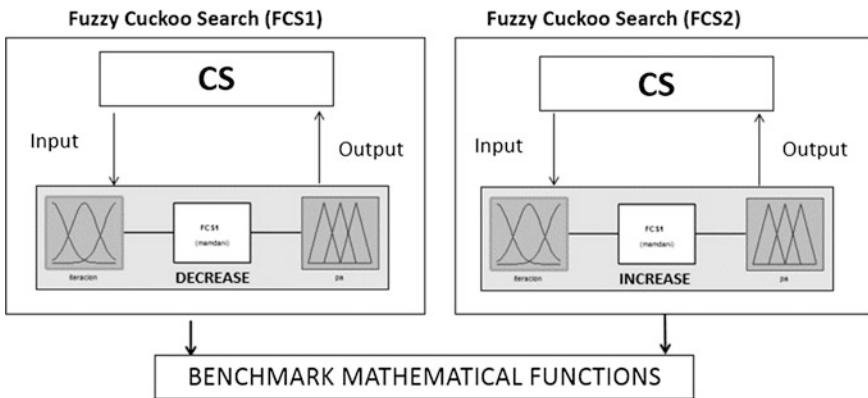


Fig. 4 Fuzzy Cuckoo search with dynamic adaptation of the Pa parameter

The difference between FCS1 and FCS2, are that the rules for FCS1 decrease and for FCS2 increase the output. Later between these two variants to be integrated in a new fuzzy system and improve the performance of the original algorithm CS.

Equation 12 is used to obtain the percentage of iterations, which is the input to the fuzzy system.

$$\text{Iteration} = \frac{\text{Current_Iteration}}{\text{Maximum_Iteration}} \quad (12)$$

Fig. 5 Input—iteration (percentage)

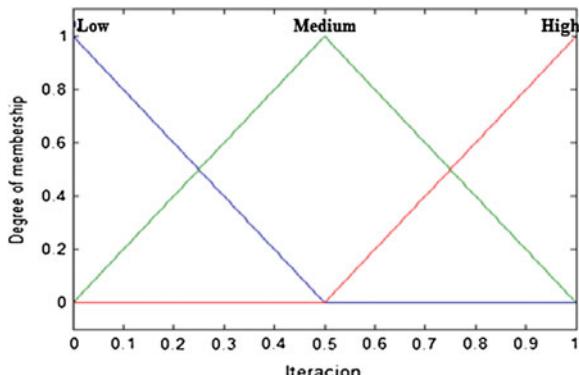
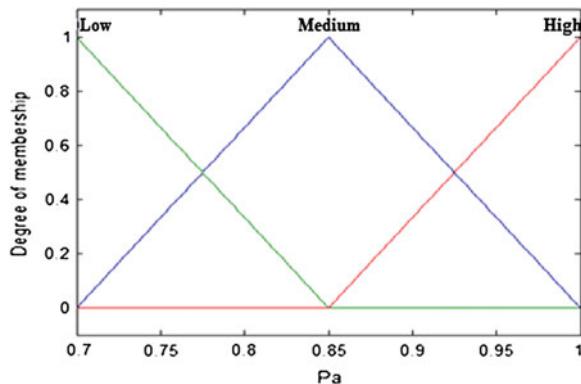


Fig. 6 Output—Pa**Fig. 7** Rules set for FCS1

1. If (Iteration is Low) the (Pa Is High)
2. If (Iteration is Medium) the (Pa Is Medium)
3. If (Iteration is High) the (Pa Is Low)

Fig. 8 Rules set for FCS2

1. If (Iteration is Low) the (Pa Is Low)
2. If (Iteration is Medium) the (Pa Is Medium)
3. If (Iteration is High) the (Pa Is High)

Figure 5 shows the input variable (*iteration*) of the fuzzy system that consists in the iterations and this is granulated into three triangular membership functions and they are: ‘Membership function: **Low** [−0.5 0 0.5]’, ‘Membership function: **Medium** [0 0.5 1]’ and ‘Membership function: **High** [0.5 1 1.5]’.

Figure 6 shows the output variable (*Pa*) of the fuzzy system, and is granulated into three triangular membership functions, which are ‘Membership function: **Low** [0.55 0.7 0.85]’, ‘Membership function: **Medium** [0.7 0.85 1]’, ‘Membership function: **High** [0.85 1 1.15].

The rules for the fuzzy system are shown in the next figures:

Figure 7 shows the fuzzy rules for the decreasing FCS1 fuzzy system and Fig. 8 for the decreasing FCS2 system.

Now in Fig. 9 we show the surfaces corresponding to these fuzzy systems.

6 Experiments and Methodology

The test of the CS, FCS1 and FCS2 all simulations are performed in a Windows 7 Ultimate Operating System with processor Intel Core 2 Duo of 64 bits that works to a frequency clock of 2.93 GHz, 4.00 GB of RAM Memory in Matlab 2009.

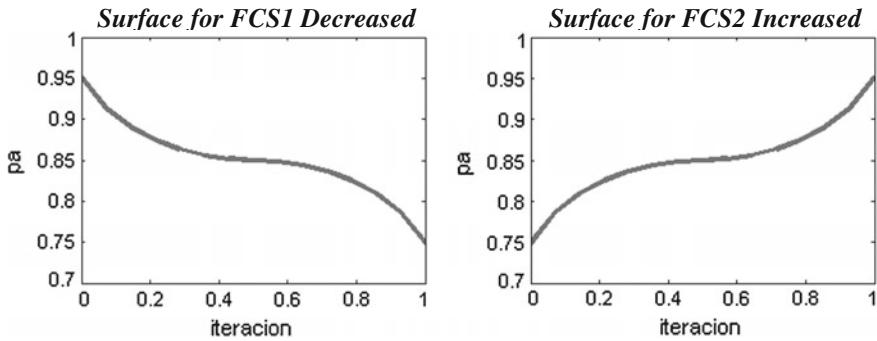


Fig. 9 Surfaces of the fuzzy system

Table 3 Parameters used in CS, FCS1 and FCS2

Parameters	CS	FCS1 AND FCS2
	Value	Value
Population size	100 Nests	100 Nests
Pa	75 %	Dynamic
α	5 %	5 %
β	1.7	1.7

In each of the algorithms, 5 Benchmark math functions are considered separately for 8, 16, 32, 64 and 128 Variables, and 50 tests are performed for each function varying the parameters of the algorithms.

Experiments with the Cuckoo Search algorithm varying the Pa variable (Probability of detection egg laying bird host, if discovered replaces nests with other) manually in a range from 0.1 to 0.9, found that the range for this variable is from 0.7 to 1, this is the reason why we implemented this range.

The parameters in the CS, FCS1 and FCS2 algorithms are as indicated in Table 3.

7 Simulation Results

In this Section we show tables with simulation results of the benchmark functions presented in Sect. 4. The Pa variable is constant in the simulation results with the CS algorithm and for FCS1 and FCS2 this parameter is dynamic.

In Sect. 7.1 we can find Tables 4, 5, 6, 7 and 8, where it can be appreciated that after executing the Cuckoo Search algorithm some good results are obtained. In the tables we can find the best value obtained, the average and the worst value of a set of 50 experiments for each of the dimensions.

Table 4 Simulation results with CS for the Sphere function

Variables	Best	Average	Worst
8	1.31902E-71	4.73265E-66	8.79146E-65
16	2.50846E-41	9.3512E-41	2.4244E-40
32	1.90197E-23	7.52953E-23	1.68211E-22
64	6.91106E-13	1.47449E-12	3.22594E-12
128	1.23772E-06	2.06675E-06	3.46609E-06

Table 5 Simulation results with CS for the Rosenbrock function

Variables	Best	Average	Worst
8	2.42696E-17	9.31316E-14	3.13512E-12
16	0.000563393	0.034374189	0.291397661
32	1.481264137	9.591505389	18.70216523
64	0.145111082	21.76065803	91.72342078
128	3.091420301	10.75405319	88.53554232

Table 6 Simulation results with CS for the Rastrigin function

Variables	Best	Average	Worst
8	0	8.2008E-12	1.37334E-10
16	7.968647975	11.10730214	14.67427655
32	49.61338869	66.91002157	88.02401868
64	181.04854	225.5827412	268.775593
128	516.7529253	662.3604611	783.9507225

- The parameters used in all the Tables 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17 and 18 are:
- Variables = number of dimensions used for the evaluation.
- Best = the best result obtained.
- Average = the average of 50 runs.
- Worst = the worst result obtained.

For Tables 4, 5, 6, 7 and 8 the algorithm used for the simulation is the original Cuckoo Search (CS), in Table 4 the results of the tests made with the Sphere function are presented, Table 5 shows the experimentation with Rosenbrock function, Table 6 shows simulation results with Rastrigin function, Table 7 with Ackley function and Table 8 experimentation with the Griewank function.

Table 7 Simulation results with CS for the Ackley function

Variables	Best	Average	Worst
8	8.88178E-16	4.22773E-15	4.44089E-15
16	2.57572E-14	1.749E-13	6.33271E-13
32	2.99693E-08	2.40971E-07	8.807E-07
64	0.00021828	0.001825476	0.014944774
128	0.067811815	0.655977252	1.858745834

Table 8 Simulation results with CS for the Griewank function

Variables	Best	Average	Worst
8	9.05115E-08	0.003605965	0.019682724
16	5.55112E-16	4.64462E-12	8.93237E-11
32	4.77396E-15	8.03426E-13	1.35695E-11
64	1.65851E-09	1.71573E-07	6.40696E-06
128	0.000219499	0.001196039	0.011598879

Table 9 Simulation results with decreasing FCS1 for the sphere function

Variables	Best	Average	Worst
8	7.43874E-73	5.59832E-68	1.94879E-66
16	4.62211E-41	2.22735E-40	8.10187E-40
32	1.49426E-23	6.07923E-23	1.41971E-22
64	7.31806E-13	1.77095E-12	3.2731E-12
128	1.09378E-06	2.12994E-06	4.27794E-06

Table 10 Simulation results with decreasing FCS1 for the Rosenbrock function

Variables	Best	Average	Worst
8	1.3614E-28	4.7036E-23	6.18954E-22
16	1.94639E-10	2.39111E-05	0.00109902
32	0.099773361	6.639269804	16.81172623
64	0.002184228	13.47391968	70.86085269
128	0.840172782	4.919593537	76.50378468

In Tables 9, 10, 11, 12 and 13 we follow the order of functions for the simulations with the Decreased Fuzzy Cuckoo Search Algorithm (FCS1) and for Tables 14, 15, 16, 17 and 18 the Increased Fuzzy Cuckoo Search Algorithm (FCS2).

The parameters used in all the Tables 19, 20, 21, 22 and 23 are:

Variables = number of dimensions used for evaluation.

CS = average for variables with the Cuckoo Search algorithm.

FCS1 decreased = average for variables with Decreased Fuzzy Cuckoo Search algorithm 1 with dynamic adaptation of Pa parameter.

Table 11 Simulation results with decreasing FCS1 for the Rastrigin function

Variables	Best	Average	Worst
8	4.2661E-11	1.98715E-06	5.16328E-05
16	7.548578173	11.63094671	15.56167089
32	33.24824105	59.14536234	79.60639448
64	129.3906971	174.6495021	228.6177993
128	392.0567959	531.7387371	692.0332701

Table 12 Simulation results with decreasing FCS1 for the Ackley function

Variables	Best	Average	Worst
8	8.88178E-16	3.73035E-15	4.44089E-15
16	7.99361E-15	7.47136E-14	4.66294E-13
32	3.8569E-09	1.24463E-07	1.39574E-06
64	9.21072E-05	0.001776161	0.024914329
128	0.030126098	0.514817218	1.955629155

Table 13 Simulation results with decreasing FCS1 for the Griewank function

Variables	Best	Average	Worst
8	9.39783E-07	0.013642345	0.03095626
16	0	6.05565E-12	1.4038E-10
32	2.22045E-16	1.70222E-13	3.36675E-12
64	2.68192E-09	1.52366E-07	3.12597E-06
128	4.98597E-05	0.000133177	0.00063412

Table 14 Simulation results with increasing FCS2 for the Sphere function

Variables	Best	Average	Worst
8	1.91939E-72	8.34777E-68	1.49149E-66
16	4.12858E-41	1.87502E-41	4.39276E-40
32	1.17756E-23	6.13849E-23	1.58306E-22
64	8.48291E-13	2.32155E-12	5.6095E-12
128	9.56958E-07	2.57884E-06	4.61942E-06

Table 15 Simulation results with increasing FCS2 for the Rosenbock Function

Variables	Best	Average	Worst
8	1.38458E-26	4.80399E-22	1.01228E-20
16	1.20387E-09	7.45285E-06	0.000134493
32	0.348927885	6.347579711	14.40746543
64	0.005996436	25.52521646	94.59883029
128	0.650072341	6.238852453	74.70529809

Table 16 Simulation results with increasing FCS2 for the Rastrigin function

Variables	Best	Average	Worst
8	4.26326E-14	7.21111E-07	1.38031E-05
16	7.498119014	11.77580857	17.84198191
32	47.01446005	63.86497875	92.50291136
64	155.179641	212.5163085	278.4747834
128	510.4692494	626.1177477	798.1824519

Table 17 Simulation results with increasing FCS2 for the Ackley function

Variables	Best	Average	Worst
8	8.88178E-16	3.44613E-15	4.44089E-15
16	7.99361E-15	1.12159E-13	4.80505E-13
32	6.04169E-09	1.05549E-07	1.17244E-06
64	0.000109998	0.001244918	0.013244132
128	0.035146143	0.482092683	2.198810171

Table 18 Simulation results with increasing FCS2 for the Griewank function

Variables	Best	Average	Worst
8	0.000684708	0.023675577	0.046122053
16	0	1.38731E-12	2.16194E-11
32	1.88738E-15	1.85547E-13	1.77225E-12
64	2.60521E-09	1.05732E-07	1.22177E-06
128	0.000296531	0.00219423	0.014511072

FCS2 increased = average for variables with Increased Fuzzy Cuckoo Search algorithm 2 with dynamic adaptation of Pa parameter.

Best Algorithm = the best algorithm for variable.

Table 19 shows a comparison for the Sphere function, the Pa variable for CS is manually defined and it remains constant throughout the execution of the algorithm, and FCS1 the value of Pa is decreased dynamically and FCS2 the value Pa is increased dynamically.

In Table 20 we can find a comparison for the Rosenbrock function, the variable Pa for CS is constant and in FCS1 the value of Pa is decreased dynamically and FCS2 the value Pa is increased dynamically.

Table 21 shows a comparison for the Rastrigin function, the Pa variable for CS is constant and in FCS1 the value of Pa is decreased dynamically and FCS2 the value Pa is increased dynamically.

Table 22 shows a comparison for the Ackley function, the Pa variable for CS is constant and in FCS1 the value of Pa is decreased dynamically and FCS2 the value Pa is increased dynamically.

Table 19 Comparison with CS, FCS1 and 2 for the Sphere function

Variables	CS	FCS1 decreased	FCS2 increased	Best algorithm
8	4.73E-66	5.60E-68	8.35E-68	FCS1 DEC
16	9.35E-41	2.23E-40	1.88E-40	FCS2 INC
32	7.53E-23	6.08E-23	6.14E-23	FCS1 DEC
64	1.47E-12	1.77E-12	2.32E-12	FCS1 DEC
128	2.07E-06	2.13E-06	2.58E-06	FCS1 DEC

Table 20 Comparison with CS, FCS1 and FCS2 for the Rosenbrock function

Variables	CS	FCS1 decreased	FCS2 increased	Best algorithm
8	9.31E-14	4.70E-23	4.80E-22	FCS1 DEC
16	0.034374189	2.39E-05	7.45E-06	FCS2 INC
32	9.591505389	6.639269804	6.347579711	FCS2 INC
64	21.76065803	13.47391968	25.52521646	FCS1 DEC
128	10.75405319	4.919593537	6.238852453	FCS1 DEC

Table 21 Comparison with CS, FCS1 and 2 for the Rastrigin function

Variables	CS	FCS1 decreased	FCS2 increased	Best algorithm
8	8.20E-12	1.99E-06	7.21E-07	CS
16	11.10730214	11.63094671	11.77580857	CS
32	66.91002157	59.14536234	63.86497875	FCS1 DEC
64	225.5827412	174.6495021	212.5163085	FCS1 DEC
128	662.3604611	531.7387371	626.1177477	FCS1 DEC

Table 22 Comparison with CS, FCS1 and FCS2 for the Ackley function

Variables	CS	FCS1 decreased	FCS2 increased	Best algorithm
8	4.23E-15	3.73E-15	3.45E-15	FCS2 INC
16	1.75E-13	7.47E-14	1.12E-13	FCS1 DEC
32	2.41E-07	1.24E-07	1.06E-07	FCS2 INC
64	0.001825476	0.001776161	0.001244918	FCS2 INC
128	0.655977252	0.514817218	0.482092683	FCS2 INC

Table 23 Comparison with CS, FCS1 and 2 for the Griewank function

Variables	CS	FCS1 decreased	FCS2 increased	Best algorithm
8	0.003605965	0.021144943	0.023675577	CS
16	4.64E-12	6.06E-12	1.39E-12	FCS2 INC
32	8.03E-13	1.70E-13	1.86E-13	FCS1 DEC
64	1.72E-07	1.52E-07	1.06E-07	FCS2 INC
128	0.001196039	0.001344208	0.00219423	CS

7.1 Experimental Results with CS

See Tables 4, 5, 6, 7 and 8.

7.2 *Simulation Results with Decreasing FCS1*

See Tables 9, 10, 11, 12 and 13.

7.3 *Simulation Results with Increasing FCS2*

See Tables 14, 15, 16, 17 and 18.

7.4 *Comparison Results with CS, FCS1 and FCS2*

We have also compared the results of FCS1 and FCS2 with the original CS Algorithm.

8 Conclusions

We can conclude that both the Fuzzy Cuckoo Search variants (decreasing and increasing) are better than the original Cuckoo Search Algorithm as are demonstrated with the tables mentioned in the previously sections to obtain.

The CS is relatively young algorithm, which has been applied to different areas, and many other researchers continue to propose new variants and our methods help the algorithm to improve the results.

The study of manual variation of the Pa parameter in previous work helped us to implement the output range for the proposed fuzzy system, finding in Tables 19, 20, 21, 22, and 23 that the proposed algorithm is better than the original CS algorithm.

Future works include continuing with the other parameters to further improve the results. We will also consider merging into a single algorithm, the different dynamic parameters to provide the better results.

Acknowledgment We thank CONACYT and Tijuana Institute of Technology for the facilities and resources granted for the development of this research.

References

1. Bacanin, N.: An object-oriented software implementation of a novel cuckoo search algorithm. In Proceedings of the 5th European Conference on European Computing Conference (ECC11), pp. 245–250 (2011)
2. Bonabeau, E., Dorigo, M., Theraulaz, G.: Swarm Intelligence: From Natural to Artificial Systems. Oxford University Press (1999)

3. Castillo, O., Neyoy, H., Soria, J., García, M., Valdez, F.: Dynamic fuzzy logic parameter tuning for ACO and its application in the fuzzy logic control of an autonomous mobile robot. *Int. J. Adv. Rob. Syst.* **10** (2013)
4. Ceviciglu P., Besdok, E.: Comparative Analysis of the Cuckoo Search Algorithm. In *Cuckoo Search and Firefly Algorithm*. Springer International Publishing, pp. 85–113 (2014)
5. Das, S., Dasgupta, P., Panigrahi, B.K.: Inter-species Cuckoo Search via Different Levy Flights. In *Swarm, Evolutionary, and Memetic Computing*, Springer International Publishing, pp. 515–526 (2013)
6. Jang, J., Sun, C., Mizutani, E.: *Neuro-fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence*. Prentice-Hall, Upper Saddle River (1997)
7. Manikandan, P., Selvarajan, S.: Data clustering using cuckoo search algorithm (CSA). In: *Proceedings of the 2nd International Conference on Soft Computing for Problem Solving (SocProS 2012)*, Dec 28–30, 2012, pp. 1275–1283. Springer, India (2014)
8. Melin, P., Olivas, F., Castillo, O., Valdez, F., Soria, J., Valdez, M.: Optimal design of fuzzy classification systems using PSO with dynamic parameter adaptation through fuzzy logic. *Expert Syst. Appl.* **40**(8), 3196–3206 (2013)
9. Patwardhan, A.P., Rohan P., Nithin V.G.: On a cuckoo search optimization approach towards feedback system identification. *Digital Signal Process.* **32**, 156–163 (2014)
10. Pavlyukevich, I.: Cooling down Lévy flights. *J. Phys. A Math. Theor.* **40**, 12299–12313 (2007)
11. Rajabioun, R.: Cuckoo optimization algorithm. *Appl. Soft Comput.* **11**(8), 5508–5518 (2011)
12. Reynolds, A.M., Frye, M.A.: Free-flight odor tracking in *Drosophila* is consistent with an optimal intermittent scale-free search. *PLoS ONE* **2**, e354 (2007)
13. Shlesinger, M.F., Zaslavsky, G.M., Frisch, U. (eds.): *Lévy Flights and Related Topics in Physics*. Springer, Berlin (1995)
14. Sombra, A., Valdez, F., Melin, P., Castillo, O.: A new gravitational search algorithm using fuzzy logic to parameter adaptation. In: *Evolutionary Computation (CEC)*, IEEE Congress on, pp. 1068–1074 (2013)
15. Valdez, F., Melin, P., Castillo, O.: A survey on nature-inspired optimization algorithms with fuzzy logic for dynamic parameter adaptation. *Expert Syst. Appl.* **41**(14), 6459–6466 (2014)
16. Walton, S., Hassan, O., Morgan, K., Brown, M.R.: Modified cuckoo search: a new gradient free optimization algorithm. *Chaos Solitons Fractals* **44**(9), 710–718 (2011)
17. Yang, X.S.: *Cuckoo Search and Firefly Algorithm*. Springer Press, Berlin (2014)
18. Yang, X.S.: *Nature-inspired Metaheuristic Algorithms*. Luniver Press (2010)
19. Zadeh, L.A.: Fuzzy sets. *Info. Control* **8**, 338–353 (1965)

Differential Evolution with Dynamic Adaptation of Parameters for the Optimization of Fuzzy Controllers

Patricia Ochoa, Oscar Castillo and José Soria

Abstract The proposal described in this paper uses the Differential Evolution (DE) algorithm as an optimization method in which we want to dynamically adapt its parameters using fuzzy logic control systems, with the goal that the fuzzy system calculates the optimal parameter values of the DE algorithm to find better results, depending on the type of problems the DE is applied. In this case we consider a fuzzy system to dynamically change the F and CR variables.

1 Introduction

The use of fuzzy logic in evolutionary computing is becoming a common approach to improve the performance of the algorithms [15–17]. Currently the parameters involved in the algorithms are determined by trial and error. In this aspect we propose the application of fuzzy logic, which is responsible in performing the dynamic adjustment of mutation and crossover parameters in the Differential Evolution (DE) algorithm. This has the goal of providing better performance to Differential Evolution.

Fuzzy logic or multi-valued logic is based on the fuzzy set theory proposed by Zadeh in 1965 which helps us in modeling knowledge, through the use of if-then fuzzy rules. The fuzzy set theory provides a systematic calculus to deal with linguistic information, and that improves the numerical computation by using linguistic labels stipulated by membership functions [12]. Differential Evolution (DE) is one of the latest evolutionary algorithms that have been proposed. It was created in 1994 by Price and Storn in an attempt to solve the problem of Chebychev polynomial. The following year these two authors proposed the DE for optimization of nonlinear and non-differentiable functions on continuous spaces.

P. Ochoa · O. Castillo (✉) · J. Soria
Tijuana Institute of Technology, Tijuana, Mexico
e-mail: ocastillo@tectijuana.mx

The DE algorithm is a stochastic method of direct search, which has proven to be effective, efficient and robust in a wide variety of applications such as learning of a neural network, a filter design of IIR, aerodynamically optimized. The DE has a number of important features which make it attractive for solving global optimization problems, among them are the following: it has the ability to handle non-differentiable, nonlinear and multimodal objective functions, usually converges to the optimal uses with few control parameters, etc.

The DE belongs to the class of evolutionary algorithms that is based on populations. It uses two evolutionary mechanisms for the generation of descendants: mutation and crossover; finally a replacement mechanism, which is applied between the father vector and son vector determining who survive into the next generation. There exist works, where they currently use fuzzy logic to optimize the performance of metaheuristic algorithms, to name a few articles such as:

Optimization of Membership Functions for Type-1 and Type 2 Fuzzy Controllers of an Autonomous Mobile Robot Using PSO [1], Optimization of a Fuzzy Tracking Controller for an Autonomous Mobile Robot under Perturbed Torques by Means of a Chemical Optimization Paradigm [2], Design of Fuzzy Control Systems with Different PSO Variants [4], A Method to Solve the Traveling Salesman Problem Using Ant Colony Optimization Variants with Ant Set Partitioning [6], Evolutionary Optimization of the Fuzzy Integrator in a Navigation System for a Mobile Robot [7], Optimal design of fuzzy classification systems using PSO with dynamic parameter adaptation through fuzzy logic [8], Dynamic Fuzzy Logic Parameter Tuning for ACO and Its Application in TSP Problems [10], Bio-inspired Optimization Methods on Graphic Processing Unit for Minimization of Complex Mathematical Functions [18].

Similarly there are papers on Differential Evolution (DE) applications that use this algorithm to solve real problems. To mention a few: A fuzzy logic control using a differential evolution algorithm aimed at modelling the financial market dynamics [5], Design of optimized cascade fuzzy controller based on differential evolution: Simulation studies and practical insights [11], Eliciting transparent fuzzy model using differential evolution [3], Assessment of human operator functional state using a novel differential evolution based adaptive fuzzy model [14, 20].

This paper is organized as follows: Sect. 2 shows the concept of the Differential Evolution algorithm. Section 3 describes the proposed methods. Section 4 the Benchmark Functions, Sect. 5 Experiments and Methodology and Sect. 6 the Conclusions.

2 Differential Evolution

The Differential Evolution (DE) is an optimization method belonging to the category of evolutionary computation applied in solving complex optimization problems.

The DE is composed of 4 steps:

- Initialization.
- Mutation.
- Crossover.
- Selection.

This is a non-deterministic technique based on the evolution of a vector population (individuals) of real values representing the solutions in the search space. The generation of new individuals is carried out by differential crossover and mutation operators [13].

The operation of the algorithm is explained below.

2.1 Population Structure

The differential evolution algorithm maintains a pair of vector populations, both of which contain N_p D-dimensional vectors of real-valued parameters [8].

$$P_{x,g} = (\mathbf{x}_{i,g}), \quad i = 0, 1, \dots, N_p, \quad g = 0, 1, \dots, g_{\max} \quad (1)$$

$$\mathbf{x}_{i,g} = (x_{j,i,g}), \quad j = 0, 1, \dots, D - 1 \quad (2)$$

where:

P_x current population.

g_{\max} maximum number of iterations.

i index population.

j parameters within the vector.

Once the vectors are initialized, three individuals are selected randomly to produce an intermediate population, $P_{v,g}$, of N_p mutant vectors, $v_{i,g}$.

$$P_{v,g} = (v_{i,g}), \quad i = 0, 1, \dots, N_p - 1, \quad g = 0, 1, \dots, g_{\max} \quad (3)$$

$$v_{i,g} = (v_{j,i,g}), \quad j = 0, 1, \dots, D - 1 \quad (4)$$

Each vector in the current population are recombined with a mutant vector to produce a trial population, P_u , the NP , mutant vector $u_{i,g}$:

$$P_{v,g} = (u_{i,g}), \quad i = 0, 1, \dots, N_p - 1, \quad g = 0, 1, \dots, g_{\max} \quad (5)$$

$$u_{i,g} = (u_{j,i,g}), \quad j = 0, 1, \dots, D - 1 \quad (6)$$

2.2 Initialization

Before initializing the population, the upper and lower limits for each parameter must be specified. These 2D values can be collected by two initialized vectors, D-dimensional, b_L and b_U , to which subscripts L and U indicate the lower and upper limits respectively. Once the initialization limits have been specified a number generator randomly assigns each parameter in every vector a value within the set range. For example, the initial value ($g = 0$) of the j th vector parameter is i th:

$$x_{j,i,0} = \text{rand}_j(0, 1) \cdot (b_{j,U} - b_{j,L}) + b_{j,L} \quad (7)$$

2.3 Mutation

In particular, the differential mutation uses a random sample equation showing how to combine three different vectors chosen randomly to create a mutant vector.

$$v_{i,g} = x_{r0,g} + F \cdot (x_{r1,g} - x_{r2,g}) \quad (8)$$

The scale factor, $F \in (0, 1)$ is a positive real number that controls the rate at which the population evolves. While there is no upper limit on F , the values are rarely greater than 1.0.

2.4 Crossover

To complement the differential mutation search strategy, DE also uses uniform crossover, sometimes known as discrete recombination (dual). In particular, DE crosses each vector with a mutant vector:

$$U_{i,g} = (u_{j,i,g}) = \begin{cases} v_{j,i,g} & \text{if } (\text{rand}_j(0, 1) \leq Cr \text{ or } j = j_{rand}) \\ x_{j,i,g} & \text{otherwise.} \end{cases} \quad (9)$$

2.5 Selection

If the test vector, $U_{i,g}$ has a value of the objective function equal to or less than its target vector, $X_{i,g}$, it replaces the target vector in the next generation; otherwise, the target retains its place in population for at least another generation [2]

$$X_{i,g+1} = \begin{cases} U_{i,g} & \text{if } f(U_{i,g}) \leq f(X_{i,g}) \\ X_{i,g} & \text{otherwise.} \end{cases} \quad (10)$$

The process of mutation, recombination and selection are repeated until the optimum is found, or terminating pre criteria specified is satisfied. DE is a simple, but powerful search engine that simulates natural evolution combined with a mechanism to generate multiple search directions based on the distribution of solutions in the current population. Each vector i in the population at generation G , x_i, G , called at this moment of reproduction as the target vector will be able to generate one offspring, called trial vector (u_i, G). This trial vector is generated as follows: First of all, a search direction is defined by calculating the difference between a pair of vectors r_1 and r_2 , called “*differential vectors*”, both of them chosen at random from the population. This difference vector is also scaled by using a user defined parameter called “ $F \geq 0$ ”. This scaled difference vector is then added

```

Begin
    G=0
    Create a random initial population  $\mathbf{x}_{i,G} \forall i, i = 1, \dots, NP$ 
    Evaluate  $f(\mathbf{x}_{i,G}) \forall i, i = 1, \dots, NP$ 
    For G=1 to MAX_GEN Do
        For i=1 to NP Do
            Select randomly  $r_1 \neq r_2 \neq r_3$  :
             $j_{rand} = \text{randint}(1,D)$ 
            For j=1 to n Do
                If  $(rand_j[0,1] < CR \text{ or } j = j_{rand})$  Then
                     $u_{i,j,G+1} = \mathbf{x}_{r_3,j,G} + F(\mathbf{x}_{r_1,j,G} - \mathbf{x}_{r_2,j,G})$ 
                Else
                     $u_{i,j,G+1} = \mathbf{x}_{i,j,G}$ 
                End If
            End For
            If  $(f(u_{i,G+1}) \leq f(\mathbf{x}_{i,G}))$  Then
                 $\mathbf{x}_{i,G+1} = u_{i,G+1}$ 
            Else
                 $\mathbf{x}_{i,G+1} = \mathbf{x}_{i,G}$ 
            End If
        End For
        G = G + 1
    End For
End
```

Fig. 1 “DE/rand/1/bin” pseudocoder and $[0, 1]$ is a function that returns a real number between 0 and 1. Randint(min, max) is a function that returns an integer number between min and max. NP , $MAX\ GEN$, CR and F are user-defined parameters n is the dimensionality of the problem [9]

to a third vector $r3$, called “*base vector*”. As a result, a new vector is obtained, known as the mutation vector. After that, this mutation vector is recombined with the target vector (also called parent vector) by using discrete recombination (usually binomial crossover) controlled by a crossover parameter $0 \leq CR \leq 1$ whose value determines how similar the trial vector will be with respect to the target vector. There are several DE variants. However, the most known and used is DE/rand/1/bin, where the base vector is chosen at random, there is only a pair of differential vectors and a binomial crossover is used. The detailed pseudocode of this variant is presented in Fig. 1 [9].

3 Proposed Method

The Differential Evolution (DE) Algorithm is a powerful search technique used for solving optimization problems. In this paper a new algorithm called Fuzzy Differential Evolution (FDE) with dynamic adjustment of parameters for the optimization of controllers is proposed. The main objective is that the fuzzy system will provide us with the optimal parameters for the best performance of the DE algorithm. In addition the parameters that the fuzzy system optimizes are the crossover and mutation, as shown in Fig. 2.

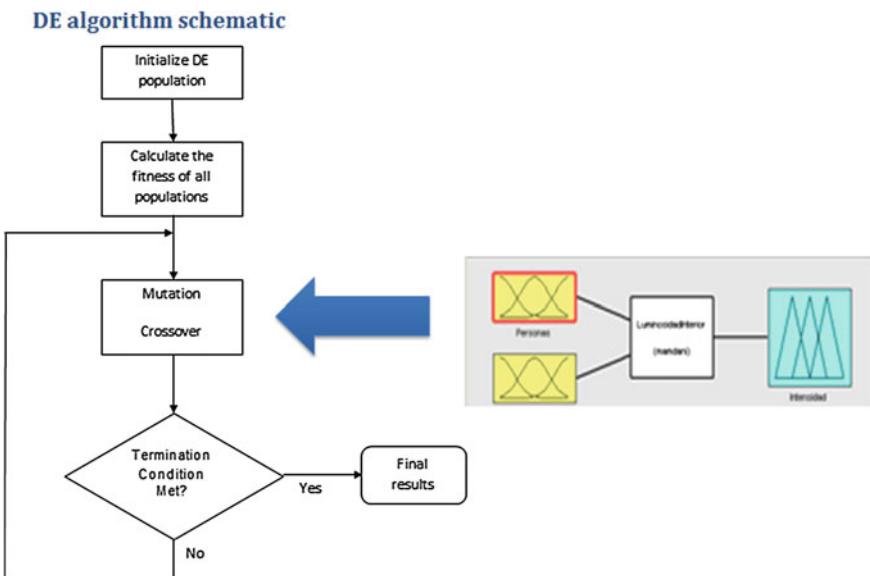


Fig. 2 The proposed differential evolution (DE) algorithm by integrating a fuzzy system to dynamically adapt parameters

4 Benchmark Function

In this paper we consider 6 Benchmark functions, which are briefly explained below [19].

- **Sphere Function**

Number of variables: n variables.

Definition:

$$f(x) = \sum_{i=1}^n x_i^2. \quad (11)$$

Search domain: $-5.12 \leq x_i \leq 5.12, i = 1, 2, \dots, n$.

Number of local minima: no local minimum except the global one.

The global minima: $x^* = (0, \dots, 0), f(x^*) = 0$

Function graph: for $n = 2$ presented in Fig. 3

- **Griewank Function**

Number of variables: n variables.

Definition:

$$f(x) = \frac{1}{4000} \sum_{i=1}^n x_i^2 - \prod_{i=1}^n \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1. \quad (12)$$

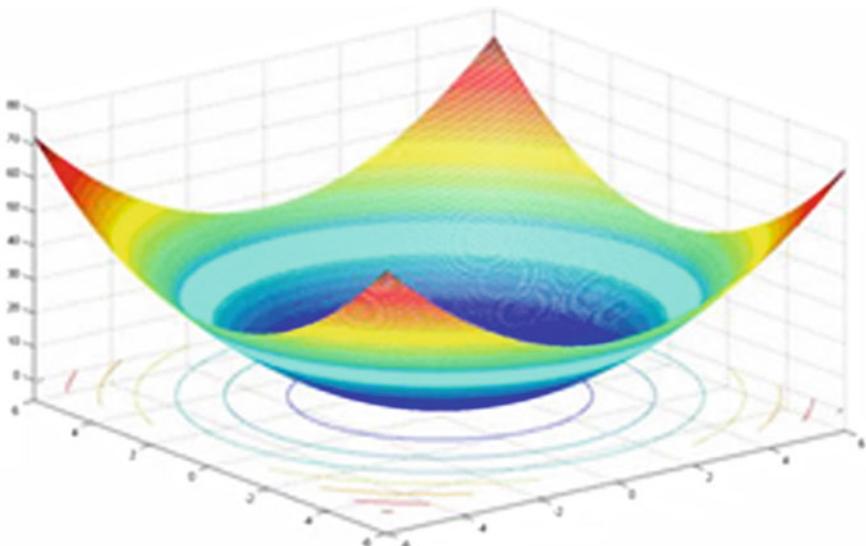


Fig. 3 Sphere function

Search domain: $-600 \leq x_i \leq 600, i = 1, 2, \dots, n$.

Number of local minima: no local minimum except the global one.

The global minima: $x^* = (0, \dots, 0), f(x^*) = 0$

Function graph: for $n = 2$ presented in Fig. 4

• Schwefel Function

Number of variables: n variables.

Definition:

$$f(x) = \sum_{i=1}^n \left[-x_i \sin(\sqrt{|x_i|}) \right]. \quad (13)$$

Search domain: $-500 \leq x_i \leq 500, i = 1, 2, \dots, n$.

Number of local minima: no local minimum except the global one.

The global minima: $x^* = (0, \dots, 0), f(x^*) = 0$

Function graph: for $n = 2$ presented in Fig. 5

• Rastrigin Function

Number of variables: n variables.

Definition:

$$f(x) = 10n + \sum_{i=1}^n [x_i^2 - 10 \cos(2\pi x_i)]. \quad (14)$$

Search domain: $-5.12 \leq x_i \leq 5.12, i = 1, 2, \dots, n$.

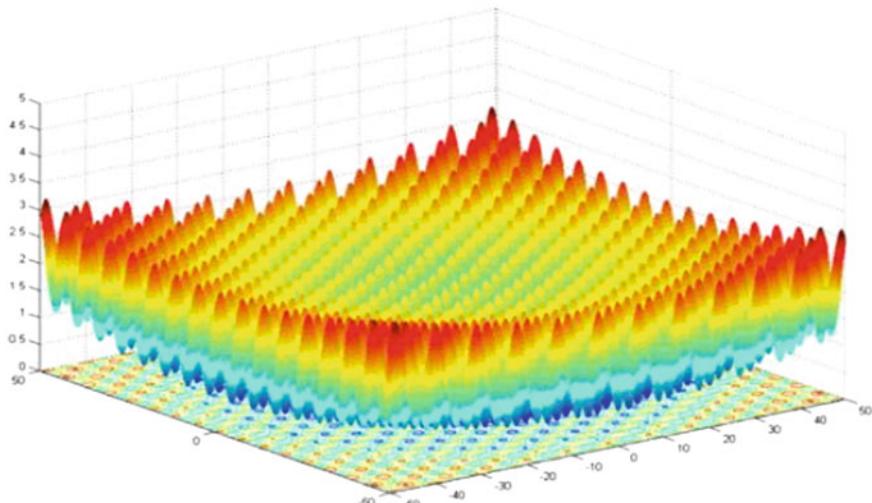


Fig. 4 Griewank function

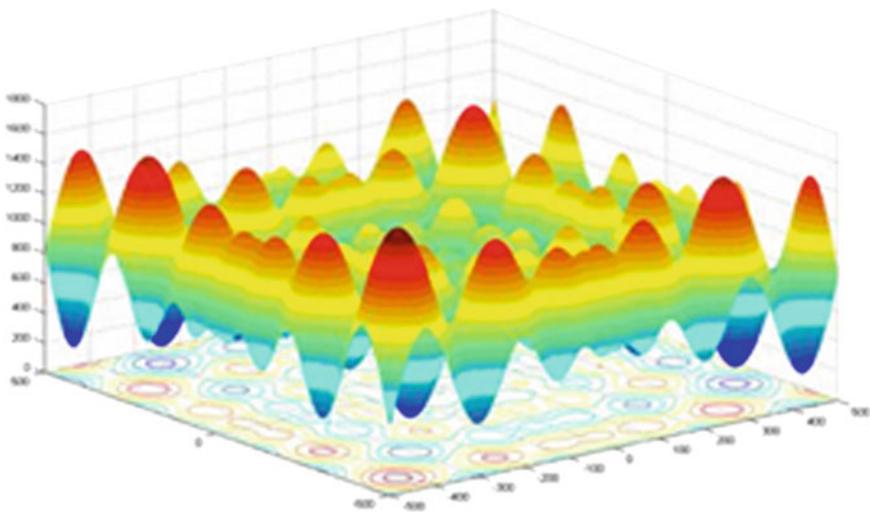


Fig. 5 Schwefel function

Number of local minima: no local minimum except the global one.

The global minima: $x^* = (0, \dots, 0)$, $f(x^*) = 0$

Function graph: for $n = 2$ presented in Fig. 6

- **Ackley Funcion**

Number of variables: n variables.

Definition:

$$f(x) = -a \cdot \exp\left(-b \cdot \sqrt{\frac{1}{n} \sum_{i=1}^n x_i^2}\right) - \exp\left(\frac{1}{n} \sum_{i=1}^n \cos(cx_i)\right) + a + \exp(1) \quad (15)$$

Search domain: $-15 \leq x_i \leq 30$, $i = 1, 2, \dots, n$.

Number of local minima: no local minimum except the global one.

The global minima: $x^* = (0, \dots, 0)$, $f(x^*) = 0$

Function graph: for $n = 2$ presented in Fig. 7

- **Rosenbrock Funcion**

Number of variables: n variables.

Definition:

$$f(x) = \sum_{i=1}^{n-1} \left[100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2 \right]. \quad (16)$$

Search domain: $-5 \leq x_i \leq 10$, $i = 1, 2, \dots, n$.

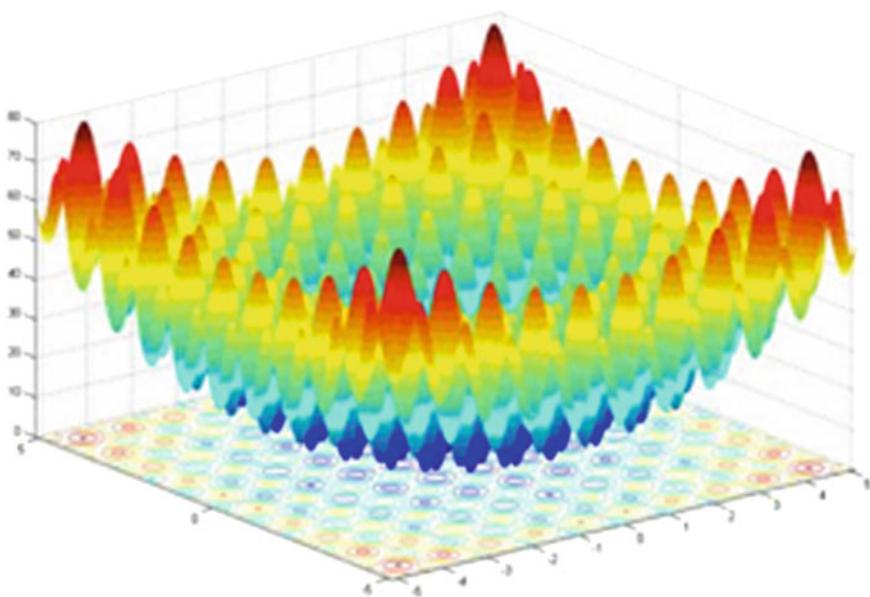


Fig. 6 Rastrigin function

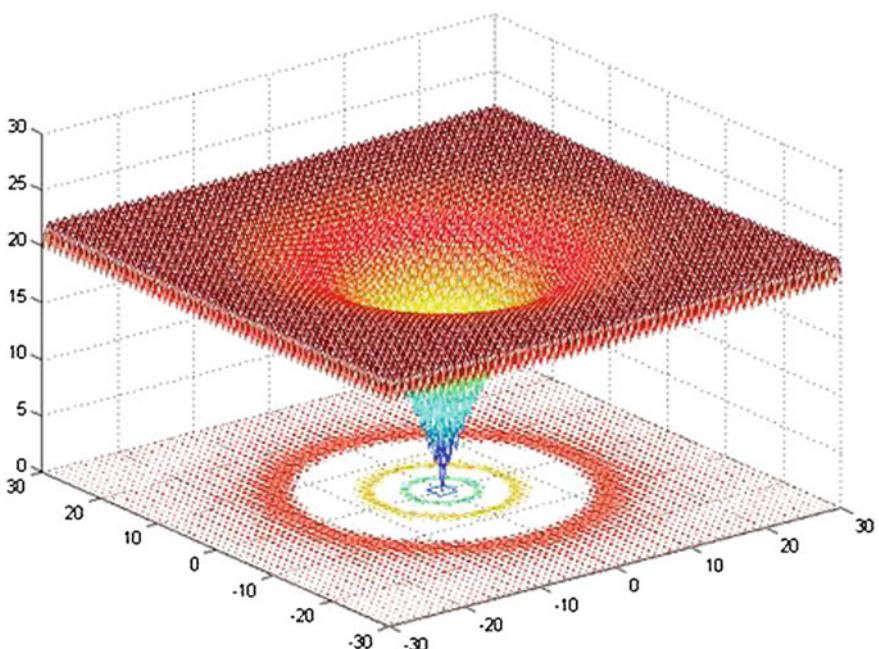


Fig. 7 Ackley function

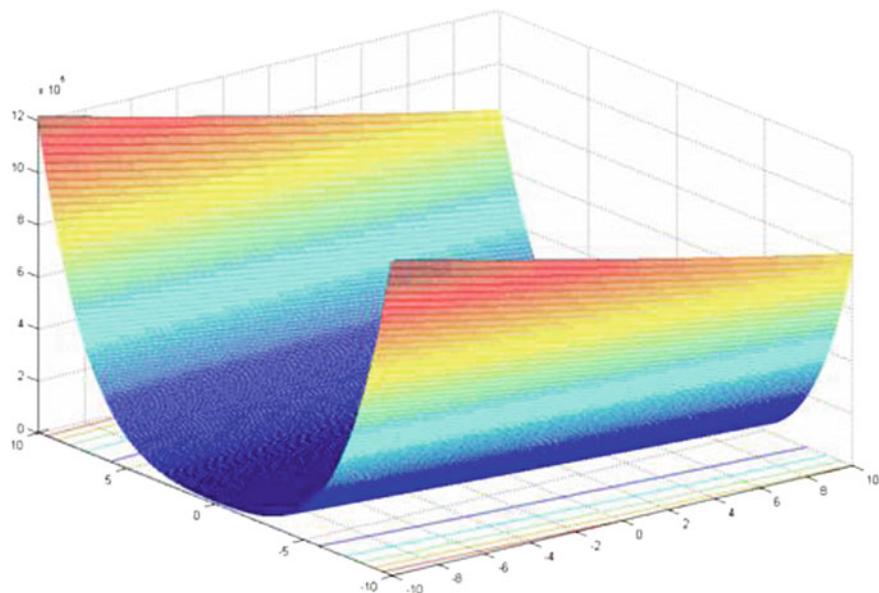


Fig. 8 Rosenbrock function

Number of local minima: no local minimum except the global one.

The global minima: $x^* = (0, \dots, 0)$, $f(x^*) = 0$

Function graph: for $n = 2$ presented in Fig. 8.

5 Experiments and Methodology

Experiments with the Differential Evolution algorithm by manually varying the F value (variable mutation) manually in a range from 0.1 to 0.9, were performed with several generations values (GR) 100, 500, 1000, 2000, 3000, 4000 and 5000. Population variables are held constant (NP) = 250, dimensions (D) = 50, crossover (CR) = 0.1, upper limit (H) = 500 and lower (L) = -500.

To make a comparison with the original differential evolution algorithm the averages of the experiments are obtained for each generation number. Experiments are performed by varying F from 0.1 to 0.9 and 30 experiments for each F. then an overall average is obtained for comparison with the Fuzzy Differential Evolution with increase and decrease of the F value.

For experiments with the Fuzzy Differential Evolution algorithm F changes dynamically increasing and decreasing F between 0 and 1, 30 experiments for each number of generations are performed and an average is obtained by generation.

For the Ackley and Rosenbrock the functions search space, since the are we were using was very large for the Ackley function, the upper limit (H) = 32.768 and

the lower limit (L) = -32.768, for the Rosenbrock function the upper limit (H) = 2.048 and the lower limit (L) = -2.048.

This paper mentions two fuzzy systems with which the experiments were performed. It has a fuzzy system which increases the F variable and another variable decreases F .

Then the fuzzy system, in which F is increased dynamically.

- Contains one input and one output
- Is of Mamdani type.
- All membership functions are triangular.
- The input of the fuzzy system is defined by the generations and is granulated into three membership functions they are: MF_1 = ‘Low’ [-0.5 0 0.5], MF_2 = ‘Medium’ [0 0.5 1], MF_3 = ‘High’ [0.5 1 1.5].
- The output of the fuzzy system and the variable F is granulated in three membership functions which are: MF_1 = ‘Low’ [-0.5 0 0.5], MF_2 = ‘Medium’ [0 0.5 1], MF_3 = ‘High’ [0.5 1 1.5].
- The fuzzy system uses 3 rules and what it does is increased the value of the F variable in a range of (0.1).

The fuzzy rules are shown in Fig. 9.

Then the fuzzy system, in which F is dynamically decreased is described as follows:

- Contains one input and one output
- Is Mamdani type.
- All functions are triangular.
- The input of the fuzzy system is generations and divided into three membership functions they are: MF_1 = ‘Low’ [-0.5 0 0.5], MF_2 = ‘Medium’ [0 0.5 1], MF_3 = ‘High’ [0.5 1 1.5].
- The output of the fuzzy system and the variable F is divided in three membership functions which are: MF_1 = ‘Low’ [-0.5 0 0.5], MF_2 = ‘Medium’ [0 0.5 1], MF_3 = ‘High’ [0.5 1 1.5].
- The fuzzy system uses 3 rules and what it does is decreased the value of the F variable in a range of (0.1).

The fuzzy rules are shown in Fig. 10.

In this paper we show six tables to compare the results of the Benchmark functions mentioned above, where the variable F is modified manually in the original Difference Evolution algorithm and F is increased and described dynamically with the proposed Fuzzy Differential Evolution algorithm.

- | |
|--|
| 1. - If (Generations is Low) then (F is Low) (1)
2. - If (Generations is Medium) then (F is Medium) (1)
3. - If (Generations is High) then (F is High) (1) |
|--|

Fig. 9 Rules of the fuzzy system

- | |
|--|
| 1. - If (Generations is Low) then (F is High) (1) |
| 2. - If (Generations is Medium) then (F is Medium) (1) |
| 3. - If (Generations is High) then (F is Low) (1) |

Fig. 10 Rules of the fuzzy system

Table 1 shows a comparison for the Sphere function, F is manually varied in the Differential Evolution algorithm and in the Fuzzy Differential Evolution algorithm F is increased and decreased dynamically with the number of generations with which the experiments are performed.

Table 2 shows a comparison of the Griewank function, F is manually varied in the Differential Evolution algorithm and in the Fuzzy Differential Evolution algorithm F is increased and decreased dynamically with the number of generations with which the experiments are performed.

Table 3 shows a comparison of the Schwefel function, F is manually varied in the Differential Evolution algorithm and in the Fuzzy Differential Evolution algorithm F is increased and decreased dynamically with the number of generations with which the experiments are performed.

Table 4 shows a comparison of the Rastrigin function, F is manually varied in the Differential Evolution algorithm and in the Fuzzy Differential Evolution algorithm F is increased and decreased dynamically with the number of generations with which the experiments are performed.

Table 5 shows a comparison of the Ackley function, F is manually varied in the Differential Evolution algorithm and in the Fuzzy Differential Evolution algorithm F is increased and decreased dynamically with the number of generations with which the experiments are performed.

Table 6 shows a comparison of the Rosenbrock function, F is manually varied in the Differential Evolution algorithm and in the Fuzzy Differential Evolution algorithm F is increased and decreased dynamically with the number of generations with which the experiments are performed.

Table 1 Simulation results of the Sphere function

Sphere function			
Generations	Differential evolution	Fuzzy differential evolution with increasing F	Fuzzy differential evolution with decreasing F
100	274,810.954	264,876.554	191,810.002
500	3416.56935	35.9248183	19.9166364
1000	3.91E+01	0.00048186	0.00024121
2000	9.61E-03	7.2858E-14	3.6657E-14
3000	2.39E-06	1.3168E-23	5.5608E-24
4000	6.10E-10	1.8705E-33	9.4497E-34
5000	1.54E-13	3.1962E-43	1.36E-43

Table 2 Simulation results of the Griewank function

Griewank function			
Generations	Differential evolution	Fuzzy differential evolution with increasing F	Fuzzy differential evolution with decreasing F
100	69.9332605	67.0770188	48.1462646
500	1.37127437	0.6940305	0.51379811
1000	2.07E-01	3.4223E-05	1.836E-05
2000	0.00087914	5.2217E-15	2.2797E-15
3000	2.2279E-07	0	0
4000	5.78723E-11	0	0
5000	1.52742E-14	0	0

Table 3 Simulation results of the Schwefel function

Schwefel function			
Generations	Differential evolution	Fuzzy differential evolution with increasing F	Fuzzy differential evolution with decreasing F
100	10,421.718	11,169.3347	10,932.22075
500	8100.8323	5193.63165	4842.340175
1000	6388.8028	14.5133895	3.958285778
2000	4043.1945	0.00063638	0.000636378
3000	548.84107	0.00063638	0.000636378
4000	0.1285108	0.00063638	0.000636378
5000	0.1323711	0.00063638	0.000636378

Table 4 Simulation results of the Rastrigin function

Rastrigin function			
Generations	Differential evolution	Fuzzy differential evolution with increasing F	Fuzzy differential evolution with decreasing F
100	277,759.787	268,207.392	184,622.012
500	3774.94974	379.027991	342.483337
1000	2.25E+02	144.499278	150.754739
2000	63.1322564	76.3943085	84.6821796
3000	43.6992981	47.4723588	57.9202967
4000	31.1808589	24.2475708	37.6116134
5000	16.6249025	1.2237E-05	1.1348E-04

Table 5 Simulation results of the Ackley function

Ackley function			
Generations	Differential evolution	Fuzzy differential evolution with increasing F	Fuzzy differential evolution with decreasing F
100	13.5629222	14.568239	13.3167462
500	1.89192803	0.4362296	0.26047838
1000	2.77E-01	0.00084351	0.00059808
2000	1.43E-03	1.0485E-08	7.5391E-09
3000	2.07E-05	1.2961E-13	1.02E-13
4000	3.27E-07	8.941E-15	7.52E-15
5000	5.11E-09	7.99E-15	6.81E-15

Table 6 Simulation results of the Rosenbrock function

Rosenbrock function			
Generations	Differential evolution	Fuzzy differential evolution with increasing F	Fuzzy differential evolution with decreasing F
100	193.195633	165.930737	141.722514
500	3.91E+01	44.4584465	34.4383295
1000	1.46E+01	0.30151117	1.54954836
2000	0.05458507	4.3226E-12	2.2078E-10
3000	9.6435E-06	7.95E-23	2.0198E-20
4000	1.7784E-09	0	0
5000	3.4709E-13	0	0

Experiments with the Differential Evolution algorithm varying the CR value (variable crossover) manually in a range from 0.1 to 0.9, were performed with several generations values (GR) 100, 500, 1000, 2000, 3000, 4000 and 5000. Population variables are held constant (NP) = 250, dimension (D) = 50, crossover (CR) = 0.1, upper limit (H) = 500 and lower (L) = -500.

To make a comparison with the Differential Evolution algorithm the averages of the experiments are obtained for each generation number. Experiments are performed by varying F from 0.1 to 0.9 and 30 experiments for each F. then an overall average is obtained for comparison with the Fuzzy Differential Evolution with increase and decrease of the F value.

For experiments with the Fuzzy Differential Evolution algorithm CR changes dynamically increasing and decreasing CR between 0 and 1, 30 experiments for each number of generations are performed and an average is obtained by generation.

For the Ackley and Rosenbrock functions the search space, since the are we were using previously was very large for the Ackley function, the upper limit (H) = 32.768 and the lower limit (L) = -32.768, for the Rosenbrock function the upper limit (H) = 2.048 and the lower limit (L) = -2.048.

This paper mentions two fuzzy systems with which the experiments were performed. It has a fuzzy system which increase the CR variable and another variable decrease CR.

Then the fuzzy system, in which CR is increased dynamically.

- Contains one input and one output
- Is of Mamdani type.
- All membership functions are triangular.
- The input of the fuzzy system is defined by the generations and granulated into three membership functions they are: MF1 = 'Low' [-0.5 0 0.5], MF2 = 'Medium' [0 0.5 1], MF3 = 'High' [0.5 1 1.5].
- The output of the fuzzy system and the variable F is granulated in three membership functions which are: MF1 = 'Low' [-0.5 0 0.5], MF2 = 'Medium' [0 0.5 1], MF3 = 'High' [0.5 1 1.5].
- The fuzzy system uses 3 rules and what it does is increased the value of the CR variable in a range of (0.1).

The fuzzy rules are shown in Fig. 11.

Then the fuzzy system, in which CR is dynamically decreased is described as follows:

- Contains one input and one output
- Is Mamdani type.
- All functions are triangular.
- The input of the fuzzy system is the generations and divided into three membership functions they are: MF1 = 'Low' [-0.5 0 0.5], MF2 = 'Medium' [0 0.5 1], MF3 = 'High' [0.5 1 1.5].
- The output of the fuzzy system and the variable F is divided in three membership functions which are: MF1 = 'Low' [-0.5 0 0.5], MF2 = 'Medium' [0 0.5 1] MF3 = 'High' [0.5 1 1.5].
- The fuzzy system uses 3 rules and what it does is decreased the value of the CR variable in a range of (0.1).

The fuzzy rules are shown in Fig. 12.

We show six tables to compare the results of the Benchmark functions mentioned above, where the variable CR is modified manually in the Difference

1. - If (Generations is Low) then (CR is Low) (1)
2. - If (Generations is Medium) then (CR is Medium) (1)
3. - If (Generations is High) then (CR is High) (1)

Fig. 11 Rules of the fuzzy system

- | |
|---|
| 1. - If (Generations is Low) then (CR is High) (1) |
| 2. - If (Generations is Medium) then (CR is Medium) (1) |
| 3. - If (Generations is High) then (CR is Low) (1) |

Fig. 12 Rules of the fuzzy system

Evolution algorithm, F is increased and described dynamically with the proposed Fuzzy Differential Evolution algorithm.

Table 7 shows a comparison for the Sphere function, CR is manually varied in the Differential Evolution algorithm and in the Fuzzy Differential Evolution algorithm CR is increased and decreased dynamically with the number of generations with which the experiments are performed.

Table 8 shows a comparison of the Griewank function, CR is manually varied in the Differential Evolution algorithm and in the Fuzzy Differential Evolution algorithm CR is increased and decreased dynamically with the number of generations with which the experiments are performed.

Table 9 shows a comparison of the Schwefel function, CR is manually varied in the Differential Evolution algorithm and in the Fuzzy Differential Evolution algorithm CR is increased and decreased dynamically with the number of generations with which the experiments are performed.

Table 10 shows a comparison of the Rastrigin function, CR is manually varied in the Differential Evolution algorithm and in the Fuzzy Differential Evolution algorithm CR is increased and decreased dynamically with the number of generations with which the experiments are performed.

Table 11 shows a comparison of the Ackley function, CR is manually varied in the Differential Evolution algorithm and in the Fuzzy Differential Evolution algorithm CR is increased and decreased dynamically with the number of generations with which the experiments are performed.

Table 7 Simulation results of the Sphere function

Sphere function			
Generations	Differential evolution	Fuzzy differential evolution with increasing CR	Fuzzy differential evolution with decreasing CR
100	1.18E+04	1380.76752	1258.8279
500	1.6E+03	1.0368E-05	64.771525
1000	1.69E+03	6.4725E-07	242.82146
2000	1.98E+03	3.5841E-17	652.19192
3000	2.05E+03	2.5951E-11	383.90060
4000	2.00E+03	1.1499E-18	551.13508
5000	2.36E+03	5.6242E-05	736.02919

Table 8 Simulation results of the Griewank function

Griewank function			
Generations	Differential evolution	Fuzzy differential evolution with increasing CR	Fuzzy differential evolution with decreasing CR
100	3.8301098	1.3683675	1.32397927
500	0.565403	1.141E-06	0.04646616
1000	6.70E-01	2.388E-08	0.25672309
2000	5.300E-10	0	0.4297082
3000	0	0	0.57680736
4000	0	0	0.69710299
5000	0	0	0.89795967

Table 9 Simulation results of the Schwefel function

Schwefel function			
Generations	Differential evolution	Fuzzy differential evolution with increasing CR	Fuzzy differential evolution with decreasing CR
100	8368.7379	11,073.1048	6572.79132
500	260.2027	5141.49869	0.03911202
1000	110.05795	15.2042732	0.00063638
2000	109.0257	0.00063638	0.00063638
3000	116.9357	0.00063638	0.00063638
4000	116.625	0.00063638	0.00063638
5000	122.7558	0.00063638	0.00063638

Table 10 Simulation results of the Rastrigin function

Rastrigin function			
Generations	Differential evolution	Fuzzy differential evolution with increasing CR	Fuzzy differential evolution with decreasing CR
100	277,759.787	268,207.392	184,622.012
500	3774.94974	379.027991	342.483337
1000	2.25E+02	144.499278	150.754739
2000	63.1322564	76.3943085	84.6821796
3000	43.6992981	47.4723588	57.9202967
4000	31.1808589	24.2475708	37.6116134
5000	16.6249025	1.2237E-05	1.1348E-04

Table 11 Simulation results of the Ackley function

Ackley function			
Generations	Differential evolution	Fuzzy differential evolution with increasing CR	Fuzzy differential evolution with decreasing CR
100	4.226611	2.9761272	2.7902716
500	0.674526	3.372E-07	0.13584804
1000	6.50E-01	6.089E-06	0.33358914
2000	6.07E-01	8.768E-12	0.63796586
3000	6.27E-01	4.218E-11	1.07084894
4000	6.10E-01	3.434E-08	1.12410262
5000	6.20E-01	4.440E-15	1.0788061

Table 12 Simulation results of the Rosenbrock function

Rosenbrock function			
Generations	Differential evolution	Fuzzy differential evolution with increasing CR	Fuzzy differential evolution with decreasing CR
100	3.40E+01	34.5356455	22.2152155
500	4.24E+00	0.03299656	4.68563992
1000	3.76E+00	8.1217E-08	318,026.696
2000	3.82E+00	0.03299656	361,504.63
3000	3.73E+00	0	8.31425219
4000	3.86E+00	3.9008E-29	9.0920495
5000	3.95E+00	0	8.6020874

Table 12 shows a comparison of the Rosenbrock function, CR is manually varied in the Differential Evolution algorithm and in the Fuzzy Differential Evolution algorithm CR is increased and decreased dynamically with the number of generations with which the experiments are performed.

6 Conclusions

We conclude that dynamically adjusting the parameters for an optimization method (in this case the Differential Evolution algorithm), we can improve the quality of results, in our case fuzzy logic is used to vary the parameters F and Cr algorithm, we made several modifications to the algorithm dynamically and we observed that the best results are achieved when F is decreased dynamically.

Another important aspect to consider for the experiments is the search space that is given to each function, as happened with Ackley and Rosenbrock functions in which the search space is modified to obtain better results.

References

1. Aguas-Marmolejo, S.J., Castillo, O.: Optimization of membership functions for Type-1 and Type 2 fuzzy controllers of an autonomous mobile robot using PSO. *Recent Adv. Hybrid Intell. Syst.* 97–104 (2013)
2. Astudillo, L., Melin, P., Castillo, O.: Optimization of a fuzzy tracking controller for an autonomous mobile robot under perturbed torques by means of a chemical optimization paradigm. *Recent Adv. Hybrid Intell. Syst.* 3–20 (2013)
3. Eftekhari, M., Katebi, S.D., Karimi, M., Jahanmir, A.H.: Eliciting transparent fuzzy model using differential evolution. *Appl. Soft Comput.* **8**, 466–476 (2008)
4. Fierro, R., Castillo, O.: Design of fuzzy control systems with different PSO variants. *Recent Adv. Hybrid Intell. Syst.* 81–88 (2013)
5. Hachicha, N., Jarboui, B., Siarry, P.: A fuzzy logic control using a differential evolution algorithm aimed at modelling the financial market dynamics. *Inf. Sci.* **181**, 79–91 (2011)
6. Lizárraga, E., Castillo, O., Soria, J.: A method to solve the traveling salesman problem using ant colony optimization variants with ant set partitioning. *Recent Adv. Hybrid Intell. Syst.* 237–2461 (2013)
7. Melendez, A., Castillo, O.: Evolutionary optimization of the fuzzy integrator in a navigation system for a mobile robot. *Recent Adv. Hybrid Intell. Syst.* 21–31 (2013)
8. Melin, P., Olivas, F., Castillo, O., Valdez, F., Soria, J., García, J.: Optimal design of fuzzy classification systems using PSO with dynamic parameter adaptation through fuzzy logic. *Expert Syst. Appl.* **40**(8), 3196–3206 (2013)
9. Mezura-Montes, E., Palomeque-Ortiz, A.: Self-adaptive and Deterministic Parameter Control in Differential Evolution for Constrained Optimization. Laboratorio Nacional de Informática Avanzada (LANIA A.C.), Xalapa (2009)
10. Neyoy, H., Castillo, O., Soria, I.: Dynamic fuzzy logic parameter tuning for ACO and its application in TSP problems. *Recent Adv. Hybrid Intell. Syst.* 259–271 (2013)
11. Oh, S.-K., Kim, W.-D., Pedrycz, W.: Design of optimized cascade fuzzy controller based on differential evolution: simulation studies and practical insights. *Eng. Appl. Artif. Intell.* **25**, 520–532 (2012)
12. Olivas, F., Castillo, O.: Particle swarm optimization with dynamic parameter adaptation using fuzzy logic for benchmark mathematical functions. *Recent Adv. Hybrid Intell. Syst.* 247–258 (2013)
13. Price, K.V., Storn, R.M., Lampinen, J.A.: Differential Evolution. Springer, Berlin (2005)
14. Raofen, W., Zhang, J., Zhang, Y., Wang, X.: Assessment of human operator functional state using a novel differential evolution optimization based adaptive fuzzy model. *Biomed. Signal Process. Control* **7**, 490–498 (2012)
15. Sombra, A., Valdez, F., Melin, P., Castillo, O.: A new gravitational search algorithm using fuzzy logic to parameter adaptation. *IEEE congress on evolutionary computation*, pp. 1068–1074 (2013)
16. Valdez, F., Melin, P., Castillo, O.: Evolutionary method combining particle swarm optimization and genetic algorithms using fuzzy logic for decision making. In: *Proceedings of the IEEE International Conference on Fuzzy Systems*, pp. 2114–2119 (2009)
17. Valdez, F., Melin, P., Castillo, O.: Parallel particle swarm optimization with parameters adaptation using fuzzy logic. *MICAI* (2), 374–385 (2012)
18. Valdez, F., Melin, P., Castillo, O.: Bio-inspired optimization methods on graphic processing unit for minimization of complex mathematical functions. *Recent Adv. Hybrid Intell. Syst.* 313–322 (2013)
19. Valdez, F., Melin, P., Castillo, O.: An improved evolutionary method with fuzzy logic for combining particle swarm optimization and genetic algorithms. *Appl. Soft Comput.* **11**, 2625–2632 (2011)
20. Vucetic, D.: Fuzzy Differential Evolution Algorithm. The University of Western Ontario, Ontario (2012)

Ant Colony Optimization with Parameter Adaptation Using Fuzzy Logic for TSP Problems

Frumen Olivas, Fevrier Valdez and Oscar Castillo

Abstract In this paper we propose a method for parameter adaptation in Ant Colony Optimization (ACO); with the use of a fuzzy system we dynamically adapt the “rho” parameter which is responsible for the evaporation of the pheromone trails in ACO. The main goal is to improve the results of ACO; basically the fuzzy system controls the ACO abilities for exploration and exploitation of the search space. The best problems to test ACO algorithms are the TSP problems, so a comparison with the proposed approach and others methods is performed and results discussed.

1 Introduction

Since Ant Colony Optimization (ACO) was developed by Dorigo in 1992 [1], it has the problem of the adjustment of their parameters depending on the problem that is applied, our proposed method tries to attack this problem with the help of the fuzzy logic proposed by Zadeh [21–23]. This is our proposed approach uses a fuzzy system to dynamically adapt the parameters of ACO, and self-adapt ACO to the problem in which is applied. To do this, we are based on some knowledge about the behavior of ACO and their parameters; first we apply ACO to different problems and by doing several experiments, extract the behavior of the parameters and now we can model this behavior with the use of fuzzy rules from a fuzzy system.

The goal of this paper is to propose a new method for dynamic parameter adaptation in ACO, with the help of fuzzy logic, we can model the way in which ACO explore the space of search, also the way in which exploits the best area of the

F. Olivas · F. Valdez (✉) · O. Castillo
Tijuana Institute of Technology, Tijuana, Mexico
e-mail: fevrier@tectijuana.mx

F. Olivas
e-mail: frumen@msn.com

O. Castillo
e-mail: ocastillo@tectijuana.mx

space of search found so far. So we want to control the abilities of ACO to search in global and local, with a fuzzy system.

ACO was introduced by Dorigo in 1992 [1], this method uses the idea of collective intelligence, but more particularly, the behavior of ant colonies to find food, through the use of a trail substance or pheromone in real ants. In this case ACO use artificial ants and each one represent a solution to the problem in which it is applied, to update the position of the artificial ants, ACO use Eq. 1 to calculate the probabilities of an ant k to select the next node from city i to city j . Equation 2 represents the evaporation of trail between time t and $t + 1$. Equation 3 indicates how the pheromone trail is updated. And Eq. 4 defines how much pheromone is deposited in the arcs of the tour constructed by the ant k

$$P_{ij}^k = \frac{[\tau_{ij}]^\alpha [\eta_{ij}]^\beta}{\sum_l \in N_i^k [\tau_{il}]^\alpha [\eta_{il}]^\beta}, \quad \text{if } j \in N_i^k \quad (1)$$

where in Eq. 1, τ_{ij} is the pheromone trail associated to the arc (i, j) , α determines the influence of the pheromone trail, η_{ij} is an heuristic value related to the distance between cities i and j , β determine the effects of the heuristic information, and N is the neighborhood of the ant k when it is city i , this is, the set of cities that the ant k haven't visited yet.

$$\tau_{ij} \leftarrow (1 - \rho)\tau_{ij}, \quad \forall (i, j) \in L \quad (2)$$

In Eq. 2, ρ is the percentage of evaporation of the pheromone trail, and L is a set of all arcs in the graph.

$$\tau_{ij} \leftarrow \tau_{ij} + \sum_{k=1}^n \Delta\tau_{ij}^k, \quad \forall (i, j) \in L \quad (3)$$

In Eq. 3, Δ is the amount of pheromone deposited in the arc (i, j) by the ant k , defined in Eq. 4.

$$\Delta\tau_{ij} = \begin{cases} \frac{1}{C^k}, & \text{if the arc } (i, j) \text{ belongs to } T^k \\ 0, & \text{otherwise} \end{cases} \quad (4)$$

In Eq. 4, C is the length of the tour constructed by the ant k , and T is the tour constructed by the ant k .

The parameter that we want to control or dynamically adapt is ρ (rho), which is the rate of evaporation of the pheromone trails, through Eq. 2. By adjusting the rho parameter, we help ACO to “forget” bad solutions in early iterations, and to reinforce the best solutions in final iterations.

The variation of ACO chosen for this investigation is Rank-based ACO, and this is based on several experiments that we perform previously, to select the best version of ACO to make a parameter adaptation.

The problem, in which the tests were performed, is the traveling salesman problem (TSP), in which the salesman requires visit each of the given n cities only once, starting from any city and return to the same city. The cities are connected with an arc that has a cost or distance. So we need to find the best tour to minimize the cost or the total travel distance. In other words we need to find a Hamiltonian cycle with minimal cost.

There are some improvements to the original method of ACO made by Dorigo the author of the method, and has been applied his method to the TSP problems, such as: Ant colony system: a cooperative learning approach to the traveling salesman problem [2], Ant colonies for the traveling salesman problem [3], The ant colony optimization meta-heuristic [4], and The ant system: optimization by a colony of cooperating agents [5], in which have been some improvements to the original method and experimentation with its parameters.

Our investigation is based in the ant colony optimization algorithm published by Haupt and Haupt in Practical genetic algorithms second edition [10], and taking some ideas from the book by Engelbrecht in Fundamentals of computational swarm intelligence [9]. Also some ideas of the hybridization of a fuzzy system and the ant colony optimization method have been taken from the book by Jang et al. Neuro-fuzzy and soft computing: a computational approach to learning and machine intelligence [11].

The ACO algorithm has been compared against the genetic algorithms (GA), simulated annealing (SA) and particle swarm optimization (PSO), and can be found in [14], in which can be observed that ACO don't have the best results, so this is one of the reasons of why we want to improve the quality of the results of ACO with the inclusion of a fuzzy system. Also, in the past, we improve the method of PSO in [16] with a fuzzy system, and in [17] with a type-2 fuzzy system. ACO has an improvement with a fuzzy system that can be found in [15]. So we want to level up the ACO algorithm with fuzzy logic in a better way.

There are also some other improvements in ant colony optimization in the literature, for example: Van Ast et al. [19] proposed an ACO algorithm that has a fuzzy partitioning of the state space of the system, Yu et al. [20] proposed an ACO with fuzzy pheromone laying mechanism, Einipour [7] a fuzzy-ACO method for detect breast cancer, Elloumi et al. [8] proposed an hybridation of fuzzy PSO and fuzzy ACO applied to TSP problems, Khan and Engelbrecht [12] proposed a fuzzy ant colony optimization for topology design of distributed local area networks.

2 Methodology

To perform the dynamic parameter adaptation in ACO, we use a fuzzy system that contains two inputs and one output. In this case, as inputs we use a percentage of the elapsed iterations described in Eq. 5, and a measure of the dispersion of the ant colony relative to the best ant described in Eq. 6.

$$\text{Iteration} = \frac{\text{Current Iteration}}{\text{Maximum of Iterations}} \quad (5)$$

where in Eq. 5, current iteration is the number of elapsed iterations, and maximum of iterations in the number established for ACO to find the best solution.

$$\text{Diversity } (S(t)) = \frac{1}{n_s} \sum_{i=1}^{n_s} \sqrt{\sum_{j=1}^{n_x} (x_{ij}(t) - \bar{x}_j(t))^2} \quad (6)$$

In Eq. 6, S is the colony or the population of ACO; t is the current iteration or time, n_s is the size of the colony, i is the number of the ant, n_x is the total number of dimensions, j is the number of the dimension, x_{ij} is the j dimension of the ant i , \bar{x}_j is the j dimension of the current best ant of the colony.

The input and output variables are illustrated in Figs. 1, 2 and 3.

Fig. 1 Iteration as input variable

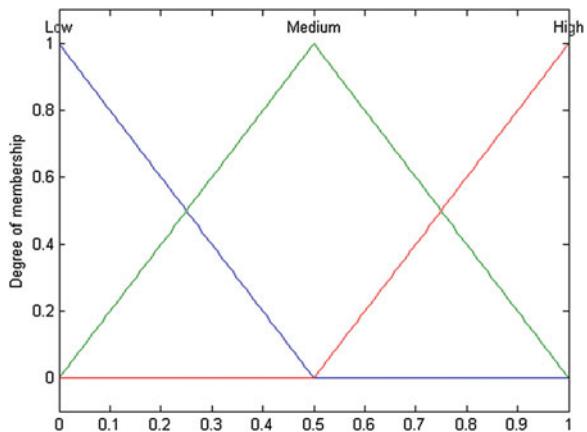


Fig. 2 Diversity as input variable

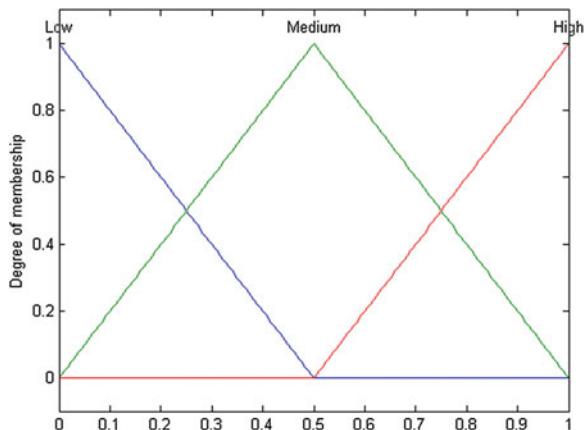


Fig. 3 Rho parameter as output variable

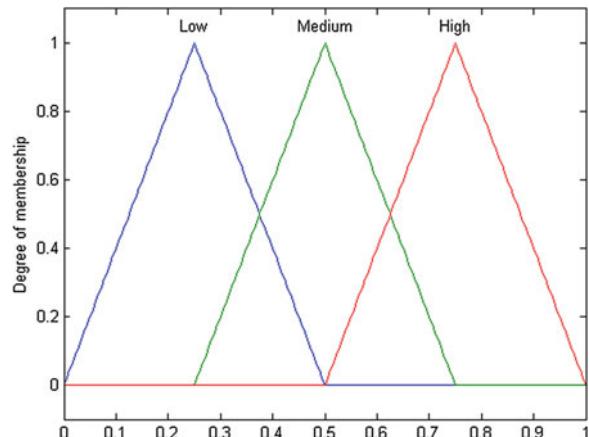


Figure 1 shows the iteration variable, with a range from 0 to 1, and granulated into three triangular membership functions.

In Fig. 2 the diversity variable is shown, with a range from 0 to 1, and granulated into three triangular membership functions.

Rho as output variable is illustrated in Fig. 3, with a range from 0 to 1, and granulated into three triangular membership functions. The range used for “rho” is taken from [6], where it suggests values bigger than 0 and lower than 1.

With the possible inputs and outputs of the fuzzy system, we decide to combine these fuzzy variables, to design three fuzzy systems, all of them with rho as output variable. The first system is using the iteration variable as input, as can be noted in Fig. 4, the second system is using the diversity as input, illustrated in Fig. 5, and finally using the iteration and diversity variable as inputs, shown in Fig. 6.

The fuzzy system from Fig. 4, uses three fuzzy rules that were developed using previous knowledge of the problem, this is, in early iterations the rho parameter must be high, and with this increase the exploration of the ants by allowing forget bad solutions, and in final iterations rho must be low, so the ants can exploit the best area, by reinforcement of the best solutions found so far.

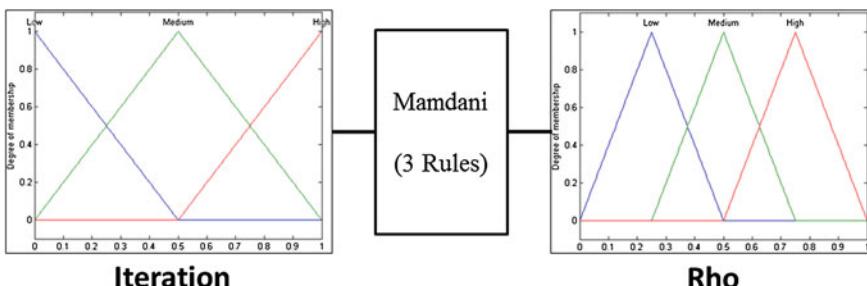


Fig. 4 Fuzzy system for parameter adaptation in ACO with iteration as input

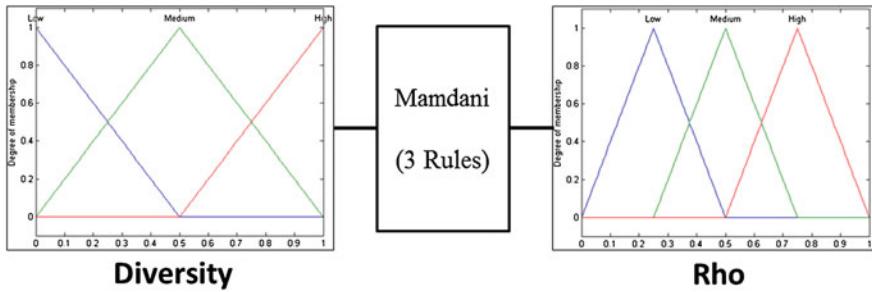


Fig. 5 Fuzzy system for parameter adaptation in ACO with diversity as input

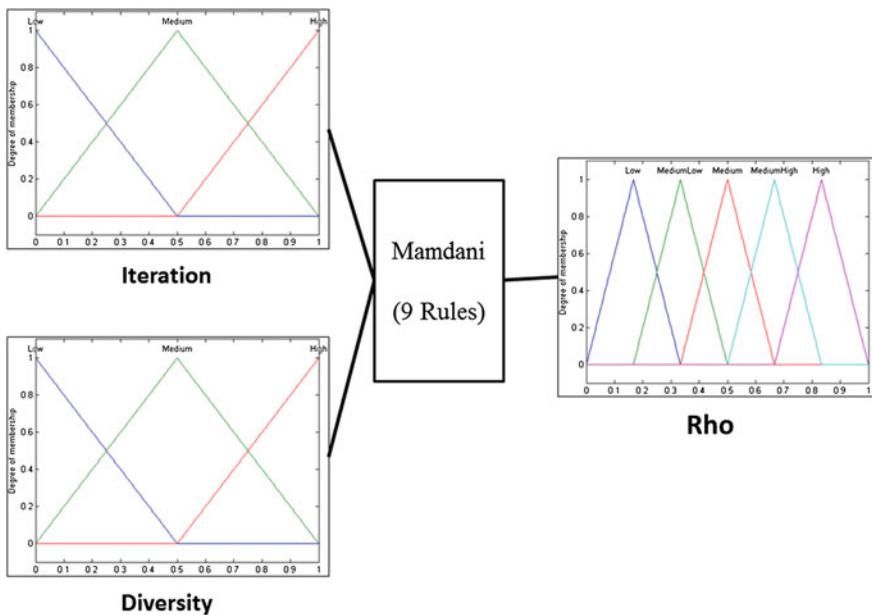


Fig. 6 Fuzzy system for parameter adaptation in ACO with iteration and diversity as inputs

The fuzzy rules for the system illustrated in Fig. 5, were developed taking into account that when diversity is low the “rho” parameter must be high, this is when ants are taking the same tour we need an rho high so the ants can forget bad tours or in another words increase the exploration of the colony, and when diversity is high rho must be low, this is, when ants are too much separated we want that exploit the best area found so far.

In the fuzzy system shown in Fig. 6, the output variable rho has a different granularity, this is, now it has five triangular membership functions, we decided to increase the number of membership functions because the use of two inputs, which

1. If (Iteration is Low) and (Diversity is Low) then	(Rho is High)
2. If (Iteration is Low) and (Diversity is Medium) then	(Rho is MediumHigh)
3. If (Iteration is Low) and (Diversity is High) then	(Rho is Medium)
4. If (Iteration is Medium) and (Diversity is Low) then	(Rho is MediumHigh)
5. If (Iteration is Medium) and (Diversity is Medium) then	(Rho is Medium)
6. If (Iteration is Medium) and (Diversity is High) then	(Rho is MediumLow)
7. If (Iteration is High) and (Diversity is Low) then	(Rho is Medium)
8. If (Iteration is High) and (Diversity is Medium) then	(Rho is MediumLow)
9. If (Iteration is High) and (Diversity is High) then	(Rho is Low)

Fig. 7 Fuzzy rule set for the fuzzy system with iteration and diversity as inputs

cause that the rules increase, and now has nine rules illustrated in Fig. 7, and taking into account these factors we decide to increase the number of membership functions, and with this allow a better model of the parameter adaptation.

The fuzzy rule set shown in Fig. 7, were develop based into a previous knowledge, and in a combination of the previous systems, but basically is to control the search abilities of ACO, now having two metrics about ACO that are percentage of the iterations and diversity of the colony. To give you an idea of how we develop this fuzzy rule set, here we have an example: if the percentage of iterations is low (this means on early iteration), and when the degree of dispersion of the particles is low (this means the colony of ants are closer to the best ant), then we need to increase the diversity or in other words use exploration, because we want to allow that the ants find another solutions, so to do that, we set rho in high, because this parameter helps ACO to forget bad solutions and explore new solutions of the space of search.

3 Experimentation, Results and Comparisons

Each of the three proposed fuzzy systems are integrated in ACO to dynamically adapt the rho parameter, and were tested in some TSPs, which characteristics are described in Table 1, and the parameters for ACO are described in Table 2. The TSPs in Table 1 were chosen from the TSPLIB [18], and because we want to compare with other works that use the same TSPs. In TSPs the order of complexity is given by the number of cities, this is a lower number of cities more easy is the

Table 1 Characteristics of the TSPs

TSP	Number of cities	Minimum
Burma14	14	3323
Ulysses22	22	7013
Berlin52	52	7542
Eil76	76	538
KroA100	100	21,282

Table 2 Parameters for ACO

Parameter	ACO_{Rank}	$\text{ACO}_{\text{Rank}} + \text{FS1}$	$\text{ACO}_{\text{Rank}} + \text{FS2}$	Proposed
Alpha α	1	1	1	1
Beta β	2	2	2	2
Rho ρ	0.1	Dynamic	Dynamic	Dynamic
Iterations	1000	1000	1000	1000
Ants	Num of cities	Num of cities	Num of cities	Num of cities

problem, in this case the TSP more easy has 14 cities and the harder TSP has 100 cities.

The parameters in Table 2, were chosen based on several experiments, that we performed to find the best parameters for ACO Rank-based without the parameter adaptation, and we use the same parameters for the proposed methods and only rho parameter is dynamic, because will be changed each iteration.

The methods in Table 2 are described below:

ACO_{Rank} is the original ACO Rank-based method without the parameter adaptation.

$\text{ACO}_{\text{Rank}} + \text{FS1}$ is ACO Rank-based that use the fuzzy system that has iteration as input.

$\text{ACO}_{\text{Rank}} + \text{FS2}$ is ACO Rank-based that use the fuzzy system that has diversity as input.

The proposed Rank-based ACO that uses the fuzzy system that has iteration and diversity as inputs.

Each result in Table 3 is an average of 30 experiments of applying the methods to the TSPs, with the parameters in Table 2, the results in bold are the best average results.

From Table 3 it can be observed that the proposed approach obtains on average better results when compared with the other methods, and only in the first TSP, cannot obtain the best result, but this is because the problem is the easiest and in all 30 experiments obtain the global minimum.

Table 3 Average results from applying the fuzzy systems for the parameter adaptation on ACO to the TSPs

TSP	ACO_{Rank}	$\text{ACO}_{\text{Rank}} + \text{FS1}$	$\text{ACO}_{\text{Rank}} + \text{FS2}$	Proposed
Burma14	3350.3	3327.96	3323	3323
Ulysses22	7089.1	7043.70	7021.26	7013
Berlin52	7850.4	7596.16	7795.96	7552.56
Eil76	554.76	587.93	608.46	545.13
KroA100	22,591	23,658.30	24,634.13	21,577.30

Table 4 Standard deviation from the average results of Table 3

TSP	ACO _{Rank}	ACO _{Rank} + FS1	ACO _{Rank} + FS2	Proposed
Burma14	80.43	10.35	0	0
Ulysses22	50.09	35.74	22.51	0
Berlin52	179.25	95.73	125.17	46.84
Eil76	8.07	10.96	9.65	6.67
KroA100	486.67	454.12	427.22	175.20

The results in Table 4 are the standard deviation of the same 30 experiments that we use to obtain the results of Table 3, and each result in bold is the best result.

Also in the results from Table 4, our proposed approach obtains the lowest standard deviation in all the TSPs, and again only in the first TSP cannot obtain the best result, and this is because the method ACO Rank-based using the second fuzzy system (ACO_{Rank} + FS2) to obtain the global minimum in all the experiments, and also our proposed method obtain the same results, so the standard deviation of these experiments is zero. The results from Tables 3 and 4 show the performance from each fuzzy system for parameter adaptation developed; in this case we can see that the results from the proposed are better when compared with the others methods.

However we want to compare the methods using a statistical test, in this case, using the Z-test using the reference in [13], and the results of applying this test are shown in Table 5. And the parameters using for the tests are: the same 30 experiments, the null hypothesis says that the proposed approach obtains on average greater than or equal results when compared with the other method ($H_0: \mu_1 \geq \mu_2$), and the alternative hypothesis says that the proposed approach obtain on average better results when compared with the other method ($H_1: \mu_1 < \mu_2$), the level of significance is 5 %, and the critical value is $Z_0 = -1.96$, so the rejection region is for all values of the Z-test lowers than Z_0 . The results in bold from Table 5 are the values of the Z-test that doesn't fall into the rejection region; all other values are in the rejection region.

From the results in Table 5 only in 2 results our proposed method fail to reject the null hypothesis, and this is in the first problem, which is the easiest problem, however in all other results our proposed approach has enough evidence with a level of significance of 5 % to reject the null hypothesis, this is our proposed method can obtain on average better results when compared with the other three methods, including the original ACO Rank-based, and the other two proposed methods.

Table 5 Results of the comparisons using the Z-test

TSP	ACO _{Rank}	ACO _{Rank} + FS1	ACO _{Rank} + FS2
Burma14	-1.861	-2.628	NaN
Ulysses22	-8.321	-4.703	-2.010
Berlin52	-8.805	-2.240	-9.974
Eil76	-5.036	-18.264	-29.562
KroA100	-10.731	-23.416	-36.259

4 Conclusions

We can conclude that we have proposed a method for the parameter adaptation in Ant colony optimization rank-based or ASRank, and we could improve the quality of the results when compared with the original algorithm, this is, our proposed approach obtains on average better results when compared with the other proposed method and also with the original algorithm. Also from the results contained in Tables 3 and 4, it can be observed that the averages and standard deviations of the proposed approach are lower than the other methods in most of the experiments with the TSPs. From the results of the Z-test from Table 5, only in the first problem, we do not find enough evidence to reject the null hypothesis, however, in all other results we find enough evidence and with a level of significance of 5 % our proposed approach has on average better results when compared with the other methods.

References

1. Dorigo, M.: Optimization, learning and natural algorithms. Ph.D. thesis, Dipartimento di Elettronica, Politecnico di Milano, Italy (1992)
2. Dorigo, M., Gambardella, L.: Ant colony system: a cooperative learning approach to the traveling salesman problem. *IEEE Trans. Evol. Comput.* **1**, 53–66 (1997)
3. Dorigo, M., Gambardella, L.: Ant colonies for the traveling salesman problem. *Biosystems* **43**, 73–81 (1997)
4. Dorigo, M., Di Caro, G.: The ant colony optimization meta-heuristic. In: Corne, D., Dorigo, M., Glover, F. (eds.) *New ideas in optimization*, pp. 11–32. McGraw-Hill, New York (1999)
5. Dorigo, M., Maniezzo, V., Colorni, A.: The ant system: optimization by a colony of cooperating agents. *IEEE Trans. Syst. Man Cybern. B* **26**(2), 29–41 (1996)
6. Dorigo, M., Stutzle, T.: *Ant colony optimization*. MIT Press, Cambridge (2004)
7. Einipour, A.: A fuzzy-ACO method for detect breast cancer. *Glob. J. Health Sci.* **3**(2), 195–199 (2011)
8. Elloumi, W., Baklouti, N., Abraham, A., Alimi, A.M.: Hybridization of fuzzy PSO and fuzzy ACO applied to TSP. In: 13th international conference on hybrid intelligent systems (HIS), pp. 106–111 (2013)
9. Engelbrecht, A.: *Fundamentals of computational swarm intelligence*. University of Pretoria, South Africa (2005)
10. Haupt R., Haupt S.: *Practical genetic algorithms*, 2nd edn. A Wiley-Interscience Publication, Hoboken (2004)
11. Jang, J., Sun, C., Mizutani, E.: *Neuro-fuzzy and soft computing: a computational approach to learning and machine intelligence*. Prentice-Hall, Upper Saddle River (1997)
12. Khan, S.A., Engelbrecht, A.P.: A fuzzy ant colony optimization algorithm for topology design of distributed local area networks. In: Swarm intelligence symposium, SIS 2008, IEEE, pp. 1–7 (2008)
13. Larson, R., Farber, B.: *Elementary statistics: picturing world*, 5th edn. Prentice Hall, Boston (2012)
14. Nedjah, N.: *Swarm intelligent systems*, vol. 26. Springer, Heidelberg (2006)
15. Neyoy, H., Castillo, O., Soria, J.: Dynamic fuzzy logic parameter tuning for ACO and its application in TSP problems. *SCI* **451**, 259–271 (2012)

16. Olivas F., Castillo O.: Optimal design of fuzzy classification systems using PSO with dynamic parameter adaptation through fuzzy logic. *Expert Syst. Appl.* **40**, 3196–3206 (2012)
17. Olivas F., Valdez F., Castillo O.: Particle swarm optimization with dynamic parameter adaptation using interval type-2 fuzzy logic for benchmark mathematical functions. In: 2013 World congress on nature and biologically inspired computing (NaBIC), pp. 36–40 (2013)
18. Reinelt, G.: TSP_LIB—a traveling salesman problem library. *ORSA J. Comput.* **3**, 376–384 (1991)
19. Van Ast J., Babuska R., De Schutter B.: Fuzzy ant colony optimization for optimal control. In: Proceedings of the 2009 American control conference, St. Louis, Missouri, pp. 1003–1008 (2009)
20. Yu, L., Yan, J.F., Yan, G.R., Yi, L.: ACO with fuzzy pheromone laying mechanism. Emerging intelligent computing technology and applications, pp. 109–117. Springer, Berlin (2012)
21. Zadeh, L.: Fuzzy sets. *Inf. Control* **8**, 338–353 (1965)
22. Zadeh, L.: Fuzzy logic. *IEEE Comput. Mag.* **1**, 83–93 (1988)
23. Zadeh, L.: The concept of a linguistic variable and its application to approximate reasoning—I. *Inform. Sci.* **8**, 199–249 (1975)

An Improved Harmony Search Algorithm Using Fuzzy Logic for the Optimization of Mathematical Functions

Cinthia Peraza, Fevrier Valdez and Oscar Castillo

Abstract In this paper a new method for dynamic parameter adaptation in harmony search (HS) algorithm is proposed. HS is a music inspired metaheuristic optimization algorithm, in particular we refer to jazz improvisation and is applied to solve complex problems. In this paper we propose an improvement to the convergence of the harmony search algorithm using fuzzy logic. Simulation results show that the propose approach improves the performance of HS, and benchmark mathematical functions are used to illustrate the feasibility of the proposed approach.

1 Introduction

The use of fuzzy logic in evolutionary computing is becoming a common approach to improve the performance of the algorithms [2, 11, 14–16]. Currently the parameters involved in the algorithms are determined by trial and error. In this aspect we propose the application of fuzzy logic which is responsible in performing the dynamic adjustment of harmony memory accepting and pitch adjustment parameters in the Harmony Search (HS) algorithm. This use of fuzzy logic has the goal of providing better performance to Harmony Search.

Fuzzy logic or multi valued logic is based on fuzzy set theory proposed by Zadeh in 1965, which helps us in modeling knowledge, through the use of if then fuzzy rules. The fuzzy set theory provides a systematic calculus to deal with linguistic information, and that improves the numerical computation by using linguistic labels stipulated by membership functions [12].

Harmony Search (HS) is one of the latest metaheuristic algorithms that have been proposed. It was created in 2001 by Geem [6], it is inspired by the observation that the aim of music is to search for a perfect state of harmony [10]. This harmony in music is analogous to finding the optimality in an optimization process.

C. Peraza · F. Valdez (✉) · O. Castillo
Tijuana Institute of Technology, Tijuana, BC, Mexico
e-mail: fevrier@tectijuana.mx

Originally, applications where HS was first assessed as an effective Meta heuristic focused mainly on the design of water distribution networks [3, 5], benchmark optimization [14] and others. Since then several studies were devoted in the development of new HS variants and hybridizations with other metaheuristic algorithms [1, 8, 9, 17]. Since then, the activity around this algorithm has increased sharply.

Similarly there are papers on Harmony Search algorithm (HS) applications that uses this algorithm to solve real problems. To mention a few: A Parameter setting free harmony search algorithm [4], A Tabu Harmony Search Based Approach to Fuzzy Linear Regression [7].

In order to get a clear overview of the classification and analysis presented in the following sections: Sect. 2 shows the concept of the harmony search algorithm as applied to the technique for parameter optimization. Section 3 described the proposed methods. Section 4 shows the methodology for parameter adaptation. Section 5 offers the simulations results. Section 6 described the conclusions.

2 Harmony Search Algorithm

Harmony search is a relatively new metaheuristic optimization algorithm inspired music and was first developed by Geem in 2001 [6].

This algorithm can be explained in more in detail with the process of improvisation that a musician uses, which consists of three options:

1. Play any song you have in your memory
2. Play a similar composition to an existing
3. Play a new song or randomly

If we formalize these three options for optimization, we have three corresponding components: memory usage of harmony, pitch adjustment and randomization [18].

2.1 Memory in the Harmony Search Algorithm

The use of harmony memory is important because it is similar to choosing the best individuals in genetic algorithms. This will ensure the best harmonies will be transferred to the new memory harmony. In order to use this memory more effectively, we can assign a parameter $r_{accept} \in [0,1]$ called acceptance rate memory. If this rate is too low, it just selects the best harmonies and may converge very slowly [18].

$$r_{accept} \in [0,1] \quad (1)$$

2.2 Pitch Adjustment

To adjust the pitch slightly in the second component, we have to use such a method can adjust the frequency efficiently. In theory, the tone can be adjusted linearly or nonlinearly, but in practice the linear approach is used. If the current solution is x_{old} (or pitch), then the new solution (tone) is generated x_{new} .

$$x_{new} = x_{old} + b_p(2rand - 1) \quad (2)$$

where “rand” is a random number drawn from a uniform distribution [0,1]. Here is it bandwidth, which controls the local range of tone adjustment in fact, we can see that the pitch adjustment (2) is a random step.

The Pitch setting is similar to the mutation operator in genetic algorithms. We can assign a pitch adjustment rate to control the degree of adjustment. If too low, there is usually no change. If too high, then the algorithm may not converge at all [18].

2.3 Randomization

The third component is a randomization component (3) that is used to increase the diversity of the solutions. Although the tone setting has a similar role, but it is limited to certain local tone adjustment and therefore correspond to a local search. The use of randomization can further push the system to explore various regions with high diversity solution in order to find the global optimum [18]. So we have:

$$P_a = P_{lower\ limit} + P_{range} * rand \quad (3)$$

where ‘An Improved Harmony Search Algorithm using Fuzzy Logic “rand” is a generator of random numbers in the range of 0 and 1’. (Search space) $P_{range} = P_{upper\ limit} - P_{lower\ limit}$.

The two remaining components in harmony search can be summarized in the pseudo code shown in Sect. 2.4, where we can find that the probability of a true randomization (4) is

$$P_{random} = 1 - r_{accept} \quad (4)$$

and the actual probability of tone adjustment (5) is

$$P_{tono} = r_{accept} * r_{pa} \quad (5)$$

2.4 Pseudo Code for Harmony Search Algorithm

The pseudo code for HS is presented below:

```

Objective function  $f(x)$ ,  $x = (x_1, \dots, x_n)^T$ 
Initial generate harmonics (matrices of real numbers)
Define pitch adjustment rate (rpa) and limits of tone
Define acceptance rate of the harmony memory (r accept)
    while ( $t <$ Maximum number of iterations)
        Generate a new harmony and accept the best harmonies
        Setting the tone for new harmonies (solutions)
        if (rand>raccept)
            Choose an existing harmony randomly
        else if (rand>rpa)
            Setting the tone at random within a bandwidth      (2)
        else
            Generate a new harmony through a randomization (3)
        End if
        Accepting new harmonies (solutions) best
    End while

```

To find the best solutions.

3 Proposed Method

The harmony search algorithm (HS) is a powerful search technique used for solving optimization problems. In this section a new algorithm called Fuzzy Harmony Search (FHS) with dynamic adjustment of parameters is proposed. The main objective is that the fuzzy system will provides us with the optimal parameters for the best performance of the HS algorithm. In addition the parameters that the fuzzy system optimizes are the harmony memory accepting and pitch adjustment, as shown in Fig. 1.

4 Methodology for Parameter Adaptation

In Sect. 2 we show the most important parameters of the algorithm, based on the literature, we decided to use the parameter rate of acceptance of memory as an outlet for our fuzzy system and must be in the range of 0.7 and 1, plus it is also suggested that changing the parameter rate of accepting of memory dynamically during the execution of this algorithm can produce better results.

In addition it is also found that the algorithm performance measures, such as: the iterations, needs to be considered to run the algorithm, among others. In our work

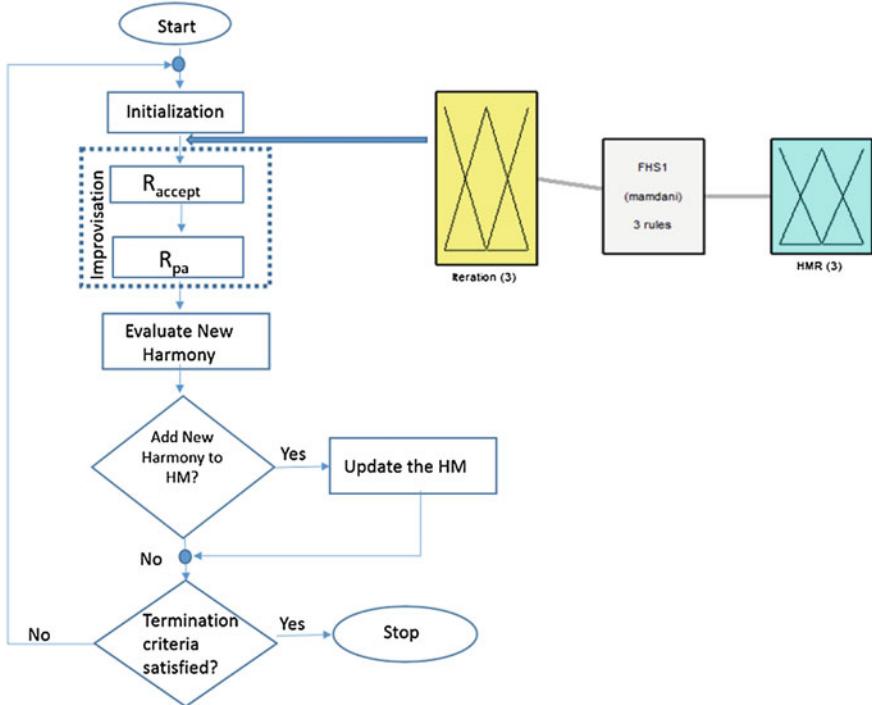


Fig. 1 Proposal for fuzzy dynamic adaptation of HS

all the above are taken in consideration for the fuzzy systems to modify the parameter rate of accepting of memory dynamically changing these parameters in each iteration of the algorithm.

For measuring the iterations of the algorithm, it was decided to use a percentage of iterations, i.e. when starting the algorithm the iterations will be considered “low”, and when the iterations are completed it will be considered “high” or close to 100 %, the design of the fuzzy system can be seen in Fig. 2 it has a one input and one output. To represent this idea we use [13]:

$$\text{Iteration} = \frac{\text{Current Iteration}}{\text{Maximum of Iterations}} \quad (6)$$

The design of the input variable can be appreciated in Fig. 3, which shows the input iteration, that input is granulated into three triangular membership functions. For the output variables, as mentioned above, the recommended values for HMR are between 0.7 and 1, so that the output variables were designed using this range of values. The output is granulated in three triangular membership functions. The design of the output variable can be seen in Fig. 4. The surface of fuzzy system is shown in Fig. 5 and the rules for the fuzzy system are shown in Fig. 6.

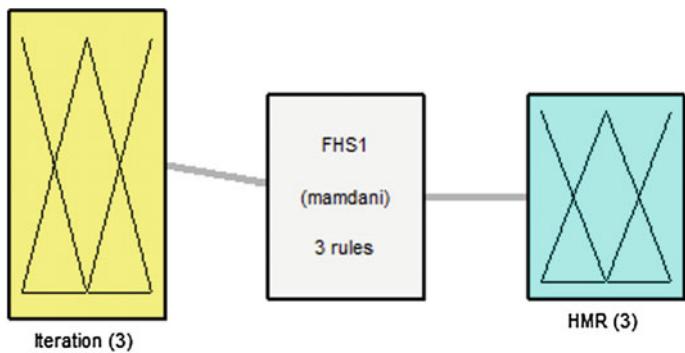


Fig. 2 Fuzzy system for parameter adaptation

Fig. 3 Input 1: Iteration

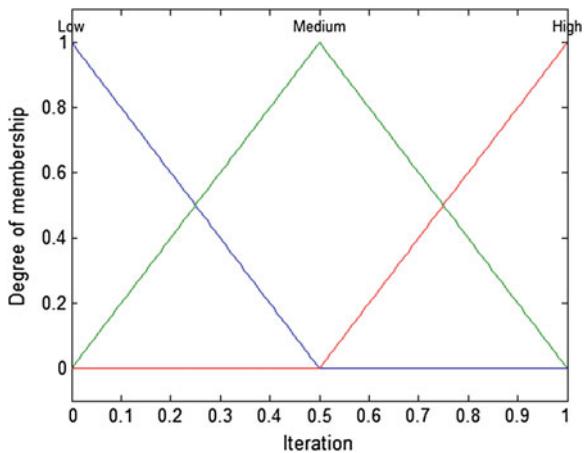
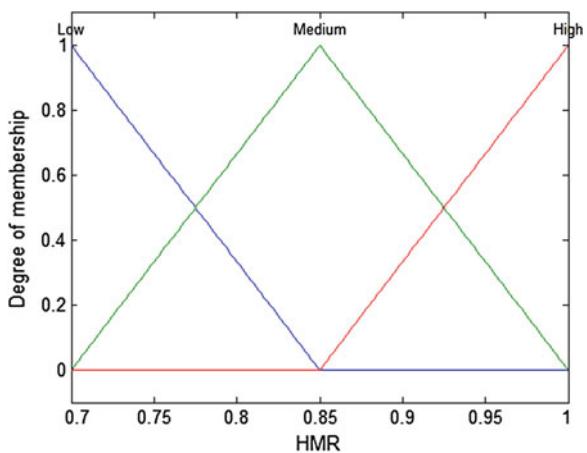


Fig. 4 Output 1: HMR



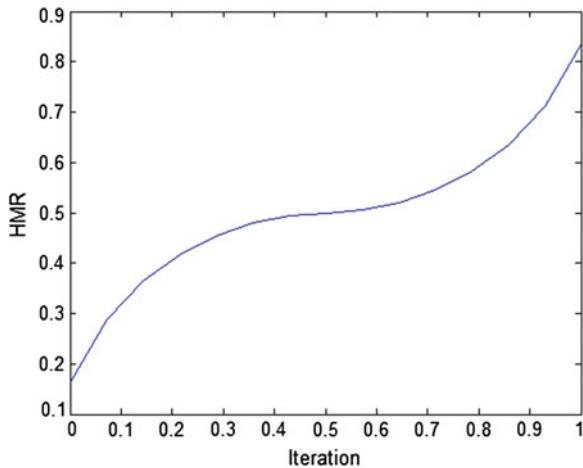


Fig. 5 Surface fuzzy system

1. If (Iteration is Low) then (HMR is Low) (1) 2. If (Iteration is Medium) then (HMR is Medium) (1) 3. If (Iteration is High) then (HMR is High) (1)
--

Fig. 6 Rules for fuzzy system FHS

5 Simulation Results

In this section the comparison of the Harmony Search algorithm is made against the proposed Fuzzy Harmony search [13]. In each of the algorithms 6 Benchmark mathematical functions were considered separately and all functions the global minimum is 0 can be appreciated in Fig. 7, a dimension of 10 variables was used with 30 runs for each function varying the parameters of the algorithms.

The parameters used in the HS are:

- Size solution harmonies: 4–40 Harmonies.
- Harmony memory accepting: 0.95.
- Pitch adjustment: 0.7.
- Pitch range: 250.

The parameters used in the Fuzzy Harmony Search are:

- Size solution harmonies: 4–40 Harmonies.
- Harmony memory accepting: Dynamic.
- Pitch adjustment: 0.7.
- Pitch range: 250.

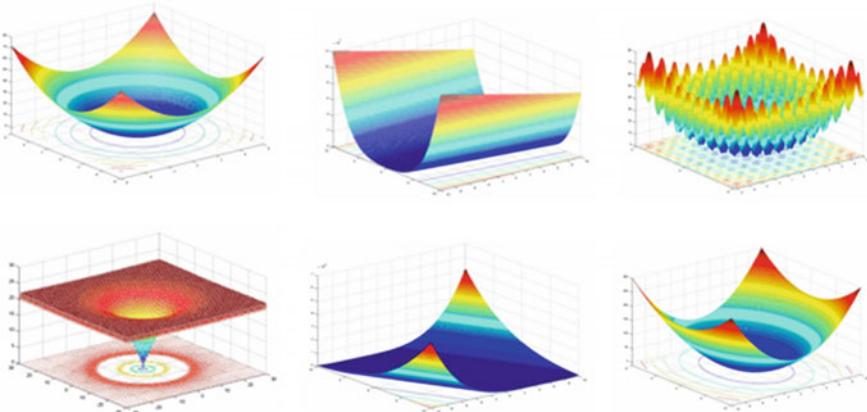


Fig. 7 Benchmark mathematical functions

5.1 Simulation Results Comparison Harmony Search Algorithm with Fuzzy Harmony Search

In this section we show the experimental results obtained by the Harmony Search algorithm and fuzzy harmony search, in separate tables of the mathematical functions. Table 1 shows the simulation results for the Spherical function.

From Table 1 it can be appreciated that after executing the HS and FHS Algorithms for 30 runs, with different parameters, we can find the average performance of each algorithm for the spherical function. Table 2 shows the simulation results for the Rosenbrock function.

From Table 2 it can be appreciated that after executing the HS and FHS Algorithms for 30 runs, with different parameters, we can find the average performance of each algorithm for the Rosenbrock function. Table 3 shows the simulation results for the Rastrigin function.

From Table 3 it can be appreciated that after executing the HS and FHS Algorithms for 30 runs, with different parameters, we can find the average performance of each algorithm for the Rastrigin function. Table 4 shows the simulation results for the Ackley function.

Table 1 Simulation results for the spherical function

Number of harmonies	Simple HS	Fuzzy HS
4	7.63E-05	6.71E-05
5	7.93E-05	7.28E-05
10	8.83E-05	7.05E-05
20	4.77E-05	2.60E-05
30	5.00E-05	3.88E-05
40	5.24E-05	4.17E-05

Table 2 Simulation results for the Rosenbrock function

Number of harmonies	Simple HS	Fuzzy HS
4	5.79E-08	1.30E-08
5	5.21E-08	1.18E-08
10	4.75E-08	2.48E-08
20	3.87E-05	4.22E-08
30	1.12E-03	1.08E-07
40	1.76E-03	3.60E-06

Table 3 Simulation results for the Rastrigin function

Number of harmonies	Simple HS	Fuzzy HS
4	1.39E-08	1.02E-08
5	1.26E-08	8.61E-09
10	3.24E-08	1.72E-08
20	1.72E-08	9.14E-09
30	3.21E-08	2.36E-08
40	2.75E-08	1.26E-08

Table 4 Simulation results for the Ackley function

Number of harmonies	Simple HS	Fuzzy HS
4	1.23E-04	9.90E-05
5	9.70E-05	5.14E-05
10	1.40E-04	8.86E-05
20	9.40E-05	7.08E-05
30	1.36E-04	9.77E-05
40	1.17E-04	6.45E-05

From Table 4 it can be appreciated that after executing the HS and FHS Algorithms for 30 runs, with different parameters, we can find the average performance of each algorithm for the Ackley function. Table 5 shows the simulation results for the Zakharov function.

From Table 5 it can be appreciated that after executing the HS and FHS Algorithms for 30 runs, with different parameters, we can find the average performance of each algorithm for the Zakharov function. Table 6 shows the simulation results for the Sum Square function.

Table 5 Simulation results for the Zakharov function

Number of harmonies	Simple HS	Fuzzy HS
4	1.95E-04	1.71E-04
5	2.22E-04	1.65E-04
10	2.72E-04	2.04E-04
20	1.92E-04	1.39E-04
30	2.25E-04	7.13E-05
40	5.21E-04	1.07E-06

Table 6 Simulation results for the Sum Square Function

Number of harmonies	Simple HS	Fuzzy HS
4	2.51E-05	1.19E-05
5	2.39E-05	8.03E-07
10	6.44E-05	1.26E-06
20	2.91E-05	4.28E-06
30	3.51E-05	2.66E-06
40	3.79E-05	1.41E-06

Table 7 Comparison between HS and FHS

Function	Simple HS	Fuzzy HS
Spherical	4.88E-04	6.33E-07
Rosenbrock	2.26E-08	1.36E-08
Rastrigin	1.18E-04	7.87E-05
Ackley	6.57E-05	5.28E-05
Zakharov	2.71E-04	1.25E-04
Sum square	3.59E-05	3.72E-06

From Table 6 it can be appreciated that after executing the HS and FHS Algorithms for 30 runs, with different parameters, we can find the average performance of each algorithm for the Sum Square function.

From Table 7 shows the results we obtained for the overall average in each function.

6 Conclusions

The harmony search algorithm is a search technique for solving optimization problems, in this case was applied fuzzy system to dynamically adapt the parameter of harmony memory, and was applied to 6 benchmark mathematical functions to validate its efficiency.

The analysis of simulation results between the HS and FHS methods considered in this work, lead us to the conclusion that for the optimization of the reference functions, the FHS method is a good alternative cause is easier for optimize. We can tentatively conclude that when Fuzzy logic is applied we can get better results from Harmony search algorithm (HS) avoiding high errors, like those obtained when trial and error is used.

We conclude that dynamically adjusting parameters of an optimization method (in this case algorithm search of harmony), can improve the quality of results and increase the diversity of solutions to a problem in some mathematical functions but will work to get better results.

As future work we want to apply a fuzzy system for it to be responsible to dynamically adjust the parameter of pitch adjustment and will make a comparison between both methods.

Acknowledgment We would like to express our gratitude to the CONACYT and Tijuana Institute of Technology for the facilities and resources granted for the development of this research.

References

1. Dexuan, Z., Yanfeng, G., Liqun, G., Peifeng, W.: A novel global harmony search algorithm for chemical equation balancing. In Computer Design and Applications (ICCPDA), 2010 International Conference on vol. 2, pp. V2–1. IEEE (2010)
2. Eberhart, R., Kennedy, J.: A new optimizer using particle swarm theory. In: Proceedings of the Sixth International Symposium on Micro Machine and Human Science, vol. 1, pp. 39–43. IEEE (1995)
3. Geem, Z., Lee, K.: A new meta-heuristic algorithm for continuous engineering optimization harmony search theory and practice. *Comput. Methods Appl. Mech. Eng.* **194**, 3902–3933 (2004)
4. Geem, Z., Sim, K.: Parameter setting free harmony search algorithm. *Appl. Math. Comput.* **7077**, 3881–3889 (2010)
5. Geem, Z.: Harmony search algorithms for structural design optimization. In: Studies in computational intelligence, vol. 239, pp. 8–121. Springer, Heidelberg, Germany (2009)
6. Geem, Z.: Music inspired harmony search algorithm theory and applications. In: Studies in Computational Intelligence, pp. 8–121. Springer, Heidelberg, Germany (2009)
7. Hadi, M., Mehmet, A., Mashinchi, M., Pedrycz, W.: A tabu harmony search based approach to fuzzy linear regression. *IEEE Trans. Fuzzy Syst.* **19**, 432–448 (2011)
8. Mahamed, G., Mahdavi, M.: Global best harmony search. *Appl. Math. Comput.* **198**, 1–14 (2008)
9. Mahdavi, M., Fesanghary, M., Damangir, E.: An improved harmony search algorithm for solving optimization problems. *Appl. Math. Comput.* **188**, 1567–1579 (2007)
10. Manjarres, D., Torres, L., Lopez, S., DelSer, J., Bilbao, M., Salcedo, S., Geem, Z.: A survey on applications of the harmony search algorithm. *Eng. Appl. Artif. Intell.* **26**(8), 3–14 (2013)
11. Ochoa, P., Castillo, O., Soria, J.: Differential evolution with dynamic adaptation of parameters for the optimization of fuzzy controllers. In: Recent Advances on Hybrid Approaches for designing intelligent systems, pp. 275–288. Springer, Heidelberg, Germany (2013)
12. Olivas, F., Melin, P., Castillo, O., et al.: Optimal design of fuzzy classification systems using PSO with dynamic parameter adaptation through fuzzy logic, pp. 2–11. Elsevier, Amsterdam (2013)
13. Perez, J., Valdez, F., Castillo O.: Bat algorithm comparison with genetic algorithm using benchmark functions. In: Recent Advances on Hybrid Approaches for designing intelligent systems, pp. 225–237. Springer, Berlin (2013)
14. Sombra, A., Valdez, F., Melin, P., Castillo, O.: A new gravitational search algorithm using fuzzy logic to parameter adaptation. In: IEEE Congress on Evolutionary Computation, pp. 1068–1074 (2013)
15. Štefek, A.: Benchmarking of heuristic optimization methods. In: Mechatronika 14th International Symposium, pp 68–71, IEEE (2011)
16. Valdez, F., Melin, P., Castillo, O.: Fuzzy control of parameters to dynamically adapt the PSO and GA algorithms. In: Fuzzy Systems International Conference, pp. 1–8. Barcelona, Spain (2010)
17. Wang, C., Huang, Y.: Self adaptive harmony search algorithm for optimization. *Expert Syst. Appl.* **37**, 2826–2837 (2010)
18. Yang, X.: Nature Inspired Metaheuristic Algorithms, pp. 73–76, 2nd edn. University of Cambridge, United Kingdom (2010)

A New Algorithm Based in the Smart Behavior of the Bees for the Design of Mamdani-Style Fuzzy Controllers Using Complex Non-linear Plants

Leticia Amador-Angulo and Oscar Castillo

Abstract A study of the behavior and evaluation of the Bee Colony Optimization algorithm (BCO) for the Mamdani Fuzzy Controllers design is presented in this paper. The Bee Colony Optimization meta-heuristic belongs to the class of Nature-Inspired Algorithms. The main objective of the work is based on the main reasons for the optimization of the design of the Mamdani Fuzzy Controllers, specifically in tuning membership functions of the fuzzy controllers for the benchmark problems known as the water tank and the temperature controller. Simulations results confirmed that using the BCO to optimize the membership functions and the scaling gains of the fuzzy system improved the controller performance. The usual five metrics of the ITAE, ITSE, IAE, ISE and MSE for the errors in control are implemented.

1 Introduction

In recent years, many works have been done on control system stabilization. However, all these control design methods require the exact mathematical model of the control systems which may not be available in practice. On the other hand, fuzzy logic control has been successfully employed for solving many nonlinear control problems. In this research, we used two benchmark problems in the area of Control to test the proposal approach. Fuzzy control systems combine information of human experts (natural language) with measurements and mathematical models. Fuzzy systems transform the knowledge base in a mathematical formulation that has proven to be very efficient [1, 2].

L. Amador-Angulo · O. Castillo (✉)
Tijuana Institute of Technology, Tijuana, Mexico
e-mail: ocastillo@tectijuana.mx

L. Amador-Angulo
e-mail: leticia.amadorangulo@yahoo.com.mx

The nonlinear characteristics of ill-defined and complex modern plants make classical controllers inadequate for such systems. However, using Fuzzy sets and fuzzy logic principles has enabled researchers to understand better, and hence control, complex systems that are difficult to model. These newly developed fuzzy logic controllers have given control system a certain degree of intelligence [5, 19, 23].

Many intelligent optimization techniques such as Ant Colony Optimization [18], and Particle Swarm Optimization [8] have been proposed to tune Fuzzy Controllers, however; the BCO algorithm is a new technique for solve complex problems.

The BCO algorithm mimics the food foraging behavior of swarms of honey bees [21]. Honey bees use several mechanisms like waggle dance to optimally locate food sources and search new ones. It is a very simple, robust and population based stochastic optimization algorithm [13].

The motivation of the practical use of the theoretic results obtained from the proposed method is the automatic optimization of the MFs in inputs and outputs for the FLC by BCO. Without trial and error and designer's experience, the FLC can be automatically tuned.

This paper focuses on tuning two Mamdani Fuzzy Controllers for benchmark problems using the Bee Colony Optimization. Section 2 presents the background. Section 3 presents the cases studies, indicating the two benchmark problems analyzed. Section 4 represents the description of the Bee colony Optimization and its implementation in this paper. Section 5 shows the results of the simulation in the model. Finally, Sect. 6 offers the conclusion and future works.

2 Background

2.1 Fuzzy Logic System

A fuzzy logic system (FLS) that is defined entirely in terms of Type-1 fuzzy sets, is known as Type-1 Fuzzy Logic System (Type-1 FLS), its elements are defined in the following Fig. 1 [10, 26, 27, 29].

A fuzzy set in the universe U is characterized by a membership function $u_A(x)$ taking values on the interval $[0,1]$ and can be represented as a set of ordered pairs of an element and the membership value to the set:

$$A = \{(x, u_A(x)) | x \in U\} \quad (1)$$

A variety of types of membership functions exist, but two of them are typically known as the Trapezoidal and Triangular, and both types of membership functions are implemented in this research [27]. The trapezoidal is a function of a variable, x , and depends on four scalar parameters a , b , c , and d , as given by (2):

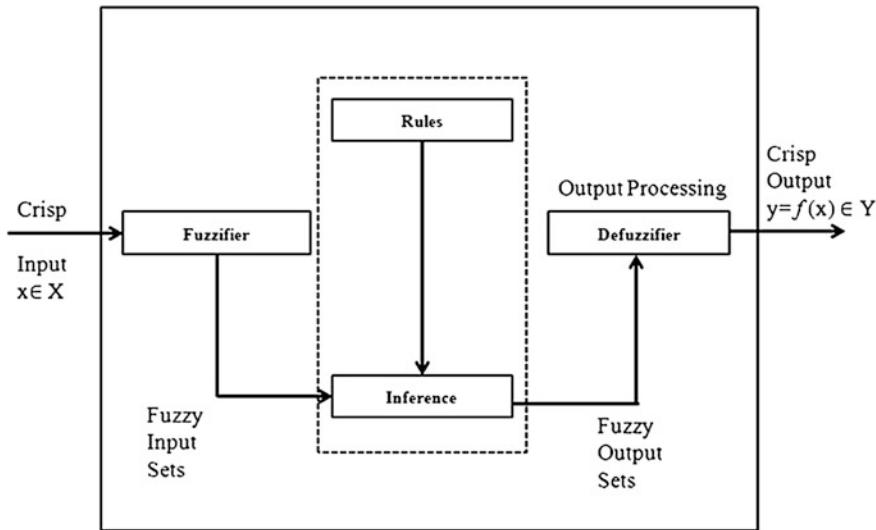


Fig. 1 Architecture of a type-1 fuzzy logic system

$$f(x; a, b, c, d) = \max \left(\min \left(\frac{x-a}{b-a}, 1, \frac{d-x}{d-c}, 0 \right) \right) \quad (2)$$

The parameters a and d locate the “feet” of the trapezoid and the parameters b and c locate the “shoulders” [27]. The triangular is a function of a vector, x , and depends on three scalar parameters a , b , and c , as given by (3):

$$f(x; a, b, c) = \max \left(\min \left(\frac{x-a}{b-a}, \frac{c-x}{c-d}, 0 \right) \right) \quad (3)$$

The parameters a and c locate the “feet” of the triangle and the parameter b locates the peak. An example of the two types the membership functions is shown in Fig. 2.

The design of each of the fuzzy logic controllers implemented in this research is described in the Sect. 3.

2.2 Fuzzy Logic Controller

Mamdani’s fuzzy inference method is the most commonly used fuzzy methodology. Mamdani’s method was among the first control systems built using fuzzy set theory. It was proposed in 1975 by Ebrahim Mamdani as an attempt to control a steam engine and boiler combination by synthesizing a set of linguistic control rules

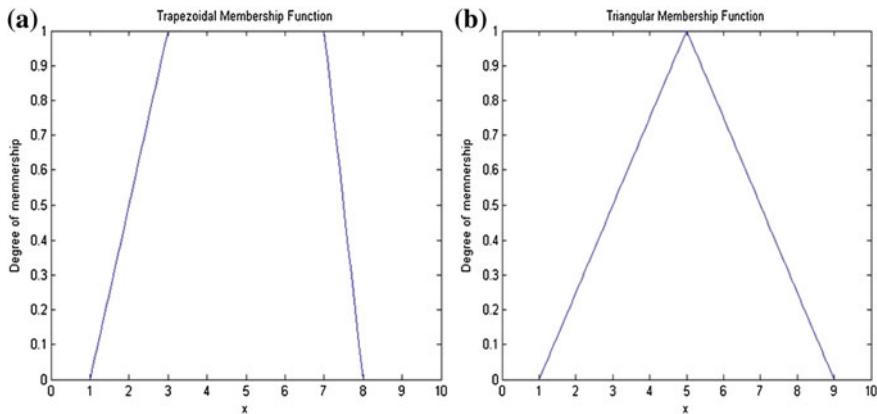


Fig. 2 Trapezoidal membership function in (a) and triangular membership function in (b)

obtained from experienced human operators. Mamdani's effort was based on Lotfi Zadeh's 1973 paper on fuzzy algorithm for complex systems and decision processes [16, 28]. Although the inference process, the model is evaluated for each of the two benchmark problems to find a fuzzy controller design with the best results in the minimization of the error.

The whole principle of automatic control systems is the application of the concept of the “feedback” shown in Fig. 3, which is a measurement taken from the process that delivers information on the current state of the variable to be controlled, as special feature is the central controller of keeping informed of the status of the variables to generate corrective actions when so required. The automatic control is maintaining a desired value for a quantity of physical condition, measuring its current value, comparing it with the reference value, and uses the difference to reduce to proceed with corrective action. Consequently, automatic control requires a closed loop action and reaction that works without human intervention [25].

The most important element of any system of automatic control is the feedback control loop, which is nothing more than a closed path formed by a sensor, a controller, and a final control element [25].

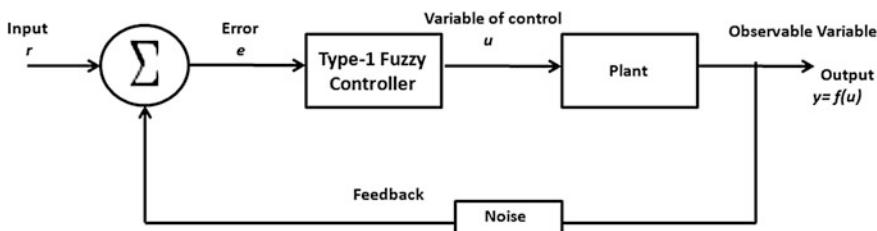


Fig. 3 Graphical representation of fuzzy controller

The BCO meta-heuristic was proposed quite recently by *Lučić* and *Teodorović*. The BCO is inspired by foraging behavior by honeybees. Is it analyzing the collective intelligent behavior that bees have for the solution of optimization problems, in this research a bee represents a solution within the search space [4, 13, 14, 20].

In this paper the optimization of the parameter values of membership functions in inputs and outputs is made with the implementation of fuzzy systems obtained with the fuzzy logic system with BCO for two benchmark problems, which are the water tank and temperature controller.

3 Bee Colony Optimization Algorithm

Many species in nature are characterized by swarm behavior. The bees have a collective movement that responds quickly to changes in the direction and speed of their neighbors. Communication between social individuals in a colony has been well known. The communication systems between insects contribute to the configuration of the “collective intelligence” of the social insects colonies. Swarm Intelligence [12, 22] is a part of the Artificial Intelligence based on study of actions of the individual in various decentralized systems, for this research we focus in the virtual (artificial) Multi Agent Systems [6, 22].

In nature a bee has the collective intelligence on the location of the food. A bee is able to leave a trail on the way in which find food [3, 7, 9, 11, 24]. Graphically shown in Fig. 4 is the representation of the smart mechanism that bees use to forage for locating the food.

Early versions of the algorithm were imitating the behavior the bees in the nature to a large extent. These versions were characterized by the *scout bees*, an important role of the hive location, and recruiting process that is more like of the natural one than it is the case in the current version of the algorithm. In this section we describe



Fig. 4 Representation of the smart intelligent of the bees

in details the current version and we will point out the different representation the BCO in Mamdani-style Fuzzy Controller.

The population of agents (artificial bees) consisting of bees collaboratively searches for the optimal solution. Every artificial bee generates one solution to the problem. The algorithm is divided into a forward pass and a backward pass constituting a single step in the BCO. In each forward pass, every artificial bee explores the search space. In this phase it applies a predefined number of moves, which construct and/or improve the solution, yielding a new solution. For example, let bees Bee 1, Bee 2.... Bee n participates in the decisions-making process on n entities. The possible situation after third forward pass is illustrated on Fig. 5.

In BCO algorithm, the hive of artificial bees contains two groups of bees, which are scout and employed bees. The scout bees have the responsibility of finding a new honey source. The responsibility of employed bees is to determine a honey in their memories and share their information with other bees within the hive [15, 19]. The best bee n is chosen according to their fitness function, for this research, the function is replaced by the resulting of the MSE in the simulation of the plant (Fig. 6).

According to the main idea in the current version of the BCO algorithm, the hive is an artificial object, without precise location and does not influence the algorithm execution. It is used only to denote the synchronization points at which bees are exchanging information about the current state of the search. The BCO algorithm is based on Eqs. (5)–(8):

$$p_{ij,n} = \frac{[\rho_{ij,n}]^\alpha \cdot \left[\frac{1}{d_{ij}}\right]^\beta}{\sum_{j \in A_{i,n}} [\rho_{ij,n}]^\alpha \cdot \left[\frac{1}{d_{ij}}\right]^\beta} \quad (5)$$

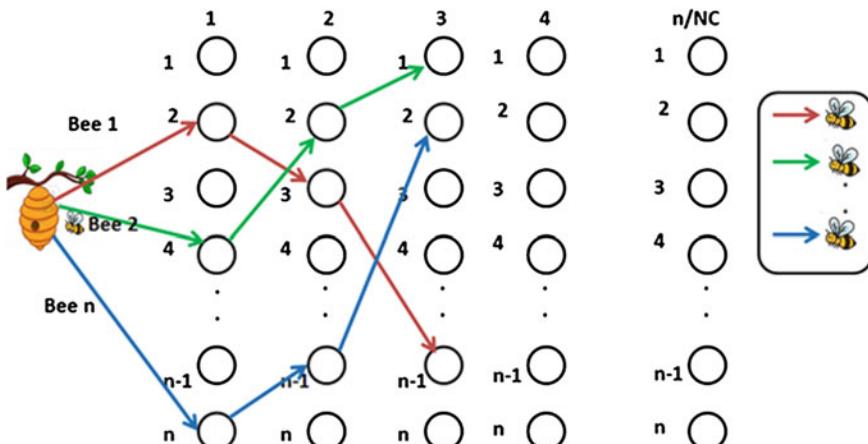
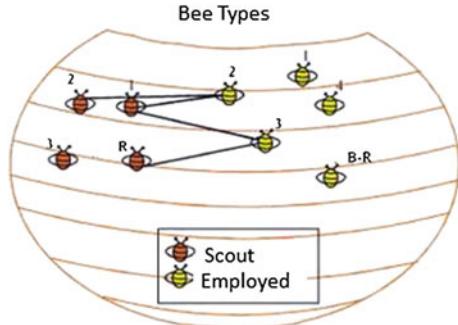


Fig. 5 An illustration of the third forward pass

Fig. 6 BCO bee types

$$D_i = K \cdot \frac{Pf_i}{Pf_{colony}} \quad (6)$$

$$Pf_i = \frac{1}{L_i}, \quad L_i = \text{Tour length} \quad (7)$$

$$Pf_{colony} = \frac{1}{N_{Bee}} \sum_{i=1}^{N_{Bee}} Pf_i \quad (8)$$

Equation (5) represents the probability of a bee k located on a node i selects the next node denoted by j , where, N_i^k is the set of feasible nodes (in a neighborhood) connected to node i with respect to bee k , ρ_{ij} is the probability of to visit of follow node. Note that the β is inversely proportional to the city distance. In this case d_{ij} represents the distance of node i until node j , and for this algorithm indicates the total the dance that a bee have in this moment. Finally α is a binary variable that is used for to find better solutions in the algorithm.

Equation (6) represents that a waggle dance will last for certain duration, determined by a linear function, where K denotes the waggle dance scaling factor, Pf_i denotes the profitability scores of bee i as defines in Eq. (7) and Pf_{colony} denotes the bee colony's average profitability as in Eq. (8) and is updated after each bee completes its tour. For this research the waggle dance is represented by the mean square error that the model to find once that is done the simulation in the iteration of the algorithm.

3.1 Basic Steps of the BCO

The basic steps are shown in this sub-section.

1. Initialization: an empty solution is assigned to every bee;
2. For every bee: //the forward pass

- a) Set $k = 1$; //counter for constructive moves in the forward pass;
 - b) Evaluate all possible constructive moves;
 - c) According to evaluation, choose one move using the roulette wheel;
 - d) $k = k+1$; if $k \leq NC$ goto step b.
3. All bees are back to the hive; //backward pass starts.
 4. Evaluate (partial) objective function value for each bee;
 5. Every bee decides randomly whether to continue its own exploration and become a recruiter, or to become a follower;
 6. For every follower, choose a new solution from recruiters by the roulette wheel;
 7. If solutions are not completed goto step 2;
 8. Evaluate all solutions and find the best one;
 9. If stopping condition is not met goto step 2;
 10. Output the best solution found.

Where:

B Represents the number of bees in the hive

NC Represents the number of constructive moves during one forward pass.

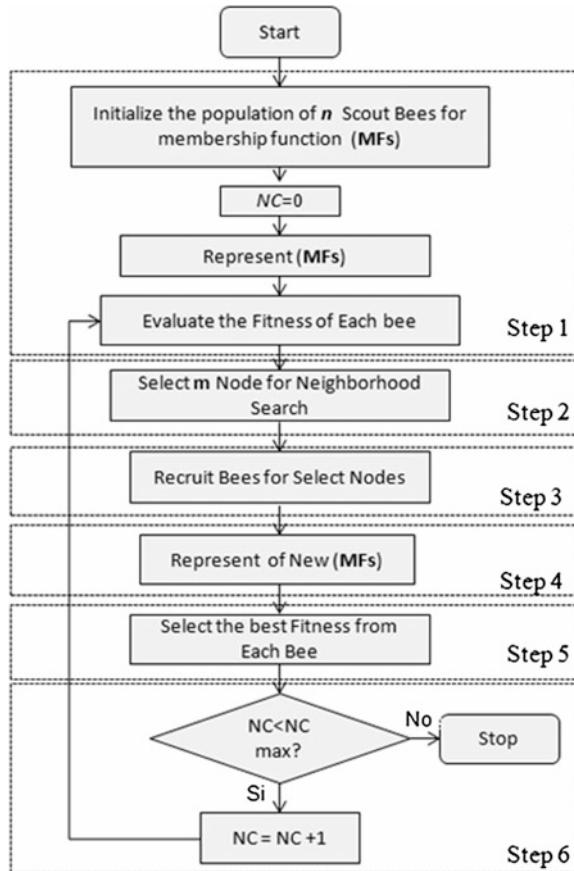
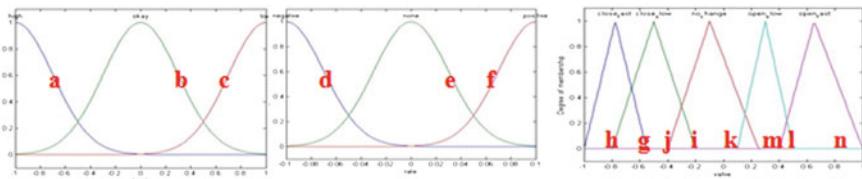
3.2 Flowchart of the BCO

The procedure of the BCO algorithm for tuning the Fuzzy Controller parameters is shown in Fig. 7.

We performed the adaptation of the algorithm for the solution of benchmark problems. Figure 8 shows the 14 parameters that the BCO algorithm searches of the tuning of the design of the membership functions of inputs and output of the Mamdani Fuzzy Controller of the water tank. For the fuzzy system of the temperature controller of each bee size is 26 and 14 for the tank controller.

4 Cases Studies

For the evaluation of the type-1 fuzzy logic controller we used two benchmark problems. The benchmark problems are mathematical models that allow us to analyze the behavior of the fuzzy controller. We describe each of the benchmark problems and the implemented fuzzy controller.

**Fig. 7** General flowchart of the BCO

```
B1 = [Vect(i,1) -1 Vect(i,2) 0 Vect(i,3) 1 Vect(i,4) -0.1 Vect(i,5) 0 Vect(i,6) 0.1.....];
```

SIZE = 14

Fig. 8 Representation of the BCO in the optimization of the fuzzy controllers

4.1 Description of the Benchmark Problems

The problems to be studied are known as the water tank controller and the temperature controller. The first benchmark problem aims at controlling the water level in a tank; therefore, one has to know the actual water level in the tank and with it has to be able to set the valve at the correct position. The second problem to be considered is the temperature controller, which aims to establish the temperature to simulate the behavior a water regulator. The graphical representation for both case studies is shown in Fig. 9.

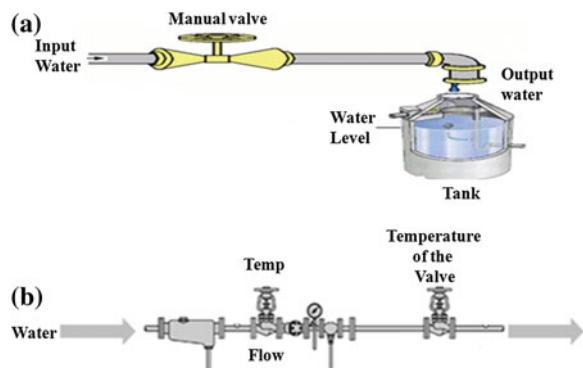
4.2 Description of the Mamdani Fuzzy Controllers

The Mamdani fuzzy controller that represents the benchmark problem known as the water tank consists of two inputs called *level* and *rate*, its memberships functions are shown in Fig. 10. It has one output called *valve* shown in Fig. 11, respectively.

The knowledge about the problem provides us with five rules, which are detailed below:

- If (*level* is *okay*) then (*valve* is *nochange*).
- If (*level* is *low*) then (*valve* is *openfast*).
- If (*level* is *high*) then (*valve* is *closefast*).
- If (*level* is *okay*) and (*rate* is *positive*) then (*valve* is *closeslow*).
- If (*level* is *okay*) and (*rate* is *negative*) then (*valve* is *openslow*).

Fig. 9 Graphical representation for the two benchmark problems. **a** Water tank controller.
b Temperature controller



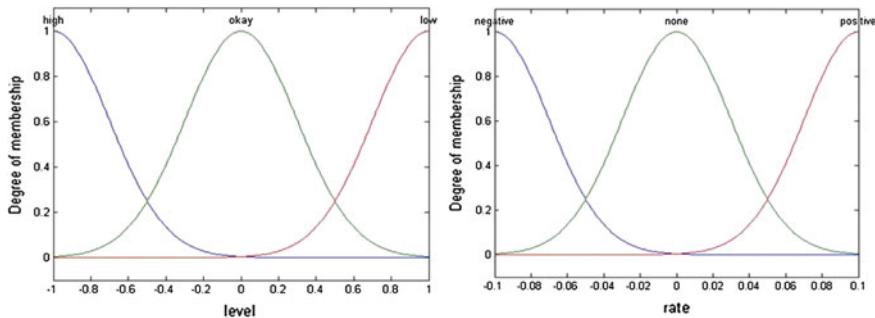
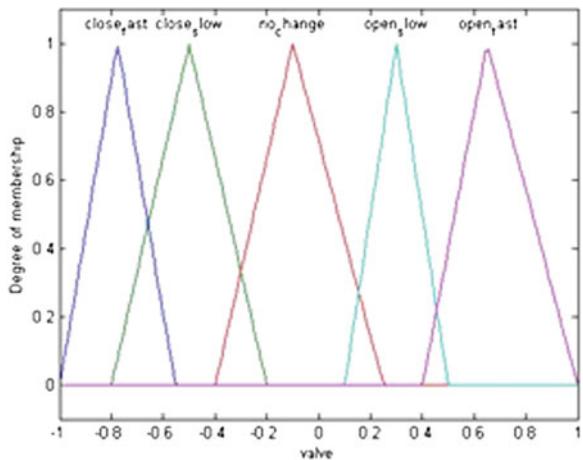


Fig. 10 Inputs of the Mamdani-style fuzzy controller for the water tank controller

Fig. 11 Output of the Mamdani fuzzy controller for the water tank controller



The combination of rules was taken from experimental knowledge according to how the process of control is performed in a water tank. We start with 5 rules to visualize the behavior of the type-1 fuzzy logic controller.

The metric on which the evaluation is being made for the fuzzy controller is by using the mean square error for measuring the behavior of the controller with respect to the reference and the error tends to be 0 (zero). The Mean Square Error (MSE) is the sum of the variance and the squared deviation of the estimator (reference) [8, 18]. The mathematical definition is presented in the following equation.

$$MSE = \frac{1}{n} \sum_{i=1}^n (\bar{Y}_i - Y_i)^2 \quad (9)$$

The test criteria consists of a series of Performance Indices; where ISE, IAE, ITSE, and ITAE were used [17], respectively shown in Eqs. (10–13).

$$ISE = \int_0^{\infty} e^2(t) dt \quad (10)$$

$$IAE = \int_0^{\infty} |e(t)| dt \quad (11)$$

$$ITSE = \int_0^{\infty} e^2(t) t dt \quad (12)$$

$$ITAE = \int_0^{\infty} |e(t)| t dt \quad (13)$$

The characteristics of the second case study are 2 inputs and two outputs, called temp, flow, cold and hot respectively. It has 9 rules, and its memberships functions are Trapezoidal and Triangulars. The Structure of the Fuzzy Controller is shown in Fig. 12.

The inference is determined with the following rules:

1. If (temp is cold) and (flow is soft) then (cold is openslow)(hot is openfast)
2. If (temp is cold) and (flow is good) then (cold is closeslow)(hot is openslow)
3. If (temp is cold) and (flow is hard) then (cold is closefast)(hot is closeslow)



Fig. 12 Structure of the fuzzy controller for the temperature controller

4. If (temp is good) and (flow is soft) then (cold is openslow)(hot is openslow)
5. If (temp is good) and (flow is good) then (cold is steady)(hot is steady)
6. If (temp is good) and (flow is hard) then (cold is closeslow)(hot is closeslow)
7. If (temp is hot) and (flow is soft) then (cold is openfast)(hot is openslow)
8. If (temp is hot) and (flow is good) then (cold is openslow)(hot is closeslow)
9. If (temp is hot) and (flow is hard) then (cold is closeslow)(hot is closefast)

5 Simulations Results

All simulations were performed in the Simulink application [17]. In Fig. 13 the block diagram of the water controller in (a) and temperature controller in (b) are shown.

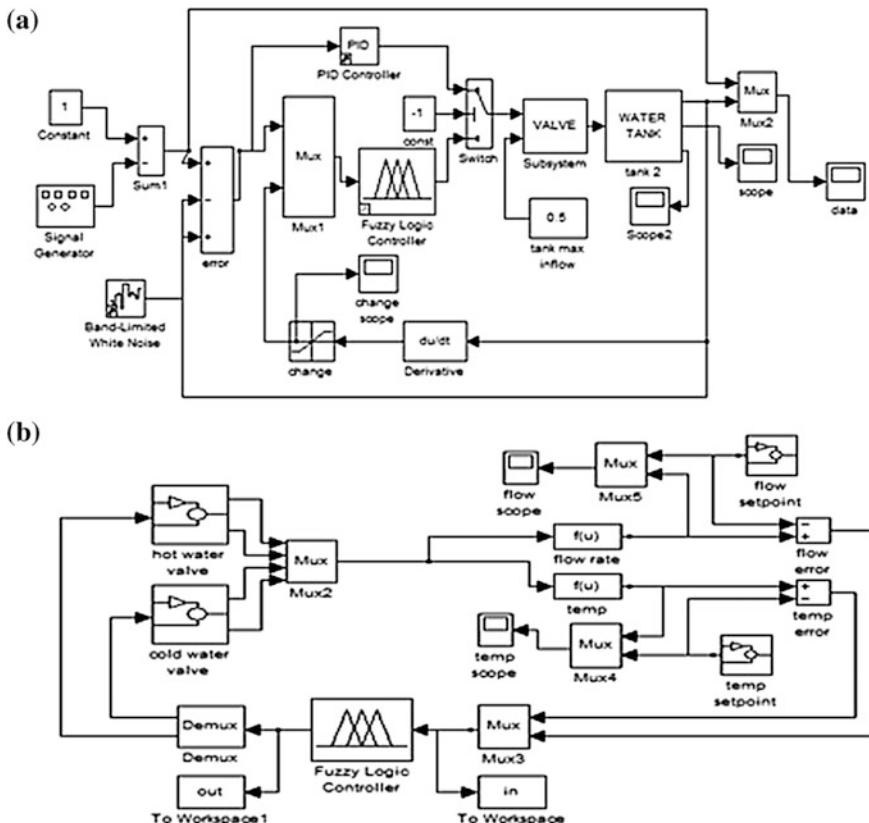


Fig. 13 Evaluation of control diagram with type-1 fuzzy inference system

Table 1 Results of the fuzzy controller of the water tank using BCO

#	Population	Iterations	Follower bees	Alpha (α)	Beta (β)	Average error	BCO time (min.)
1	90	20	10	0.4	0.5	0.080920	26:24
2	80	20	10	0.4	0.5	0.076116	06:57
3	70	25	20	0.4	0.5	0.072430	18:23
4	60	30	25	0.4	0.5	0.07576	21:36
5	50	30	30	0.4	0.5	0.080448	23:14
6	70	25	15	0.4	0.5	0.098945	12:43
7	70	25	20	0.4	0.5	0.075186	15:04
8	70	25	25	0.4	0.5	0.084104	18:57
9	70	25	30	0.5	0.6	0.072270	31:52
10	90	30	60	0.5	0.6	0.070211	58:22
11	90	30	20	0.5	0.6	0.068650	39:25
12	90	20	80	0.5	0.6	0.072144	19:05
13	80	20	80	0.5	0.6	0.068682	01:45
14	80	25	85	0.5	0.6	0.072575	26:54
15	70	20	70	0.5	0.6	0.071754	26:41

Simulation for 15 experiments have also been performed with different characteristics of the algorithm, by changing the size of population, number of iterations, number of follower bees, values of *alpha* and *beta*, and the experiments are presented in Table 1.

Table 1 shows the results in which the computational time increases when there are more iterations and the number of follower bees in the search space is increased, experiments were performed by changing the value of the follower bees and size of population and iterations, thereby obtaining the experiment 11 is the one with the minimum error for all testing and the experiment 6 is the one with the maxime error.

Figure 14 shows on top the inputs of fuzzy controller and below the output of the Mamdani Fuzzy Controller that the BCO algorithm found as the best of all experiments performed, the membership functions are Gaussian because they were the ones that showed the best error of simulation.

Simulating the fuzzy controller shown in Fig. 15, we are representing the yellow color the reference value and control the behavior pink optimized fuzzy controller, the simulation was performed in 100 iterations.

In Fig. 16 we can note that when applying the optimization with BCO the behavior of the algorithm tuning the error simulations, because the design of the

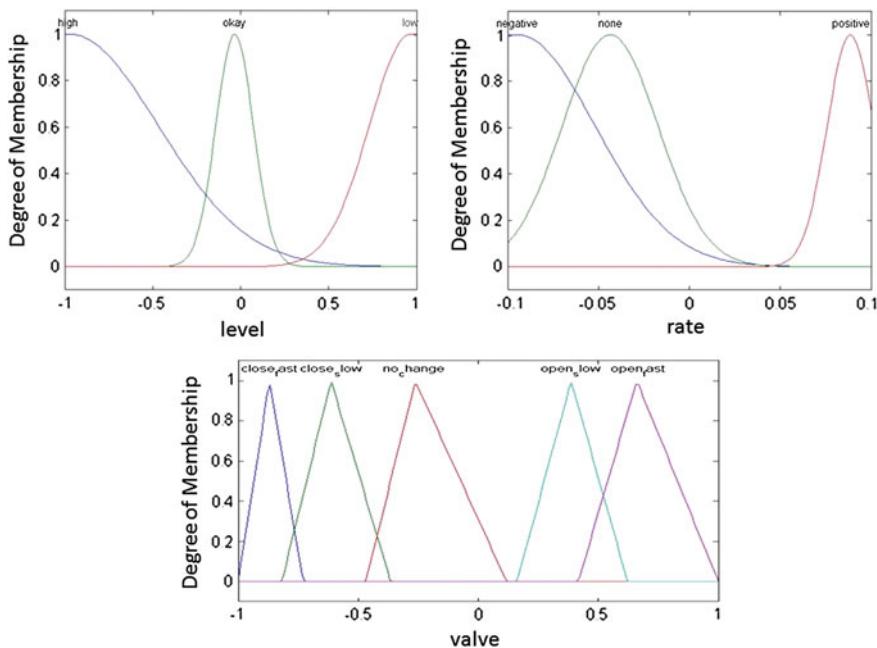


Fig. 14 Distribution of memberships functions of the Mamdani fuzzy controller found by the BCO

Fuzzy Logic System in the simulations model found is better. With the BCO algorithm the fuzzy controller has a better stability in the simulation.

The BCO convergence of the algorithm is presented in Fig. 17, showing the behavior of the control reference to the behavior of the algorithm; this was performed for 30 iterations in the algorithm.

With the same methodology, we implemented the optimization of the Fuzzy Controller of the second benchmark problem called the temperature controller using BCO. The fuzzy system for the temperature controller has two outputs in the diagram block called flow and temp, for the experiments made, we using the scope flow for the calculated of the Mean Square Error (MSE), in Table 2 results are presented.

Table 2 shows the simulation results in which with 25 iterations and 70 bees of population the best is found, that is to say, in experiment six with the same model the MSE the error found is **0.050901**.

Figure 17 shows above the inputs of the fuzzy controller and below the outputs of the Mamdani Fuzzy Controller that the BCO algorithm found as the best of all performed experiments.

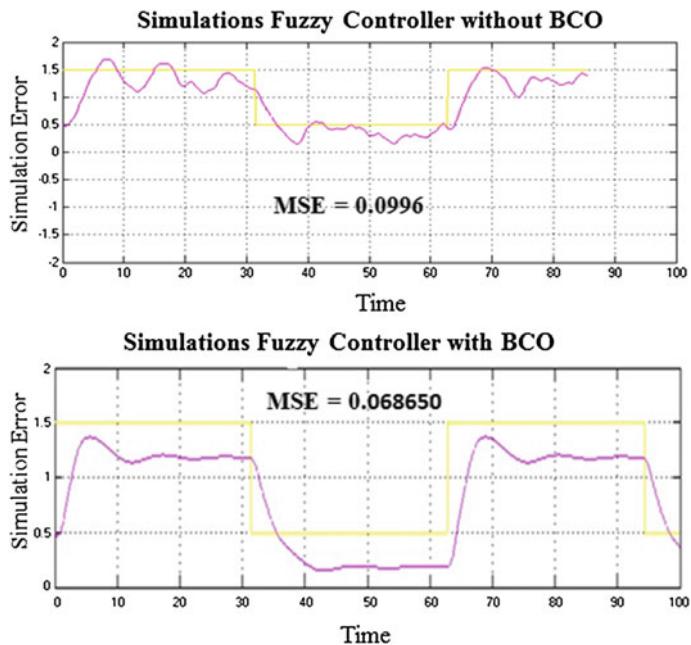


Fig. 15 Simulation model using type-1 fuzzy controller optimization with BCO

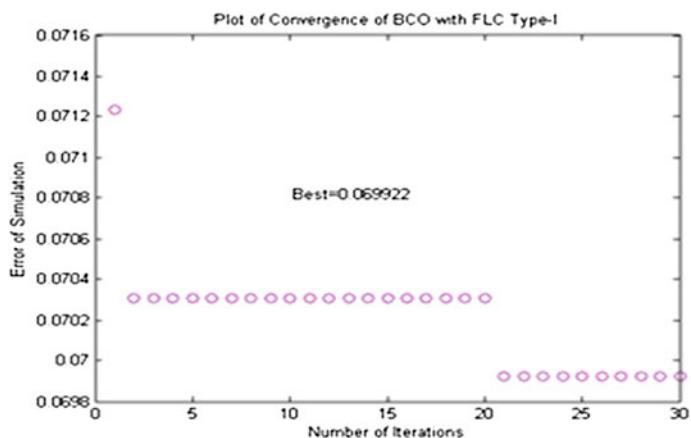


Fig. 16 Optimization behavior for the BCO on type-1 FLC

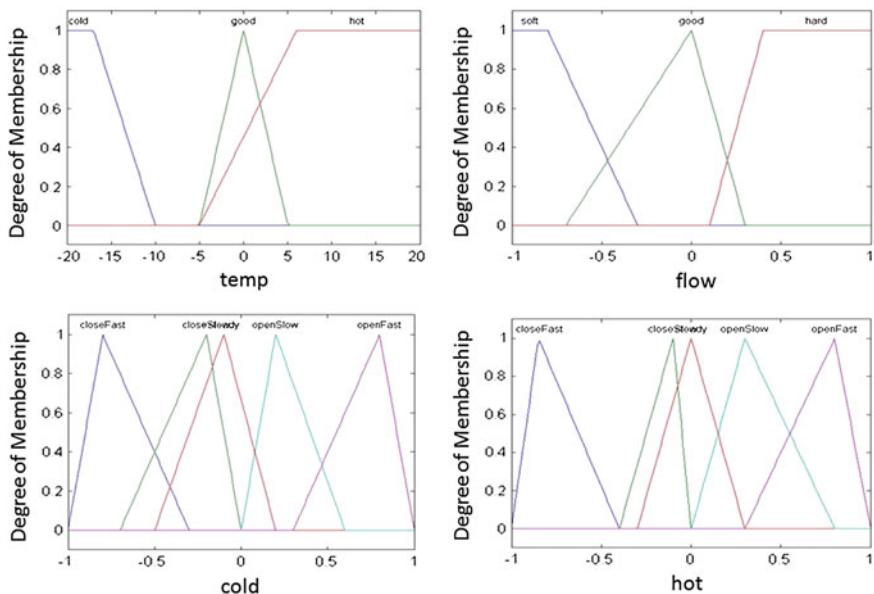


Fig. 17 Distribution of memberships functions of the Mamdani fuzzy controller found by the BCO

Table 2 Results of the fuzzy controller of the temperature controller using BCO

#	Population	Iterations	Follower bees	Alpha (α)	Beta (β)	Average error	BCO time (min.)
1	90	20	10	0.4	0.5	0.078910	36:24
2	80	20	10	0.4	0.5	0.078116	20:58
3	70	25	20	0.4	0.5	0.079020	28:21
4	60	30	25	0.4	0.5	0.075761	21:36
5	50	30	30	0.4	0.5	0.081428	26:14
6	70	25	15	0.4	0.5	0.050901	22:43
7	70	25	20	0.4	0.5	0.075186	19:04
8	70	25	25	0.4	0.5	0.074987	28:37
9	70	25	30	0.5	0.6	0.064571	31:52
10	90	30	60	0.5	0.6	0.070011	58:22
11	90	25	70	0.5	0.6	0.069650	32:05
12	90	20	80	0.5	0.6	0.071146	29:05
13	80	20	80	0.5	0.6	0.072270	31:42
14	80	25	85	0.5	0.6	0.070175	25:34
15	70	20	70	0.5	0.6	0.069724	46:41

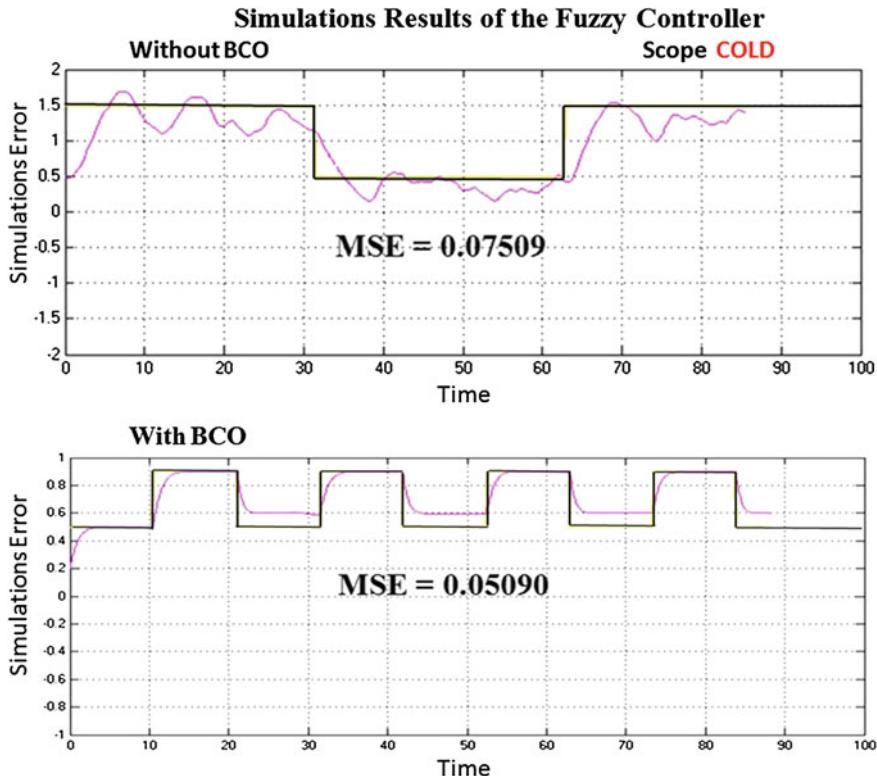


Fig. 18 Simulation model using fuzzy controller optimization for the output flow

Simulating the fuzzy controller shown in Fig. 18 we were representing the yellow color the reference value and control the behavior pink without optimizing fuzzy controller, the simulation was performed in 100 iterations.

In Figs. 18 and 19 we can note that when applying the optimization with BCO the behavior of the algorithm tuning the error simulations, because the design of the Fuzzy Logic System in the simulations model found is better. The behavior of the fuzzy controller with the design of the fuzzy logic system found for the BCO is closer to the reference, and therefore, the simulations error is lower.

We implemented others four metric for the evaluate the model in the fuzzy controller, the best experiment for each fuzzy controller whit BCO are shown in Table 3, where the Performance Indices are calculated with each experiment and the error of simulation is using the BCO.

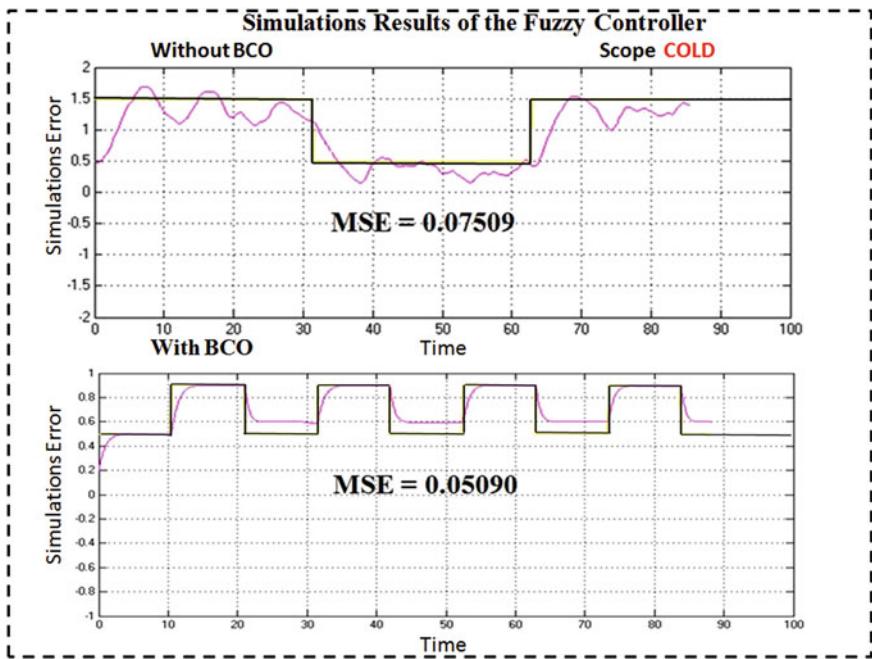


Fig. 19 Simulation model using fuzzy controller optimization with BCO

Table 3 Performance index results for the two fuzzy controllers using BCO

Metric	Without optimizing			With BCO		
	Tank	Temperature controller		Tank	Temperature controller	
	Valve	Cold	Hot	Valve	Cold	Hot
ISE	150.5	51.69	1.851e+004	134.6	51.67	1.106e+004
IAE	112.8	69.07	1312	109.5	69.05	1441
ITSE	7165	2668	7.417e+005	7148	2660	1.065e+006
ITAE	5458	3513	5.999e+004	4985	3512	2.274e+004
MSE	0.096	0.075	0.08129	0.0686	0.0509	0.0408

6 Conclusions and Future Work

The results show that performing the optimization of the parameters of the membership functions in the Mamdani fuzzy controllers with the BCO algorithm indicate that the error is minimized in the simulation, thus the BCO algorithm is considered an efficient optimization technique for the design of Mamdani Fuzzy Controllers. We observed with the simulations and experiments when we increase

the size of the follower bees and decrease the value of the population in the algorithm to find the better errors. This is because the algorithm performs better find local optimum exploitation.

In the future, we could considerer include changing the parameters of BCO algorithm such as; values of *alpha*, *beta*, *population size* and number of *followers bees* to thereby find better designs fuzzy controllers. An interesting aspect is to perform the dynamic adjustment of these parameters with the help of fuzzy logic. Finally, we look forward to improve the Intervals type-2 fuzzy logic system with BCO algorithm and it use in other areas of application.

References

1. Amador-Angulo, L., Castillo, O.: Comparison of fuzzy controllers for the water tank with type-1 and type-2 fuzzy logic. In: Proceedings in NAFIPS, Edmonton, Canada (2013)
2. Amador-Angulo, L., Castillo, O.: Comparison of the optimal design of fuzzy controllers for the water tank using ant colony optimization, transactions on engineering technologies. In: International Multi-Conference of Engineers and Computer Scientists, pp. 1–2. Tijuana, B.C., Mexico (2013)
3. Biesmeijer, J.C., Seeley, T.D.: The use of waggle dance information by honey bees throughout their foraging careers. *Behav. Ecol. Sociobiol.* **59**(1), 133–142 (2005)
4. Bonabeau, E., Dorigo, M., Theraulaz, G.: *Swarm Intelligence*. Oxford University Press, Oxford (1997)
5. Cervantes L., Castillo, O., Melin, P.: Intelligent control of nonlinear dynamic plants using a hierarchical modular approach and type-2 fuzzy logic. In: MICAI, pp. 1–12 (2011)
6. Chong, C.S., Low, M.Y.H., Sivakumar, A.I., Gay, K.L.: A bee colony optimization algorithm to job shop scheduling. In: Proceedings of the 2006 Winter Simulation Conference, pp. 13–25 (2006)
7. Dyler, F.C.: The biology of the dance language. *Annu. Rev. Entomol.* **47**, 917–949 (2002)
8. Fierro, R., Castillo, O., Váldez, F.: Optimization of fuzzy control systems with different variants of particle swarm optimization. In: 2013 IEEE Symposium Series on Computational Intelligence, pp. 51–56 (2013)
9. von Frisch, K.: Decoding the language of the bee. *Sciences* **185**(4152), 663–668 (1974)
10. Klir, G.J., Yuan, B.: *Fuzzy Sets and Fuzzy Logic: Theory and Applications*. Prentice Hall, New Jersey (1995)
11. Karaboga, D., Basturk, B.: A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. *J. Global Optim.* **39**(3), 459–471 (2007)
12. Lučić, P., Teodorović, D.: Bee system: modeling combinatorial optimization transportation engineering problems by swarm intelligence. In: Preprints of the TRISTAN IV Triennial Symposium on Transportation Analysis, São Miguel, Azores Islands, Portugal, pp. 441–445 (2001)
13. Lučić, P., Teodorović, D.: Computing with bees; attacking complex transportation engineering problems. *Int. J. Artif. Intell. Tools* **12**(3), 375–394 (2003)
14. Lučić, P., Teodorović, D.: Transportation modeling: an artificial life approach. In: Proceedings of the 14th IEEE International Conference on Tools with Artificial Intelligence, Washington, D.C., pp. 216–223 (2002)
15. Lučić, P., Teodorović, D.: Vehicle routing problem with uncertain demand at nodes: the bee system and fuzzy logic approach. In: Verdegay, J.L. (ed.) *Fuzzy Sets in Optimization*, pp. 67–82. Springer, Berlin (2003)

16. Mamdani, E.H., Assilian, S.: An experiment in linguistic synthesis with fuzzy logic controller. *Int. J. Man Mach. Stud.* **7**, 1–13 (1975)
17. MATLAB: MATrix LABoratory, <http://www.mathworks.com/products/simulink/>
18. Naredo, E., Castillo, O.: ACO-tuning of a fuzzy controller for the ball and beam problem. In: MICAI, pp. 58–69 (2011)
19. Pham, D.T., Darwish, A.H., Eldukhri, E.E., Otri, S.: Using the bees algorithm to tune a fuzzy logic controller for a robot gymnast. *Innovative Production Machines and Systems* (online), pp. 1–2, July 2007
20. Teodorović, D., Lučić, P., Marcković, G., Dell'Orco, M.: Bee colony optimization: principles and applications. In: Reljin, B., Stanković, S. (eds.) *Proceedings of the Eight Seminar on Neural Network Applications in Electrical Engineering—NEUREL*, pp. 151–156. University of Belgrade, Belgrade (2006)
21. Teodorović, D.: Transport modeling by multi-agent systems: a swarm intelligence approach. *Transport. Plan. Techn.* **26**, 289–312 (2003)
22. Teodorović, D.: Swarm intelligence systems for transportation engineering: principles and applications. *Transp. Res. Pt. C-Emerg. Technol.* **16**, 651–782 (2008)
23. Tiacharoen, S., Chatchanayuenyong, T.: Design and development of an intelligent control by using bee colony optimization technique. *Am. J. Appl. Sci.* **9**(9), 1467 (2012)
24. Wong, L.P., Low, M.Y.H., Chong, C.S.: A Bee colony optimization algorithm for traveling salesman problem. In: *Proceedings of Second Asia International Conference on Modelling and Simulation*, pp. 818–823 (2008)
25. Yen, J., Langari, R.: *Fuzzy Logic: Intelligence, Control and Information*. Prentice Hall, New Jersey (1999)
26. Zadeh, L.A.: Fuzzy sets. *Inf. Control* **8**, 338–353 (1965)
27. Zadeh, L.A.: The concept of a linguistic variable and its application to approximate reasoning, Part I. *Inf. Sci.* **8**, 199–249 (1975)
28. Zadeh, L.A.: The concept of a linguistic variable and its application to approximate reasoning, Part II. *Inf. Sci.* **8**, 301–357 (1975)
29. Zadeh, L.A.: Toward a theory of fuzzy information granulation and its centrality in human reasoning and fuzzy logic. *Fuzzy Sets and Syst.* **90**, 117 (1997)