STUDIES IN *FUZZINESS*
AND *SOFT COMPUTING*

Patricia Melin
Oscar Castillo (Eds.)

Soft Computing Applications in Optimization, Control, and Recognition

Springer

Studies in Fuzziness and Soft Computing 294

Editor-in-Chief

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

For further volumes:
http://www.springer.com/series/2941

Patricia Melin and Oscar Castillo (Eds.)

Soft Computing Applications in Optimization, Control, and Recognition

Springer

Editors
Prof. Patricia Melin
Tijuana Institute of Technology
Graduate Program of Computer Science
Tijuana
Mexico

Prof. Oscar Castillo
Tijuana Institute of Technology
Graduate Program of Computer Science
Tijuana
Mexico

ISSN 1434-9922
e-ISSN 1860-0808
ISBN 978-3-642-35322-2
e-ISBN 978-3-642-35323-9
DOI 10.1007/978-3-642-35323-9
Springer Heidelberg New York Dordrecht London

Library of Congress Control Number: 2012953383

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

We describe in this book the application of soft computing techniques for intelligent control, pattern recognition, and optimization of complex problems. Soft Computing (SC) consists of several intelligent computing paradigms, including fuzzy logic, neural networks, and bio-inspired optimization algorithms, which can be used to produce powerful hybrid intelligent systems. The book is organized in four main parts, which contain a group of papers around a similar subject. The first part consists of papers with the main theme of nature-inspired optimization methods and their applications, which are basically papers that propose new models and concepts, which can be the basis for achieving intelligent optimization in diverse areas of application. The second part contains papers with the main theme of hybrid intelligent systems for achieving intelligent control, which are basically papers using nature-inspired techniques, like evolutionary algorithms, fuzzy logic and neural networks, for the optimal design of intelligent controllers in diverse areas of application. The third part contains papers with the theme of pattern recognition based on SC techniques, which basically consider the proposal of new methods and their applications to solve complex pattern recognition problems. The fourth part contains papers that deal with the application of intelligent optimization techniques in real world problems. The fifth part contains papers with the theme of new theoretical concepts and methods in SC, which are papers considering soft computing methods for applications related to diverse areas, such as natural language processing, clustering and optimization.

In the part of nature-inspired optimization methods there are 3 papers that describe different contributions of new algorithms for optimization and their application to diverse complex optimization problems. The nature-inspired methods include the chemical optimization paradigm, and cellular genetic algorithms. In the part of hybrid intelligent systems for control and robotics there are 3 papers that describe different contributions that propose new models and concepts, which can be the considered as the basis for achieving intelligent control and mobile robotics for real world problems. In the part of hybrid intelligent systems for pattern recognition there are 3 papers that describe different contributions on achieving efficient pattern recognition using hybrid intelligent systems based on soft computing techniques.

In the part of new theoretical concepts and methods in SC, there are 4 contributions that describe the development of new models and algorithms relevant to complex problems, such as natural language processing, clustering and optimization.

The papers in this book consist of selected and extended versions of papers presented by advanced doctoral students at the ISCI 2012 meeting held in Tijuana, Mexico in May of 2012. The papers have been accepted after a strict peer review process by a review committee of well-known experts in the respective fields. The committee has selected 13 contributions by the best doctoral students and their supervisors from various top Mexican universities.

In conclusion, the edited book comprises papers on diverse aspects of nature-inspired models, soft computing and hybrid intelligent systems for control, mobile robotics, pattern recognition, and other complex real world problems. There are covered theoretical aspects as well as applications.

September 28, 2012

Patricia Melin
Tijuana Institute of Technology
Mexico

Oscar Castillo
Tijuana Institute of Technology
Mexico

Contents

Part III: Soft Computing in Pattern Recognition Applications

Part IV: Soft Computing: Theory and New Models

Part I

Optimization Methods and Applications

Optimization of Type-2 and Type-1 Fuzzy Tracking Controllers for an Autonomous Mobile Robot under Perturbed Torques by Means of a Chemical Optimization Paradigm

Leslie Astudillo, Patricia Melin, and Oscar Castillo

Tijuana Institute of Technology, Tijuana México
{epmelin,ocastillo}@hafsamx.org

Abstract. This paper addresses the tracking problem for the dynamic model of a unicycle mobile robot. A novel optimization method inspired on the chemical reactions is applied to solve this motion problem by integrating a kinematic and a torque controller based on fuzzy logic theory. Computer simulations are presented confirming that this optimization paradigm is able to outperform other optimization techniques applied to this particular robot application.

1 Introduction

Optimization is an activity carried out in almost every aspect of our life, from planning the best route in our way back home from work to more sophisticated approximations at the stock market, or the parameter optimization for a wave solder process used in a printed circuit board assembly manufacturer optimization theory has gained importance over the last decades. From science to applied engineering (to name a few), there is always something to optimize and of course, more than one way to do it.

In a generic definition, we may say that optimization aims to find the "best" available solution among a set of potential solutions in a defined search space. For almost every problem exists a solution, not necessarily the best, but we can always find an approximation to the "ideal solution", and while in some cases or processes is still common to use our own experience to qualify a process, a part of the research community have dedicated a considerably amount of time and efforts to help find robust optimization methods for optima finding in a vast range of applications.

It has been stated the difficulty to solve different problems by applying the same methodology, and even the most robust optimization approaches may be outperformed by other optimization techniques depending on the problem to solve.

P. Melin, O. Castillo (Eds.): *Soft Computing Appli. in Optimization,* STUDFUZZ 294, pp. 3–26.
DOI: 10.1007/978-3-642-35323-9_1

When the complexity and the dimension of the search space make a problem unsolvable by a deterministic algorithm, probabilistic algorithms deal with this problem by going through a diverse set of possible solutions or candidate solutions. Many metaheuristic algorithms can be considered probabilistic, while they apply probability tools to solve a problem, metaheuristic algorithms seek good solutions by mimicking natural processes or paradigms. Most of these novel optimization paradigms inspired by nature were conceived by merely observation of an existing process and their main characteristics were embodied as computational algorithms.

The importance of the optimization theory and its application has grown in the past few decades, from the well known Genetic Algorithm paradigm to PSO, ACO, Harmonic Search, DNA Computing, among others, they all were introduced with the expectation of improving the results obtained with the existing strategies.

There's no doubt that there could be some optimization strategies presented at some point that were left behind due their complexity and poor performance. Novel optimization paradigms should be able to perform well in comparison with another optimization techniques and must be "easily adaptable" to different kinds of problems.

Optimization based on chemical processes is a growing field that has been satisfactorily applied to several problems. In [25] A DNA based algorithm was to solve the small hitting set problem. A catalytic search algorithm was explored in [30], where some physical laws such as mass and energy conservation were taken into account. In [19], the potential roles of energy in algorithmic chemistries were illustrated. An energy framework was introduced, which keeps the molecules within a reasonable length bounds, allowing the algorithm to behave thermodynamically and kinetically similar to real chemistry. A chemical reaction optimization was applied to the grid scheduling problem in [29], where molecules interact with each other aiming to reach the minimum state of free potential and kinetic energies. The main difference between these metaheuristics is the parameter representation, which can be explicit or implicit.

In this paper we introduce an optimization method inspired on the chemical reactions and its application for the optimization of the tracking controller for the dynamic model of the unicycle mobile robot.

The importance of applying this chemical optimization algorithm is that different methods have been applied to solve motion control problems. Kanayama et al. [13] propose a stable tracking control method for a non-holonomic vehicle using a Lyapunov function. Lee et al. [15] solved tracking control using backstepping and in [17] with saturation constraints. Furthermore, most reported designs rely on intelligent control approaches such as fuzzy logic control [3][12][16][23][27][28] and neural networks [10][26].

However the majority of the publications mentioned above, have concentrated on kinematic models of mobile robots, which are controlled by the velocity input, while less attention has been paid to the control problems of nonholonomic dynamic systems, where forces and torques are the true inputs: Bloch and Drakunov [4] and Chwa [8], used a sliding mode control to the tracking control problem. Fierro and Lewis [9] propose a dynamical extension that makes possible

the integration of kinematics and torque controller for a nonholonomic mobile robot. Fukao et al. [11], introduced an adaptive tracking controller for the dynamic model of mobile robot with unknown parameters using backstepping methodology, which has been recognized as a tool for solving several control problems [24] [31].Motivated by this, a mamdani fuzzy logic controller is introduced in order to drive the kinematic model to a desired trajectory in a finite-time, considering the torque as the real input, a chemical reaction optimization paradigm is applied and simulations are shown.

Further publications [2][18][6] have applied bio-inspired optimization techniques to find the parameters of the membership functions for the fuzzy tracking controller that solves the problem for the dynamic model of a unicycle mobile robot, using a fuzzy logic controller that provides the required torques to reach the desired velocity and trajectory inputs.

In this paper, the main contribution is the representation of the fuzzy controller in the chemical paradigm to search for the optimal parameters. Simulation results show that the proposed approach outperforms other nature inspired computing paradigms, such as genetic algorithms, particle swarm and ant colony optimization.

The rest of this paper is organized as follows. Section 2 illustrates the proposed methodology. Section 3 describes the problem formulation and control objective. Section 4 describes the proposed fuzzy logic controller of the robot. Section 5 shows some experimental results of the tracking controller and in section 6 some conclusions and future work are presented.

2 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. The general flowchart of the algorithm is shown in Figure 1.

The main difference with other optimization techniques [25][30][19][29] is that no external parameters are taken into account to evaluate the results, while other algorithms introduce additional parameters (kinetic/potential energies, mass conservation, thermodynamic characteristics, etc), this is a very straight forward methodology that takes the characteristics of the chemical reactions (synthesis, decomposition, substitution and double-substitution) to find for optimal solution.

This approach is a static population-based metaheuristic that applies an abstraction of the 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 every iteration. The algorithm may trigger only one reaction or all of them, depending on the nature of the problem to solve, in example; we may use only the decomposition reaction sub-routine to find the minimum value of a mathematical function.

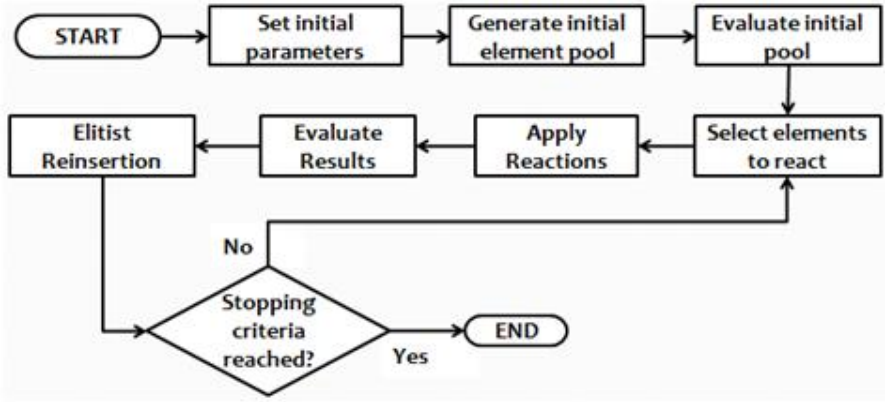


Fig. 1 General flowchart of the chemical reaction algorithm

The pseudocode for the chemical reaction algorithm is as follows:

Chemical_Reaction _Algorithm
Input: *problem_definition*, *objective_function*, *dimensions*,
1. Assign values to variables: *pool_size*, *Trials*, *upper_boundary*, *lower_boundary*, *synthesis_rate*, *decomposition_rate*, *singlesubstitution_rate*, *doublesubstitution_rate*. 2. Generate Randomly *Initial_Pool* in interval [*lower_boundary*, *upper_ boundary*] 3. Evaluate *Initial_Pool* *4.* Identify *best_solution* *5.* **while** (stopping criteria not met) **do** 6. Perform *Synthesis_Procedure*; Get *Synthesis_vector* 7. Perform *Decomposition_Procedure*; Get *Decomposition_vector* 8. Perform *SingleSubstitution_Procedure*; Get *SingleSubstitution_vector* 9. Perform *DoubleSubstitution_Procedure*; Get *DoubleSubstitution_vector* 10. Evaluate *Synthesis_vector*, *Decomposition_vector*, *SingleSubstitution_ vector*, *DoubleSubstitution_vector* 11. Apply *elitist_reinsertion*; Get *improved_pool* 12. Update *best_solution* *13.* **end while**
Output: *best_solution*

Every nature inspired paradigm has their own way to encode candidate solutions. When these parameters are defined, a set of processes or procedures are applied to lead the population to an optimal result. The main components of this chemical reaction algorithm are described below.

Elements/Compounds
These are the basic components of the algorithm. Each element or compound represents a solution within the search space. The initial definition of elements and/or compounds depends on the problem itself and can be represented as binary numbers, integer, floating, etc. They interact with each other implicitly; this is, the definition of the interaction is independent of the real molecular structure; in this approach the potential and kinetic energies and other molecular characteristics are not taken into account.

Chemical Reactions
A chemical reaction is a process in which at least one substance changes its composition and its sets of properties, in this approach, the chemical reactions behave as intensifiers (substitution, double substitution reactions) and diversifying (synthesis, decomposition reactions) mechanisms. The 4 chemical reactions considered in this approach are the *synthesis, decomposition, single and double substitution reactions.* The objective of these operators is exploring or exploiting new possible solutions within a slightly larger hypercube than the original elements/compounds, but within the previously specified range.

The *synthesis* and *decomposition* reactions are used to diversify the resulting solutions; these procedures showed to be highly effective to rapidly lead the results to a desired value. They can be described as follows.

Synthesis Reactions
Is a reaction of two reactants to produce one product. By combining two (or more) elements, this procedure allows to explore higher valued solutions within the search space. The result can be described as a compound *(B+C → BC)*. The pseudocode for the synthesis reaction procedure is as follows:

Synthesis_Procedure
Input: *selected_elements*, *synthesis_rate* 1. n = size (*selected_elements*) 2. i = floor (n / 2) 3. **for** j = 1 to i – 1 *4.* *Synthesis* = $selected\_elements_j + selected\_elements_{j+1}$ 5. $j = j + 2$ 6. **end for**
Output: *Synthesis_vector*

Decomposition Reactions
In this reaction, typically, only one reactant is given, it allows a compound to be decomposed into smaller instances *(BC → B+C)*. The pseudocode for the decomposition reaction procedure is as follows:

Decomposition_Procedure
Input: *selected_elements*, *decomposition_rate* 1. n = size (*selected_elements*) 2. Get *randval* randomly in interval [0, 1] 3. **for** i = 1 to n *4.* $Deco_1 = selected\_elements_i$ x *randval* *5.* $Deco_2 = selected\_elements_i$ x (1 – *randval*) 6. $i = i + 1$ 7. **end for**
Output: *Decomposition_vector* ($Deco_1, Deco_2$)

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.

Single-Substitution Reactions
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 more reactive than the element it replaces. A new compound and a new free element are produced; during the algorithm, a compound and an element are selected and a decomposition reaction is applied to the compound; two elements are generated from this operation. Then, one of the new generated elements is combined with the non-decomposed selected element *(*$C + AB \rightarrow AC + B$*)*. The pseudocode for the single-substitution reaction procedure is as follows:

SingleSubstitution_Procedure
Input: *selected_elements*, *singlesubstitution_rate* 1. n = size (*selected_elements*) 2. i = floor (n / 2) 3. $a = selected\_elements_1, selected\_elements_2, \ldots, selected\_elements_i$ 4. $b = selected\_elements_{i+1}, selected\_elements_{i+2}, \ldots, selected\_elements_{ix2}$ 5. Apply *Decomposition_Procedure* to a; Get $Deco_1, Deco_2$ 6. Apply *Synthesis_Procedure* ($b + Deco_1$); Get *Synthesis_vector*
Output: *SingleSubstitution _vector* (*Synthesis_vector*, $Deco_2$)

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. In this type of reaction, the cations simply swap anions; during the algorithm, a similar process that in the previous reaction happens, the difference is that in this reaction both of the selected compounds are decomposed and the resulting elements are combined between each other (*$AB + CD \rightarrow CB + AD$*)*. The pseudocode for the double-substitution reaction procedure is as follows:

DoubleSubstitution_Procedure
Input: *selected_elements*, *doublesubstitution_rate* 1. n = size (*selected_elements*) 2. i = floor (n / 2) 3. a = $selected\_elements_1$, $selected\_elements_2$, …, $selected\_elements_i$ 4. b = $selected\_elements_{i+1}$, $selected\_elements_{i+2}$, …, $selected\_elements_{ix2}$ 5. Apply *Decomposition_Procedure* to a and b; Get ($Deco_1$, $Deco_2$), ($Deco_1$', $Deco_2$') 6. Apply *Synthesis_Procedure* ($Deco_1$ + $Deco_1$'), ($Deco_2$ + $Deco_2$') Get $Synthesis\_vector_1$, $Synthesis\_vector_1$'
Output: *SingleSubstitution _vector* ($Synthesis\_vector_1$, $Synthesis\_vector_1$')

In this chemical reaction algorithm we may trigger only one reaction or all of them, depending on the nature of the problem to solve, e.g., we can apply only the decomposition reaction sub-routine to find the minimum value of a mathematical function.

Throughout the execution of the algorithm, whenever a new set of elements/compounds are created, an elitist reinsertion criteria is applied, allowing the permanence of the best elements and thus the average fitness of the entire element pool increases through iterations.

In order to have a better picture of the general schema for this proposed chemical reaction algorithm, a comparison with other nature inspired paradigms is shown in Table 1.

Table 1 Main elements of several nature inspired paradigms

Paradigm	Parameter Representation	Basic Operations
GA	Genes	Crossover, Mutation
ACO	Ants	Pheromone
PSO	Particles	Cognitive, Social Coefficients
GP	Trees	Crossover, Mutation (In some cases)
CRM	Elements, Compounds	Reactions (Combination, Decomposition, Substitution, Double-substitution)

3 The Mobile Robot

Mobile robots are non-holonomic systems due to the constraints imposed on their kinematics. The equations describing the constraints cannot be integrated symbolically to obtain explicit relationships between robot positions in local and global coordinate's frames. Hence, control problems that involve them have attracted attention in the control community in recent years [14].

The model considered is that of a unicycle mobile robot (see Figure 2) that has two driving wheels fixed to the axis and one passive orientable wheel that are placed in front of the axis and normal to it [5].

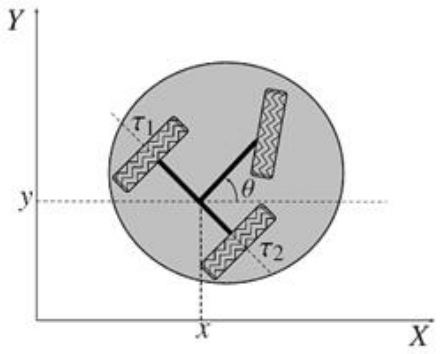


Fig. 2 Diagram of a wheeled mobile robot

The two fixed wheels are controlled independently by the motors, and the passive wheel prevents the robot from overturning when moving on a plane.

It is assumed that the motion of the passive wheel can be ignored from the dynamics of the mobile robot, which is represented by the following set of equations [9]:

$$\dot{q} = \begin{vmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{vmatrix} \begin{vmatrix} v \\ w \end{vmatrix} \tag{1}$$

$$M(q)\dot{v} + V(q,\dot{q})v + G(q) = \tau$$

Where $q = [x, y, \theta]^T$ is the vector of generalized coordinates which describes the robot position, *(x,y)* are the Cartesian coordinates, which denote the mobile center of mass and θ is the angle between the heading direction and the *x*-axis (which is taken counterclockwise form); $v = [v, w]^T$ is the vector of velocities, *v* and *w* are the linear and angular velocities respectively; $\tau \in R^r$ is the input vector, $M(q) \in R^{nxn}$ is a symmetric and positive-definite inertia matrix, $V(q,\dot{q}) \in R^{nxn}$ is the centripetal and Coriolis matrix, $G(q) \in R^n$ is the gravitational vector. Equation (1.a) represents the kinematics or steering system of a mobile robot.

Notice that the no-slip condition imposed a non holonomic constraint described by (2), that it means that the mobile robot can only move in the direction normal to the axis of the driving wheels.

$$\dot{y}\cos\theta - \dot{x}\sin\theta = 0 \tag{2}$$

The control objective will be established as follows: Given a desired trajectory $q_d(t)$ and the orientation of the mobile robot we must design a controller that applies an adequate torque τ such that the measured positions $q(t)$ achieve the desired reference $q_d(t)$ represented as (3):

$$\lim_{t\to\infty}\|q_d(t)-q(t)\|=0 \tag{3}$$

To reach the control objective, the method is based on the procedure of [9], we are deriving a $\tau(t)$ of a specific vc(t) that controls the steering system (1.a) using a Fuzzy Logic Controller (FLC). A general structure of tracking control system is presented in Figure 3.

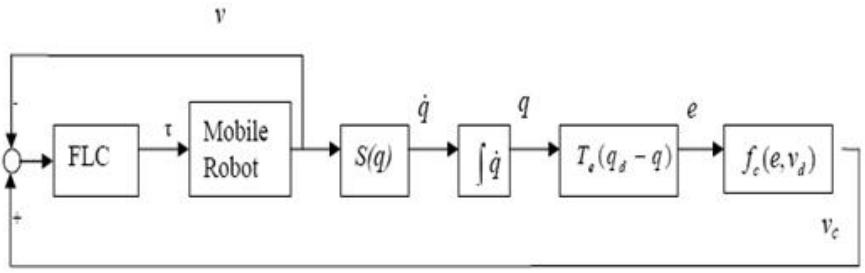


Fig. 3 Tracking control structure

The control is based on the procedure proposed by Kanayama et al. [13] and Nelson et al. [21] to solve the tracking problem for the kinematic model $v_c(t)$. Suppose that the desired trajectory q_d satisfies (4):

$$\dot{q}_d=\begin{vmatrix}\cos\theta_d & 0\\ \sin\theta_d & 0\\ 0 & 1\end{vmatrix}\begin{vmatrix}v_d\\ w_d\end{vmatrix} \tag{4}$$

Using the robot local frame (the moving coordinate system x-y in figure 1), the error coordinates can be defined as (5):

$$e=T_e(q_d-q),\begin{vmatrix}e_x\\ e_y\\ e_\theta\end{vmatrix}=\begin{vmatrix}\cos\theta & \sin\theta & 0\\ -\sin\theta & \cos\theta & 0\\ 0 & 0 & 1\end{vmatrix}\begin{vmatrix}x_d-x\\ y_d-y\\ \theta_d-\theta\end{vmatrix} \tag{5}$$

And the auxiliary velocity control input that achieves tracking for (1.a) is given by (6):

$$v_c = f_c(e, v_d), \begin{vmatrix} v_c \\ w_c \end{vmatrix} = \begin{vmatrix} v_d + \cos e_\theta + k_1 e_x \\ w_d + v_d k_2 e_y + v_d k_3 \sin e_\theta \end{vmatrix} \tag{6}$$

Where k_1, k_2 and k_3 are positive gain constants.

The first part for this work is to apply the proposed method to obtain the values of k_i (i = 1, 2, 3) for achieving the optimal behavior of the controller, and the second part is to optimize the fuzzy controller.

4 Fuzzy Logic Controller

The purpose of the fuzzy logic controller (FLC) is to find a control input τ such that the current velocity vector v is able to reach the velocity vector v_c and this is denoted as:

$$\lim_{t \to \infty} \| v_c - v \| = 0 \tag{7}$$

The inputs variables of the FLC correspond to the velocity errors obtained of (10) (denoted as e_v and e_w: linear and angular velocity errors respectively), and 2 outputs variables, the driving and rotational input torques τ (denoted by F and N respectively). The initial membership functions (MF) are defined by 1 triangular and 2 trapezoidal functions for each variable involved. Figure 4 depicts the MFs in which N, Z, P represent the fuzzy sets (Negative, Zero and Positive respectively) associated to each input and output variable.

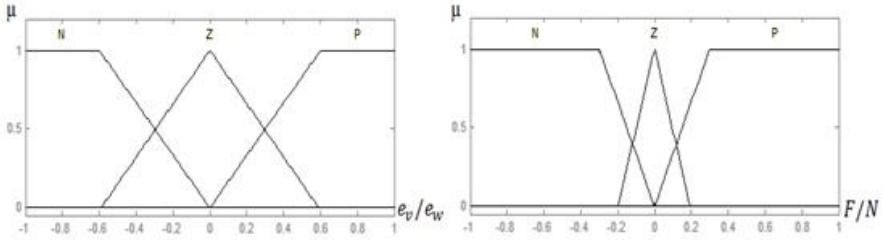


Fig. 4 Membership functions of the (a) input e_v and e_w, and (b) output variables *F* and *N*

The rule set of the FLC contain 9 rules, which govern the input-output relationship of the FLC and this adopts the Mamdani-style inference engine. We use the center of gravity method to realize defuzzification procedure. In Table 2, we present the rule set whose format is established as follows:

Rule i: If e_v is G1 and e_w is G2 then F is G3 and N is G4

Where G1…G4 are the fuzzy sets associated to each variable and i= 1 ... 9. In this case, P denotes “Positive”, N denotes “Negative”, and Z denotes “Zero”.

Table 2 Fuzzy rule set

e_v/e_w	N	Z	P
N	N/N	N/Z	N/P
Z	Z/N	Z/Z	Z/P
P	P/N	P/Z	P/P

5 Experimental Results

Several tests of the chemical optimization paradigm were made to test the performance of the tracking controller. First, we need to find the values of k_i (i = 1, 2, 3) showed in equation 6, which shall guarantee convergence of the error e to zero.

To evaluate the constants obtained by the algorithm, the mobile robot tracking system, which consists in equations 5 and 6 was modeled using Simulink®. Figure 5 shows the closed loop for the tracking controller.

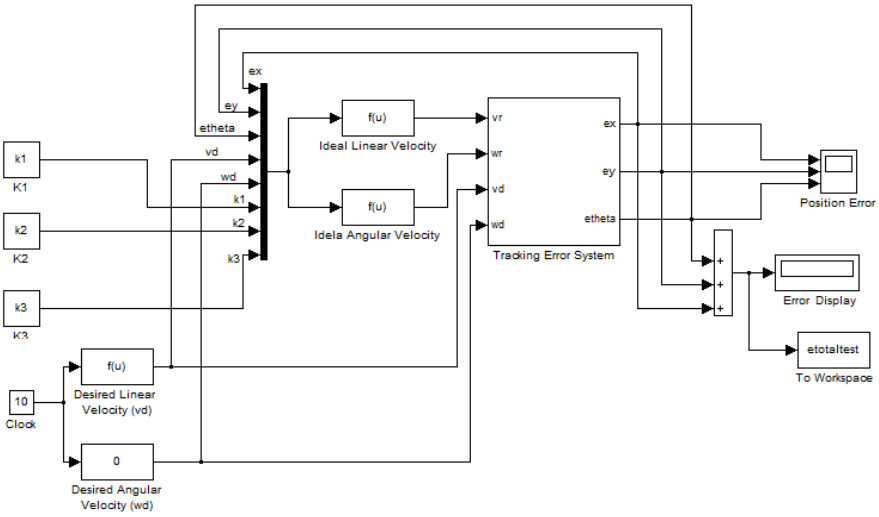


Fig. 5 Closed loop for the tracking controller system

The conditions to evaluate each result, which correspond to the final position error, are given by equation 12.

$$EP = \sum_{i=1}^{n} \frac{e_x(i) + e_y(i) + e_\theta(i)}{n} \tag{12}$$

For the first set of experiments only the decomposition reaction mechanism was triggered and the decomposition factor was varied; this factor is the quantity of resulting elements after applying a decomposition reaction to a determined "compound"; the only restriction here is that let x be the selected compound and x'_i ($i=1\ 2,\ \ldots,\ n$), the resulting elements; the sum of all values found in the decomposition must be equal to the value of the original compound. This is shown in equation 13.

$$\sum_{i=1}^{n} x'_i = x \tag{13}$$

Each experiment was executed 35 times and the test parameters for each set of experiments can be observed in Table 3.

The decomposition rate (Dec. Rate) represents the percentage of the pool to be candidate for the decomposition and the decomposition factor (Dec. Factor) is the number of elements to be decomposed into.

The selection strategy applied was the stochastic universal sampling, which uses a single random value to sample all of the solutions by choosing them at evenly spaced intervals.

In example, for a pool containing 5 initial compounds, the vector length of decomposed elements when the decomposition factor is 3 and the decomposition rate is 0.4 will be of 6 elements.

Table 3 Parameters of the Chemical Reaction Optimization

No.	Elements	Iterations	Dec. Factor	Dec. Rate
1	2	10	2	0.3
2	5	10	3	0.3
3	2	10	2	0.4
4	2	10	3	0.4
5	5	10	2	0.4
6	5	10	3	0.4
7	5	10	2	0.5
8	10	10	2	0.5

By applying this criterion, the initial pool of elements increased with every iteration; this is why the initial element pool was set to 10 elements as maximum. Table 4 shows the results after applying the chemical optimization paradigm.

Table 4 Experimental Results of the proposed method for optimizing the values of the gains k_1, k_2, k_3

No.	Best Error	Mean	k_1	k_2	k_3
1	0.0086	1.1568	519	46	8
2	4.79e-04	0.1291	205	31	31
3	0.0025	0.5809	36	328	88
4	0.0012	0.5589	2	206	0
5	0.0035	0.0480	185	29	5
6	8.13e-005	0.0299	270	53	15
7	0.0066	0.1440	29	15	0
8	0.0019	0.1625	51	3	0

As it is observed in Table 4, experiment number 6 seems to be the best result because it reached the smaller final error among all experiments.

Figure 6 shows the final position errors in *x*, *y* and *theta* for experiment no. 6.

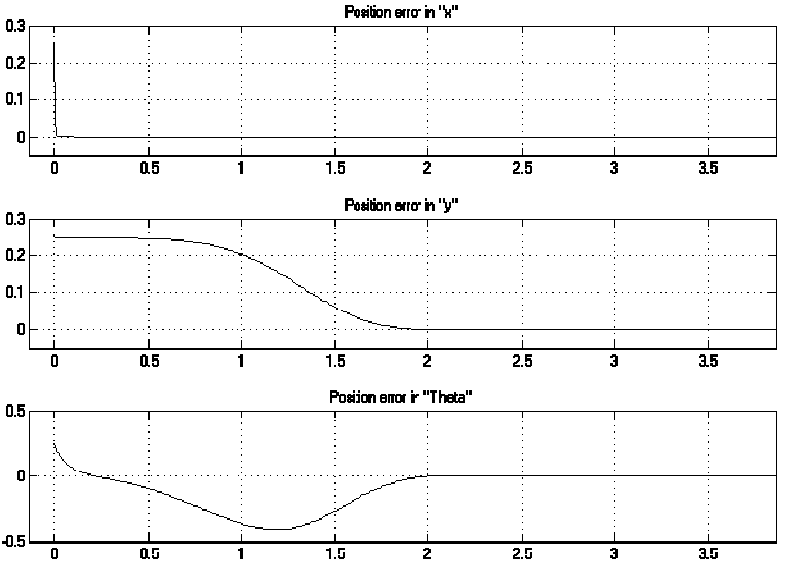


Fig. 6 Final position errors in *x*, *y* and *theta* for experiment no. 6

By analyzing the graphical results of several set of exercises, we noticed that the control obtained for some of them was "smoother" despite the average error value. This was the case for experiment no. 3, in which the final error value was significantly higher than the obtained in experiment no. 6. Figure 7 shows the final position errors in *x*, *y* and *theta* for experiment no. 3.

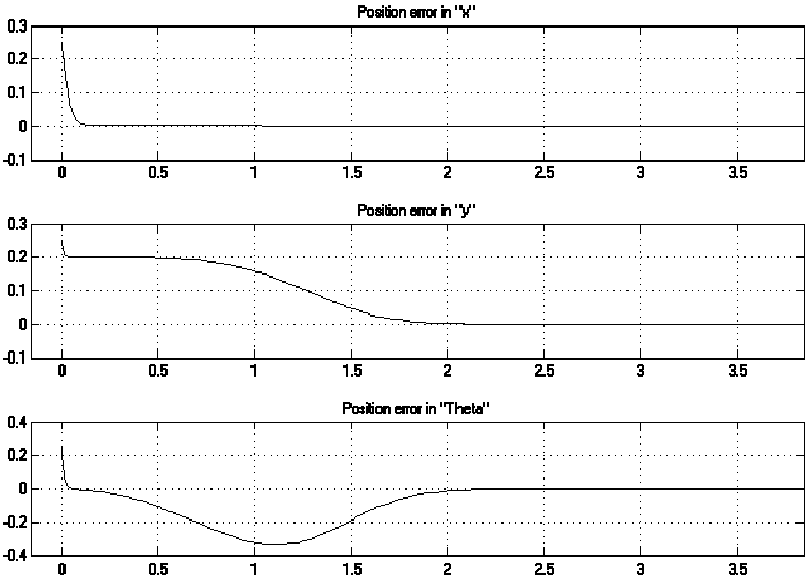


Fig. 7 Final position errors in *x*, *y* and *theta* for experiment no. 6

Making a comparison between both graphics, we can observe that the average error obtained for *theta* is 0.0338 for experiment no. 6 and 0.0315 for experiment no. 3.

This smoother control of the tracking system could make a big difference in the complete dynamic system of the mobile robot.

In previous work [22], the gain constant values were found by means of genetic algorithms. In Table 5 we have a comparison of the best results obtained with both algorithms, we can observe that the result with the chemical optimization outperforms the GA in finding the best gain values.

Table 5 Comparison of the Best Results

Parameters	Genetic Algorithm	Chemical Optimization Algorithm
Individuals	5	2
Iterations	15	10
Crossover Rate	0.8	N/A
Mutation Rate	0.1	N/A
Synthesis Rate	N/A	0.2
Decomposition Rate	N/A	0.8
Substitution Rate	N/A	0.6
Double Substitution Rate	N/A	0.6
k1, k2, k3	43, 493, 195	36, 328, 88
Final Error	0.006734	0.0025

Figure 8 shows the result in Simulink for the experiment with the best overall result, applying GAs as optimization method.

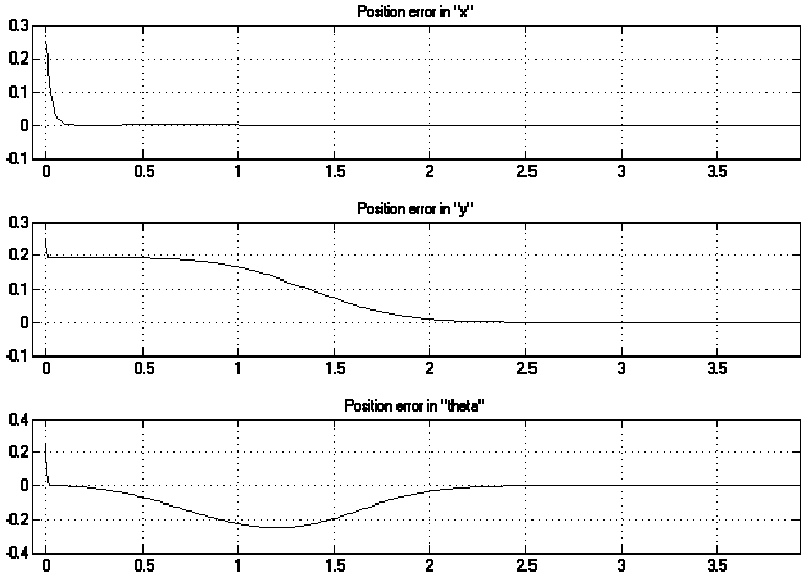


Fig. 8 Position errors in *x*, *y* and *theta* of best result applying GAs

Once we have found optimal values for the gain constants, the next step is to find the optimal values for the input/output membership functions of the fuzzy controller. Our goal is that in the simulations, the lineal and angular velocities reach zero. Table 6 shows the parameters of the simulateons for typ-1 FLC.

Table 6 Parameters of the simulations for Type-1 FLC

Parameters	Value
Elements	10
Trials	15
Selection Method	Stochastic Universal Sampling
k_1	117
k_2	226
k_3	137
Error	0.077178

Figure 9 shows the behavior of the chemical optimization algorithm throughout the experiment.

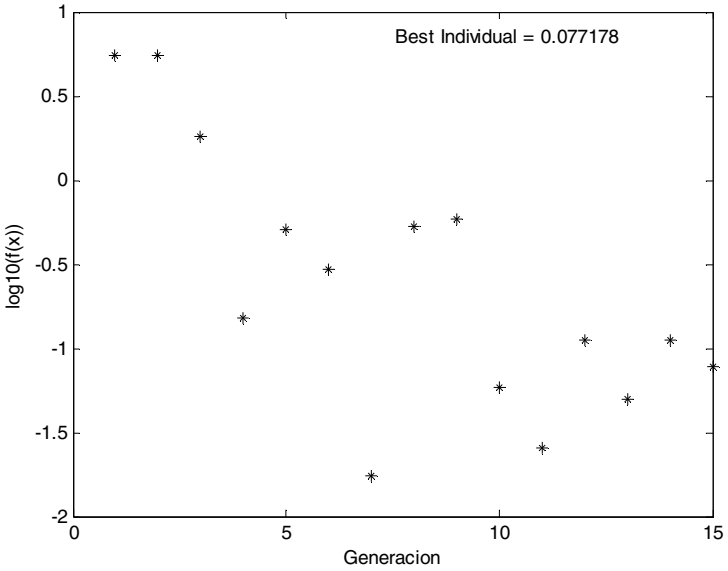


Fig. 9 Best simulation of experiments with the chemical optimization method

Figure 10 shows the resulted input and output membership functions found by the proposed optimization algorithm.

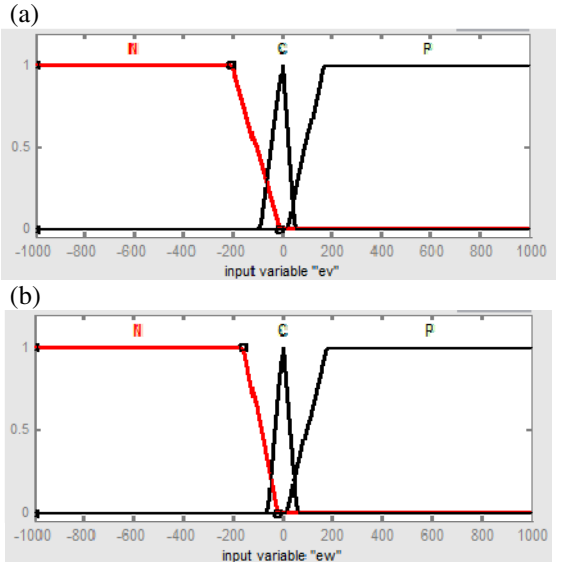


Fig. 10 Resulting input membership functions: (a) linear and (b) angular velocities and output (c) right and (d) left torque

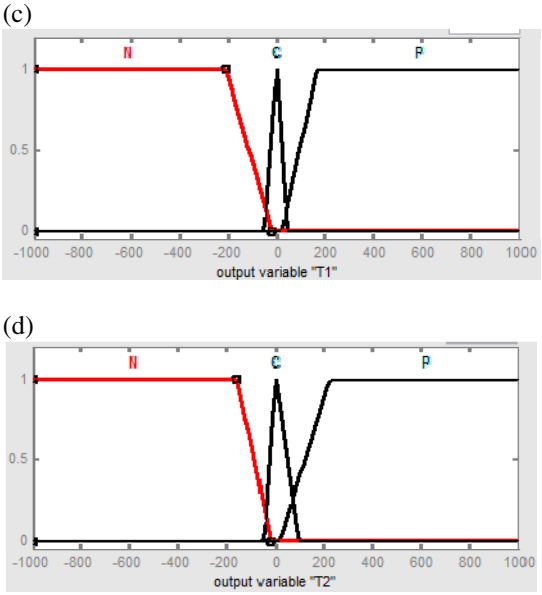


Fig. 10 (*continued*)

Figure 11 shows the obtained trajectory when simulating the mobile control system including the obtained input and output membership functions.

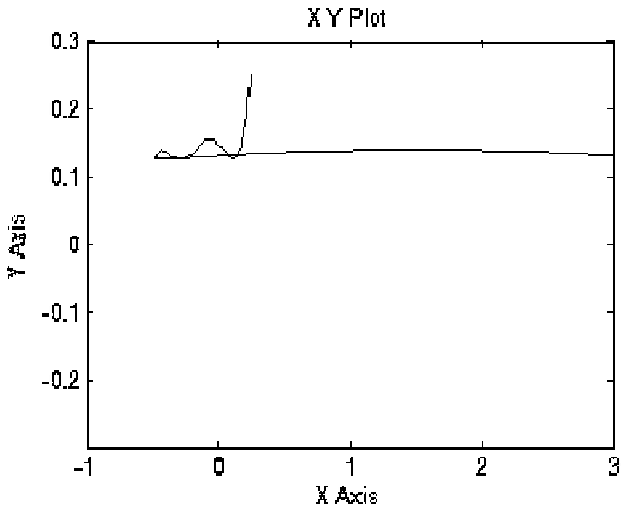


Fig. 11 Obtained trajectory when applying the chemical reaction algorithm

Figure 12 shows the best trajectory reached by the mobile when optimizing the input and output membership functions using genetic algorithms.

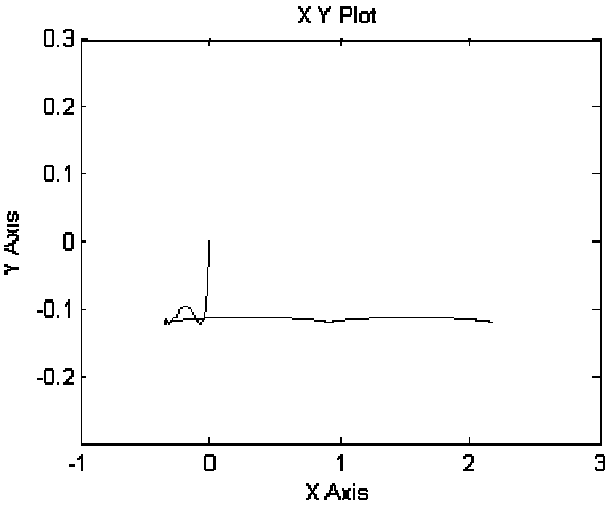


Fig. 12 Obtained trajectory using genetic algorithms

A Type-2 fuzzy logic controller was developed using the parameters of the membership functions found for the FLC type-1. The parameters searched with the chemical reaction algorithm were for the footprint of uncertainty (FOU).

Table 7 shows the parameters used in the simulations and Figure 13 shows the behavior of the chemical optimization algorithm throughout the experiment.

Table 7 Parameters of the simulations for Type-2 FLC

Parameters	Value
Elements	10
Trials	10
Selection Method	Stochastic Universal Sampling
k_1	117
k_2	226
k_3	137
Error	2.7736

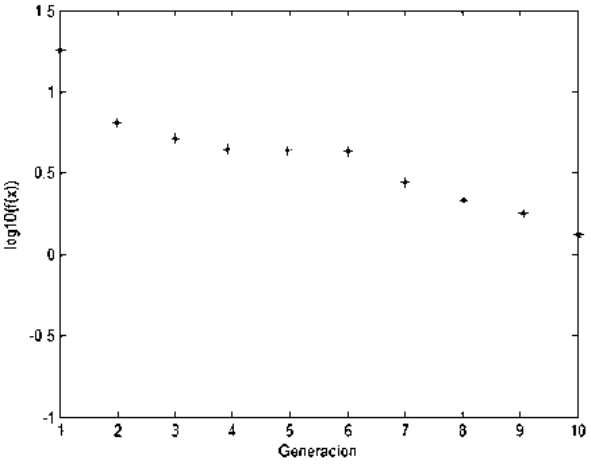


Fig. 13 Behavior of the algorithm when optimizing the type-2 FLC

Figure 14 shows the resulting type-2 input and output membership functions found by the proposed optimization algorithm and Figure 15 shows the obtained trajectory reached by the mobile robot.

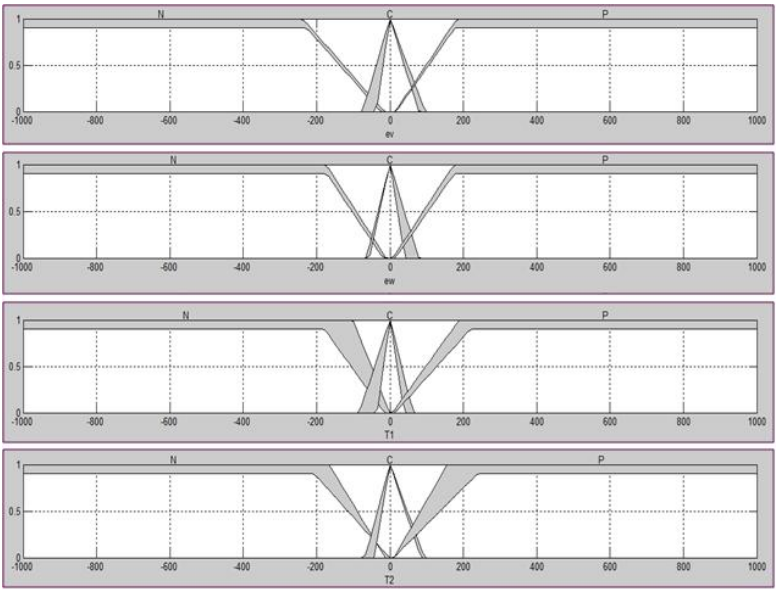


Fig. 14 Resulting type-2 input membership functions, from top to bottom: (a) linear and (b) angular velocities and output (c) right and (d) left torque

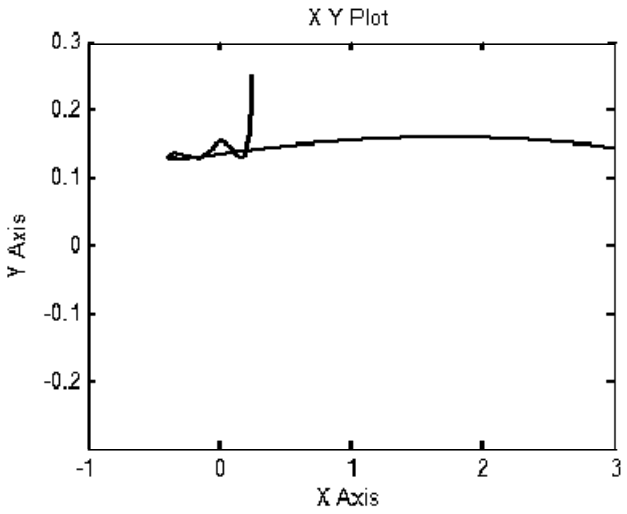


Fig. 15 Obtained trajectory for the mobile robot when applying the chemical reaction algorithm to the type-2 FLC

As observed in Table 7, the final error obtained is not smaller that the final error found for the type-1 FLC. Despite this, the trajectory obtained and showed in Figure 15 is acceptable taking into account that the reference trajectory is a straight line. In Figure 16 we can observe an "unacceptable" trajectory that was found in the early attempts of optimization for the type-1 FLC applying this chemical reaction algorithm. Here, we can observe that the parameters found were not the adequate to make the FLC follow the desired trajectory.

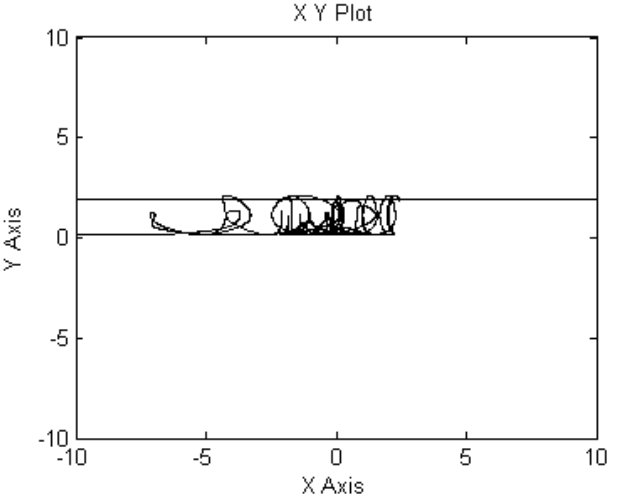


Fig. 16 Unaccepted resulting trajectory in early optimization trials

In order to test the robustness of the type-1 and type-2 FLC, we added an external signal given by equation (13).

$$F_{ext}(t) = \varepsilon \times \sin \omega \times t \tag{13}$$

This represents an external force applied in a period of 10 seconds to the obtained trajectory that will make the mobile robot to be out of its path. The idea of adding this disturbance is to measure the errors obtained with the FLC and to test the behavior of the mobile robot under perturbed torques. Table 8 shows the parameters for the simulations and the errors obtained during the run of the simulation.

Table 8 Simulation parameters and errors obtained under disturbed torques

ε	Velocity errors	Type-1 (GA)	Type-1 (CRA)	Type-2 (CRA)
0.05	Final error	4.0997	0.9815	29.5115
	Average error	4.1209	1.5823	26.6408
5	Final error	4.1059	0.9729	29.52
	Average error	3.1695	1.8679	26.1646
10	Final error	4.1045	0.9745	29.51
	Average error	3.0985	1.7438	24.9467
30	Final error	4.0912	0.9783	29.51
	Average error	2.2632	1.9481	24.6032
32	Final error	3273	0.9748	29.52
	Average error	3.4667e+003	2.8180	24.6465
34	Final error	1.5705e+004	566.8	29.51
	Average error	1.1180e+004	215.8198	24.9211
40	Final error	2.534e+004	3.5417e+04	29.51
	Average error	186.0611	5.7492e+003	23.8938
41	Final error	8839	3168	685.1
	Average error	2.0268e+004	0.0503e+003	16.5257

Figure 17 show the obtained trajectories for the type-1 FLC optimized with Genetic Algorithms.

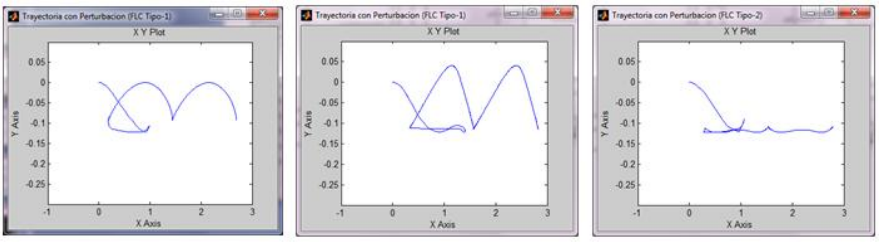


Fig. 17 From left to right, trajectory obtained with the type-1 FLC optimized with GA's. (a) ε = 30, (b) ε = 32, (c) ε = 34.

Figure 18 shows the obtained trajectories for the type-1 FLC optimized with the chemical reaction algorithm.

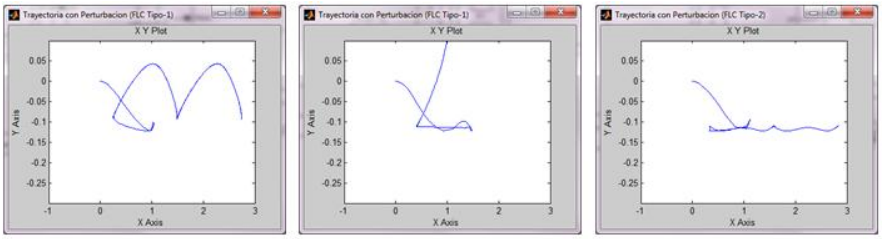


Fig. 18 From left to right, trajectory obtained with the type-1 FLC optimized with CRA. (a) ε = 30, (b) ε = 32, (c) ε = 34.

Figure 19 shows the obtained trajectories for the type-2 FLC optimized with the CRA method.

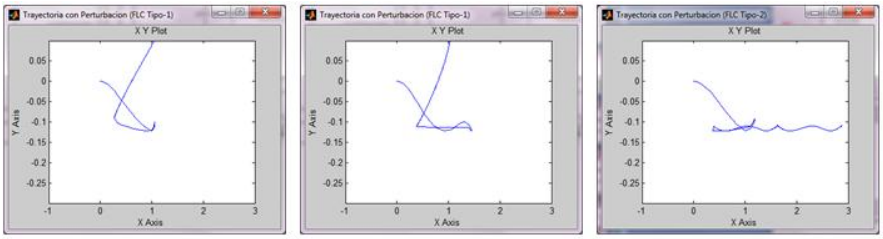


Fig. 19 From left to right, trajectory obtained with the type-2 FLC optimized with CRA. (a) ε = 30, (b) ε = 32, (c) ε = 34.

When observing Table 8 and Figures 17 to 19 we can observe that the type-2 FLC was able to maintain a more controlled trajectory in despite of the "large" error found by the algorithm (e=2.7736). For larger epsilon (ε) values, it was difficult for the type-1 FLC's to keep in the path and in a determined time, the controller was not able to return to the reference trajectory.

6 Conclusions

In this paper, we presented simulation results from an optimization method that mimics chemical reactions applied to the problem of tracking control. The goal was to find the gain constants involved in the tracking controller for the dynamic model of a unicycle mobile robot. In the figures of the experiments we are able to note de behavior of the algorithm and the solutions found through all the iterations. Simulation results show that the proposed optimization method is able to outperform the results previously obtained applying a genetic algorithm optimization technique. The optimal fuzzy logic controller obtained with the

proposed chemical paradigm has been able to reach smaller error values in less time than genetic algorithms. Also, the type-2 fuzzy controller was able to perform better under the presence of disturbance for this problem in despite of the "large" error obtained (e=2.7736). The design of optimal type-2 fuzzy controllers is being performed at the time.

Acknowledgement. The authors would like to thank CONACYT and Tijuana Institute of Technology for the facilities and resources granted for the development of this research.

References

[1] Aliev, R.A., Pedrycz, W., Guirimov, B.G., Aliev, R.R., Ilhan, U., Babagil, M., et al.: Type-2 fuzzy neural networks with fuzzy clustering and differential evolution optimization. Information Sciences 181(9), 1591–1608 (2011)

[2] Astudillo, L., Castillo, O., Aguilar, L.: Intelligent Control for a Perturbed Autonomous Wheeled Mobile Robot: a Type-2 Fuzzy Logic Approach. Nonlinear Studies 14(1) (2007)

[3] Bentalba, S., El Hajjaji, A., Rachid, A.: Fuzzy Control of a Mobile Robot: A New Approach. In: Proc. IEEE Int. Conf. on Control Applications, Hartford, CT, pp. 69–72 (October 1997)

[4] Bloch, A.M., Drakunov, S.: Tracking in NonHolonomic Dynamic System Via Sliding Modes. In: Proc. IEEE Conf. on Decision & Control, Brighton, UK, pp. 1127–1132 (1991)

[5] Campion, G., Bastin, G., D'Andrea-Novel, B.: Structural Properties and Classification of Kinematic and Dynamic Models of Wheeled Mobile Robots. IEEE Trans. on Robotics and Automation 12(1) (February 1996)

[6] Lopez, M., Melin, P., Castillo, O.: Comparative Study of Fuzzy Methods for Response Integration in Ensemble Neural Networks for Pattern Recognition. In: Melin, P., Kacprzyk, J., Pedrycz, W. (eds.) Bio-inspired Hybrid Intelligent Systems for Image Analysis and Pattern Recognition. SCI, vol. 256, pp. 123–140. Springer, Heidelberg (2009)

[7] Cazarez-Castro, N.R., Aguilar, L.T., Castillo, O.: Fuzzy logic control with genetic membership function parameters optimization for the output regulation of a servomechanism with nonlinear backlash. Expert Systems with Applications 37(6), 4368–4378 (2010)

[8] Chwa, D.: Sliding-Mode Tracking Control of Nonholonomic Wheeled Mobile Robots in Polar coordinates. IEEE Trans. on Control Syst. Tech. 12(4), 633–644 (2004)

[9] Fierro, R., Lewis, F.L.: Control of a Nonholonomic Mobile Robot: Backstepping Kinematics into Dynamics. In: Proc. 34th Conf. on Decision & Control, New Orleans, LA (1995)

[10] Fierro, R., Lewis, F.L.: Control of a Nonholonomic Mobile Robot Using Neural Networks. IEEE Trans. on Neural Networks 9(4), 589–600 (1998)

[11] Fukao, T., Nakagawa, H., Adachi, N.: Adaptive Tracking Control of a NonHolonomic Mobile Robot. IEEE Trans. on Robotics and Automation 16(5), 609–615 (2000)

[12] Ishikawa, S.: A Method of Indoor Mobile Robot Navigation by Fuzzy Control. In: Proc. Int. Conf. Intell. Robot. Syst., Osaka, Japan, pp. 1013–1018 (1991)

[13] Kanayama, Y., Kimura, Y., Miyazaki, F., Noguchi, T.: A Stable Tracking Control Method For a Non-Holonomic Mobile Robot. In: Proc. IEEE/RSJ Int. Workshop on Intelligent Robots and Systems, Osaka, Japan, pp. 1236–1241 (1991)
[14] Kolmanovsky, I., McClamroch, N.H.: Developments in NonholonomicNontrol Problems. IEEE Control Syst. Mag. 15, 20–36 (1995)
[15] Lee, T.-C., Lee, C.H., Teng, C.-C.: Tracking Control of Mobile Robots Using the Backsteeping Technique. In: Proc. 5th. Int. Conf. Contr., Automat., Robot. Vision, Singapore, pp. 1715–1719 (December 1998)
[16] Lee, T.H., Leung, F.H.F., Tam, P.K.S.: Position Control for Wheeled Mobile Robot Using a Fuzzy Controller, pp. 525–528. IEEE (1999)
[17] Lee, T.-C., Tai, K.: Tracking Control of Unicycle-Modeled Mobile robots Using a Saturation Feedback Controller. IEEE Trans. on Control Systems Technology 9(2), 305–318 (2001)
[18] Martinez, R., Castillo, O., Aguilar, L.: Optimization of type-2 fuzzy logic controllers for a perturbed autonomous wheeled mobile robot using genetic algorithms. Information Sciences 179(13), 2158–2174 (2009)
[19] Meyer, T., Yamamoto, L., Banzhaf, W., Tschudin, C.: Elongation Control in an Algorithmic Chemistry. In: Kampis, G. (ed.) ECAL 2009, Part I. LNCS, vol. 5777, pp. 273–280. Springer, Heidelberg (2011)
[20] Mohammadi, S.M.A., Gharaveisi, A.A., Mashinchi, M., Vaezi-Nejad, S.M.: An evolutionary tuning technique for type-2 fuzzy logic controller. Transactions of the Institute of Measurement and Control 33(2), 223–245 (2011)
[21] Nelson, W., Cox, I.: Local Path Control for an Autonomous Vehicle. In: Proc. IEEE Conf. on Robotics and Automation, pp. 1504–1510 (1988)
[22] Oh, S., Jang, H., Pedrycz, W.: A comparative experimental study of type-1/type-2 fuzzy cascade controller based on genetic algorithms and particle swarm optimization. Expert Systems with Applications 38(9), 11217–11229 (2011)
[23] Pawlowski, S., Dutkiewicz, P., Kozlowski, K., Wroblewski, W.: Fuzzy Logic Implementation in Mobile Robot Control. In: 2nd Workshop on Robot Motion and Control, pp. 65–70 (October 2001)
[24] Sahab, A.R., Moddabernia, M.R.: Backstepping method for a single-link flexible-joint manipulator using genetic algorithm. IJICIC 7(7B), 4161–4170 (2011)
[25] Shi, N.-Y., Chu, C.-P.: A molecular solution to the hitting-set problem in DNA-based supercomputing. Information Sciences 180, 1010–1019 (2010)
[26] Song, K.T., Sheen, L.H.: Heuristic fuzzy-neural Network and its application to reactive navigation of a mobile robot. Fuzzy Sets Systems 110(3), 331–340 (2000)
[27] Tsai, C.-C., Lin, H.-H., Lin, C.-C.: Trajectory Tracking Control of a Laser-Guided Wheeled Mobile Robot. In: Proc. IEEE Int. Conf. on Control Applications, Taipei, Taiwan, pp. 1055–1059 (September 2004)
[28] Ulyanov, S.V., Watanabe, S., Ulyanov, V.S., Yamafuji, K., Litvintseva, L.V., Rizzotto, G.G.: Soft Computing for the Intelligent Robust Control of a Robotic Unicycle with a New Physical Measure for Mechanical Controllability. Soft Computing 2, 73–88 (1998)
[29] Xu, J., Lam, A.Y.S., Li, V.O.K.: Chemical Reaction Optimization for the Grid Scheduling Problem. In: IEE Communication Society, ICC 2010, pp. 1–5 (2010)
[30] Yamamoto, L.: Evaluation of a Catalytic Search Algorithm. In: Proc. 4th Int. Workshop on Nature Inspired Cooperative Strategies for Optimization, NICSO 2010, pp. 75–87 (2010)
[31] Yu, J., Ma, Y., Chen, B., Yu, H., Pan, S.: Adaptive Neural Position Tracking Control for Induction Motors via Backstepping. IJICIC 7(7B), 4503–4516 (2011)

A Genetic Algorithm for the Problem of Minimal Brauer Chains for Large Exponents

Arturo Rodriguez-Cristerna and Jose Torres-Jimenez

Information Technology Laboratory, CINVESTAV-Tamaulipas Km. 5.5 Carretera Cd. Victoria-Soto la Marina, 87130, Cd. Victoria Tamps., Mexico
arodriguez@tamps.cinvestav.mx, jtj@cinvestav.mx

Abstract. Exponentiation is an important and complex task used in cryptosystems such RSA. The reduction of the number of multiplications needed during the exponentiation can significantly improve the execution time of cryptosystems. The problem of determining the minimal sequence of multiplications required for performing a modular exponentiation can be formulated using the concept of Brauer Chains.

This paper, shows a new approach to face the problem of getting Brauer Chains of minimal length by using a Genetic Algorithm (GA). The implementation details of the GA includes a representation based on the Factorial Number System (FNS), a mixture of Neighborhood Functions (NF), a mixture of Distribution Functions (DF) and a fine-tuning process to set the parameter values. We compare the proposed GA approach with another relevant solutions presented in the literature by using three benchmarks considered difficult to show that it is a viable alternative to solve the problem of getting shortest Brauer Chains.

Keywords: Addition Chains, Genetic Algorithms.

1 Introduction

Exponentiation is an important and complex task used in cryptosystems such as RSA. The reduction of the number of multiplications needed during the exponentiation can significantly improve the execution time of cryptosystems [3]. The problem of determining the minimum operations required for the exponentiation of x^n have been searched with different strategies and a *naive* solution is to apply a sequence of *n-1* multiplications of x such that $x^n = x \cdot x \cdot \ldots \cdot x$. For example, if we want to compute x^{23} using the naive way we have to apply 22 multiplications. Another possibility to compute x^n is by applying the *binary method* [14], which is showed in a recursive description in the Equation 1.

P. Melin, O. Castillo (Eds.): *Soft Computing Appli. in Optimization,* STUDFUZZ 294, pp. 27–51.
DOI: 10.1007/978-3-642-35323-9_2

$$x^{\alpha}\begin{cases} x & \text{if } \alpha = 1 \\ x^{\frac{\alpha}{2}} \cdot x^{\frac{\alpha}{2}} & \text{if } \alpha \text{ is even} \\ x^{\alpha-1} \cdot x & \text{otherwise} \end{cases} \quad (1)$$

For the example of the exponentiation x^{23}, using the *binary method* we only need to perform 7 multiplications, as can be seen next:

$$xx = x^2, x^2x^2 = x^4, xx^4 = x^5, x^5x^5 = x^{10}, xx^{10} = x^{11}, x^{11}x^{11} = x^{22}, xx^{22} = x^{23}$$

But it is possible to reduce further the number of multiplications needed for the exponentiation than those used by the *naive* way or the *binary method*, and there have been reported different strategies to simplify this complex task like the *m-ary method* and the *window-based* method [7,14].

Also, the problem of determining the minimal sequence of multiplications required to perform an exponentiation can be formulated with the concept of Addition Chain (AC). The addition chain-based methods for exponentiation use a sequence of positive integers such that the first number of the chain is 1 and the last number is the exponent *n*. Therefore, the length of an addition chain *n* is equal to the corresponding number of multiplications required for the computation of x^n. Thus, the smallest of such multiplications, given by the chain length *l(n)*, makes the exponentiation task faster.

Given a positive integer *n*, an AC is a sequence of integers $C = \{\alpha_0, \alpha_1, \dots, \alpha_r\}$ such that:

$$\alpha_i = \begin{cases} 1 & \text{if } i = 0 \\ \alpha_j + \alpha_k & \text{if } i > 0 \text{ for some } j, k < i \end{cases} \quad (2)$$

The length of the AC is $|C| = r$, the chain length *l(n)* is the minimal length of all possible ACs for *n*, and the smaller addition chain is called Minimal Addition Chain (MAC).

An example of an AC for the exponent 5 with length 4 is the sequence of numbers shown in the Figure 1, where can be see how each non root member in the chain are composed by two previous members and the element *r* is an integer equal to *n*.

In an AC every set of values *{j,k}* is called step, and according with their properties along the chain, the steps and the complete chain takes some particular name. A first instance is if *j* and *k* are equal to *i*-1, it its called "double step". Another instance is when *j* is equal to *i*-1 and *k*<*j*, i.e. if one of the addends is the previous member, then it is called a star step, and "an addition chain that consists entirely of star steps is called a star chain" [27] or Brauer Chain (BC) [15] in honor of Brauer (1937). We denote $l^*(n)$ the minimal BC length for a number *n*, and where a BC *C* has the smallest length *r* for a number *n* we can say that *C* is a Minimal Brauer Chain (MBC) for *n*.

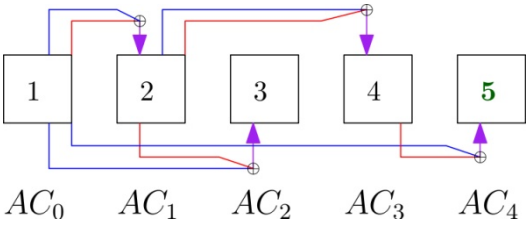


Fig. 1 Example of an AC for the number 5

An example of a BC is (1,2,4,5,9,18,23), which leads to the following scheme for the computation of x^{23} using only 6 multiplications:

$$xx = x^2, x^2x^2 = x^4, xx^4 = x^5, x^5x^4 = x^9, x^9x^9 = x^{18}, x^{18}x^5 = x^{23}$$

A difference between an AC and a BC is the search space, because in an AC the search space grows in a way at least factorial according to its length but in a BC it is only grow in a factorial way. In fact, the search space for BCs can be seen as a tree of r levels (Fig. 2). For that reason the searching process for a MBC for small numbers like 19 is relatively easy, but for bigger numbers is not because the search space becomes very large. According to Knuth [17] and Mignotte [19] another remarkable difference is that $l(n) \leq l^*(n)$. The smallest exponent, taken from [11], with we can observe the difference lengths of MCAs and MBCs is 12509 (Table 1).

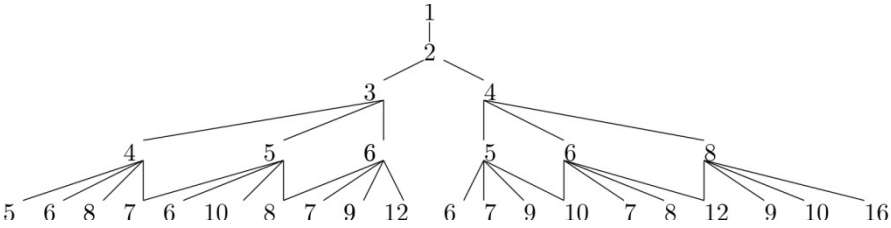


Fig. 2 Search space for Brauer Chains with length 4

Table 1 Example of difference between $l(n)$ and $l^*(n)$ with the exponent n=12509

Type	Chain	r
MAC	1→2→3→6→12→13→24→48→96→192→384→ 768→781→1562→3124→6248→12496→12509	17
MBC	1→2→3→6→12→24→48→96→192→384→768→ 1536→1560→3120→3126→6252→6254→12508→ 12509	18

In this paper we propose a Genetic Algorithm (GA) to face the problem of getting MBCs, using a representation based on the Factorial Number System (FNS), a mixture of neighborhood functions and a mixture of distribution functions. The remaining of this paper is organized as follows. Section 2 gives a brief description

of a variety of approaches proposed to find MACs and MBCs, Section 3 describes our proposed approach, Section 4 shows how the fine-tuning process was done and the parameter configuration used, Section 5 shows the results obtained and finally Section 6 gives the reached conclusions.

2 Relevant Related Work

In the last years, it has been shown that metaheuristic strategies find near optimal solutions for a wide variety of combinatorial problems in a reasonable time. In this section are described some approaches that have been designed for the problem of finding MACs and MBCs.

Bleichenbacher and Flammenkamp [3] search for MACs by using Direct Acyclic Graphs (DAG) to represent the chains and a BackTracking (BT) search to construct the graph. Their backtrack approach uses a stage where special cases of addition chains are replaced with another equivalent ones in order to get smaller ACs.

Thurber in 1999 [27] proposed a BT algorithm to find MACs, by using a representation based on a tree of k levels and branch and bound methods to explore the tree of a size at least $k!$.

Nedja and Moruelle in 2002 [20] designed an approach based in the *m-ary* method using a parallel implementation to compute MACs by decomposing the exponent, in its binary representation, in blocks (also called windows) containing successive digits of ones that results in variable length zero-partitions and one-partitions. They found ACs with a lower number of elements than the binary method does.

Nedja and Moruelle in 2003 [23] used large windows inside a genetic algorithm. Their optimal parameter settings found were: 50 individuals; a double-points crossover; a mutation rate between 0.4 and 0.7; and a mutation degree of about 1% of the last value in the binary encoding sequence.

Cruz-Cortes *et. al* in 2005 [7] proposed a GA to solve the problem of finding MACs with the following features: each solution is represented as a sequence of integers, where each gene is related to one step of the AC, so each time that their algorithm apply a crossover operator it have to assure that the resulting sequence is a valid AC and the fitness is the AC length. The remaining features of their proposed GA are: a "non elitist" survivor selection; a population size of 100; a number of generations of 300; a mutation rate of 0.5; a selection of parents pairs to be recombined by a binary tournament; and an one point crossover. Their results suggest that this kind of algorithm can be a good alternative to solve the problem of finding MBCs.

Nedja and Moruelle in 2004 [21] and 2006 [22] proposed Ant Colony Systems (ACS) to obtain MACs that use a bi-dimensional triangular array to store the ACS global memory. The local memory is divided in two parts: a vector of length n and the fitness of the path traveled by the ant to construct the AC.

Gelgi and Onus in 2006 [12], proposed some heuristics approaches for the problem of getting an MBC. They present five approaches: three greedy heuristics and two dynamic programming approaches. They found empirically, that their

dynamic heuristic approach has an approximation ratio (obtained length / minimum length) of 1.1 with $0 \leq n \leq 20000$.

Cruz-Cortes *et. al* in 2008 [6] presented an algorithm using the metaheuristic approach known as Artificial Immune System (AIS) to tackle the problem of finding short BCs, which uses a cloning operator and a hypermutation operator (the hypermutation operator is inversely proportional to the clones fitness) over the best solutions; and an "elitist" selection. The values of the parameter used are: a population size of 45; a number of best individuals to be cloned of 11; a number of replaced antibodies of 4 and 25 iterations of the main cycle. Also they combine their AIS system with a slide-window method to deal with large numbers. They use a function named *fill* to search for a valid BC, but its important to say that they do not mention about how much effort is done inside the *fill* function either in terms of time or evaluations.

Osorio-Hernandez *et. al* in 2009 [24] design a GA to find minimal length Brauer Chains. They use a repairing process; local search; and integer representation. They were able to find short BCs with the following parameters: population size of 200; maximum number of generations of 300; and binary tournament selection. However they don't were able to find all minimal length BCs with only one configuration and they had to use two different configurations to find some of them. Other important aspects of this work are that: the parameter configuration was obtained by trial and error, and there is no mention about how many operations are used or how much time is used inside the *fill* function (which is used to search for a valid BC).

Dominguez-Isidro *et. al* in 2011 [8,9] designed an algorithm to face the problem of finding MBCs using an evolutionary programming approach, where mutation is the only variation operator. Each individual k in the current population generates by mutations t mutants, and the best of them is chosen as k's offspring. As parameter values they use: population size of 100; maximum number of generations of 230; 4 mutants per individual; and a survivor selection based on a tournament of size 10.

Jose-Garcia *et. al* in 2011 [16] designed an interesting algorithm based on Simulated Annealing (SA) approach for the problem of finding MBCs, although the name of the work mentions addition chains, which uses a mixture of 4 neighborhood functions and a representation based on a chain in FNS. This algorithm works only with one solution at a time due to Simulated Annealing specification and the parameter values were: an initial temperature of 10, a cooling rate of .85 and a length of a Markov chain of 10. Also, with the purpose of reducing consuming time of the experimentation they performed and used a parallel test scenario.

Rodriguez-Cristerna *et. al* in 2011 [26] reports a Mutation-Selection (MS) algorithm, based on the general scheme of an evolutionary algorithm but without the recombination stage to find MBCs. This algorithm uses: a representation based on FNS, which is adequate to apply genetic operators and always generate a valid solution; a fitness function based in the n achieved and the length of the chain; a "non elitist" survivor selection; $3\log_2 n$ parents; $7\log_2 n$ mutated children per each parent; and $n \times 1000$ iterations.

Clift in 2011 [4], designed a BT strategy based on a graph representation and a novel prune criterion. This approach was able to find all $l(n)$ with $n \leq 2^{32}$ using a computational time of about a month using 12 processors.

There have been proposed more approaches to find MACs and MBCs, however here we list some important metaheuristic works. A brief enumeration of the main features of the presented strategies is showed in the Table 2.

Table 2 Summary of main features for some proposed approaches to find MACs and MBCs

Approach	Repair process	Use of past memory	Representation	Iterative operatorss	Parallell	Chain type
BT [3]	no	no	graph	no	no	AC
BT [27]	no	no	graph	no	no	AC
m-ary [20]	no	no	binary	no	yes	AC
GA [23]	no	no	binary	no	no	AC
GA[7]	yes	no	integer	no	no	AC
ACS [21,22]	no	no	integer	no	no	AC
AIS [6]	no	yes	integer	yes	no	BC
GA [24]	yes	no	integer	yes	no	BC
SA [16]	no	no	FNS	yes	no	BC
MS [26]	no	no	FNS	yes	no	BC

3 Proposed Approach

3.1 Genetic Algorithm

In order to present the GA proposed, a brief description of how it works is given. GA uses one or more points in the search space, called *parent-points*, to generate multiple points through recombination and then apply to them a mutation process (many times implemented as a local search procedure) to the new points. The generated points are called *children-points* and are evaluated in search of an optimal point. When no optimal point is found, the whole process is repeated replacing the *parent-points* with some points of the population according to a rule, which is called survivor selection. This cycle is repeated until an optimal point is found or certain termination criterion is met.

The algorithm proposed is based on the general scheme of an evolutionary algorithm [10], and its pseudocode is showed in the Algorithm 1.

Algorithm 1. *General scheme of the GA algorithm proposed.*

```
INITIALIZE parents
EVALUATE parents
REPEAT
  SELECT pairs of parents
  RECOMBINE pairs of parents
```

```
    MUTATION through local search with the resulting
  offspring
    EVALUATE new candidates
    parents =  survivor selection
  UNTIL the number of evaluations functions are done
```

Contextualizing the GA for MBC computation, we have to address the next points:

- The representation and the search space used by the proposed algorithm (described in Subsection 3.2).
- The distribution functions used to select a position in a BC (in FNS representation) and the distribution function used to pickup values for a given position in the BC (described in Subsection 3.3).
- The *survivor selection* methods used (described in Subsection 3.4).
- The *children-points* generated through selection and recombination (detailed in Subsection 3.5).
- The local search process with children-points using Neighborhood Functions (detailed in Subsection 3.6).
- The evaluation function used to measure the quality of the potential solutions and the termination condition (described in Subsection 3.7).

3.2 Representation and Search Space

The representation used is based on the FNS and the search space is $r!$ where r is the length of the BC. We use a lower bound denoted by φ and an upper bound denoted by ψ, defined in the Equations 3 and 4 respectively. The lower bound is defined as the barrier of minimum number of multiplications that are needed if we could apply only double steps and the upper bound is defined as the maximum number of multiplications needed by the binary method.

$$\varphi = \lfloor \log_2(n) \rfloor \tag{3}$$

$$\psi = 2 \cdot \lfloor \log_2(n) \rfloor \tag{4}$$

In FNS we can describe a BC C with a chain of numbers, that we refer as C', by taking a value from the set $\{0,1,...,i\text{-}1\}$ for each node of C' with an index position i greater than 0, because the representation in FNS for the node 0 of C has no value. We can rebuild the original BC from the chain C' applying the Equation 5. The Figure 3 shows an example of how to represent a BC C for n=23 using a chain of numbers C' in the FNS.

It's necessary to mention that the population is initialized only with double steps nodes, and the chain C' is always translated to a BC until the last node which its translation is equal or lower than n.

index	0	1	2	3	4	5	6
C	1	2	4	5	9	18	23
C'	-	0	1	0	2	4	3

Fig. 3 Example of representation of a BC C for n=23 in a chain of numbers C' using the FNS

$$C(i) = \begin{cases} C(i-1) + C\big(C'(i)\big) & \text{if } i > 0 \\ 1 & \text{if } i = 0 \end{cases} \tag{5}$$

3.3 Distribution Functions for Selecting a Position in the BC

The distribution functions allows to select an i position in a C' chain, such that $1 \leq i \leq r$, with a non Gaussian distribution in two steps: first is calculated a random value x with $0 \leq x \leq \tau$ (Equation 6) and second is used one of the two distribution functions in Equation 7 to calculate the selected i position.

The main purpose of the two distribution DF_1 and DF_2 is the selection of a point in the BC. DF_1 allows to select with more probability the left most positions while the distribution function DF_2 allows to select with more probability the right most positions. The importance of using DF_1 and DF_2 is to establish a balance between exploration and exploitation. DF_1 enables exploitation and DF_2 enables exploration. The distribution functions DF_1 and DF_2 also has the property to select with a small probability a higher position, but not more than one unit far from the chain length, which increases the possibility of modifying the chain length.

$$\tau = \frac{(r+1) \times (r+2)}{2} \tag{6}$$

$$i = \begin{cases} DF_1 = \left\lfloor \dfrac{1 + \sqrt{1 + 8(x+1)}}{2} \right\rfloor - 1 \\ DF_2 = r + 3 - \left\lfloor \dfrac{1 + \sqrt{1 + 8(x+1)}}{2} \right\rfloor \end{cases} \tag{7}$$

Once defined the criteria for selecting a position within the BC, we are able to define the criterion to choose a valid value for the positions. The steps that follow a neighborhood function to select a valid value for an i position in a BC are two: first is selected a random x' such that $0 \leq x' \leq \tau'$ (Equation 8); and second is used the distribution function DF_1. This methodology ensure that the i chain member has a value from the set {0,1,...,i-1} and also has the property to set in the i position a double step with high probability.

$$\tau' = \frac{(i+1) \times i}{2} \tag{8}$$

3.4 Survivor Selection

It is important to select the best survivor selection type because it allows to manage the memory of the GA in different ways. For that reason we explore the use of three types of survivor selection: the first one is called (μ+λ) and takes the best points from the set of *parent-points* and the *children-points* using a random criterion as a tiebreaker.

The second one is called (μ+λ) with no repetitions, which takes the best points from whole population but discards the points with the same fitness. In the cases where the *parent-points* can not be full filled with different individuals, the *parent-points* are completed with the the best remaining individuals.

The third one is called (μ,λ) and takes the best points from the set of the *children-points*, in case of ties its used a random selection criterion. The *parent-points* are randomly replaced in case that they were fewer *children-points* than *parent-points*.

3.5 Selection and Recombination

The selection of parents pairs to be recombined were explored with the following strategies: random selection, tournament of size two and tournament of size three. In the random selection both parents are randomly chosen. For the tournament of size two, there are selected two groups of two parents points and only the best one of each group is selected to be recombined. Finally for the tournament of size three, there are selected groups of three parents points and the best parent point for each group is selected.

For the recombination, we use a simple recombination of one point scheme. It means, that if we choose the point t for the recombination, the *children-point* C''_1 is constructed with the first t elements of the *parent-point* C'_1 and the remaining elements of the *parent-point* C'_2, in complement, the *children-point* C''_2 is constructed with the first t elements of the *parent-point* C'_2 and the remaining elements of the parent-point C'_1. This process is exemplified in the Figure 4.

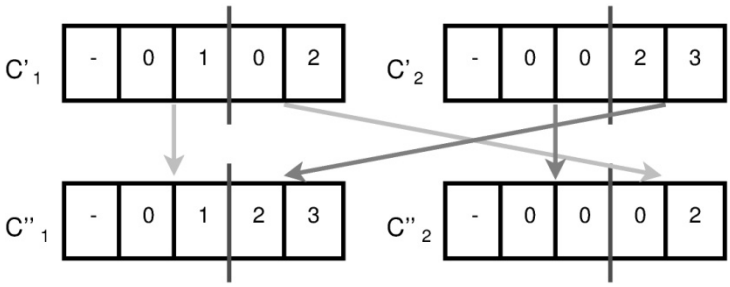


Fig. 4 Example of recombination of *parents-point* C'_1 and C'_2

To pickup the t point, we explore five mixtures of use of the distribution functions DF_1 and DF_2 which are listed in Table 3.

Table 3 Mixtures of use of the distribution functions DF_1 and DF_2 inside neighborhood functions

Mixture no.	DF_1	DF_2
0	0%	100%
1	25%	75%
2	50%	50%
3	75%	25%
4	100%	0%

3.6 Neighborhood Functions and Local Search

Inside of the local search we explore the use of a mixture of four neighborhood functions, which are described below:

- $NF_1(s)$. Select a random index position i from s, and pickup another FNS value different from the original with the use of DF1, the change is made only if the resulting BC is better than the original.
- $NF_2(s)$. Select a random index position i from s, and pickup a FNS value different from the original in i. Then select another different random position j from s, and pickup a FNS value different from the original in j. The change is applied only if the resulting BC is better than the original.
- $NF_3(s)$. Select a random index position i from s, and pickup the best FNS value.
- $NF_4(s)$. Select a random index positions i and j from s, and pickup the best FNS values.

To select the i and j positions inside the pairs of neighborhood functions NF_1- NF_3 and NF_2- NF_4, we test the use of a mixture of distribution functions as in Section 3.5, because the use of different distribution functions modify the behavior of the neighborhood functions. For example, the use of the distribution function DF_1 implies exploitation while the use of the distribution function DF_2 implies exploration. Additionally NF_1 and NF_2 use the DF_1 to pickup a value for the selected position.

The local search consists in applying iteratively the neighborhood functions. To explore the behavior of the amount of resources wasted during the local search in the algorithm, we try five differents limits of iteratively applications of the neighborhood functions: 1, $0.10 \cdot r+1$, $0.20 \cdot r+1$, $0.50 \cdot r+1$, $r+1$.

Looking for the performance, their probability of use each one of the neighborhood functions was modeled according to the solution of a Diophantine Equation (DE) with four variables (Equation 9), where each variable could take a value of the set {0.1, 0.2, 0.3, ... , 1.0}.

$$p_1 + p_2 + p_3 + p_4 = 1.0 \tag{9}$$

3.7 Evaluation Function and Termination Condition

The evaluation function ς used in the GA is shown in Equation 10.

$$\varsigma = |n' - n|(r + 1) + r \tag{10}$$

In Equation 10 r represents the length of the BC that is evaluated, n' is the value of the evaluated chain in its r position and n is the searched value. In this way, solutions whose n' is near to the n searched, have an evaluation only determined by its length, but solutions whose difference between n' and n, have an evaluation determined by its difference multiplied by its length plus its length which mean that those solutions are penalized, making possible to discriminate between the quality of chains.

The termination condition is met when certain limit number of evaluation functions are done. We established the limit of evaluations functions as $\log_2(n)^2 * 26^{\log2(\log2(n))}$. The evaluation functions are considered according to each time that is checked the fitness in the chain, thus NF_3 and NF_4 use a number of evaluation functions that grows with the position or positions that their are modifying while NF_1 and NF_2 only use one evaluation function each time that are called. This criterion make us able to distinguish the potential of the different configurations of the full system using the same computational resources.

4 Fine-Tuning Process

The selection of the best configuration parameters is a problem that is addressed in different ways: by using some parameter configuration taken from the literature, using a configuration defined by guessing, using a exhaustive search, using Mixed Covering Array (MCA), among others.

We will get very unlikely the best system configuration with the first two strategies, because there can be significance differences between problems and algorithm implementations or our guessing can be wrong. On the other hand we will be able to find the bet best system configuration with an exhaustive search, but we need to consider the amount of time that it requires and most of the times its an infeasible amount because the number of parameter makes the possible configuration system grow exponentially.Another strategy is to use MCAs, having the advantage of use fewer test cases than exhaustive search, but still be able to find good parameter settings and at the same time the test suite is reusable, i.e. the same MCA could be used for different systems.

The use of MCAs has been reported for fine-tuning process to establish parameter configurations [13, 16, 25, 26], however the most reported application of MCAs is in software interaction testing testing [5], which based in the concept that the software faults are caused by unexpected interactions between components. Thus to test a system instead of test all possible combinations between interactions, its used a *t*-way testing. Some empirical studies have found that software interaction testing is feasible [1,18].

According to Gonzalez-Hernandez *et al.* [13] a MCA, represented as $MCA\ (N; t, k, v_0 v_1 \ldots v_{k-1})$, is an $N \cdot k$ array where $v_0 v_1 \ldots v_{k-1}$ are the cardinalities for the vectors or alphabets that indicates the values for the i column. The value t is the interaction degree between parameters covered and is called the strength of the MCA. The value k is the number of columns or parameters. The rows are the specific system configurations to be tested and the complete MCA is the test suite. Also, the MCA has the follow properties:

1. Each column i $(0 \leq \mathrm{i} < k)$ contains only elements from a set (alphabet) S_i with $|S_i| = v_i$.
2. The rows of each $N \cdot t$ sub-array cover all t-tuples from t columns at least once.

To set the parameters of the GA proposed and get a good performance we use a Mixed Covering Array (MCA) and the solutions of the Equation 9, relying the methodology followed to tune the values of the parameters on the study of the effect over the quality of the solutions generated by tests that cover the t interaction degree between parameters.

First we choose an MCA that describe the system to be tested, in our case we select a $MCA\ (32; 2, 8, 5^4 3^4)$[1], which has 32 test cases, 8 parameters, 4 of them with five possible values and the remaining parameters with three possible values. The MCA used could be see in Table 5a. The strength level is 2, means that the MCA contains every interaction between pairs of parameters, and therefore if the best configuration is determined only by pairs of parameters we are able to find it with t=2.

The mapping between the numbers inside the MCA (Table 5a) and their corresponding parameter values can be done with the Table 5b where the parameters selected for the fine tuning are listed in the Table 4.

Table 4 Mixtures of use of the distribution functions DF_1 and DF_2 inside neighborhood functions

Parameter	Description
P1	Probability of use DF1 and DF2 inside NF1-NF3
P2	Probability of use DF1 and DF2 inside NF2-NF4
P3	Probability of use DF1 and DF2 inside the recombination strategy
P4	Iteration amount of neighborhood function inside the local search
P5	Recombination selection type
P6	Parent points
P7	Children points
P8	Survivor selection type

We use an MCA combined with the solutions of a DE, because we want to explore the performance of the system using different ratios of use each neighborhood function. Since was tested each combination of the MCA rows with all the possible solutions of the DE (Equation 9), the Equation 11 represents the

[1] A repository of MCAs is available in [28].

total of the experiments that we ran during the tuning process, where M represents the number of rows of the MCA used, D is the number of possible solutions of the DE (Equation 9), B is the number of times that each $M \cdot D$ experiment was done to obtain statistical significance, and I is the number of instances to be tested during the fine tuning process. Since M=32, D=286, B=31 and I=2 then the total number of experiments is 32x286x31x2 = 567,424. The fine tuning was done by trying to obtain the MBC for the numbers 457 and 14143037.

$$T = M\text{x}D\text{x}B\text{x}I \tag{11}$$

Table 5 MCA transposed matrix and their corresponding parameter values for parameter optimization

(a). MCA values for GA algorithm

No	P1	P2	P3	P4	P5	P6	P7	P8
1	0	0	2	1	2	0	2	1
2	0	1	0	2	0	0	1	0
3	0	2	1	4	1	2	0	2
4	0	3	3	0	1	0	0	0
5	1	0	4	3	0	1	1	1
6	1	1	1	1	1	2	0	0
7	1	2	2	2	2	1	1	0
8	1	3	0	1	0	0	2	1
9	1	4	3	4	0	2	1	2
10	2	0	1	0	0	1	1	2
11	2	2	3	1	2	1	0	2
12	2	3	4	2	1	0	0	1
13	2	4	0	3	1	2	2	0
14	3	0	0	4	1	1	0	0
15	3	2	4	0	0	2	2	1
16	3	3	2	3	1	2	1	2
17	3	4	1	2	1	0	0	1
18	4	0	3	2	1	2	2	2
19	4	5	5	4	0	1	2	2
20	4	2	0	0	2	0	1	2
21	4	3	1	3	2	1	2	1
22	4	4	2	1	0	0	1	2
23	0	4	4	1	2	1	1	2
24	3	4	1	0	1	2	0	1
25	3	1	3	1	2	1	1	1
26	2	3	2	4	2	2	0	1
27	2	1	4	4	1	0	2	2
28	2	1	4	4	1	0	2	2
29	1	1	2	0	0	0	2	1
30	4	1	4	3	0	0	0	0
31	0	4	4	4	1	2	2	2
32	4	0	1	4	1	2	2	2

(b). Parameter values

Values	0	1	2	3	4
P1	$0, \frac{4}{4}$	$\frac{1}{4}, \frac{3}{4}$	$\frac{2}{4}, \frac{2}{4}$	$\frac{1}{4}, \frac{3}{4}$	$0, \frac{4}{4}$
P2	$0, \frac{4}{4}$	$\frac{1}{4}, \frac{3}{4}$	$\frac{2}{4}, \frac{2}{4}$	$\frac{1}{4}, \frac{3}{4}$	$0, \frac{4}{4}$
P3	$0, \frac{4}{4}$	$\frac{1}{4}, \frac{3}{4}$	$\frac{2}{4}, \frac{2}{4}$	$\frac{1}{4}, \frac{3}{4}$	$0, \frac{4}{4}$
P4	1	.1r+1	.2r+1	.5r+2	r+1
P5	rnd.	tourney size 2	tourney size3		
P6	$\lceil \log_2 n \rceil$	$\lceil 2\log_2 n \rceil$	$\lceil 3\log_2 n \rceil$		
P7	$\lceil \log_2 n \rceil$	$\lceil 2\log_2 n \rceil$	$\lceil 3\log_2 n \rceil$		
P8	(μ,λ)	(μ+λ) diversity	(μ+λ)		

4.1 Parameters Used

Te best configuration obtained as a result of a fine-tuning process is described in Table 6.

Table 6 Parameter configuration obtained during the fine-tuning process

Parameter	Configuration
P1	0% of use DF1 and 100% of use DF2 inside NF1-NF3.
P2	100% of use DF2 and 0% of use DF2 inside NF1-NF3
P3	25% of use DF2 and 75% of use DF2 inside of the recombination strategy
P4	r+1 iterations of neighborhood functions during the local search.
P5	Random parent-point selection for the recombination strategy
P6	⌈3log_2 n⌉ parent-points and children-points
P7	20% of use of NF3 and 80% of use of DF4
P8	Survivor selection type (μ+λ)

4.2 Implementation Note

The proposed GA was coded using C language and compiled with GCC 4.3.5 with -O3 optimization flag. The algorithm has been run on a single core of a cluster with 4 processor six-core AMD® 8435 (2.6 Ghz), 32 GB RAM, and Operating System Red Hat Linux Enterprise 4.

5 Results

In order to measure the performance of the GA proposed, the first experiment was to search the MBC for a benchmark made of 27 different numbers n that satisfy the restriction of be $c(r)$, available in [11] along a database of many $l(n)$ values computed by Neill Clift [4], and to get statistical significance each experiment was tested 31 times with different random seeds. $c(r)$ is the smallest number which have an addition chain of lenght r, and the set of numbers that accomplish the $c(r)$ property are a special class of numbers.

The main results of the first experiment are shown in Table 7, where we can see the set of n's tried; the minimal, average and maximum length obtained; the average and standard deviation of the time to conclude each experiment; and the hits (times that a MBC was found). It can be seen that for the first experiment (Table 7) all the MBC was obtained and the average length and the standard deviation obtained indicates the reliability of the GA. The behavior of the proposed GA for the first experiment also can be seen in the Figure 4 that shows the difference between minimum, average and maximum length with their respective standard deviation versus optimal lengths, where is evident that our approach can get easily most of the Brauer Chains with $l^*(n)<22$ and for the other cases the average length obtained are not too far from the optimal. Table 9 shows

Table 7 Summary of results to compute MBC for some n for which is hard to find their MBC (part 1)

id	n	Min. r	Average r	Max. r	Std. Dev. r	Average time (s)	Std. Dev. time (s)	hits
1	7	4	4	4	0	0	0	31
2	11	5	5	5	0	0.0003	0	31
3	19	6	6	6	0	0.0010	0	31
4	29	7	7	7	0	0.0023	0	31
5	47	8	8	8	0	0.0055	0	31
6	71	9	9	9	0	0.0107	0.0001	31
7	127	10	10	10	0	0.0259	0.0021	31
8	191	11	11	11	0	0.0442	0.0006	31
9	379	12	12	12	0	0.1031	0.0007	31
10	607	13	13	13	0	0.1760	0.0017	31
11	1087	14	14	14	0	0.3241	0.0076	31
12	1903	15	15	15	0	0.5503	0.0034	31
13	3583	16	16.0323	17	0.1767	0.1767	0.0075	30
14	6271	17	17	17	0	1.5671	0.0117	31
15	11231	18	18	18	0	2.4806	0.0176	31
16	18287	19	19	19	0	3.5497	0.0250	31
17	34303	20	20.1613	21	0.3678	5.5663	0.0553	26
18	65131	21	21	21	0	8.5120	0.1364	31
19	110591	22	22.2903	23	0.4539	11.9941	0.1586	22
20	196591	23	23.0323	24	0.1767	17.0883	0.2624	30
21	357887	24	24.1613	26	0.4470	24.0470	0.3448	27
22	685951	25	25.6129	27	0.5493	34.3607	0.6828	13
23	1176431	26	27.0645	29	0.6689	46.5622	1.0647	5
24	2211837	27	27.9677	30	0.6949	63.2554	0.8555	7
25	4169527	28	28.3226	29	0.4675	86.3890	2.1981	21
26	7624319	29	30.0968	34	1.1175	114.9951	2.2392	11
27	14143037	30	30.9677	32	0.6949	152.4645	2.2392	8

the MBCs found for $n \in \{2211837, 4169527, 7624319, 14143037\}$ where can be verified the restriction of the BC's.

A second benchmark, taken from [18], is composed by 20 different numbers hard to optimize, because its minimal addition chains currently has not been generated by deterministic methods (i.e. binary method or window-based) or some other non deterministic methods. For the second benchmark, every instance was tried 31 times with different random seeds and the results are showed in Table 8 where can be seen the set of n's tried; the minimal, average and maximum length obtained; the average and standard deviation time to finish the tests; and the hits. In the second benchmark only for one instance was not obtained its MBC (3926651), however the minimum size found is not to far from the optimal, in fact it is only at one unit of distance. Additionally, with the complete results we observe that the worst cases and the average cases are not far from the optimal solution, getting in the worst case BC's only three units far from the optimal and in average at 1.4 units plus the optimal length. One of the MBCs found for the numbers 3459835, 3493799, 3704431 and 3922763. are presented in Table 10.

Table 8 Summary of results to compute MBC for some n for which is hard to find their MBC (part 2)

id	n	Min. r	Average r	Max. r	Std. Dev. r	Average time (s)	Std. Dev. time (s)	hits
1	2948207	27	28.2258	30	0.5512	73.3253	0.9955	1
2	3093839	27	28.2903	30	0.6812	75.2953	0.9979	3
3	3167711	27	27.9355	30	0.7155	74.5330	1.0593	7
4	3182555	27	27.7419	28	0.4376	75.1344	0.7311	8
5	3190511	27	27.9355	29	0.5643	74.7987	1.0623	6
6	3230591	27	27.9677	29	0.3094	75.5409	0.7089	2
7	3234263	27	27.3548	29	0.5983	75.3779	0.7897	22
8	3235007	27	28.0645	30	0.6188	75.8259	0.9108	2
9	3243679	27	28.0000	29	0.5080	75.8844	0.8510	4
10	3243931	27	28.0000	29	0.5680	75.7768	0.8483	5
11	3266239	27	28.3548	30	0.6500	76.5093	1.1414	1
12	3287999	27	28.1935	30	0.6435	76.5596	1.0754	3
13	3325439	27	28.3871	30	0.6052	77.0552	1.0106	1
14	3352927	27	27.6774	29	0.5895	76.5714	0.9955	12
15	3440623	27	28.0968	30	0.7769	78.6647	0.9695	7
16	3459835	27	28.0968	29	0.3898	78.6540	0.8115	1
17	3493799	27	28.0968	29	0.5301	79.6454	0.9094	3
18	3704431	27	27.9355	30	0.5643	81.2270	1.3532	5
19	3922763	27	28.4839	30	0.6154	84.6750	1.3315	1
20	3926651	28	28.3548	29	0.4785	84.2965	1.4677	0

Table 9 Some MBCs found (part 1)

n	MBC found	$l^*(n)$
2211837	1→2→4→8→16→32→33→65→130→260→520→ 1040→1073→2146→4292→8584→8617→17234→34468→ 68936→137872→137937→275809→551618→ 552691→1105382→1106455→2211837	27
4169527	1→2→3→6→7→14→28→56→112→113→226→ 452→904→1808→3616→7232→7238→14476→28952→ 28955→57910→115820→231640→260595→ 521190→1042380 →2084760→4169520→4169527	28
7624319	1→2→4→6→7→14→28→30→58→116→232→464→ 928→1856→3712→7424→14848→29696→29724→ 59448→118896→237792→475584→951168→953024→ 1906048→1906078→3812156→7624312→7624319	29
14143037	1→2→3→5→10→13→26→52→104→208→ 416→832→858→1716→1726→3452→6904→ 13808→27616→55232→110464→220928→441856→ 883712→1767424→3534848→3535706→3535758→7071516→ 14143032→14143037	30

In order to contrast the obtained results, Table 11 presents a comparison between the obtained results of our proposed approach in the first experiment and other five approaches reported in the literature: BackTracking [3], Genetic Algorithm [7], Artificial Inmune System [6], Mutation Selection [26] and Simulated Annealing [16]. The comparison is made using the minimum length obtained, the average time (in seconds) and the average hits, where the symbol ⊕

means that it is a worse result than the one given by our approach in both time and/or number of hits. It can be observed that our results have the same or better quality than the procedures presented in the state of the art. It could be useful a more detailed comparison against one of the best reported works [6], however they don't report time spent; number of evaluation functions; or number of hits for each case.

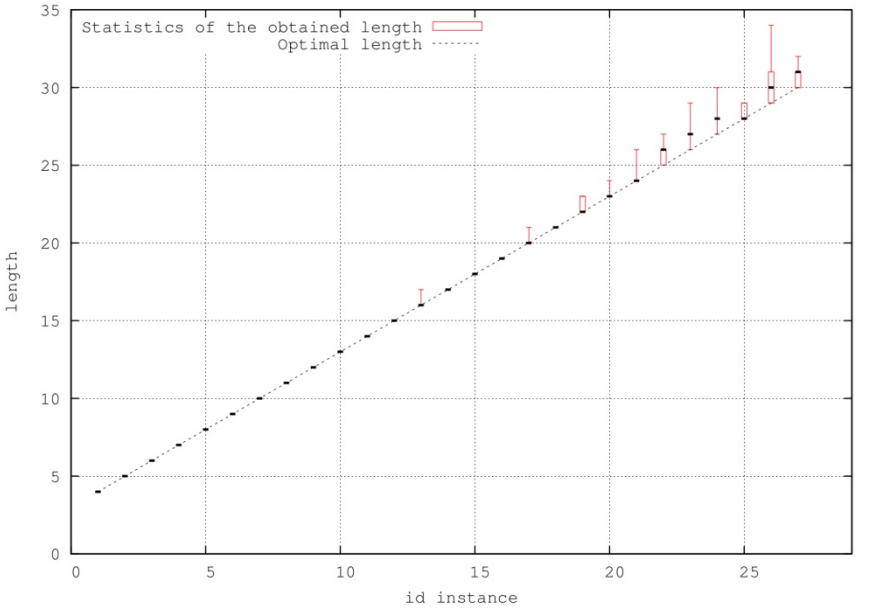


Fig. 5 Comparison of box-and-whisker plot results for the first experiment versus optimal lengths for some n's for which is hard to find their MBC (part 1)

Table 10 Some MBCs found (part 2)

n	MBC found	$l^*(n)$
3459835	1→2→4→8→16→32→33→66→99→ 198→396→792→1584→1683→1691→3374→6748→ 13496→26992→53984→107968→215936→216332→432268→ 864536→1729072→3458144→3459835	27
3493799	1→2→3→5→8→13→26→27→53→106→ 212→424→848→1696→3392→6784→13568→27136→ 54272→108544→217088→434176→868352→868365→875149→ 1309325→2184474→3493799	27
3704431	1→2→4→6→12→13→25→50→63→113→ 226→452→904→1808→3616→7232→14464→28928→ 28940→57880→115760→231520→463040→463052→926092→ 1852184→3704368→3704431→	27
3922763	1→2→4→8→9→17→34→68→136→272→ 281→349→698→1396→1532→3064→6128→12256→ 24512→49024→98048→196096→392192→784384→784665→ 1569049→2353714→3922763	27

Table 11 Comparison of results to compute hard MBC

id	n	$l(n)$	BT [2]	GA [5]	AIS [4]	MS [19] (time s.) (hits)	SA [13] (hits)	GA proposed hits (time s.) (hits)
3	19	6	6	6	6	⊕6 (9.629s) (20)	6 (32)	6 (0.001s) (31)
4	29	7	7	7	7	⊕7 (7.470s) (27)	7 (32)	7 (0.002s) (31)
5	47	8	8	8	8	⊕8 (9.477s) (10)	8 (32)	8 (0.005s) (31)
6	71	9	9	9	9	⊕9 (13.408s) (8)	9 (32)	9 (0.01s) (31)
7	127	10	10	10	10	⊕10 (12.282s) (4)	⊕10 (30)	10 (0.025s) (31)
8	191	11	11	11	11	⊕11 (14.816s) (2)	⊕11 (30)	11 (0.044s) (31)
9	379	12	12	12	12	⊕12 (14.057s) (29)	⊕12 (30)	12 (0.103s) (31)
10	607	13	13	13	13	⊕13 (21.635s) (21)	⊕13 (28)	13 (0.176s) (31)
11	1087	14	14	14	14	⊕14 (26.326s) (8)	⊕14 (13)	14 (0.324s) (31)
12	1903	15	15	15	15	⊕15 (18.538s) (26)	⊕15 (6)	15 (0.550s) (31)
13	3583	16	16	17	16	-	⊕16 (1)	16 (0.973s) (30)
14	6271	17	17	17	17	⊕17 (23.032s) (9)	⊕17 (1)	17 (1.567s) (31)
15	11231	18	18	18	18	⊕18 (30.650s) (20)	⊕18 (1)	18 (2.480s) (31)
16	18287	19	19	19	19	⊕19 (28.136s) (4)	⊕19 (2)	19 (3.549s) (31)
17	34303	20	20	20	20	⊕20 (29.623s) (0)	⊕20 (1)	20 (5.566s) (26)
18	65131	21	21	21	21	⊕21 (32.396s) (3)	⊕21 (2)	21 (8.512s) (31)
19	110591	22	22	22	22	-	⊕22 (2)	22 (11.994s) (22)
20	196591	23	23	23	23	-	⊕23 (2)	23 (17.088s) (30)
21	357887	24	24	24	24	-	⊕24 (1)	24 (24.047s) (27)
22	685951	25	25	25	25	⊕25 (37.696s) (2)	⊕26 (0)	25 (34.360s) (13)
23	1176431	26	26	⊕27	26	⊕27 (46.717s) (0)	⊕26 (1)	26 (46.562s) (5)
24	2211837	27	27	⊕28	27	-	⊕27 (2)	27 (63.255s) (7)
25	4169527	28	-	⊕29	28	28 (33.291s) (1)	⊕28 (1)	28 (86.3890s) (21)
26	7624319	29	-	⊕30	29	29 (42.237s) (1)	⊕30 (0)	29 (114.9951s) (11)
27	14143037	30	-	⊕31	30	30 (64.844s) (1)	⊕30 (2)	30 (152.4645s) (8)

A third experiment, consisted in calculate the accumulated addition chain length for the range of exponents [1,*Z*] for $Z \in \{512, 1000, 1024, 2000, 2048, 4096\}$, where each exponent was tested 31 times with different random seeds. The best results are calculated according to Equation 12 and the worst case according to Equation 13, where GA is a function that receives an integer number and returns a BC generated through our proposed approach. The average results are computed according to Equation 14 and the standard deviation is based on the Equation 15. *CV* is the coefficient of variation calculated as *CV=stddev/average* and δ is the difference between the best case and the optimal case.

$$\sum_{i=1}^{i=Z} \min \{GA_j(i) | 0 \leq j \leq 30\} \tag{12}$$

$$\sum_{i=1}^{i=Z} \max \{GA_j(i) | 0 \leq j \leq 30\} \tag{13}$$

$$average\left\{AS_j = \sum_{i=1}^{i=Z} GA_j(i) \min | 0 \leq j \leq 30\right\} \tag{14}$$

$$stddev\left\{AS_j = \sum_{i=1}^{i=Z} GA_j(i) \min | 0 \leq j \leq 30\right\} \tag{15}$$

Table 12 and 13 presents the results of the third experiment where it is observed the optimal case calculated using the public database of addition chains of Achim Flammenkamp [11]; the best case; the worst case; the average case; and the standard deviation. Additionally it is presented the accumulated time spent for all the cases together with the standard deviation and the difference between the optimal case and our best case (δ). Our results for $Z \in \{512, 1000, 1024\}$ exhibits a δ=0, meaning that all the Minimal Brauer Chains were constructed, but δ is greater than 0 for $Z \in \{2000, 2048, 4096\}$ where the list of numbers (with their corresponding $l(n)$) for which the GA was not able to construct their MBC were: (1063,13), (1143,13), (1387,13), (2011,14), (2087,14), (2091,14), (2135,14), (2151,14), (2251,14), (2285,14), (2507,14), (2617,14), (2647,14), (2774,14), (2957,14), (3199,15), (3559,15), (3707,15), (3801,15), (3803,15), (3819,15), (3829,15), (4051,15) and (4070,15). However this does not mean that our approach with different configurations cannot reach that MBCs. Also is visible in Table 12 that the best case is very close or equal to optimum case, the average case is also competitive and the coefficient of variation is small which means that our results have a high degree of confidence and are not dependent on the random seeds.

Table 14 allows to see a comparison between classic approaches ([2]) and the GA proposed for the calculation of the accumulated addition chains where *n* is in the range [1,1000]. There is a remarkable difference between the accumulated

addition chain computed by the binary and dichotomic method versus the optimal accumulated sum, however both are fast options to compute addition chains since they are deterministic approaches that do not imply any heuristic search or backtrack.

A comparison between the GA proposed and the more recently metaheuristics are presented in Table 15 and it is done using the best results for an accumulated addition chain in the range [1,*Z*] for $Z \in \{512, 1000, 1024, 2000, 2048, 4096\}$, where Δ is the difference between the best reported results and our best result. The metaheuristics chosen for the comparatison, that are part of the best strategies to construct Minimal Addition Chains or Minimal Brauer Chains, are: AIS [6], SA [16], EP [8,9], GA [7] and GA [24].

Table 12 Accumulated addition chain length and time obtained with the GA proposed for *n* in the ranges [1,512], [1,1000], [1,1024], [1,2000], [1,2048] and [1,4096]

n ∈	[1,512]	[1,1000]	[1,1024]	[1,2000]	[1,2048]	[1,4096]
AS						
Optimal	4924	10808	11115	24063	24731	54408
δ	0	0	0	3	4	24
Best	4924	10808	11115	24066	24735	54432
Average	4926.2581	10817.0645	11124.3226	24106.2581	24777.6129	54572.5791
Worst	4932	10838	11147	24185	24860	54807
Std. Dev.	1.1062	1.9333	1.8384	3.8266	4.1557	11.5808
CV	0.000224	0.000178	0.000158	0.000158	0.000167	0.000212
Time (s)						
Best	33.4870	138.9694	146.0097	567.8268	595.2965	2291.0749
Average	34.4035	141.9297	149.0738	577.4052	605.2810	2326.0918
Worst	39.0730	153.831	161.2811	602.2832	630.98	2338.8750
Std. Dev.	0.1024	0.3386	0.3412	0.8298	0.8331	2.1918
CV	0.002976	0.002385	0.002288	0.001437	0.001376	0.000942

Table 13 Accumulated addition chain length and time obtained with the GA proposed for *n* in the ranges [1,8192], [1,16384], [1,32768] and [1,65536]

n ∈	[1, 8192]	[1, 16384]	[1, 32768]	[1, 1180974]
AS				
Optimal	118624	256723	552119	24063
δ	86	296	1002	2992
Best	118710	257019	553121	1183966
Average	119166.1290	258315.1613	556553.7742	1192822.2258
Worst	119946	260417	561994	1206321
Std. Dev.	14.0339	14.4827	25.8029	57.0001
CV	0.000117	0.000056	0.000046	0.000047
Time (s)				
Best	8397.2797	29423.2867	99413.2074	323831.4285
Average	8525.8082	29879.2562	101004.4887	329262.0376
Worst	8704.8499	30449.9889	102897.8341	335349.5347
Std. Dev.	6.3230	15.76311	37.8138	98.9806
CV	0.000741	0.000547	0.000374	0.000300

Table 14 Comparison between classics approaches and the GA proposed for accumulated addition chains in the range [1,1000]

Strategy	Total length	Difference with l(n)
Binary [1]	11925	1117
Factor [1]	11088	280
Dichotomy [1]	11064	256
Fermat [1]	10927	119
Dyadic [1]	10837	29
Total [1]	10821	13
Proposed GA	10808	0
Optimal	10808	-

Table 15 Comparison of the best results of accumulated addition chains lengths

n ∈	optimal	AIS [4]	SA [13]	EP [6,7]	GA [5]	GA [18]	Proposed	Δ
[1,512]	4924	4924	-	4924	4925	4924	4924	0
[1,1000]	10808	10808	10823⊕	10808	-	10809⊕	10808	0
[1,1024]	11115	11120⊕	-	11115	-	-	11115	0
[1,2000]	24063	24108⊕	-	24070⊕	24124⊕	24076⊕	24066	4
[1,2048]	24731	4778⊕	-	24737⊕	-	24748⊕	24735	2
[1,4096]	54408	54617⊕	-	54487⊕	54648⊕	54487⊕	54432	55
total Δ								61

In Table 15 we can see with the symbol ⊕ when the difference between the accumulated addition chain and the results of other strategy is positive, meaning that our result is better. It can be seen that the quality of our result are better than the best reported in the state of art, however it is not possible to conclude that our approach have a better efficiency than the approaches used for the comparison due to lack of necessary information. Here it's necessary to remark the difference that we only use one configuration for the reported results, and the complexity of the system as the number of times that is checked the fitness of a BC, taken as as evaluation function or the test of how long is the chain and what is the obtained number in the r position, is limited to $\log_2(n)^2 * 26^{\log 2(\log 2(n))}$. A more detailed comparison can be done between our approach and other approaches, but they do not report accumulated time or how many operations are done inside the working chain or chains. It is need to say that: AIS [6], EP [8,9] and GA [24] do not report evaluations functions used, however in [8,9] are used 92000 comparison between individuals and in GA [24] are used 300000 comparisons.

But it is not only important the best results, it is also is important how much confidence we can have in the average case, taken as the coefficient of variation. For this reason we have made a comparison using the coefficient of variation between the GA proposed and the metaheuristics compared in Table 15. The comparisons of coefficients of variation for the accumulated addition chains lengths is presented in Table 16, where can be observed that all approaches have a high degree of confidence, and the one with the highest confidence is EP [8,9]. Also, with ⊕ are marked the coefficients of variation that have less confidence than our results.

Table 16 Comparison of coefficient of variation (*CV=stddev/average*) of the of accumulated addition chains lengths

n ∈	AIS [4]	SA [13]	EP [6,7]	GA [5]	GA [18]	Proposed
[1,512]	0.0001	-	0	0.0009⊕	0	0.0002
[1,1000]	0.0002⊕	0.0002⊕	0	-	0.0001	0.0001
[1,1024]	0.0002⊕	-	0	-	-	0.0001
[1,2000]	0.0002⊕	-	0	0.0002⊕	0.0001	0.0001
[1,2048]	0.0002⊕	-	0	-	0.0001	0.0001
[1,4096]	0.0002	-	0	0.0002⊕	0.0001	0.0002

6 Conclusions

In this paper, we have presented a novel approach to find Minimum Brauer Chains based on a GA with the following features: a representation based in FNS which has proven to be well suited to the problem of addition chains [16,26]; the use of distribution functions used by the neighborhood functions to focus their behavior; and a limit of evaluation functions according to the BC searched.

The use of a representation based on the FNS allows the implementation of neighborhood functions and recombination without a repairing process, giving the advantage to construct hard MBCs in a fast way, without modifying the search space. This properties allow to preserve the characteristics of individuals inside the recombination strategy and the local search. In this sense, the use of FNS open the possibilities of using other neighborhood functions or recombination strategies, like recombination of two points or uniform crossover.

Each one of the neighborhood functions used have distinct properties: NF_1 and NF_2 can be named as exploration functions while NF_3 and NF_4 can be seen as exploitation functions. Also NF_1 and NF_3 produce less dramatic changes than NF_2 or NF_4.

The problem of how to distribute the use of each neighborhood function inside the local search was delegated to a solution of the Diophantine equation 9, where the variable p_1 with $1 \leq i \leq 4$ represents the probability of use of the i neighborhood function. The use of mixture of neighborhood functions, allows to explore a wide range of behaviors of the algorithm and turned out to conclude that is better the use of a mixture of neighborhood functions than the use of a single neighborhood function.

The use of distribution functions allowed to focus the section of the chain where its necessary to make a change with more probability. For this reason we used a mixture of distribution functions inside the recombination strategy and neighborhood functions. In the case of the single point recombination, the position of point used could determine if it is going to be an exploration or an exploitation so the use of a distribution functions let center the main work that need to be done by the crossover.

In the case of the neighborhood functions, the use of a distribution functions to choose the i and j members of the chain allows to focus even more the goal of each neighborhood function. For the NF_1 and NF_2, the results allowed us to

conclude that the use of the DF_1 to set the random value of the chosen member in the chain is better than the use of a Gaussian distribution.

The setting of the optimal parameter values is a complex problem in the algorithm design task, that use to consume a lot of the development time. For example, the use of an exhaustive search need 5^4x3^4x286x31x2 tests, where 5^4x3^4 are all the possible combination between parameters, 286 the number of solutions of the Diophantine equation 9, 31 the number of times that each experiment should be repeated to get statistical significance and 2 the number of instances to be tested.

However the use of a fine tuning based on MCAs, relies to tune the values of the parameters on the study of the effect over the quality of the solution by the interaction between parameters, where the degree of interaction between parameters and the number of parameter values determine the number of experiment. In our case, the number of test using MCAS and the solution of a DE resulted in 32x286x31x2 number of experiments, which are fewer than the number of test in an exhaustive search. Another advantage of the use of MCAs is that the behavior of the algorithm is searched by the interaction between parameters and not just by guessing or feelings. For this reasons we recommend a fine-tuning process with the use of MCAs and DEs. In our case, as others [13,16,25,26], their use gave the possibility to discover excellent parameter values in an easy way, for that reason we suggest their use in order to save time and effort in the search of good parameters values.

The experimental results of the three benchmarks used demonstrated the strength of the GA proposed, in terms of the quality of the solutions. For the first benchmark consisting of 27 hard instances (that meet the requirement to be $c(r)$), we obtained the same or better results in all the instances as others competitive approaches reported in the literature, in fact all $l(n)$ was reached.

For the second experiment which consisted on 20 instances hard to optimize, we could reach 19 chains with length equal to $l(n)$.

For the third experiment which was done through calculating the accumulated addition chain length for a sequence of number in the range $[1,Z]$ for $Z \in \{512, 1000, 1024, 2000, 2048, 4096\}$, our proposed approach gives a competitive solution against the classics approaches: binary, factor, dichotomic, fermat, dyadic and total [2]. Additionally our approach is strong versus new metaheuristic strategies, achieving better quality results than: BackTracking [3], Genetic Algorithm [7], Artificial Inmune System [6], Mutation Selection [26] and Simulated Annealing [16].

Due to overall results in the three benchmarks, we can conclude that the proposed approach is a feasible solution to get MBC, however the search of efficient strategies to get MACs and MBCs is not finished and need more research and that the practitioners of cryptosystems use MBC in a more thoroughly way.

Acknowledgments. The authors acknowledge the support of access to the infrastructure of high performance computing of the Laboratory of Information Technologies Unit (Hidra) at CINVESTAV-Tamaulipas and to the hybrid cluster for supercomputing (Xiuhcoatl) at CINVESTAV.

This research was partially funded by the following projects: CONACyT 58554 - Cálculo de Covering Arrays, 51623 - Fondo Mixto CONACyT y Gobierno del Estado de Tamaulipas.

References

1. Bell, K.: Optimizing effectiveness and efficiency of software testing: A hybrid approach. Ph.D. thesis, North Carolina State University (2006)
2. Bergeron, F., Berstel, J., Brlek, S.: Efficient computation of addition chains. Journal de Théorie des Nombres de Bordeaux 6(1), 21–38 (1994)
3. Bleichenbacher, D., Flammenkamp, A.: An efficient algorithm for computing shortest addition chains. SIAM Journal of Discrete Mathematics 10(1), 15–17 (1997)
4. Clift, N.: Calculating optimal addition chains. Computing 91, 265–284 (2011) 10.1007/ s00607-010-0118-8, http://dx.doi.org/10.1007/s00607-010-0118-8
5. Cohen, M., Gibbons, P., Mugridge, W., Colbourn, C.: Constructing test suites for interaction testing. In: Proceedings of the 25th International Conference on Software Engineering, Portland, Oregon, USA, pp. 38–48 (May 2003)
6. Cruz-Cortés, N., Rodríguez-Henríquez, F., Coello Coello, C.A.: An artificial immune system heuristic for generating short addition chains. IEEE Transactions on Evolutionary Computation 12(1), 1–24 (2008)
7. Cruz-Cortés, N., Rodríguez-Henríquez, F., Juárez-Morales, R., Coello Coello, C.A.: Finding Optimal Addition Chains Using a Genetic Algorithm Approach. In: Hao, Y., Liu, J., Wang, Y.-P., Cheung, Y.-M., Yin, H., Jiao, L., Ma, J., Jiao, Y.-C. (eds.) CIS 2005, Part I. LNCS (LNAI), vol. 3801, pp. 208–215. Springer, Heidelberg (2005)
8. Domínguez-Isidro, S., Mezura-Montes, E., Osorio-Hernández, L.: Addition chain length minimization with evolutionary programming. In: Proceedings of the I Congreso Internacional de Electrónica, Instrumentación y Computación, Minatitlan, Veracruz, México (2011)
9. Domínguez-Isidro, S., Efren, M.M.: Addition chain length minimization with evolutionary programming. In: Proceedings of the 13th Annual Conference Companion on Genetic and Evolutionary Computation, Dublin, Ireland, pp. 59–60. ACM (2011)
10. Eiben, A., Smith, J.: Introduction to evolutionary computing. Springer (2003)
11. Flammenkamp, A.: Shortest addition chains repository (June 2012), http://wwwhomes.uni-bielefeld.de/achim/addition_chain.html
12. Gelgi, F., Onus, M.: Heuristics for Minimum Brauer Chain Problem. In: Levi, A., Savaş, E., Yenigün, H., Balcısoy, S., Saygın, Y. (eds.) ISCIS 2006. LNCS, vol. 4263, pp. 47–54. Springer, Heidelberg (2006), http://dx.doi.org/10.1007/
13. Gonzalez-Hernandez, L., Torres-Jimenez, J.: MiTS: A New Approach of Tabu Search for Constructing Mixed Covering Arrays. In: Sidorov, G., Hernández Aguirre, A., Reyes García, C.A. (eds.) MICAI 2010, Part II. LNCS, vol. 6438, pp. 382–393. Springer, Heidelberg (2010), http://dx.doi.org/10.1007/978-3-642-16773-7_33
14. Gordon, D.M.: A survey of fast exponentiation methods. Journal of Algorithms 27(1), 129–146 (1998), http://www.sciencedirect.com/science/article/pii/S0196677497909135
15. Guy, R.: Unsolved problems in number theory, 3rd edn., vol. 1. Springer (2004)

16. Jose-Garcia, A., Romero-Monsivais, H., Hernandez-Morales, C.G., Rodriguez-Cristerna, A., Rivera-Islas, I., Torres-Jimenez, J.: A Simulated Annealing Algorithm for the Problem of Minimal Addition Chains. In: Antunes, L., Pinto, H.S. (eds.) EPIA 2011. LNCS (LNAI), vol. 7026, pp. 311–325. Springer, Heidelberg (2011), http://dx.doi.org/10.1007/978-3-642-24769-9_23
17. Knuth, D.: The Art of Computer Programming. Seminumerical Algorithms, vol. 2. Addison-Wesley (1997)
18. Kuhn, D., Wallace, D., Gallo Jr., A.: Software fault interactions and implications for software testing. IEEE Transactions on Software Engineering 30(6), 418–421 (2004)
19. Mignotte, M., Tall, A.: A note on addition chains. International Journal of Algebra 5(6), 269–274 (2011)
20. Nedjah, N., de Macedo Mourelle, L.: Efficient Parallel Modular Exponentiation Algorithm. In: Yakhno, T. (ed.) ADVIS 2002. LNCS, vol. 2457, pp. 405–414. Springer, Heidelberg (2002), http://dx.doi.org/10.1007/3-540-36077-8_43
21. Nedjah, N., de Macedo Mourelle, L.: Finding Minimal Addition Chains Using Ant Colony. In: Yang, Z.R., Yin, H., Everson, R.M. (eds.) IDEAL 2004. LNCS, vol. 3177, pp. 642–647. Springer, Heidelberg (2004), http://dx.doi.org/10.1007/978-3-540-28651-6_94
22. Nedjah, N., de Macedo Mourelle, L.: Towards minimal addition chains using ant colony optimization. Journal of Mathematical Modelling and Algorithms 5, 525–543 (2006), http://dx.doi.org/10.1007/s10852-005-9024-z
23. Nedjah, N., Mourelle, L.: Efficient Pre-Processing for Large Window-Based Modular Exponentiation Using Genetic Algorithms. In: Chung, P.W.H., Hinde, C.J., Ali, M. (eds.) IEA/AIE 2003. LNCS, vol. 2718, pp. 165–194. Springer, Heidelberg (2003), http://dx.doi.org/10.1007/3-540-45034-3_63
24. Osorio-Hernandez, L., Mezura-Montes, E., Cruz-Cortes, N., Rodriguez-Henriquez, F.: A genetic algorithm with repair and local search mechanisms able to find minimal length addition chains for small exponents. In: IEEE Congress on Evolutionary Computation, CEC 2009, pp. 1422–1429 (May 2009)
25. Rangel-Valdez, N., Torres-Jiménez, J., Bracho-Ríos, J., Quiz-Ramos, P.: Problem and algorithm fine-tuning - a case of study using bridge club and simulated annealing. In: Dourado, A., Rosa, A.C., Madani, K. (eds.) IJCCI, pp. 302–305. INSTICC Press (2009)
26. Rodriguez-Cristerna, A., Torres-Jiménez, J., Rivera-Islas, I., Hernandez-Morales, C.G., Romero-Monsivais, H., Jose-Garcia, A.: A Mutation-Selection Algorithm for the Problem of Minimum Brauer Chains. In: Batyrshin, I., Sidorov, G. (eds.) MICAI 2011, Part II. LNCS, vol. 7095, pp. 107–118. Springer, Heidelberg (2011), http://dx.doi.org/10.1007/
27. Thurber, E.: Efficient generation of minimal length addition chains. SIAM Journal on Computing 28(4), 1247–1263 (1999)
28. Torres-Jimenez, J.: Covering array repository (June 2012), http://www.tamps.cinvestav.mx/~jtj/

Cellular Processing Algorithms

J. David Terán-Villanueva[1], Héctor Joaquín Fraire Huacuja[2],
Juan Martín Carpio Valadez[1], Rodolfo A. Pazos Rangel[2],
Héctor José Puga Soberanes[1], and José Antonio Martínez Flores[2]

[1] Instituto Tecnológico de León (ITL), Avenida Tecnológico s/No,
C.P. 37290, León, Gto. Mexico
david_teran01@yahoo.com.mx, jmcarpio61@hotmail.com,
pugahector@yahoo.com
[2] Instituto Tecnológico de Ciudad Madero (ITCM), Av. 1o. de Mayo s/No esq. Sor Juana Inés de la Cruz, C.P. 89440, Cd. Madero, Tam. Mexico
{automatas2002,r_pazos_r}@yahoo.com.mx,
jose.mtz@itcm.edu.mx

Abstract. In this chapter we propose a new class of cellular algorithms. There exists a variety of cellular algorithm approaches but most of them do not structure the search process. In this work we propose a cellular processing approach to solve optimization problems. The main components of these algorithms are: the processing cells (PCells), the communication between PCells, and the global and local stagnation detection. The great flexibility and simplicity of this approach permits pseudo-parallelization of one or several different metaheuristics. To validate our approach, the linear ordering problem with cumulative costs (LOPCC) was used to describe two cellular processing algorithms, whose performance was tested with standard instances. The experimental results show that the cellular processing algorithms increase solution quality up to 3.6% and reduce time consumption up to 20% versus the monolithic approach. Also the performance of these algorithms is statistically similar to those of the state-of-the-art solutions, and they were able to find 38 new best-known solutions (i.e., not previously found by other algorithms) for the instances used. Finally, it is important to point out that these encouraging results indicate that the field of cellular processing algorithms is a new and rich research area.

1 Introduction

The goal of cellular computing is providing new means for doing computation more efficient [20]. The main characteristics of this approach are: simplicity, vast parallelism and locality. Simplicity means that processing is carried out by a set of simple structured cells. The massive parallelism indicates that a high number of

P. Melin, O. Castillo (Eds.): *Soft Computing Appli. in Optimization,* STUDFUZZ 294, pp. 53–74.
DOI: 10.1007/978-3-642-35323-9_3

cells are used. And locality implies that each cell can communicate with a few cells, commonly close to it.

Recently several cellular algorithm approaches to solve optimization problems have been proposed. This research area could be called cellular optimization, which includes cellular genetic algorithms, co-evolutionary algorithms, island evolutionary algorithms, and ant colony algorithms.

Cellular genetic algorithms use a cell structured population which allows communication among the locally near cells, aiming at exploring different sections of the search space with an appropriate exploration and exploitation balance [10] [1] [12]. The main limitation of this approach is that the search process is carried out based on a single heuristic strategy.

In other hand, co-evolutionary algorithms divide a solution in species, where a complete solution is composed from one element of each species. Where the term co-evolutionary refers to simultaneous evolution of several species, and usually the fitness of a specific species depends on the evolution of other species. Co-evolutionary algorithms can be cooperative or competitive, where the cooperative ones reward or punish the interaction among the different species, and the competitive ones reward or punish each element inside one species without considering their interaction with other species [19] [14] [23]. As we can see, in this case the algorithm evolves multiple populations which are constituted by sub-parts of the solutions. The application of this approach is limited to a single population heuristics.

Sipper [20] states that the ants in an ant colony system [4] are like processing cells; where each cell (ant) builds a solution, and they all share knowledge through a global memory structure. However, this approach is limited to a specific constructive population algorithm.

As we can see, the cellular genetic and the co-evolutionary algorithms use cell structured populations and a single search space process. In another hand, the ant colony system uses an unstructured population and multiple processing cells, but it has low flexibility to adapt to other metaheuristics.

In this work we propose a new class of cellular processing algorithms; which include multiple processing cells that explore different regions of the search space. Each processing cell can be implemented using population or search based heuristics or a mixture of them. The linear ordering problem with cumulative costs (LOPCC) will be solved for exemplifying this new class of algorithms.

This chapter is organized as follows. In Section 2 the linear ordering problem with cumulative costs is reviewed. The proposed new class of cellular algorithms is described in Section 3. Section 4 contains a review on two metaheuristics used to implement the cellular processing algorithms. Sections 5 and 6 describe the two cellular processing algorithms used to validate the new cellular processing proposal. Finally, Sections 7 and 8 contain the experimental results and the conclusions of this work respectively.

2 The Linear Ordering Problem with Cumulative Costs (LOPCC)

The linear ordering problem with cumulative costs (LOPCC) is originated in wireless communications. In this context, the wireless devices have to communicate to

a base station in order to be identified. To this end, the Universal Mobile Telecommunication Standard (UMTS) adopted the code division multiple access technique where each device has a specific code.

However, due to simultaneous communication and radio propagation, a distortion is induced on all the wireless devices. Then, each device produces distortion on the rest of the devices in different proportions. And so the need arises to keep the distortion as low as possible, while ensuring reception for each wireless device.

There is a technique designed to keep a low level distortion called successive interference cancellation (SIC). This technique detects one device at a time and then its interference is removed, so the rest of the devices would have less interference [3].

Then the problem arises of finding the order of detection for the devices that produces the minimal overall interference, while keeping a desirable level of reception for each device. This problem is addressed as a joint power-control and receiver optimization (JOPCO), which is equivalent to the NP-hard linear ordering problem with cumulative costs (LOPCC) [2][3]. Formally LOPCC is defined as:

Given a complete digraph $G = (V, A)$ with no negative arc costs $c_{\pi_i \pi_j}$ and nodes with no negative costs d_{π_i}, the problem is to find a permutation $\pi = (1, 2, \cdots, n-1, n)$ that minimizes:

$$LOPCC(\pi) = \sum_{i=1}^{n} \alpha_{\pi_i}$$

where

$$\alpha_{\pi i} = d_{\pi_i} + \sum_{j=i+1}^{n} \alpha_{\pi_i} + c_{\pi_i \pi_j} \text{ for } i = n, n-1, n-2, \cdots, 1$$

With respect to the wireless devices application, $c_{\pi_i \pi_j}$ represents the interference of device π_j on device π_i, and α_{π_i} represents the power of the signal emitted by device π_j. In [3] is proved that this problem is NP-hard.

3 Cellular Processing Approach

The main idea in the cellular processing approach is to split a sequential algorithm into several pseudo-parallel processing modules. The pseudo-parallel execution permits exploring different regions of the solution space. Also the continuous verification of the stagnation conditions avoids wasting time on unnecessary tasks.

The main components of a cellular processing algorithm are: processing cells (PCells), the communication between PCells, and the global and local stagnation detection.

Cellular processing algorithms simulate a parallel execution of a set of PCells that explore different regions of the solution space. A cellular algorithm can be homogeneous or heterogeneous, depending on the kind of processing cells that compose it. We say that it is homogeneous if all the PCells are implemented using the same heuristics, and heterogeneous otherwise.

Also the computational effort of the processing cells can be balanced or unbalanced. In order to induce diversification through the communication between processing cells, an unbalanced configuration might be used. Each processing cell improves its own solution or set of solutions, and its execution is stopped once a stagnation condition is reached.

At a certain point the processing cells will communicate with each other. If the communication is carried out during the execution of the processing cells, we will call it *on-line*. And, if the communication is executed once all processing cells have reached a stagnation condition, then we will call it *off-line*. The communication could help to increase intensification or diversification in the search process.

There are several decisions to make in order to implement a cellular processing algorithm. The first issue is the selection of the heuristics to be used as processing cells (PCell) and if the PCells will be homogeneous or heterogeneous.

An advantage of our approach is the freedom of choosing any kind of heuristics as a PCell, whether a population type or search based type. Also we need to define the size of the processing cell and if the PCells are to be balanced or unbalanced. The size of the PCell can be defined taking into account the number of iterations, processed solutions, executions of local searches, execution time, etc.

Communication is another main component of cellular processing algorithms. Here we can choose to make the communication *on-line* or *off-line*. The *on-line* communication can be implemented using shared solutions, or other mechanisms of shared memory. The *off-line* communication can be implemented through crossover operators, path-relinking, combinations, etc. It is necessary to define the time in which the communication is to be carried out for both alternatives, as well as the purpose of the communication: intensification or diversification.

The last issue is local and global stagnation detection, which has two purposes in two different levels. To avoid the waste of time, local stagnation detection stops the PCells once they do not produce improvement in the local solution or population. Once a PCell reaches a stagnation condition, it is ignored by the algorithm until every PCell is stagnated. At this point a processing cell communication might occur and every PCell must be restarted. Similarly global stagnation detection has the purpose of monitoring the global algorithm contribution to the improvement of the global solution. Stagnation detection can be implemented through: number of consecutive iterations without improvement in the quality of the best (local or global) solution, number of generations without improvement in the population quality, time limit, etc.

The general structure of the cellular processing algorithm is shown in Algorithm 1. As we can see, in lines 1 to 3 all the PCells must be started. Line 5 controls the global search and stops it when the stagnation condition is reached. Line 6 verifies if any cell is not locally stagnated, so this loop will continue until every PCell is stagnated. Line 7 to 11 implements a pseudo-parallel execution of each non-stagnated PCell. Lines 12 and 14 show the place where the communication might be carried out. Finally line 15 restarts each PCell so that the global search continues.

Algorithm 1: Cellular processing algorithm

```
1.        For {i = 1 to numberOfPCells}
2.          Start(PCell_i())
3.        EndFor
4.
5.        While(GlobalSearchNotStagnated())
6.          While(AnyCellNotStagnated())
7.            For {i = 1 to numberOfPCells}
8.              If {IsNotStagnated(PCell_i())}
9.                PCell_i()
10.             EndIf
11.           EndFor
12.           PCellCommunicationOnLine()
13.         EndWhile   //Local search
14.         PCellCommunicationOffLine()
15.         ReStartPCells()
16.       EndWhile    //Global search
```

We want to emphasize that the pseudo-parallel execution of the PCells permits exploring different regions of the solution space. And the continuous verification of the stagnation conditions avoids wasting time on unnecessary tasks, helping produce high performance algorithms. The PCells can be implemented using any heuristic strategy, which provides more flexibility than other cellular approaches.

4 Scatter Search and GRASP

In this section the metaheuristics used in the cellular processing algorithms proposed are described.

4.1 Scatter Search

The scatter search heuristics was first proposed by Fred Glover in [11]. This is a procedure that tries to evolve and improve a reference set through several methods and uses a small amount of randomness in order to find global optimal solutions.

Our motivation to use a scatter search algorithm is that it is a population algorithm that has several customizable functions. Also its reference set *RefSet* is constituted by quality and diversity solutions. This property permits having better control over the intensification-diversification balance. The scatter search algorithm uses a reference set, which can be constructed using quality and diverse elements.

Also, instead of using a genetic algorithm, as an alternative we choose the scatter search approach, because of its multiple configurable methods, which permits fully customizing the algorithm.

The methods used in the scatter search algorithm implementation are described in the following sections.

4.1.1 Diversification Generation Method

A reactive greedy construction was implemented for the *diversification generation method* [17][15]. Further detail can be found in Section 4.2.

4.1.2 Improvement Methods

For the improvement methods in our cellular processing scatter search, three local searches were used: a percentage-of-critical-elements local search (PCLS), a stagnation local search (SLS) and an optimal local search (LOS). These local search algorithms work together to form two different configurations of composite local search, each one of them with a particular purpose [22].

The percentage-of-critical-elements local search (PCLS) builds a set C of critical elements of the current permutation ordered according to the α values considered to determine their objective values. The most critical element $c \in C$ is selected, inserted in a position that improves the objective value of the permutation and removed from C. This process continues until a certain percentage of critical elements are removed from C, which we set at 30%.

The stagnation local search (SLS) selects a random element of the current permutation in order to be inserted in a position that improves the objective value of the permutation. This process is carried out until a given number of iterations without improvement is reached. We define 10% of the instance size as the limit of iterations without improvement.

The local optimal search (LOS) builds a set C of critical elements of the current permutation, based on the α values considered to determine their objective values. The most critical element $c \in C$ is selected, inserted in a position that improves the objective value of the permutation and removed from C. Once C is empty, it is rebuilt if any improvement was achieved during the search process, otherwise the local optimal search ends.

The $Improvement_1$ method is carried out after the *Diversification Generation* method and is applied to all the solutions in the pool of solutions. This improvement method is formed by a composite local search of PCLS and SLS. The purpose of this improvement method is to apply a fast and diverse improvement (PCLS and SLS respectively) in order to obtain diverse good solutions for the pool of solutions.

$Improvement_2$ is carried out after the *Solution Combination* method and the *Interaction* method, and it is applied to all the solutions that result from both methods. $Improvement_2$ is formed by a composite local search: a composition of PCLS, SLS and LOS. The purpose of this improvement method is to apply a fast, diverse and local optimal search (PCLS, SLS and LOS respectively) in order to obtain a variety of local optimal solutions.

4.1.3 RefSet Update$_1$ Method

It initially builds a set Q with the best quality solutions in the *pool.* A local optimal search (LOS) is applied to these solutions when they are incorporated to the reference set. This local search was not carried out during the creation of the *pool* of solutions in order to produce diverse solutions. Once the elements in Q are selected, *RefSet Update$_1$* proceeds to build a set D of diverse solutions, including the elements of the *pool* with the highest average distance to set Q. The distance between permutations $q \in Q$ and $d \in D$ is defined as follows:

$$\sum_{i=1}^{n} ABS(Pos(i,q) - Pos(i,d))$$

where *Pos(i, p)* is the position of element *i* in permutation *p.*

4.1.4 Stagnation Condition

This condition detects if $Q \subset RefSet$ has been modified with new solutions, so if no new solution has been added to Q, then we consider $RefSet$ to be stagnated.

4.1.5 Subset Generation Method

This method chooses for combination all the pairs of solutions in Q plus all the pairs (q, d) such that $q \in Q$ and $d \in D$.

4.1.6 Combination Method

In order to describe how the permutations are combined, consider two permutations π_1 and π_2, an index Ind_r for the position of each element in π_r, and the set of differences $Dif = \{ d_k \mid d_k = \alpha_{Ind_{2_k}} - \alpha_{Ind_{1_k}} \text{ for } k = 1, \cdots, n\}$. Now element $k^* = arg_{min}(Dif)$ is selected, meaning that $\alpha_{Ind_{2_{k^*}}}$ and $\alpha_{Ind_{1_{k^*}}}$ have the largest difference. Therefore, the best improvement will be reached if element k^* is placed in position $Ind_{2_{k^*}}$. A new solution π is built inserting the element placed in position $Ind_{1_{k^*}}$ of permutation π_1, in position $Ind_{2_{k^*}}$, and d_{k^*} is eliminated from Dif. This process continues until we have taken 15% from Dif and subsequently an *Improvement$_2$* method is carried out.

4.1.7 RefSet Update$_2$ Method

It includes new solutions in Q if they have better quality than the worst solution in Q, if so then the solutions to be replaced in Q might be included in D. Also if the new solution is not better than the worst solution in Q, then it might be included into D if it has a higher average distance to set Q (See Section 4.1.3).

4.2 GRASP

Feo and Resende propose the GRASP methodology in [7] and [8], and the acronym was coined in [9]. GRASP is a multi-start algorithm which consists basically by a construction phase and a local search, see Algorithm 2.

Algorithm 2: General GRASP structure

```
bestSol = Huge_Value
For(MaxIterations)
  sol = GraspConstruction()
  sol = LocalSearch(sol)
  if (cost(sol) < bestSol)
    bestSol = cost(sol)
  endif
EndFor
```

4.2.1 GRASP Construction

The GRASP construction is one of the main parts of a GRASP algorithm. For LOPCC an initial construction consists of a permutation that is built from the last position to the first, in order to evaluate part of the objective function value due to its cumulative costs. There are several ways to create a *GraspConstruction*, for this work particularly we use a reactive GRASP.

Resende in [17] and Paris in [15] propose a reactive GRASP, whose general structure is next described. The first element (last position in the permutation) is selected randomly. And for each remaining position, the next process is carried out. Each permutation element is evaluated using a GRASP function, and a restricted candidate list is generated with the elements that surpass a shold $RCL = Cost_{min} + \beta(Cost_{max} - Cost_{min})$. Then a random element from the restricted candidate list is selected to be placed at the current position of the construction, and the same process is repeated for the next position until the solution construction is completed.

A GRASP is called reactive when it chooses the β value using statistical information of the performance of previous solutions obtained with each β value. The general structure of the greedy construction is shown in Algorithm 3.

Algorithm 3: Reactive greedy construction

```
          Output π
1         cost = 0
          InsertElement(Random, π)
3         cost += CalculateCostOfInsertion()
          While {notFinished(π)}
5           β = SelectBeta()
            RCL = CreateCandidateList(β)
7           nextElement = SelectRandomElement(RCL)
            InsertElement(nextElement, π)
```

```
9           cost += CalculateCostOfInsertion()
            cost += TryToImproveSolutionSoFar()
11          UpdateBeta()
          EndWhile
```

The selection and update of the β values are performed in lines 5 and 11 respectively; the probabilities associated with the choice of selecting one of the possible m values of β are: $p_i = 1/m$, for $i = 1, \ldots, m$. Now, let z^* be the objective function value of a new solution and Z_i be the average of the objective function value of all the solutions found using β_i, for $i = 1, \ldots, m$. The selection probabilities are periodically reevaluated to $p_i = q_i / \sum_{j=1}^{m} q_j$, where $q_i = z^*/Z_i$ for $i = 1, \ldots, m$. And so the value of q_i will be larger for those values of β that yields to the best solutions. The *CreateCandidateList()* method creates the candidate list using the β values produced by the previous process. The *SelectRandomElement()* method chooses the next element randomly from the restricted candidate list (RCL) in the construction of the solution. The *InsertElement()* function in line 8, inserts *NextElement* in the next position to be included in permutation π, it is important to remember that permutation π is constructed in reverse order due to the cumulative costs. The *TryToImproveSolutionSoFar()* method is a partial local search that tries to find a better partial solution by trying to insert the new element in other position.

5 Homogeneous Cellular Processing Algorithm

In this section a homogeneous cellular processing algorithm is described. It uses PCells based on the scatter search metaheuristic. And it was motivated for reducing the time consumption of a sequential scatter search algorithm, when used to solve large scale instances of LOPCC. The homogeneous algorithm executes multiple small scatter search PCells by reducing the size of their reference sets.

For this particular homogeneous algorithm the next configuration was used. In each iteration of the algorithm, every PCell is executed as long as it is not stagnated. The PCells are homogeneous because they all use a scatter search heuristics, and their computational effort is balanced. Each PCell evolves a different set of solutions, which must be initialized before its execution. The communication is carried out *off-line*, so it will be executed once every PCell has stagnated. Several stagnation detection strategies where applied for testing.

Algorithms 4, 5, 6 and 7 show the detailed description of the main methods used: processing cell starter, processing cell, and cellular communication.

Algorithm 4: PCell starter

```
Output Pool
For {i=0 to poolSize}
  newSol = DiversificationGeneration()
  Pool += Improvement1(newSol)
EndFor
```

Algorithm 5: PCell processing

```
Input i
PCellSol_i=SubSetGeneration(i)
NPCellS_i=SolCombination(PCellSol_i)
ImprovedPCellSol_i=Improvement_2(NPCellS_i)
RefSetUpdate(ImprovedPCellSol_i,i)
```

Algorithm 6: PCell off-line communication

```
Repeat
 For {i=0 to numberOfPCells}
  For {j=0 to numberOfPCells}
   If {i ≤ j}
    NewSol = Communication(i,j)
    UpdateWorstCell(i,j,NewSol)
   EndIf
  EndFor
 EndFor
Until {NoBetterSolFound()}
```

The communication process receives the identifiers (i, j) of the PCells to communicate. Once the PCells are identified the best solutions obtained for each one (π^i, π^j) are combined to produce a new solution π^{**}. If $Cost(\pi^{**}) < Min(Cost(\pi^i), Cost(\pi^j))$, then π^{**} will replace the worst solution in $\{\pi^i, \pi^j\}$. In this process the combination of solutions is carried out using a truncated path-relinking. In this process parameter γ determines the truncation level of the path between π^i *and* π^j, which in our algorithm is set to 10% [17].

The PCells are executed in each iteration on a fixed order from 1 to n, until every PCell stagnates. The individual stagnation technique has the advantage of low time consumption, because, once k PCells have stagnated, the rest of the process continues without wasting time on those k PCells.

Algorithm 7: Communication (truncated path-relinking)

```
Input {i, j}
Output π**
Let: Ind(π,k) be a function that returns
the position of element k in solution π
D = ∅
For {k = 1 \to n}
 If {Ind(π^i,k) ≠ Ind(π^j,k)}
  d_k=α^j_{π^j Ind(π^j,k)} - α^i_{π^i Ind(π^i,k)}
  D = D ∪ d_k
 EndIf
EndFor
For {m = 1 \to n*γ}  where 0 < γ ≤ 1
 d_{k*} = arg_min(D)
```

```
    π* = InsertMove(π^i, Ind(π^i, k*), Ind(π^j, k*)
    D = D \ {d_k*}
  EndFor
  π** = Improvement_2(π*)
```

The communication process is carried out once all the PCells have reached a stagnation condition. The communication between PCells is carried out by updating the reference set of each cell with new solutions. These new solutions are generated using a truncated path-relinking among the best elements of each PCell. Once the communication has ended and if the stop condition is not reached, a restart of the reference set for each PCell is carried out. This *Re-start* method is used to produce a new reference set of solutions for each processing cell. The new solutions are produced with the *Solution Combination* method of the scatter search, but in this case all the new solutions substitute the old ones except for the best solution of each processing cell.

The general structure of the cellular processing scatter search algorithm is shown in Figure 1.

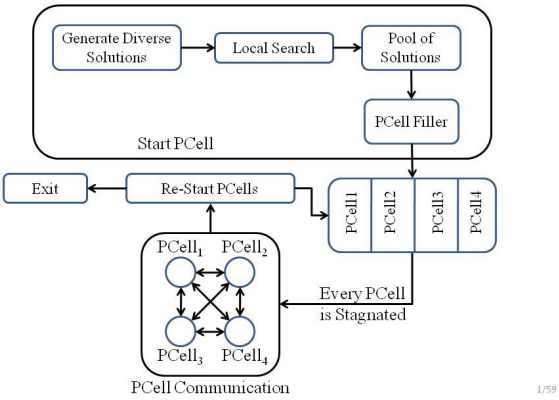


Fig. 1 Structure of the cellular processing scatter search

6 Heterogeneous Cellular Processing Algorithm

In this section a heterogeneous cellular processing algorithm is described. It uses PCells based on both the scatter search and GRASP metaheuristics. We configure it as an unbalanced algorithm, and so the GRASP processing cells (GRASP PCells) will iterate 50% more than the scatter search processing cells (SS PCells). Also as in the previous algorithm the communication is an off-line process, using the same combination technique as in the homogeneous algorithm. The stagnation detection for the GRASP PCells is determined according to a number of iterations without improvement; while for the SS PCells the local stagnation condition is

reached when in one execution of the processing cell the local best is not improved.

The SS PCells are the same as the ones used in our previous algorithm and the GRASP PCells were designed as shown in Algorithm 8.

Algorithm 8: GRASP PCell

```
1.        Input {i}
2.        Output {bestValue}
3.        If (GenerateNew_i == true)
4.          GRASPSol_i = GRASPConstruction()
5.        EndIf
6.        IterStag = 0.3 * SolSize
7.        For {k = 1 To maxIter And j < IterStag}
8.          j++
9.          GRASPSol_i = Perturbation(GRASPSol_i)
10.         GRASPSol_i = Improvement_2(GRASPSol_i)
11.         GRBest_i = Best(GRASPSol_i, GRBest_i)
12.         If (Improvement)
13.           j = 0
14.         EndIf
15.       EndFor
16.       If (j==IterStag && NoBetterSolution(i))
17.         Stagnated_i = true
18.       Else
19.         Stagnated_i = false
20.       EndIf
```

In line 3 of this algorithm we evaluate if a new construction is needed; and if that condition is true, a reactive greedy construction is carried out as shown in Algorithm 3. Then the loop in line 7 is carried out a number of iterations, or until a certain number of consecutive iterations without improvement occur. Inside this loop a perturbation followed by an *improvement*$_2$ (see section 4.1.2) are carried out, lines 9 an 10. The perturbation moves 20% of the elements of the solution *i* to random positions. Then an update of the best solution is calculated (line 11). Lines from 12 to 14 check if there is an improvement for reseting *j*. Once the loop ends, a stagnation detection of the GRASP processing cell is carried out. The algorithm considers that the *GRASP PCell* is stagnated if *j* has reached *IterStag* and if *NoBetterSolution* has been found in the last entire execution of the *GRASP PCell*.

7 Experimental Results

In this section we describe: the experimental settings, the preliminary experimentation, and the evaluation of the performance of the homogeneous and heterogeneous cellular processing algorithms.

7.1 Experimentation Settings

The algorithms were implemented in C, and two different sets of experiments were carried out. One set was designed to study the impact of the different parameters of the homogeneous cellular processing proposal. The other set was designed to compare the performance of the homogeneous cellular processing algorithm (HoCPA) and the heterogeneous cellular processing algorithm (HeCPA) with respect to the state-of-the-art algorithms to solve LOPCC.

For the first set of experiments a computer with a Phenom X4 955 3.2 GHz processor with 2 GB of RAM was used. And for the second one a computer with dual Xeon processors at 3.06 GHz and with 4 GB of RAM was used.

The instances used for the experiments are:

- **UMTS.** These are instances from the group of telecommunications of the engineering school from the University of Padua, related with the order of detection for UMTS networks [3]. These instances consist of four sets of size 16, the characteristics for each set of instances are: synchronous and asynchronous, with and without scramble; and their optimal values are known.
- **Random.** Instances generated randomly with a uniform distribution proposed by Reinelt [16]. There are three sets of random instances, each one of size 35, 100 and 150 respectively; their optimal values are unknown.
- **LOLIB.** These instances come from input-output tables from the European economy [13] and are a well-known set of LOP instances. We use 48 instances of sizes 44 to 60, 30 of size 44, 4 of size 50, 11 of size 56, and 3 of size 60; their optimal values are unknown.

7.2 Impact of Different Parameters, in the Performance of the Cellular Processing Algorithms

A preliminary experimentation was conducted in order to identify the impact of a set of parameters on the performance of the cellular processing algorithms. In this preliminary experimentation a set of 49 LOLIB instances was used. The parameters analyzed in these experiments were: *the number of processing cells (nc)*, *the processing cell size (cs)* (the running time of each processing cell) and *the stagnation detection (sd) for the processing cells.*

The number of processing cells (*nc*) test is carried out in order to observe the performance produced with different numbers of processing cells.

The processing cell size (*cs*) test is designed to determine the performance produced with different processing cell sizes. The processing cell size is related with the number of combinations of Table 1. For example, a number of combinations equal to 60 (experiments from 1 to 5) means that, in one processing cell execution, 60 combinations are produced. And therefore, we are taking this value as the size of the processing cell.

The stagnation detection for the processing cells (*sd*) test considers two main factors: the percentage of improvement and the amount of new solutions produced. The percentage of improvement tries to identify stagnation if no better

solution has been produced with at least a certain percentage of improvement. And the amount of new solutions produced aims at detecting stagnation when a certain amount of new better solutions has not been produced.

Table 1 shows the configurations used for the three parameters in the preliminary experimentation. The first column indicates the number of experiment, the second column shows the amount of processing cells used for the experiment, while the third and forth columns show the quantity of quality and diversity solutions in each processing cell. The fifth column contains the overall number of combinations ($PCells * (Q * (Q - 1) + Q * D)$), the sixth one indicates the overall number of solutions used ($PCells * Q + D$), and the seventh column shows the stagnation detection criterion used by each processing cell.

For the stagnation detection column, 1Q and 2Q mean finding at least one or two solutions respectively that replace a quality element in the processing cell. The values 0.1%, 0.01% and 0.001% Improvement means finding a solution that is at least 0.1%, 0.01% and 0.001% better than any quality solution. If these criteria are met then we state that the processing cell is not stagnated.

Table 1 Configurations used in the experiments: *number of processing cells (nc), processing cell size (cs)* and *stagnation detection for the processing cells (sd)*

Experiment No.	PCells	Q	D	Total of Combinations	Total of Solutions	Stagnation Detection
1	1	6	5	60	11	1Q
2	2	5	2	60	14	1Q
3	3	4	2	60	18	1Q
4	4	3	3	60	24	1Q
5	5	3	2	60	25	1Q
6	8	3	3	120	48	1Q
7	10	3	2	120	50	1Q
8	10	3	2	90	50	1Q
9	10	4	2	200	60	1Q
10	10	4	2	140	60	1Q
11	10	4	3	240	70	1Q
12	10	4	3	180	70	1Q
13	10	4	2	200	50	0.1% Improvement
14	10	4	2	200	50	0.01% Improvement
15	10	4	2	200	60	0.001% Improvement
16	10	4	2	200	60	2Q

Table 2 shows the results obtained for the experiments 1 to 7 of Table 1, where we study the impact of the *number of processing cells (nc)*. The first column indicates the experiment number, the second one shows the average percentage error with respect to the best-known solutions reported in [6], the third column presents the number of best-known solutions found, and the last one shows the average time for instance solution in CPU seconds.

In this table we can see that experiments 5, 6 and 7 are the ones with the best average error. The experiments with the best times are 3 and 5. However, it is important to remember that experiments from 1 to 5 are experiments with 60 combinations, and in this group experiment 1 (one processing cell) obtains 23 best-known solutions, while experiment 5 finds 22 best-known solutions in almost half the time used in experiment 1. Furthermore, experiment 7 obtains a better average error and 23 best-known solutions and uses less time than experiment 1, despite producing 120 combinations in comparison with the 60 combinations of experiment 1. So in this experiment we can see the advantage of using a cellular processing approach instead of a monolithic approach.

Table 2 Quality and efficiency observed in the number of processing cells (*nc*) experiments

Experiment No.	Avg. Err.	# of Best-Known Solutions	CPU. Sec.
1	0.656	23	29.17
2	0.656	21	15.451
3	1.1	21	14.743
4	0.654	22	15.413
5	0.633	22	14.921
6	0.632	23	25.034
7	0.634	23	23.172

As we can see in experiments 6 and 7 the cellular processing approach increases solution quality by 3.6% and 3.3% respectively with respect to the monolithic approach (experiment 1). In another hand our approach reduces time consumption by 14% and 20% respectively.

Table 3 shows the results obtained for experiments 7 to 12 of Table 1, where we study the impact of *the processing cell size (cs)*. The first column indicates the experiment number, the second one shows the average percentage error with respect to the best-known solutions reported in [6], the third column presents the number of best-known solutions found, and the last one shows the average time for instance solution in CPU seconds.

The experiments are compared pairwise because each pair has the same number of solutions and a different number of combinations. As we can see, there is an average reduction of time of 1.5 seconds comparing experiment 7 versus 8, 9 versus 10, and 11 versus 12. Also the best average error is found with experiment 11,

which is the experiment with the largest number of combinations (240). However, experiment 12 is the one that finds the largest number of best-known solutions (24). Still, this behavior seems to be a fluke, because if we compare all the previous pairs of experiments (7,8) (9,10), they tend to reduce the number of best-known solutions found, from 23 to 21 or 22. According to these results there is a small time saving for the three comparisons.

Table 3 Quality and efficiency observed in the processing cell size (cs) experiments

Experiment No.	Avg. Err.	# of Best-Known Solutions	CPU. Sec.
7	0.634	23	23.172
8	0.638	21	22.428
9	0.632	23	25.262
10	0.657	22	23.553
11	0.043	23	28.434
12	0.632	24	26.276

Table 4 shows the results obtained for experiments 9 and 13 to 16 of Table 1, where we study the impact of the *stagnation detection* for the processing cells (*sd*). The experiments selected for this study contain the same amount of quality and diversity solutions as well as the number of combinations, the only variable is their stagnation detection criterion. Here we can see that there is not really a tendency for these experiments. We expected there should be an increasing average error, a decreasing number of best-known solutions and a decreasing CPU time. But the only tendency observed was the decreasing number of best-known solutions of experiment 16 and also a slight decrease in CPU time.

According to these results, there is not an important reduction of CPU seconds and so we will use the configuration of the experiment 9, for one of the cellular processing algorithms.

Table 4 Quality and efficiency observed in the stagnation detection (sd) experiments

Experiment No.	Avg. Err.	# of Best-Known Solutions	CPU. Sec.
9	0.632	23	25.262
13	0.632	23	25.272
14	0.632	23	25.294
15	0.632	23	25.279
16	0.634	22	24.522

7.3 Cellular Processing Algorithms Performance

Table 5 shows the comparative average performance of the homogeneous and heterogeneous cellular processing algorithms (HoCPA and HeCPA) versus a hypothetical algorithm (HA) capable of obtaining all the best-known solutions reported in [6]. This table contains the average percentage error, the number of best-known solutions found, the number of new best-known solutions produced, and the average time for instance solution in CPU seconds. The time comparison was performed against the EvPR algorithm, which is the best performance algorithm reported in [6]. We make comparisons versus the HA algorithm, because the results in [6] do not include the objective values obtained by EvPR for each instance.

Table 5 Comparative average performance of cellular processing algorithms (HoCPA, HeCPA) versus the HA algorithm

Algorithm	Avg. Err.	# Best-Known or Optimal Solutions	New Best-Known Solutions	CPU Sec.
UMTS				
HoCPA	0	100	-	0.863
HeCPA	0	100	-	0.822
HA	0	100	-	1.62 (EvPR)
LOLIB				
HoCPA	0.01	43	16	33.41
HeCPA	0.01	42	16	33.13
HA	382057.38	33	-	32.34 (EvPR)
Rnd 35				
HoCPA	0.37	22	0	4.66
HeCPA	0.39	21	0	3.59
HA	0	25	-	3.75 (EvPR)
Rnd 100				
HoCPA	1.78	10	10	431.95
HeCPA	1.97	11	11	360.29
HA	1.49	14	-	351.38 (EvPR)
Rnd 150				
HoCPA	4.75	11	11	1628.75
HeCPA	5.61	8	8	1074.24
HA	3.26	14	-	1127.24 (EvPR)

As we can see, for the UMTS instances our algorithms were able to find 100 optimal solutions out of 100. Also the time spent on these instances was almost halved for the cellular processing algorithms.

For the LOLIB instances, the HoCPA and HeCPA algorithms have lower average error than the hypothetical algorithm and find 43 and 42 best-known solutions respectively. Also both cellular processing algorithms find 16 new best-known solutions. However, they use about one second more than EvPR.

The Random instances of size 35 are the only scenario where the cellular processing algorithms were not able to find new best-known solutions. HoCPA and HaCPA find 22 and 21 best-known solutions respectively.

For the Random instances of size 100, HoCPA finds 10 new best-known solutions while HeCPA finds 11 new best-known solutions, and it uses 70 seconds less than HoCPA, but still uses 9 seconds more than EvPR.

On the Random 150 instances, HoCPA finds 11 new best-known solutions and HeCPA finds 8. In this scenario HoCPa uses 501 seconds more than EvPR but HeCPA uses 53 less seconds than EvPR. Table 7 and 8, at the end of this chapter, show an update of the best known solutions for the LOLIB and Random sets of instances respectively.

Now for the Wilcoxon test we compared HoCPA (excluding HeCPA) versus HA, because HoCPA obtains more best-known solutions than HeCPA. Table 6 shows the Wilcoxon test results when HoCPA and HA are compared for all the sets of instances.

As we can see all the tests are statistically equivalent, because the sum of ranks is not equal to or greater than the reference value for any algorithm. These results indicate that the performance of both algorithms is statistically equivalent.

Table 6 Wilcoxon test to compare the performance of the Homogeneus Cellular Processing Algorithm (HoCPA) versus the Hypothetical Algorithm (HA)

Instances	Algorithm	Sum of ranks	Reference Value (Significance 10%)
LOLIB	HoCPA	99	101
	HA	37	
Rnd 35	HoCPA	0	–
	HA	6	
Rnd 100	HoCPA	132	225
	HA	193	
Rnd 150	HoCPA	121	225
	HA	204	

8 Conclusions and Future Work

In this work we propose a new class of cellular algorithms. There are several cellular algorithms in the literature, but they are bound to their heuristics, and in most cases the cellularization is in the structure of the population and not in the process. The ant colony system does cellularize the process, but due to its low flexibility, it is difficult to adapt it to other metaheuristics.

Our cellular approach, cellularizes the process of the metaheuristic method or metaheuristic methods, in order to: explore different solution spaces looking for local optima and save time through stagnation detection for each processing cell.

To validate our approach two cellular processing algorithms were built to solve the linear ordering problem with cumulative costs (LOPCC). A homogeneous and a heterogeneous cellular processing algorithms were tested to assess their performance. Three sets of standard instances were used to this purpose (UMTS, LOLIB and Random).

The experimental results show that the performance of these algorithms is statistically similar to the state-of-the-art solutions. It is important to point to that the homogeneous cellular processing algorithm (HoCPA) was able to find 37 new best-known solutions, and the heterogeneous cellular processing algorithm (HeCPA) finds 35 new best-known solutions. However, it is important to notice that the HeCPA obtains only two new best-known solutions less than HoCPA. But in general it uses about 627 seconds less, in all the tests, than HoCPA.

As the experimental results show, the cellular processing approach was able to increase the solution quality up to 3.6% and reduce the time consumption up to 20% versus the monolithic counterpart.

The main components of these algorithms are: the processing cells (PCells), the communication between PCells, and the global and local stagnation detection. Yet even if the main components are already determined, there are still several questions that need an answer. For example: which combination of metaheuristics would produce a good combination of PCells?; which is the right balance between search-based and population-based PCells?; is balanced or unbalanced the best choice for a homogeneous cellular processing algorithm?; which kind of communication is better, *on-line* or *off-line*?; is it a good idea to implement both communication types in the same algorithm? In this work we use path-relinking as communication method, but is it a better communication technique the combination or crossover operators? Which is the appropriate stagnation criterion for detecting local and global stagnations? Should the local and global stagnations be similar or unbalanced?

As we can see the field of cellular processing algorithms is a new and rich research area. And it's great flexibility and simplicity permits that almost any metaheuristic can be transformed into a cellular processing algorithm.

Table 7 Update to the best-known solutions for the LOLIB instances

be75eec	5.085	t65n11xx	**0.319**	t75i11xx	**4454.913**
be75np	**16543405.76**	t65w11xx	19.258	t75k11xx	1.323
be75oi	2.788	t69r11xx	14.036	t75n11xx	**9.896**
be75tot	**297138.268**	t70b11xx	93.671	t75u11xxa	0.326
stabu70	13.284	t70d11xx	79.742	tiw56n54	2.645
stabu74	14.029	t70d11xxb	4.435	tiw56n58	3.62
stabu75	9.412	t70f11xx	1.267	tiw56n62	3.024
t59b11xx	76261.813	t70i11xx	**115233.4**	tiw56n66	2.687
t59d11xx	4086.303	t70k11xx	0.492	tiw56n67	1.877
t59f11xx	**61.618**	t70l11xx	798.919	tiw56n72	1.567
t59i11xx	**7917.489**	t70n11xx	**0.054**	tiw56r54	2.626
t59n11xx	1618.897	t70u11xx	**42148.82**	tiw56r58	3.602
t65b11xx	**28230.381**	t70w11xx	**0.045**	tiw56r66	2.189
t65d11xx	3898.568	t70x11xx	0.231	tiw56r67	1.541
t65f11xx	**1.245**	t74d11xx	4.756	tiw56r72	1.349
t65i11xx	**473730.893**	t75d11xx	5.059		
t65l11xx	**2657.725**	t75e11xx	**2062.246**		

Table 8 Update to the best-known solutions for the Random instances

t1d35.1	0.923	t1d100.1	253.988	t1d150.1	8588.289
t1d35.2	0.167	t1d100.2	288.372	t1d150.2	**166482.377**
t1d35.3	0.154	t1d100.3	1307.432	t1d150.3	574943.633
t1d35.4	0.196	t1d100.4	**7293.311**	t1d150.4	**74063.165**
t1d35.5	1.394	t1d100.5	**165.963**	t1d150.5	79069.363
t1d35.6	0.2	t1d100.6	395.035	t1d150.6	46829.985
t1d35.7	0.12	t1d100.7	**5656.723**	t1d150.7	161149.153
t1d35.8	0.226	t1d100.8	2760.619	t1d150.8	251940.422
t1d35.9	0.436	t1d100.9	**62.69**	t1d150.9	364320.25
t1d35.10	0.205	t1d100.10	**156.018**	t1d150.10	**121446.49**
t1d35.11	0.369	t1d100.11	233.586	t1d150.11	**13054.614**
t1d35.12	0.234	t1d100.12	236.696	t1d150.12	65717.265
t1d35.13	0.196	t1d100.13	**577.453**	t1d150.13	**104975.277**
t1d35.14	0.138	t1d100.14	**246.03**	t1d150.14	74854.867
t1d35.15	1.376	t1d100.15	406.478	t1d150.15	**329110.316**

Table 8 (*continued*)

t1d35.16	0.286	t1d100.16	707.413	t1d150.16	**16651299**
t1d35.17	0.199	t1d100.17	725.79	t1d150.17	**71190.802**
t1d35.18	0.381	t1d100.18	622.942	t1d150.18	**711011.245**
t1d35.19	0.236	t1d100.19	228.486	t1d150.19	**59594.204**
t1d35.20	0.068	t1d100.20	**241.283**	t1d150.20	1886041.88
t1d35.21	0.202	t1d100.21	228.59	t1d150.21	41453.911
t1d35.22	0.177	t1d100.22	**153.388**	t1d150.22	695751.688
t1d35.23	0.345	t1d100.23	**1588.314**	t1d150.23	22203891.8
t1d35.24	0.132	t1d100.24	469.658	t1d150.24	**100543.43**
t1d35.25	0.143	t1d100.25	644.782	t1d150.25	462316.511

References

1. Alba, E., Dorronsoro, B., Alfonso, H.: Cellular memetic algorithms. Journal of Computer Science and Technology 5(4), 257–263 (2005)
2. Benvenuto, N., Carnevale, G., Tomasin, S.: Optimum power control and ordering in SIC receivers for uplink CDMA systems. In: IEEE-ICC 2005 (2005)
3. Bertacco, L., Brunetta, L., Fischetti, M.: The linear ordering problem with cumulative costs. Eur. J. Oper. Res. 189(3), 1345–1357 (2008)
4. Dorigo, M., Gambardella, L.M.: Ant colony system: A cooperative learning approach to the traveling salesman problem. IEEE Transactions on Evolutionary Computation 1(1), 53–66 (1997), doi:10.1109/4235.585892
5. Duarte, A., Laguna, M., Marti, R.: Tabu search for the linear ordering problem with cumulative costs. Computational Optimization and Applications 48, 697–715 (2011)
6. Duarte, A., Marti, R., Alvarez, A., Angel Bello, F.: Metaheuristics for the linear ordering problem with cumulative costs. European Journal of Operational Research 216(2), 270–277 (2012)
7. Feo, T., Resende, M.: A probabilistic heuristic for a computationally difficult set covering problem. Operations Research Letters 8(2), 67–71 (1989)
8. Feo, T., Resende, M.: Greedy randomized adaptive search procedures. Journal of Global Optimization 6(2), 109–133 (1995)
9. Feo, T., Resende, M., Smith, S.: A greedy randomized adaptive search procedure for maximum independent set. Operations Research 42(5), 860–878 (1994)
10. Folino, G., Pizzuti, C., Spezzano, G., Spezzano, O.: Combining cellular genetic algorithms and local search for solving satisfiability problems. In: Proceedings of Tenth IEEE International Conference on Tools with Artificial Intelligence, pp. 192–198 (1998)
11. Glover, F.: Heuristics for integer programming using surrogate constraints. Decis. Sci. 8, 156–166 (1977)
12. Huy, N.Q., Soon, O.Y., Hiot, L.M., Krasnogor, N.: Adaptive cellular memetic algorithms. Evolutionary Computation 17(2), 231–256 (2009) ISSN:1063-6560
13. Laguna, M., Marti, R., Campos, V.: Intensification and diversification with elite tabu search solutions for the linear ordering problem. Computers & Operations Research 26(12), 1217–1230 (1999), doi: 10.1016/s0305- 0548(98)00104-x

14. Li, B., Zhao, X.-F., Z.Q.s.T.S.h: Differentiate coevolutionary algorithms. Journal of Convergence Information Technology 6(4), 3247–3259 (2011)
15. Prais, M., Ribeiro, C.: Parameter variation in GRASP procedures. Investigacion Operativa 9, 1–20 (2000)
16. Reinelt, G.: The linear ordering problem: Algorithms and applications. Mathematical Social Sciences 14(2), 199–200 (1985)
17. Resende, M., Riberio, C.: Greedy Randomized Adaptive Search Procedures: Advances, Hybridizations, and Applications. In: Handbook of Metaheurictics, vol. 146, pp. 283–319. Springer (2010)
18. Righini, G.: A branch-and-bound algorithm for the linear ordering problem with cumulative costs. European Journal of Operational Research 186, 965–971 (2008)
19. Seredynski, F., Zomaya, A., Bouvry, P.: Function optimization with coevolutionary algorithms. In: International Intelligent Information Processing and Web Mining Conference, Zakopane, Poland (June 2003)
20. Sipper, M.: The emergence of cellular computing. IEEE Computer 32(7), 18–26 (1999)
21. Teran-Villanueva, J., Fraire-Huacuja, H., Duarte, A., Pazos-Rangel, R., Carpio Valadez, J., Puga-Soberanes, H.: Improving Iterated Local Search Solution for the Linear Ordering Problem with Cumulative Costs (LOPCC). In: Setchi, R., Jordanov, I., Howlett, R.J., Jain, L.C. (eds.) KES 2010, Part II. LNCS, vol. 6277, pp. 183–192. Springer, Heidelberg (2010)
22. Teran-Villanueva, J., Pazos-Rangel, R., Martinez, J.A., Lopez-Loces, M.C., Zamarron-Escobar, D., Pineda, A.: Hybrid GRASP with composite local search and path-relinking for the linear ordering problem with cumulative costs. International Journal of Combinatorial Optimization Problems and Informatics 3(1), 21–30 (2012)
23. Wiegand, R.P.: An analysis of cooperative coevolutionary algorithms. Ph.D. thesis, Fairfax, VA, USA (2004)

Part II

Soft Computing in Intelligent Control Applications

Hierarchical Genetic Optimization of the Fuzzy Integrator for Navigation of a Mobile Robot

Abraham Meléndez and Oscar Castillo

Tijuana Institute of Technology, Tijuana México
abraham.ms@gmail.com, ocastillo@tectijuana.mx

Abstract. This paper describes the optimization of an Integrator control block within the proposed navigation control system for a mobile robot. The control blocks that the integrator will combine are two Fuzzy Inference Systems (FIS) in charge of tracking and reaction respectively. The integrator block is call Weighted Fussy Inference System (WFIS), and assigns weights to the responses on each behavior block, to combine them into a single response.

1 Introduction

The use of mobile robots has increased over the last decades in many areas from industrial work to research and household and one reason for this is that they have proved useful in each of these areas from doing very specific task to ongoing monotonous shores, they help their human counterpart be more productive and efficient. Also as hardware technology is moving forward and developing more capable robots at lower cost, this is another reason for this increase and why we are seeing them in more common places.

The mobile robot, needs to move around its environment and this is why a great deal of research has been invested on testing them with control systems that allow the robots to navigate on their own, and different methodologies have been applied from traditional control such as PD, PID [4, 9] to soft computing methods like Fuzzy Logic [10, 25, 13, 17, 15, 18, 12, 11, 21, 5, 19, 20], Neural Networks [11] and hybrid ones also [6,23,24].

In this paper, the navigation control system has been designed to combine two key behaviors that are considered to be required for any navigation control system of a mobile robot. The first one is a tracking controller, this is an obvious one since there is no point of having a navigation system on a robot, that can´t go to a desired location, the second one is a reactive controller and here we considered this one to be off great importance also, since the tracking controller can get the robot to the destination, but that will be on an ideal situation where there are no obstacles present on the robots path.

P. Melin, O. Castillo (Eds.): *Soft Computing Appli. in Optimization,* STUDFUZZ 294, pp. 77–96.
DOI: 10.1007/978-3-642-35323-9_4

The reactive controller is for those cases where an obstacle free path cannot be guaranteed; this is where the reactive controller will do its work providing a behavior that will make the robot react to any type of obstacle so that the robot can continue on its journey. In this paper we describe the integration method for these two controls as part of the complete Navigation Control System, the control blocks are fuzzy inference systems of type-1 and type-2, and a general GA (Genetic Algorithm) is applied to the optimization of each of the controller blocks with a specific fitness function for each part that will evaluate the corresponding individual performance.

As related work, we can find that of Cupertino et al [8] developed a Fuzzy controller of a mobile robot, based on 3 FLCs (Fuzzy Logic Controller) and one Fuzzy Supervisor that was in charge of determining which FLC behavior will be active, there the FLCs are of Type-1 and the Fuzzy Supervisor mainly acts as a switch. In our proposed method the fuzzy integrator acts more like a fusion block. S. Coupland et al [7] proposed a Type-2 Fuzzy Control of a Mobile Robot, which is based on W Payton et al [22] Command Fusion, where the idea is that a behavior should work with others to find a mutually beneficent solutions, where each behavior takes into consideration every possible output with its corresponding activation value (positive or negative), and a winner takes all network is use to select the winning responses for each behavior. Coupland suggests using two FISs one for goal seeking and another for obstacle avoidance. The activation value for each of the FIS output will be a Fuzzy set that will be passed to the command fusion block to later be defuzzifed and that crisp value pass to the Actuator block, being a difference with our proposed control the integration method of the two behaviors. The control navigation of a mobile robot is a topic that has been extensively investigated over the years, the method proposed in this paper is based on the idea that separation and the cooperation between key behaviors produces a better result than the use of a single behavior and it differs from previous approaches from the integration perspective done by a FIS that is in charge of the weighted system, that will assign a weight to each response from each controller by each control step that is combined to obtain a unified single response to the robot.

This paper is organized as follows: In section 2 we describe the mobile robot used in these experiments, section 3 describes the development of the evolutionary method. Section 4 shows the simulation results. Finally, section 5 shows the Conclusions.

2 Mobile Robot

The particular mobile robot considered in this work. The robot is based on the description of the Simulation toolbox for mobile robots [26], which assumes a wheeled mobile robot consisting of one conventional, steered, unactuated 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 unactuated steered wheels. The Robot has two degrees of freedom (DOFs): y-translation and either x-translation or z-rotation [26]. Fig. 1 shows the robot's configuration, it has

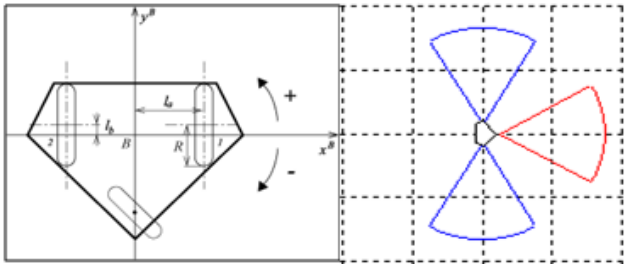


Fig. 1 Kinematic coordinate system assignments[26]

2 independent motors located on each side of the robot and one castor wheel for support located at the front of the robot.

The kinematic equations of the mobile robot are as follows:

Eq. 1: The sensed forward velocity solution [26]

$$\begin{pmatrix} V_{B_x} \\ V_{B_y} \\ \omega_{B_z} \end{pmatrix} = \frac{R}{2l_a} \begin{bmatrix} -l_b & l_b \\ -l_a & -l_a \\ -1 & -1 \end{bmatrix} \begin{pmatrix} \omega_{W_1} \\ \omega_{W_2} \end{pmatrix} \tag{1}$$

Eq. 2: The Actuated Inverse Velocity Solution [26]

$$\begin{pmatrix} \omega_{W_1} \\ \omega_{W_2} \end{pmatrix} = \frac{1}{R(l_b^2+1)} \begin{bmatrix} -l_a l_b & -l_b^2 - 1 & -l_a \\ l_a l_b & -l_b^2 - 1 & l_a \end{bmatrix} \begin{pmatrix} V_{B_x} \\ V_{B_y} \\ \omega_{B_z} \end{pmatrix} \tag{2}$$

Under the Metric system are define as:

V_{B_x}, V_{B_y}Translational velocities $[\frac{m}{s}]$,
ω_{B_z}Robot z-rotational velocity $[\frac{rad}{s}]$,
ω_{W_1}, ω_{W_1}Wheel rotational velocities $[\frac{rad}{s}]$,
R Actuated wheel radius[m],
l_a , l_bDistances of wheels from robot's axes [m].

3 Navigation Control System

The proposed control system consists of three main fuzzy blocks, two are behavior based and the other one is in charge of the response integration, the behaviors are the reactive and tracking blocks, and each one will provide its specific behavior that will be combined into one response by the integration block.

Each behavior block is in charge of its own task, the problem is that they seem to be in conflict with each other when an unexpected obstacle arises, because if at the time of planning the route the obstacles are present then the route can be designed to avoid them, but when there are obstacles that we where un aware off, the

two behaviors enter in contradiction one is designed to avoid the object and the other to keep the robot on its track.

The most common solution will be to just switch between controllers when need it, however, this approach is not very efficient due to the lack of awareness the two blocks have of each other, the reactive will effectively keep the robot from the collision but it may redirect the robot farther away from its destination to a point where the tracking controller can no longer find its way back to the reference, or the tracking controller can guide the robot straight into the obstacle if the reactive control is not activated on time. The proposed referral for control navigation is to always have both controls active and their responses are combined and generate the movement of the robot, the integration is done with another fuzzy block call WFIS[15] (Weight-Fuzzy Inference System) and what this controller does is to assign response weights to each of the controllers crisp response value.

The inputs are gathered from the information that we can collect from the robot (sensors) or the environment by other means (cameras) and from this we need to create the knowledge rule base to give higher activation values to the response we want to take the lead on the robot movement one example of the rule is the following (*if Front_Sensor_Distance is Close Then TranckingWeight is Medium and ReactiveWeight is Medium*), both off our controls provide the right and left motor speed and we combine each one with the weight given by the WFIS block. Figure 2 shows the proposed navigation control.

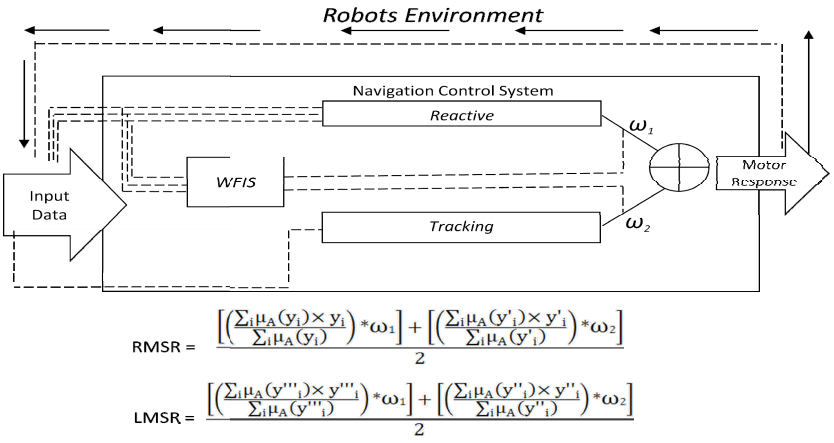


Fig. 2 Navigation Control System [15]

4 Genetic Algorithms

The Genetic Algorithm (GA) was applied to each of the design problems, of finding the best fuzzy reactive and tracking controllers [16]; however this paper will only focus on the WFIS controller.

The purpose of using an evolutionary method is to find the best possible controllers of each type and this can be obtained using the GA, as it searches along the solution space, combining the attributes from the best controllers in generating new ones, this concept taken from the building blocks theory.

The idea was to optimize the parameters in the Membership Functions, but also the number of Membership functions and this means to also optimize the number of rules making this a multi objective problem. For this we will take advantage of the HGA (Hierarchical Genetic Algorithm) intrinsic characteristic to solve multi objective problems.

The work of the GA was divided in two main modules, one that handles all the operations related to the selection and chromosome manipulation, which we use for all our controllers that we work on, the other module is the one where we evaluated the performance of each chromosome and this part is different on each case. With this approach we utilize the generality of the GA and just have a specific evaluation method for each controller. Figure 3 shows the 2 main modules.

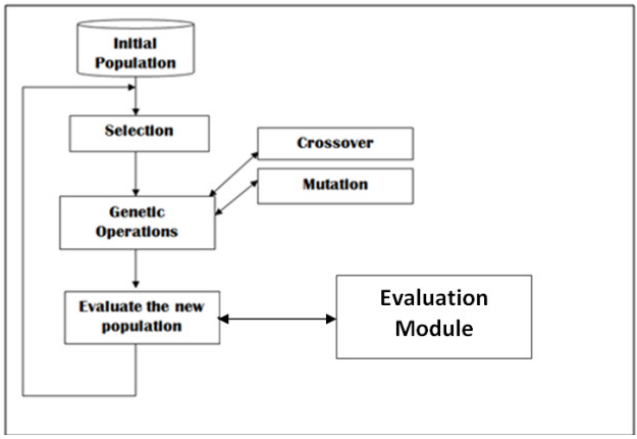


Fig. 3 Genetic Algorithm process

The GA module is in charge of initializing the population, selecting the chromosomes that will be used for the genetic operations and letting the Evaluation Module know which chromosomes are ready to be evaluated and reinserting them to the population pool.

4.1 Chromosome Encoding

Each individual on the population will represent a FIS controller, each of which will be encoded on a vectorial structure that will have “n” main sections, one for each variable (input and output). Each main section will contain 2 subsections (control genes, Connection genes). The section and subsection sizes depend on the controller that they represent.

4.2 Reactive Controller

The function of the reactive control is to give the same ability that we apply when we are driving, that is to react to unexpected situations, traffic jams, stop lights, etc, but in a more basic concept and ability, to the problem that is the navigation of the robot. A forward moving behavior response out off the control is desired. The objective is to guide the robot through the maze avoiding any collision. It's not our objective to optimize the robot to find the maze exit, we use a maze to optimize the reactive control because of the characteristic it offers to the simulation, i.e. it is a closed space where the robot cannot easily wonder off and each wall is considered an obstacle to the robot that it must avoid while it moves around. The FIS´s are interval Mamdani type-1 fuzzy system [26], each consisting of 3 inputs that are the distances obtained by the robots sensors described on section 2, and 2 outputs that control the velocity of the servo motors on the robot, all this information is encoded into each chromosome.

4.3 Tracking Controller

The tracking controller has the responsibility of keeping the robot on the correct path, this is when a reference is provided, it will move the robot to the reference and keep it on track and this is will allow the robot to move from point A to B, with in a obstacle free environment is possible.

The controller will work by keeping the error (Δep, $\Delta\theta$) to minimum values, which represents the error relative to the position and the error relative to the orientation of the front of the robot to a minimum value, the fuzzy system is a Mamdani type-1 FIS and consists of 2 inputs that are (Δep, $\Delta\theta$) and 2 outputs that control the velocity of the servo motors on the robot, see Figure 4.

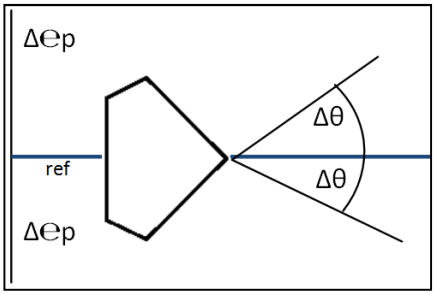


Fig. 4 Fuzzy controller inputs ep, $e\theta$

4.4 WFIS Controller

The function of the WFIS control is to correctly combine the 2 behaviors of tracking and reaction and obtain a new global behavior that resembles the same ability that we apply when we are driving, that is to react to unexpected objects, but in a

more basic concept and ability, to the problem that is the navigation of the robot. A forward moving behavior response out off the global control is desired. The objective is to guide the robot through the reference avoiding any collision with any obstacle present. It's not our objective to optimize the robot to find the maze exit, we use a closed space where the robot cannot easily wonder off and each wall is considered an obstacle to the robot that it must avoid while it moves around. The FISs are Mamdani type-1 fuzzy systems [14], each consisting of 3 inputs, which are the distances obtained by the robots sensors described on section 2, and 2 outputs that are the weights that will be used to integrate the responses of the other 2 controllers, all this information is encoded into each chromosome.

4.5 Type-1 Fuzzy Weight Controller Chromosome Architecture

The control genes consist of 5 bit vectors, this will indicate which fuzzy membership is or not active, the connection genes are divided in 5 subsections, 5 is the maximum number of membership functions that are allowed per variable, each of which can be trapezoidal or triangular membership function, and each of these subsection is divided into 2 sections one that indicates the type of the membership function and the other the parameters for the function, see Figure 5.

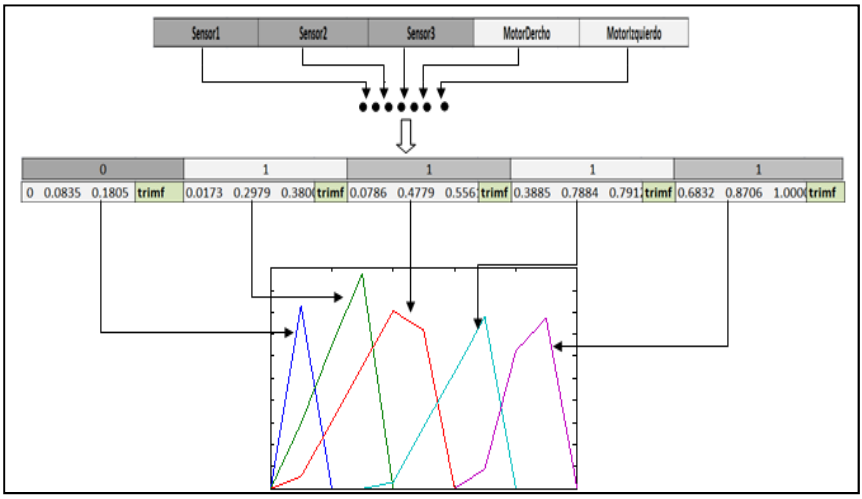


Fig. 5 Type-1 WFIS Controller Chromosome Architecture

4.6 Fuzzy Rules

The rules population is a different and separated population for each controller with respect to the control population, this is because the optimization procedure is totally different, but they are tightly related because the number of active rules depends on the number of active membership functions. In order to optimize the fuzzy rules we have a population off all the possible subsets keeping one

restriction that the number of active membership functions must be the same. With equation 3 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 3 gives a total of 625 fuzzy rules subsets.

$$rp=m*n*p*k \tag{3}$$

In this case, we will only have one active subset that can match the fuzzy controller that has the following membership functions active, $Sa,b,c,d,e,$ Where a, b, c are the number of active membership functions for the input variables and d,e for the output variable, and we use an index table for each of the fuzzy subsets, see Table 1.

Table 1 Rules Index Table

	Input 01	Input 02	Input 03	Output 01	Output 02
	1	1	1	1	1
$S_{1,3,2,2,2}=$	1	1	2	1	1
	1	2	1	1	1
	1	2	2	2	1
	1	3	1	1	2
	1	3	2	2	1

A Special mutation operator is applied (Equation 4) to find the optimal fuzzy rule set for the reactive controller, the shift operation that is used, changes the consequent part of the rule.

$$h(i,j,q,r,s)=h(i,j,q,(r+\Delta r),(s+\Delta s)) \tag{4}$$

Where $h(i,j,q,r,s)$ is the consequent of the rule that has i, j, q, r, s. active membership functions, Δr; Δs represent our shift operator, with a probability of 0.01.

4.7 Objective Function

The GA will be generating individuals 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 GA 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 our controller; if we fail to provide a proper evaluation method we can guide the population to suboptimal solutions or non solution at all.

4.7.1 Reactive Controller Objective Function

The criteria used to measure the Reactive controller performance takes into are the following

- 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 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 Figure 6.

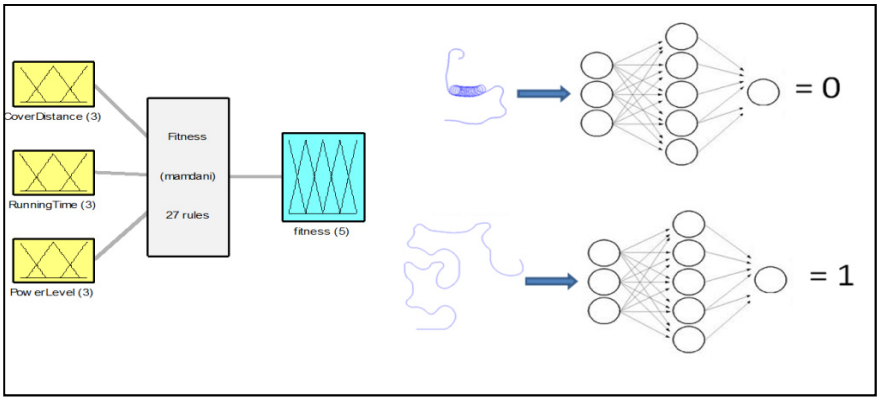


Fig. 6 Fitness Function for The Reactive Controller

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 Equation 5, based on the response off 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.

$$f(i) = \begin{cases} fv * nnv, nnv < 0.35 \\ fv, nnv \geq 0.35 \end{cases} \quad (5)$$

Where:

fi Fitness value of the i-th individual,
fv Crisp value out of the fitness FIS,
nnv Looping trajectory activation value.

4.7.2 Tracking Controller Objective Function

The Tracking controller performance is measure with the RMSE between the reference and the robots trajectory, we apply the test three times and take the average, on each of the three test the robot and the reference vertical position is random, but its ensure that on one test the robots vertical position is above the reference and on another test is below it, we do this to ensure the controller works properly for any case the robot may need it when it´s above or below (Figure 7).

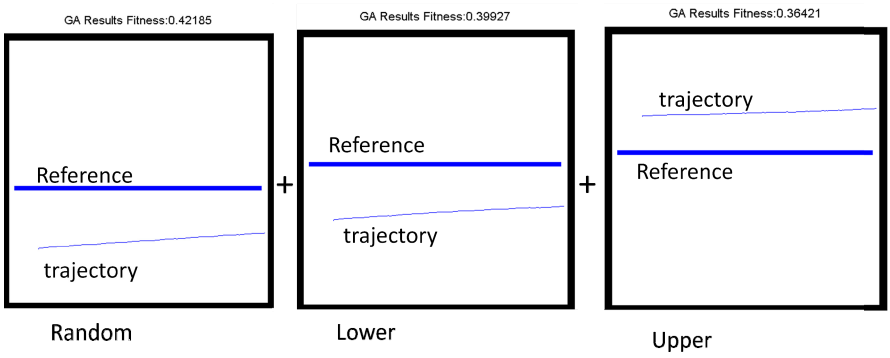


Fig. 7 Fitness Function for the Reactive Controller

4.7.3 WFIS Controller Objective Function

The WFIS controller performance is measured with the RMSE between the reference and the robots trajectory. We apply the test three times and take the average, on each of the three tests the robot and the reference vertical position is random, but we ensure that on one test the robots vertical position is above the reference and on another test is below it, we do this to ensure the controller works properly for any case the robot may need it when it´s above or below (Figure 8).

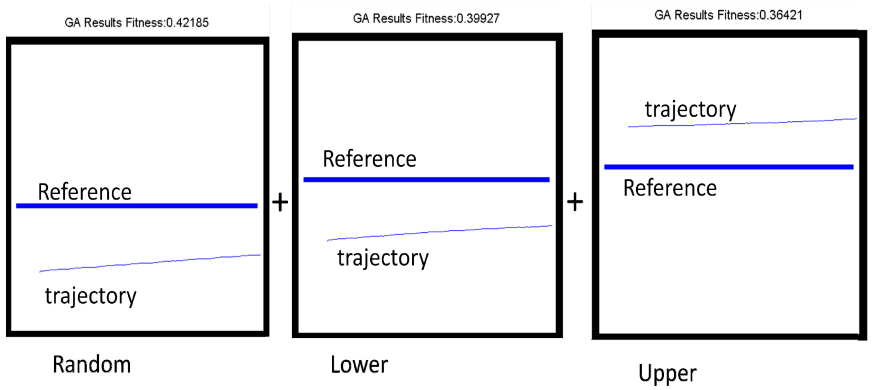


Fig. 8 Fitness Functions for the WFIS Controller

4.8 Optimization GUI

The GUI, is designed to optimize Type-1 and Type-2 FIS controllers, you can customize the number of inputs and outputs and the maximum and minimum number of Fuzzy memberships allow, and the basic parameters of the Genetic Algorithm such as number of generations, population replacement mutation and crossover rate and the type of selection, see Figure 9.

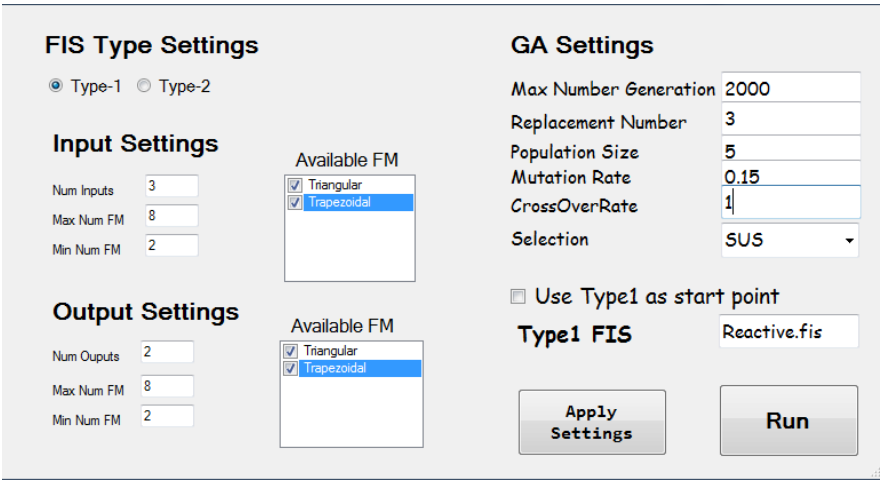


Fig. 9 Parameter selection with the GUI

As the GA is running a generation graph is plotted with the best individual found and a display area with all the population data, see Figure 10.

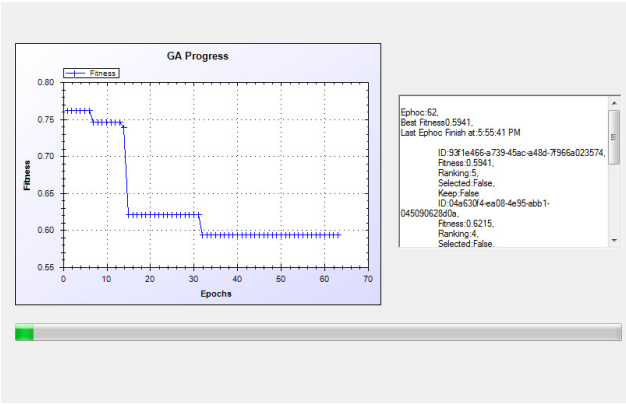


Fig. 10 Plot showing the execution of the GA

5 Simulation Results

For the simulation experiment the GA and the evaluation process were separated into two different parts, the generic GA process was developed on the C# language with .net 4, where a GA and Fuzzy System library where created with a GUI to setup the GA parameters, there the GA operations and cycle are run and the FIS are created. When a chromosome is ready to be evaluated it lets Matlab know and a modified version of the Simulation toolbox for mobile robots [26] is used to run each test, where the performance is measured and a Fitness value is returned to the GA process, and the communication between both process is done using a SQL server queue table.

5.1 Reactive Controller

For the type-1 reactive controller, a GA was setup with high number of generations and a low number of population size, this because of the large solution space the reasoning behind this is that with a relative small group of individuals it will cover focused sections of the solution and can move around the space, A constrain for inputs and outputs of maximum 10 and minimum 2 FM was set, on the outputs another constrain was set and it's that the outputs had to be the same, the evaluation as describe on section 4 is based upon each individual performance on the particular maze problem.

Table 2 shows the GA configuration and the top 9 Results, where we have the fitness value and the number of membership functions of each input and output, where the S represents the inputs and indicate the sensor number and M the outputs and indicate the Motor number, and the total rules that are active on each controller. Figure 11 shows the reactive controller results.

Table 2 Summary of Type-1 Reactive Controls Results

	Membership Chromosome		Fuzzy Rule Chromosome
	Control Genes	Connections Genes	
Representation	Binary	Real Number	Integer
Population Size	20		
No. of Offspring	5		
Crossover	One Point	One Point	
Crossover Rate	1.0	1.0	
Mutation	Bit Mutation	Random Mutation	Shift index operation
Mutation Rate	0.02	0.02	
	GA Parameters		
Generation	8000		
Selection	Roulette Wheel with Ranking		
	Results		
Rank	Fitness	Active FM's (S_1+S_2+S_3+M_1+M_2)	Active Rules
1	0.4895	(4+3+2+3+3)=15	24
2	0.4895	(4+3+2+3+3)=15	24
3	0.4895	(4+3+2+3+3)=15	24
4	0.4895	(4+3+2+3+3)=15	24
5	0.4895	(4+3+2+3+3)=15	24
6	0.4895	(4+3+2+3+3)=15	24
7	0.4895	(4+3+2+3+3)=15	24
8	0.4895	(4+3+2+3+3)=15	24
9	0.4895	(4+3+2+3+3)=15	24

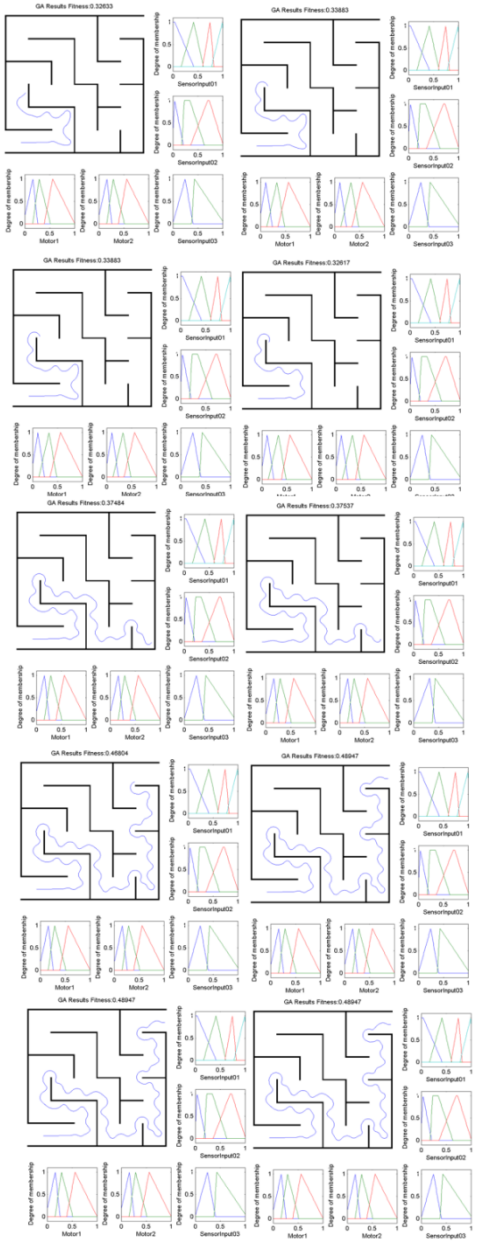


Fig. 11 Reactive Controller Results

5.2 Tracking Controller

Table 3 shows the GA configuration and the top 9 Results, where we have the fitness value and the number of membership functions of each input and output, where the e_p and $\Delta\theta$ represent the inputs and indicate the error on the position and orientation respectively, and M are the outputs and indicate the Motor number, and the total rules that are active on each controller. Figure 12 shows the tracking controller results.

Table 3 Summary of Type-1 Tracking Results

	Membership Chromosome		Fuzzy Rule Chromosome
	Control Genes	Connections Genes	
Representation	Binary	Real Number	Integer
Population Size	20		
No. of Offspring	2		
Crossover	One Point	One Point	
Crossover Rate	1.0	1.0	
Mutation	Bit Mutation	Random Mutation	Shift index operation
Mutation Rate	0.02	0.02	
GA Parameters			
Generation	4000		
Selection	Roulette Wheel with Ranking		
Results			
Rank	Fitness	Active FM's (e_p+ e_θ +M_1+M_2)	Active Rules
1	0.1907	5+4+4+3=16	20
2	0.2021	5+4+4+3=16	20
3	0.2023	5+4+4+3=16	20
4	0.2091	5+4+4+3=16	20
5	0.2124	7+2+4+5=18	14
6	0.2125	5+4+4+3=16	20
7	0.2182	6+2+5+2=15	12
8	0.2199	5+4+4+3=16	20
9	0.225	5+4+4+3=16	20

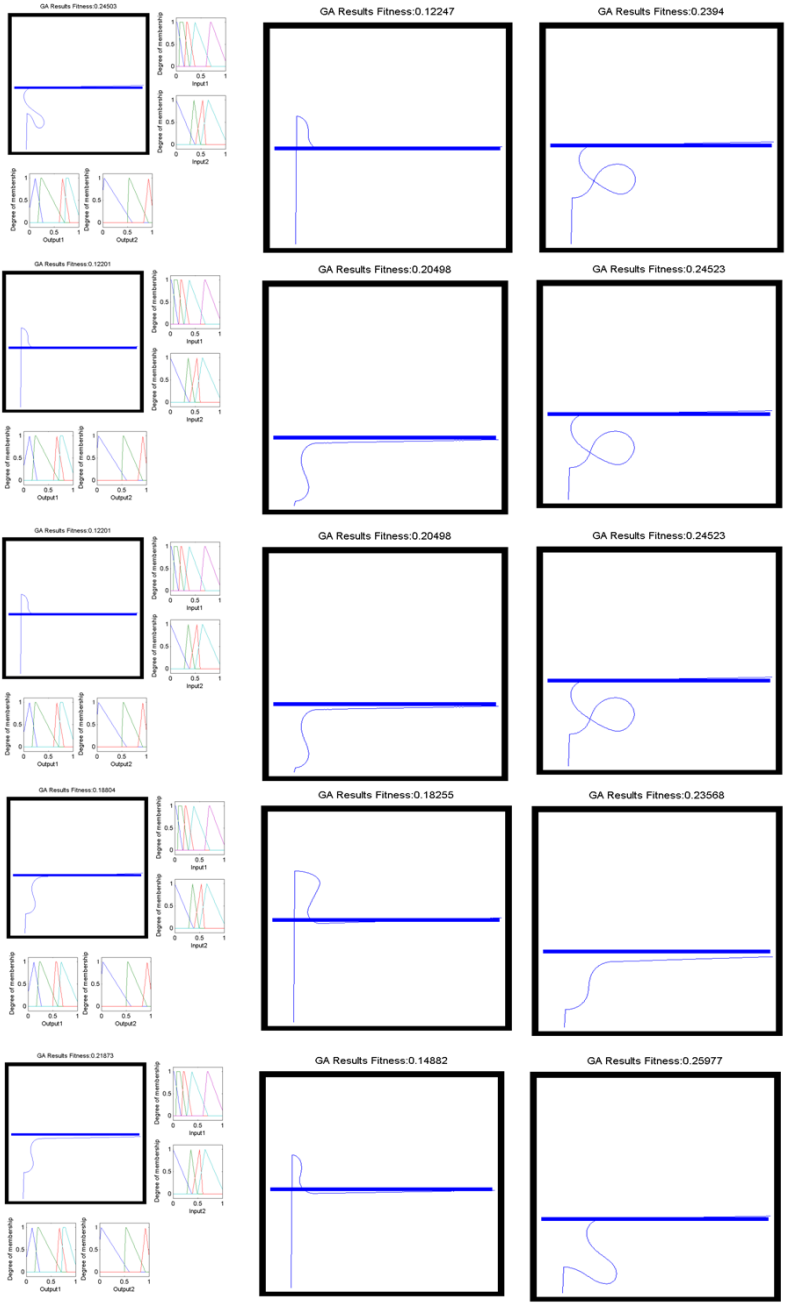


Fig. 12 Tracking Controller Results

5.3 WFIS Controller

For the type-1 WFIS controller, a GA was setup with high number of generations and a low value of population size, this because of the large solution space. The reasoning behind this is that with a relative small group of individuals it will cover focused sections of the solution and can move around the space, A constrain for inputs and outputs of maximum 10 and minimum 2 FMS was set, on the outputs, the evaluation as described on section 4 is based upon each individual performance on the particular maze problem.

Table 4 Summary of Type-1 WFIS Results

	Membership Chromosome		Fuzzy Rule Chromosome
	Control Genes	Connections Genes	
Representation	Binary	Real Number	Integer
Population Size	10		
No. of Offspring	3		
Crossover	One Point	One Point	
Crossover Rate	1.0	1.0	
Mutation	Bit Mutation	Random Mutation	Shift index operation
Mutation Rate	0.02	0.02	
GA Parameters			
Generation	1500		
Selection	Roulette Wheel with Ranking		
Results			
Rank	Fitness	Active FM's (S_1+S_2+S_3+W_1+W_2)	Active Rules
1	0.2746	3+5+2+2=12	15
2	0.4170	2+4+3+2=11	8
3	0.4553	3+5+2+2=12	15
4	0.4740	3+5+2+2=12	15
5	0.4856	2+4+2+2=10	8
6	0.5043	2+4+2+2=10	8
7	0.5197	2+4+2+2=10	8
8	0.5233	2+4+2+2=10	8
9	0.5277	2+4+2+2=10	8

Table 4 shows the GA configuration and the top 9 Results, where we have the fitness value and the number of membership functions of each input and output, where the S represents the inputs and indicate the sensor number and W the outputs and indicate the Weight number, and the total rules that are active on each controller.

Figure 13 shows the 3 tests during the evaluation process of the GA, where the red line is the reference, the blue dotted line is robot path on each run and the gray squares are obstacle located around the reference path.

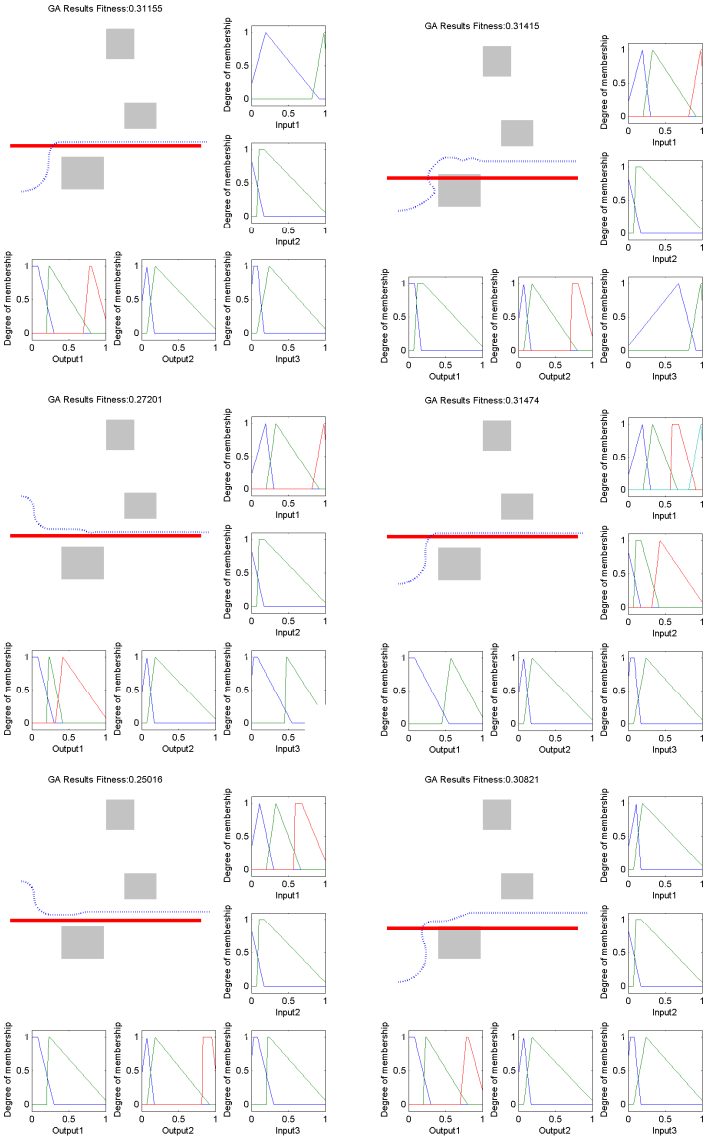


Fig. 13 Type-1 WFIS Controller Results

6 Conclusions

In this paper we have been able to optimize the Type-1 Reactive, Tracking and the WFIS controllers, and developed a GUI to optimize the Fuzzy Inference System, the results obtained on the proposed control system, show good performance on integrating the 2 behavior into a single response that was able to take the robot to the reference and avoid any collisions with the obstacles present on the map. Future work will consist in the Optimization of the WFIS based on a Type-2 fuzzy system.

Acknowledgment. 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] Aceves, A., Aguilar, J.: A Simplified Version of Mamdani's Fuzzy Controller the Natural Logic Controller. IEEE Transactions on Fuzzy Systems 14(1), 16–30 (2006)
[2] Aguilar, L., Melin, P., Castillo, O.: Intelligent Control of a stepping motor drive using a hybrid neuro-fuzzy ANFIS approach. Applied Soft Computing 3(3), 209–219 (2003)
[3] Astudillo, L., Castillo, O., Aguilar, L.: Intelligent Control of an Autonomous Mobile Robot Using Type-2 Fuzzy Logic. Engineering Letters 13(2), 93–97 (2006)
[4] Bell, M., Toriu, T., Nakajima, S.: Image-Based Robot Map Building and Path Planning with an Omnidirectional Camera Using Self-Organising Maps. International Journal of Innovative Computing, Information and Control, 3845–3852 (2011)
[5] Castillo, O., Melin, P.: New fuzzy-fractal-genetic method for automated mathematical Modeling and Simulation of Robotic Dynamic Systems. In: IEEE, International Conference on Fuzzy Systems, vol. 2, pp. 1182–1118 (1998)
[6] Castillo, O., Melin, P.: New fuzzy-fractal-genetic method for automated mathematical Modeling and Simulation of Robotic Dynamic Systems. In: IEEE, International Conference on Fuzzy Systems, vol. 2, pp. 1182–1118 (1998)
[7] Coupland, S.: Type-2 Fuzzy Control of a Mobile Robot, PhD Transfer Report, De Montfort University, UK (2003)
[8] Cupertino, F., Giordano, V., Naso, D., Delfine, L.: Fuzzy control of a mobile robot. IEEE Robotics & Automation Magazine, 74–81 (2006)
[9] Fate, M.: Robust Voltage Control of Electrical Manipulators in Task-Space. International Journal of Innovative Computing, Information and Control, 2691–2700 (2010)
[10] Ishikawa, S.: A Method of Indoor Mobile Robot Navigation by Fuzzy Control. In: Proc. Int. Conf. Intell. Robot. Syst., Osaka, Japan, pp. 1013–1018 (1991)
[11] Klir, J., Yuan, G.: Fuzzy Sets, Fuzzy Logic, and Fuzzy Systems: Selected Papers by Lotfi A. Zadeh. In: Advances in Fuzzy Systems: Application and Theory, vol. 6. World Scientific Publishing Company (1996)
[12] Kim, C., Lee, K.: Robust Control of Robot Manipulators Using Dynamic Compensators under Parametric Uncertainty. International Journal of Innovative Computing, Information and Control, 4129–4137 (2011)

[13] Leyden, M., Toal, D., Flanagan, C.: A Fuzzy Logic Based Navigation System for a Mobile Robot. In: Proceedings of Automatisierungs Symposium (1999)
[14] Mamdani, E.: Applications of fuzzy logic to approximate reasoning using linguistic synthesis. In: Proc. 6th Int. Symp. on Multiple Value Logic, Utah State University, pp. 196–202 (1976)
[15] Melendez, A., Castillo, O., Soria, J.: Reactive and Tracking Control of a Mobile Robot in a Distributed Environment Using Fuzzy Logic. In: FUZZ, IEEE International Conference, pp. 1–5 (2010)
[16] Melendez, A., Castillo, O., Melin, P.: Evolutionary Optimization of the Fuzzy Controllers in a Navigation System for a Mobile Robot. International Journal of Innovative Computing, Information and Control (in press)
[17] Meléndez, A., Castillo, O., Soria, J.: Reactive Control of a Mobile Robot in a Distributed Environment Using Fuzzy Logic. In: Annual Meeting of the North American Fuzzy Information Processing Society, NAFIPS 2008, May 19-22, pp. 1–5 (2008)
[18] Melin, P., Castillo, O.: Intelligent Systems with Interval Type-2 Fuzzy Logic. International Journal of Innovative Computing, Information and Control, 771–784 (2008)
[19] Melin, P., Castillo, O.: Intelligent control of aircraft dynamic systems with a new hybrid neuro-fuzzy fractal approach. Information Sciences 142(1-4), 161–175 (2002)
[20] Melin, P., Castillo, O.: Adaptive Intelligent control of aircraft systems with a hybrid approach combining neural networks, fuzzy logic and fractal theory. Applied Soft Computing 3(4), 353–362 (2003)
[21] Pishkenari, H.N., Mahboobi, S.H., Meghdari, A.: On the Optimum Design of Fuzzy Logic Controller for Trajectory Tracking Using Evolutionary Algorithms. In: 2004 IEEE Conference on Publication Date Cybernetics and Intelligent Systems, vol. 1(1-3), pp. 660–665 (2004)
[22] Payton, D.W., Rosenblatt, J.K., Keirsey, D.M.: Plan guided reaction. IEEE Transactions on Systems, Man and Cybernetics 20(6), 1370–1382 (1990)
[23] Shafiei, S., Soltanpour, M.: Neural Network Sliding-Mode-PID Controller Design for Electrically Driven Robot Manipulators. International Journal of Innovative Computing, Information and Control, 511–524 (2011)
[24] Shafiei, S., Soltanpour, M.: Robust Task-Space Control of Robot Manipulators under Imperfect Transformation of Control Space. International Journal of Innovative Computing, Information and Control, 3949–3960 (2009)
[25] Thomson, A., Baltes, J.: A path following system for autonomous robots with minimal computing power. University of Auckland, Private Bag 92019, Auckland, New Zealand, Technical Report (2001)
[26] Mobile robotics toolbox for Matlab 5 (2001), http://www.uamt.feec.vutbr.cz/robotics/simulations/amrt/simrobot_en.html

Particle Swarm Optimization for Multi-objective Control Design Using AT2-FLC in FPGA Device

Yazmin Maldonado, Oscar Castillo, and Patricia Melin

Tijuana Institute of Technology, Calzada Tecnológico S/N, Tijuana, México
ocastillo@hafsamx.org

Abstract. This research proposes the design, simulation and implementation of the optimization of type-2 membership functions for the Average Approximation of an Interval of Type-2 Fuzzy Logic Controller (AT2-FLC) using bio-inspired algorithms, such as Particle Swarm Optimization (PSO). The optimization only considers certain points of the membership functions, the fuzzy rules are not modified, so that the algorithm minimizes the runtime. Based on the concept of swarm intelligence, PSO is applied to membership functions parameter optimization of the AT2-FLC. Implementations and simulations are carried out on the FPGA device using the Xilinx System Generator. The optimization method was coded in Matlab. Comparisons were made between simulation and implementation of the AT2-FLC, to regulate the velocity of a DC motor. We compared the results of the AT2-FLC under uncertainty and the results are discussed. Experiments were performed by changing the number of bits for encoding the AT2-FLC in VHDL.

The main contribution of this research is the design, simulation and implementation of PSO of the AT2-FLC for real applications in FPGA. The AT2-FLC is targeted to a Xilinx Spartan 3AN XC3S700A device using Xilinx Foundation Environment.

Keywords: AT2-FIS, AT2-FLC, PSO, VHDL, FPGA, ReSDCM.

1 Introduction

Nowadays the use of fuzzy logic controllers is more common, because of the way of processing information, primarily in type-2 fuzzy logic controllers because they manage uncertainty and they are considered to be robust when compared with others [1].

P. Melin, O. Castillo (Eds.): *Soft Computing Appli. in Optimization,* STUDFUZZ 294, pp. 97–124.
DOI: 10.1007/978-3-642-35323-9_5

The use of optimization strategies applied to type-2 fuzzy logic controllers, such as genetic algorithms, particle swarm optimization, among others, make them more attractive [4].

With the optimization of the type-2 fuzzy systems arises the problem of processing time, which can be solved by processing in parallel, but in a physical implementation in particular this problem is not solved. For this reason we propose the optimization of type-2 fuzzy systems with bio-inspired or genetic algorithms for applications in FPGAs, the latter processed in parallel and speed is higher when compared to other electronic devices.

Fuzzy inference systems are based on rules, these rules incorporate linguistic variables, linguistic terms and fuzzy rules. The acquisition of rules is not an easy task for the expert and is of vital importance in the operation of the controller. The process of adjusting these linguistic terms and rules is usually done by trial and error, which implies a difficult task, and for this reason there have been methods proposed to optimize those elements that over time have taken importance, such as particle swarm optimization [5].

Most of the fuzzy logic applications with physical systems require a real time operation, the simple way to implement these systems is to realize them as software programs on a personal computer or higher density programmable logic devices, such as the field programmable gate array (FPGA).

The research of different optimization techniques for type-2 fuzzy systems have increased, however there is the problem of runtime, and the runtime decreases when the implementation is processed in parallel, as in the FPGA. There are some works related to the optimization of a particular problem [13][14].

This paper explains the design of T2-MFs optimization of the AT2-FLC for regulation speed of a DC motor (ReSDCM) in FPGA, based on an a-verage approximation of interval type-2 fuzzy systems method [9]. The main goal of this paper is to compare the results (average errors, runtime and resolution for number of bits) of the AT2-FLC optimized with PSO.

The proposed methodology is to synthesize the AT2-FLC in FPGA. The optimization of the T2-MFs takes place outside the FPGA, i.e. on a PC via serial port, the optimized parameters of the T2-MFs are sent to the FPGA, this with the idea that once the AT2-FLC was optimized, the optimization process is disconnected from the PC and the AT2-FLC is ready for use.

Figure 1 shows the methodology diagram used for the optimization of the T2-MFs for the AT2-FIS in the FPGA.

This paper is organized as follows. In section 2 we present an introduction to type-2 FISs, FPGA and the PSO method, in section 3 we present the description of the problem, in this case regulation of speed of the DC motor in VHDL for FPGA. The AT2-FIS in VHDL code is present in section 4. The design of the PSO method for the AT2-FLC for ReSDCM in shown in section 5, the T2-MFs optimization results for AT2-FLC in XSG versus T2- MFs optimization results for AT2-FLC in FPGA device are shown in section 6, and finally section 7 offers conclusions about this work.

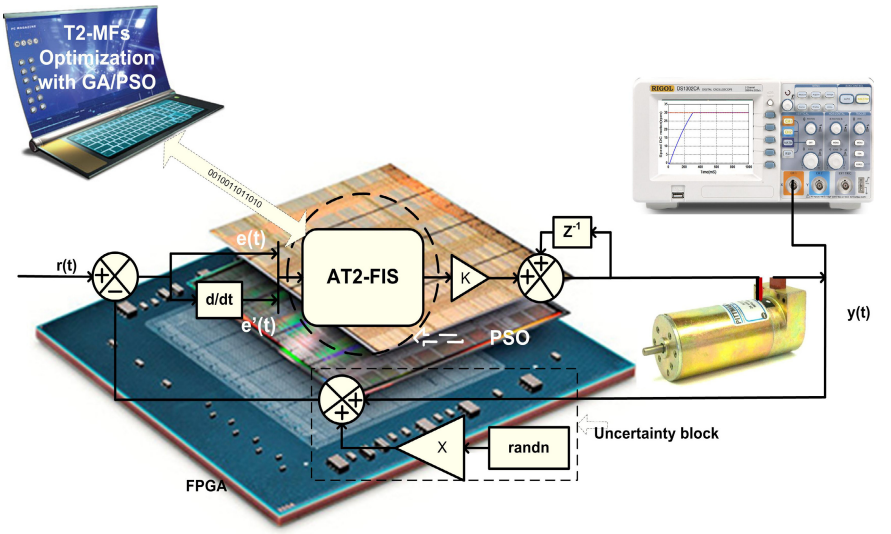


Fig. 1 Methodology used for PSO of the AT2-FLC for ReSDCM

2 Type-2 Fuzzy Inference Systems and Optimization Method

Fuzzy systems are being used more frequently, because they tolerate imprecise information and can be used to model nonlinear functions of arbitrary complexity [1-3]. T1-FIS have exact membership functions, while interval type-2 fuzzy systems (IT2-FIS) are described by membership functions with uncertainty [8][11]. The uncertainty in a fuzzy system can occur when:

- There is uncertainty in the words that are used in the rules.
- Uncertainty about the consequent to be used with a rule.
- Uncertainty about the measurements that activate a fuzzy system.
- Uncertainty about the data that are used to tune the parameters of a fuzzy system.

For example, the knowledge that is often used to create fuzzy rules is uncertain, this uncertainty leads to rules whose antecedents or consequents are uncertain, which translates into uncertainty in the membership functions.

The interval type-2 fuzzy inference systems (IT2-FIS) consist of four stages: Fuzzification, Inference, Type Reduction and Defuzzification.

The fuzzification stage maps a numeric value $\times=(\times_1..\times_p)^T \in X_1 \times X_2 \times..X_p \equiv X$, into a type-2 fuzzy set $\tilde{A}_x$ in X, where $\tilde{A}_x$ is a singleton fuzzy set, if $\mu_{\tilde{A}_x}(\times)=1/1$ for $\times=\times'$ and $\mu_{\tilde{A}_x}(\times)=1/0$ for all others $\times\neq\times'$ [12].

The inference stage consists of two blocks, the rules and the inference engine; it works the same way as for type-1 fuzzy systems, except the antecedent fuzzy sets and the consequent are represented by type-2 fuzzy sets. The process consists of

combining the rules and maps the input to the output (interval type-2 fuzzy sets), using the Join and Meet operations [11]. For an IT2-FIS with p inputs $x_1 \in X_1, x_2 \in X_2, \ldots, x_p \in X_p$ and one output $y \in Y$, it is assumed that there are M rules, the lth rule in an IT2-FIS and can be written as:

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

where l=1,…,M. Once we have the rules, it is necessary to calculate the operations Join(ц) and Meet(п) as well as sup-star composition (★)[12].

The type reductor stage is used to convert all type- 2 fuzzy sets to type-1 fuzzy intervals on the output. There are several methods to calculate the reduced set, such as the joint center, center of sums, height, among others.

The Defuzzification stage consists in obtaining a numeric value for the output. Using the COS type reductor, the defuzzification is an average value since the range is given by $[y_l, y_r]$ [12].

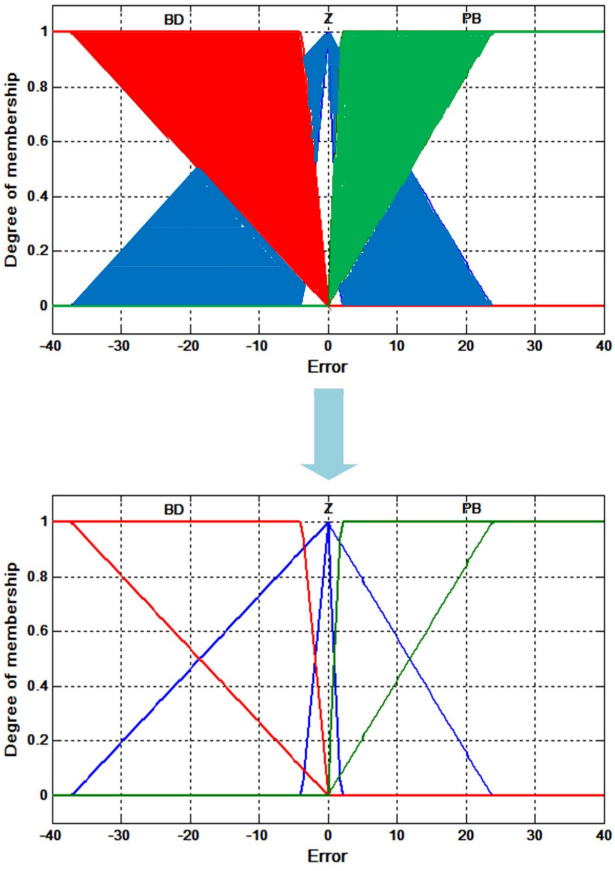


Fig. 2 T2-MFs using AT2-FIS method

In the method of average approximation of an interval type-2 fuzzy system (AT2-FIS), the AT2-FIS is replaced by a type-2 fuzzy system using the average of two T1-FIS, this method [14] is performed as follows:

1. Replace each T2-MF with two T1-MFs using different degrees of membership in order to obtain the footprint of uncertainty. Figure 2 shows this process.
2. To replace the type-2 inference stage, the inference from each T1-FIS must be obtained.
3. To replace the type-reduction system and defuzzification stage of the IT2-FIS, we obtain the defuzzification of each T1-FIS and the results of the two systems are averaged.

An IT2-FIS and AT2-FIS can be implemented on a general purpose computer, or by a specific use of a microelectronics realization such as the FPGA. In this work we use the AT2-FIS for FPGA synthesis. Figure 3 shows the block diagram of the AT2-FIS.

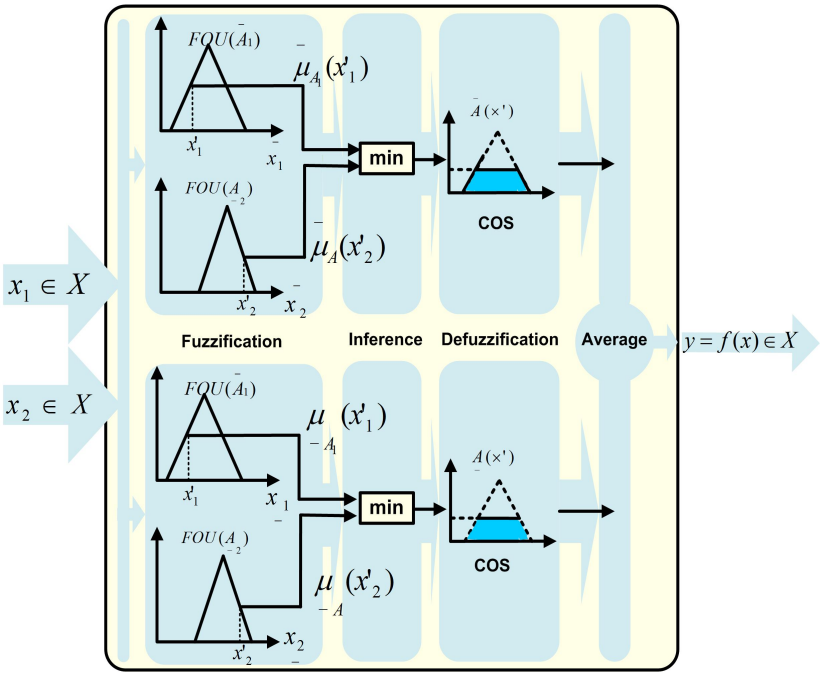


Fig. 3 Average approximation of an interval type-2 fuzzy system

A FPGA is a semiconductor device that contains in its interior components such as gates, multiplexers, etc. These are interconnected with each other, according to a given particular design. These devices use the VHDL programming language, which is an acronym that represents the combination of VHSIC (Very High Speed Integrated Circuit) and HDL (Hardware Description Language).

The design of a FPGA implementation is done by specifying the logic function to develop, either by a CAD (computer aided design) or through a hardware description language. Having defined the function to perform, the design is transferred to the FPGA.

This process consists in programming the configurable logic blocks (CLBs) to perform a specific function (there are thousands of configurable logic blocks in the FPGA). The configuration of these blocks and their interconnections are the reasons why it can achieve very complex designs. The interconnections enable connecting the CLBs. Finally, it has configuration memory cells (CMC, Configuration Memory Cell) distributed throughout the chip, which store all information necessary for programming of the programmable elements mentioned. These cells usually consist of a configuration RAM and are initialized in the process of loading of the configuration. The programmable elements of an FPGA are: Configurable Logic Blocks (CLBs), In/Out Blocks (IOBs) and Programmable Interconnection (by fuse technology and be of OTP and by antifuses or by type SRAM cells).

Figure 4 shows the basic elements of a FPGA.

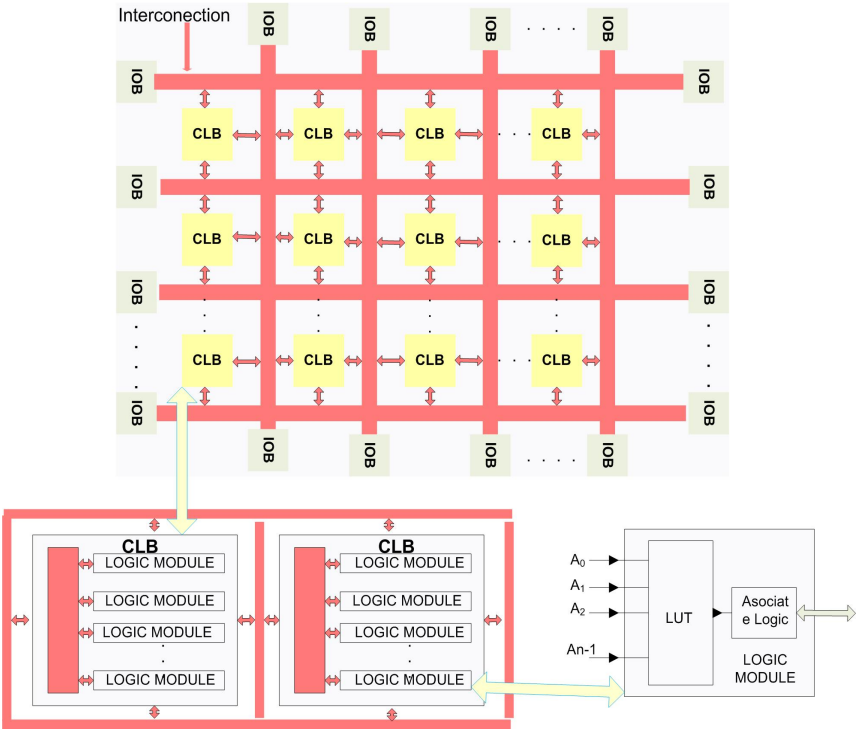


Fig. 4 FPGA basic elements

The FPGAs can be used to implement specific architectures to accelerate a particular algorithm. Applications that require a great number of simple operations are suitable for implementation on FPGAs.

FPGAs have been increasingly applied to high performance embedded systems because FPGAs are configured after fabrication and they also can be reconfigured. This is done with HDL, which is compiled to a bit stream and download to the FPGA device.

A processing element can be designed to perform this operation and several instances of it can be used to perform parallel processing [6][10].

The easiest way to get a design of a type-2 fuzzy system is to use software, the problem arises when you have a particular application and the response is not the best, this is when there is the need to optimize the original design. There are many optimization methods [5][9], such as particle swarm optimization.

Particle Swarm Optimization (PSO) is a bio-inspired optimization method. PSO finds the optimal solution by simulating social behavior. PSO is developed through simulation of birds that come in two-dimensional space, each particle has position and speed.

A PSO algorithm maintains a swarm of particles, where each particle represents a possible solution. In analogy with the paradigms of evolutionary computation, the particles are transported through a multidimensional search space, where the position of each particle is adjusted according to their experience and of their neighbors, $x_i(t)$ represents the position of particle i in the search space at time t, t denotes the discrete time. The position of the particle is modified by the addition of a velocity $v_i(t)$, i.e. the current position [5], Equation 2 shows the position of the particle.

$$x_i(t+1) = x_i(t) + v_i(t+1) \tag{2}$$

where $x_i(0) \sim U(x_{min}, x_{max})$. The velocity vector reflects both the experimental knowledge of the particle and the exchanged social information. The experimental knowledge of a particle is often referred to as the cognitive component, which is proportional to the distance of the particle from its best position (referred to as the best personal position of the particle) found from the beginning.

PSO can be described as follows, each swarm knows the best position of the particle (*Plbest*) and the best global position of the swarm (*Pgbest*). The speed of each particle can be calculated using the Equation 3 [5].

$$v_{ij}(t+1) = v_{ij}(t) + c_1 r_{1j}(t)\left[y_{ij}(t) - x_{ij}(t)\right] + c_2 r_{2j}(t)\left[\hat{y}_j(t) - x_{ij}(t)\right] \tag{3}$$

where $v_{ij}(t)$ is the velocity of the particle i from $j = 1, \ldots, n_x$ at time t, $x_{ij}(t)$ is the position of particle i in dimension j at time t, c_1 and c_2 are the positives constants acceleration used for cognitive and social components respectively, $r_{1j}(t)$, $r_{2j}(t) \sim U(0,1)$, which are random values in the range [0,1]. These random values in the algorithm introduce stochastic elements. y_{ij} is the *Plbest*, is associated with the particle i, is the best position of the particle, is the best global position of the particle swarm *Pgbest.*

3 Description of the Problem

This paper proposes the T2-MF optimization with the PSO method for the AT2-FIS codified to VHDL for FPGA. To validate the optimized AT2-FIS we applied

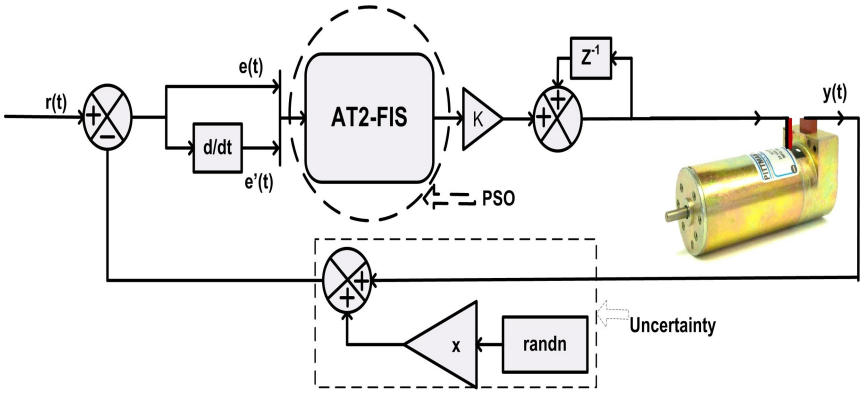


Fig. 5 AT2-FLC for ReSDCM

the proposed approach to a real problem, which is the regulation of speed of a DC motor (ReSDCM). Figure 5 shows the AT2-FLC for ReSDCM.

In Figure 3 the AT2-FLC [7] has the following inputs, error ($e_{(t)}$) and change of error ($e'_{(t)}$), and the output is the control signal ($y_{(t)}$), the control objective AT2-FLC is:

$$\lim_{t\to\infty} \left| y_{(t)} - r_{(t)} \right| = 0 \tag{4}$$

where t is the sampling time.

The inputs are calculated as follows:

$$e_{(t)} = r_{(t)} - y_{(t)} \tag{5}$$

$$e'_{(t)} = e_{(t)} - e_{(t-1)} \tag{6}$$

The reference signal $r(t)$, is given by [0,70] revolutions per minute (rpm).

The uncertainty block represents an external perturbation, the goal is to disrupt the AT2-FLC and then the AT2-FLC is expected to retrieve its desired path. The uncertainty block is represented by:

$$\hat{y}_{(t)} = y_{(t)} + x * randn \tag{7}$$

where x is the uncertainty level factor [0,1].

4 AT2-FIS Design in VHDL Code

For the design for AT2-FIS in VHDL the average approximation for interval type-2 fuzzy systems was used [14]. The AT2-FIS has four stages, which are fuzzification, inference, defuzzification and average. Figure 6 shows the AT2-FIS implementation.

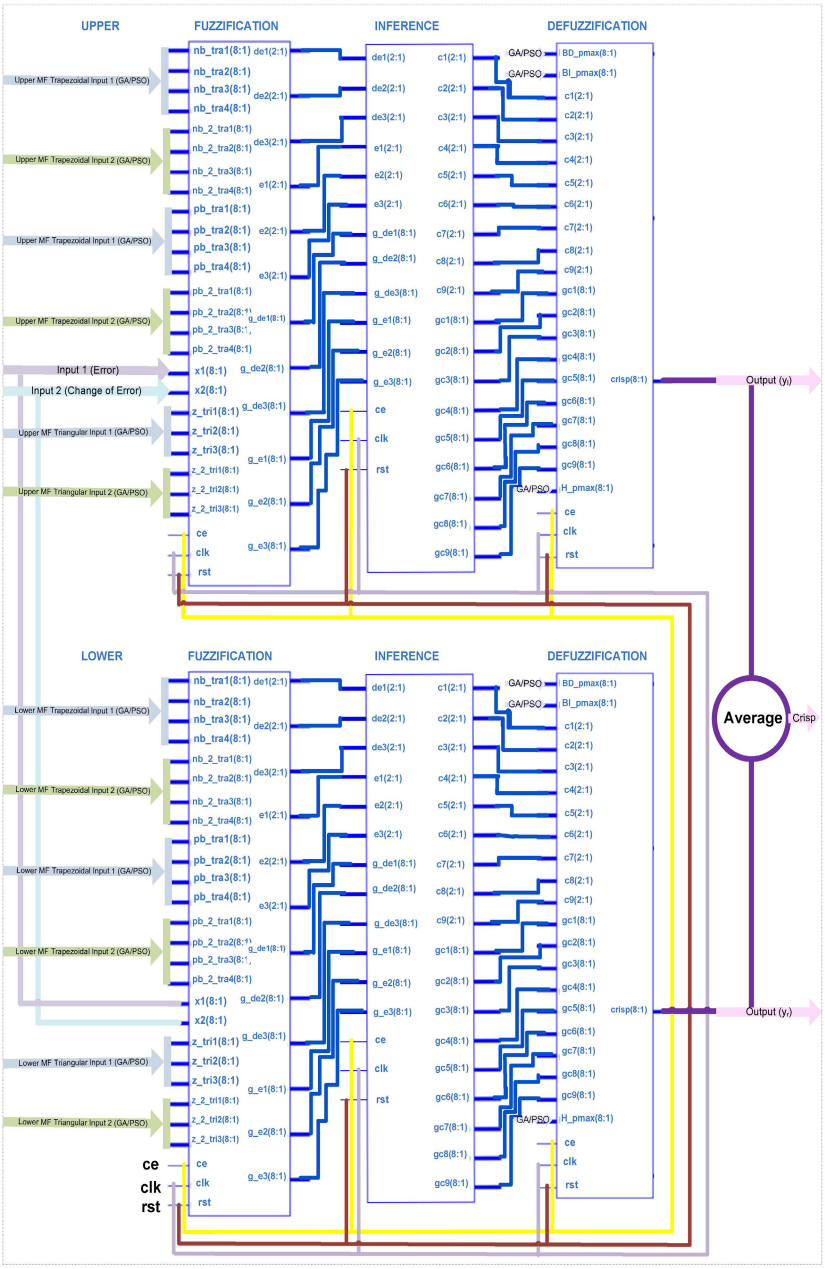


Fig. 6 AT2-FIS diagram for FPGA

The AT2-FIS has three common inputs for each stage (`ce, clk and rst`). The clock enable (`ce`) input, to simulate VHDL code in XSG, all blocks should have this input. `Clk` input is a clock for FPGA, we used Spartan 3AN with clock chip is 50 MHz. Reset (`rst`) input is a reset for all stages of the AT2-FIS. Below the explanation of the stages of AT2-FIS is given.

4.1 Fuzzification

We present an algorithm that works with the calculation of the slopes of the triangular and trapezoidal MFs. The main advantages of this algorithm is that it works for symmetrical and non symmetrical T2-MFs, the value of the slope is calculated on line, therefore it is possible to optimize the T2-MF using this method because most of the time in the optimization of membership functions not symmetrical T2-MFs are obtained. A disadvantage of our algorithm is that it only considers membership functions of triangular and trapezoidal form.

The procedure of the algorithm is summarized in three steps: calculate the slope values, calculate the degree values of the membership functions and send to the inference stage the membership degrees and the linguistic terms.

The fuzzification stage algorithm calculates the value of the membership degree with the equation of the line $y = mx + b$, where m is the slope of the line and b is the y intercept of the graph of the line.

The fuzzification stage is discretized in bits selected by the user, i.e. the number of bits is adjustable to the characteristics required by the problem to solve.

For the AT1-FIS, the fuzzification stage has inputs such as error ($e = x1$) and change of error ($de = x2$), each with three membership functions (Negative Big (**NB**="01"), Zero (**Z**="10"), Positive Big (**PB**="11")), two trapezoidal MFs and one triangular MF.

The universe of discourse as the degree of membership are designed for 8 bits, however, simply change one variable in the VHDL code to increase or decrease the number of bits.

The Fuzzification stage has outputs such as degree and linguistic terms for the error input (g e1, g e2, g e3, e1, e2, e3) and the change of error input (g de1, g de2, g de3, de1, de2, de3).

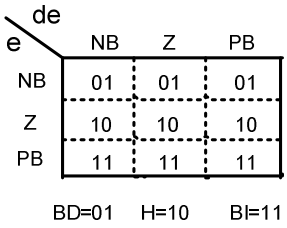
The Fuzzification stage has two outputs, μ_x is the membership degree and L_x is the linguistic tag, these outputs are sent to directly to the inference stage.

4.2 Inference

The inference stage receives the data sent from the fuzzification stage, which are labels and the membership degrees of each input are: g_e1, g_e2, g_e3, e1, e2, e3, g_de1, g_de2, g_de3, de1, de2, de3, so that multiplexes labels and evaluates the rule base and this is illustrated in Table 1.

An example of rules using this codification is: If e is "PB" and de is "Z" then C (consequent) is BI. For each of these rules the max-min operation is calculated of labels (c1, c2, c3, c4, c5, c6, c7, c8, c9) and the firing forces (gc1, gc2, gc3, gc4, gc5, gc6, gc7, gc8, gc9) are sent to the defuzzification stage.

Table 1 Rule Matriz



We have nine labels (c1, c2, c3, c4, c5, c6, c7, c8, c9) and nine firing forces (gc1, gc2, gc3, gc4, gc5, gc6, gc7, gc8, gc9) because we have three T1-MFs for each input and output. Figure 7 shows the inference stage process.

4.3 Defuzzification

The Defuzzification stage is calculated using the Height´s method as shown in Equation 8 [16].

$$y_{(x)} = \frac{\sum_{m=1}^{n} C_m o_m}{\sum_{m=1}^{n} o_m} \tag{8}$$

where C is the consequent (firing forces) and o is the consequent tags (labels). Once the consequent is calculated using Equation 8 the defuzzification stage sends the crisp value to the output.

4.4 Average

The average stage receives the interval defined by y_l and y_r, later to calculate the crisp value with an average. Equation 9 computes the crisp value.

$$y_{(x)} = \frac{y_l + y_r}{2} \tag{9}$$

Figure 7 shows the AT2-FIS architecture in VHDL for FPGA.

In our algorithm for the fuzzification stage, y is the degree of membership (μ_x), *Count* is a counter, which identifies the linguistic tag (L_x, which is stored in a register) and the slope (m), if the account number is pair then the slope is negative, if the account number is odd then the slope is positive.

The inference stages receive the membership degrees (*ge* and *gde*) and linguistic labels (*e* and *de*) and using the min operation calculates the for-ces firing (*gc*) and consequents (*c*).

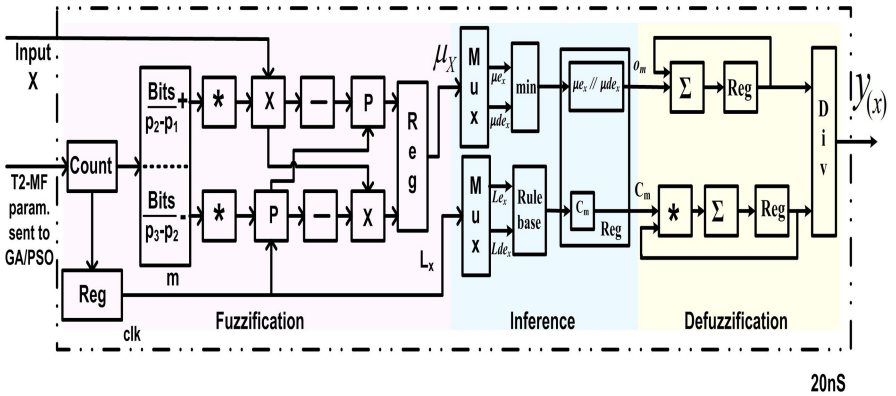


Fig. 7 AT2-FIS architecture in VHDL for FPGA

The four stages are targeted on a FPGA Xilinx Spartan 3AN XC3S700A device. Table 2 shows the device utilization summary for these stage, and we can see that after having synthesized in VHDL the AT2-FIS there is space that is available on the FPGA.

Table 2 Device utilization summary for the Fuzzification (F), Inference (I) and Defuzzification (D) stages

Logic Utilization	Used			Available			Utilization (%)		
	F	I	D	F	I	D	F	I	D
No. of 4 Input LUTs	5234	181	104	15360	15360	15360	34	1	0
No. of Bonded IOBs	155	153	2057	173	173	15360	99	88	13

5 Particle Swarm Optimization for T2-MFs of the Average Approximation of an Interval Type-2 Fuzzy Inference System

We optimized the type-2 membership functions (T2-MFs) of the AT2-FIS with PSO. Figure 8 shows the triangular and trapezoidal T2-MFs that are used.

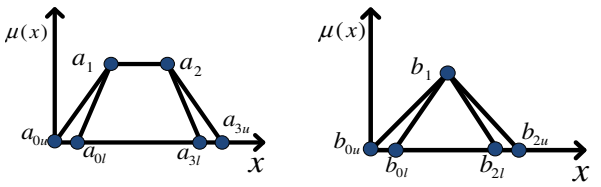


Fig. 8 Triangular and trapezoidal T2-MFs parameters

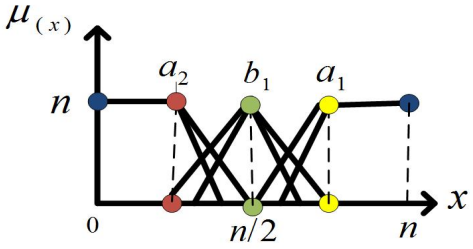


Fig. 9 Inputs and output of the AT2-FIS design

The design of the AT2-FIS only considers triangular and trapezoidal T2-MFs for each input and output, and Figure 9 shows the inputs and output design.

Figure 9 shows the design of the inputs and output T2-MFs of the AT2-FLC with fixed and variable parameters, where the universe of discourse and the degree of membership are divided into 8 bits (for example, as already mentioned this number may be changed, if n is 8 bits). The blue points are fixed, the red dots represent the parameter a_2, the green dots are fixed (b_1) and the yellow dots represent the parameter a_1.

The T2-MFs parameters for each and output are moved by the optimization method according to Table 3.

In Table 3, *U* corresponds to the ranges of the upper T2-MF and *L* correspond to the ranges of the lower T2-MF. The conditions of the T2-MF lower boundary are very important because if we are not cautions about the T2-MF low boundary it can be converted to the T2-MF high boundary and vice versa.

For example, if n=8 bits, the a_{2U} parameter should be within the range of 0 – 128, if we consider that is 100, then the a_{2L} parameter must be greater than 100 and less than 128, these conditions are necessary to achieve the upper and lower values of the T2-MF.

Table 3 Boundary T2-MFs parameters

	Input 1	**Input 2**	**Output**
Upper T2-MFs Parameters	**First T2-FM** $a_{0U} = a_{1U} = 0$ $0 < a_{2U} < n/2$ $a_{3U} = n/2$	**First T2-FM** $a_{0U} = a_{1U} = 0$ $0 < a_{2U} < n/2$ $a_{3U} = n/2$	**First T2-FM** $a_{0U} = a_{1U} = 0$ $0 < a_{2U} < n/2$ $a_{3U} = n/2$
	Second T2-MF $0 < b_{0U} < n/2$ $b_{1U} = n/2$ $n/2 < b_{2U} < n$	**Second T2-MF** $0 < b_{0U} < n/2$ $b_{1U} = n/2$ $n/2 < b_{2U} < n$	**Second T2-MF** $0 < b_{0U} < n/2$ $b_{1U} = n/2$ $n/2 < b_{2U} < n$
	Third T2-MF $a_{0U} = n/2$ $n/2 < a_{1U} < n$ $a_{2U} = a_{3U} = n$	**Third T2-MF** $a_{0U} = n/2$ $n/2 < a_{1U} < n$ $a_{2U} = a_{3U} = n$	**Third T2-MF** $a_{0U} = n/2$ $n/2 < a_{1U} < n$ $a_{2U} = a_{3U} = n$

Table 3 (*continued*)

Lower T2-MFs Parameters	**First T2-FM** $a_{0L} = a_{1L} = 0$ $a_{2U} < a_{2L} < n/2$ $a_{3L} = n/2$	**First T2-FM** $a_{0L} = a_{1L} = 0$ $a_{2U} < a_{2L} < n/2$ $a_{3L} = n/2$	**First T2-FM** $a_{0L} = a_{1L} = 0$ $a_{2U} < a_{2L} < n/2$ $a_{3L} = n/2$
	Second T2-FM $b_{0U} < b_{0L} < n/2$ $b_{1L} = n/2$ $b_{2U} > b_{2L} < n$	**Second T2-FM** $b_{0U} < b_{0L} < n/2$ $b_{1L} = n/2$ $b_{2U} > b_{2L} < n$	**Second T2-FM** $b_{0U} < b_{0L} < n/2$ $b_{1L} = n/2$ $b_{2U} > b_{2L} < n$
	Third T2-FM $a_{0L} = n/2$ $a_{1U} < a_{1L} < n$ $a_{2L} = a_{3L} = n$	**Third T2-FM** $a_{0L} = n/2$ $a_{1U} < a_{1L} < n$ $a_{2L} = a_{3L} = n$	**Third T2-FM** $a_{0L} = n/2$ $a_{1U} < a_{1L} < n$ $a_{2L} = a_{3L} = n$

The PSO design only moves 24 parameters of the 66 parameters to optimize, of which 8 parameters are for each input and output (a_{2U}, a_{2L}, b_{0U}, b_{0L}, b_{2U}, b_{2L}, a_{1U} and a_{1L}) the remaining 42 parameters are fixed and therefore these are not considered for the particle design.

In reaching this conclusion we conducted several experiments to test that with only the optimization of 24 points it was sufficient for a better response in the AT2-FLC in a lesser runtime.

After designing the AT2-FLC and taking into account their characteristics we arrived to the conclusion that the PSO algorithm is of a multiobjective type [9], because they are based on evaluating three characteristics, minimum steady state error, minimum overshoot and minimum undershoot, and these three characteristics help us to determine the best AT2-FLC solution.

The minimum overshoot is given by Equation 10.

$$if\ y_{(t)} > r_{(t)} \quad \rightarrow o_1 = \min(y_{(t)}) - r_{(t)} \tag{10}$$

The minimum undershoot is given by Equation 11.

$$o_2 = \left|\min(y_{(t)}) - r_{(t)}\right| \tag{11}$$

The minimum output of steady state error (sse) is given by Equation 12.

$$sse = \sum_{t=201}^{1000} y_{(t)} - r_{(t)} \tag{12}$$

where $y_{(t)}$ is the output of the system and $r_{(t)}$ is reference. The three objective functions are evaluated for fitness evaluation.

For the optimization of the T2-MFs using PSO, we need to define the number of particles in the swarm, the calculation of the position and the initial and final velocity, given by Equations 2 and 3. The PSO process starts by generating the initial swarm with 10 particles, and these particles are evaluated once for initial

selection of the best global particle (*Pgbest*) and best local particle (*Plbest*). If a better particle is found, the T2-MF parameters are sent to the AT2-FLC into the FPGA, if a better particle is not found then each one is again updated by equations for the position and velocity of the particle. Then the T2-MF parameters are downloaded to AT2-FLC in the FPGA, if it meets the convergence criteria (iterations number) then the cycle ends, if the optimization cycle is not fulfilled the particle swarm is evaluated by selecting the *Pgbest* and *Plbest* until the of end the cycle of optimization.

Figure 10 shows the PSO process for the AT2-FLC.

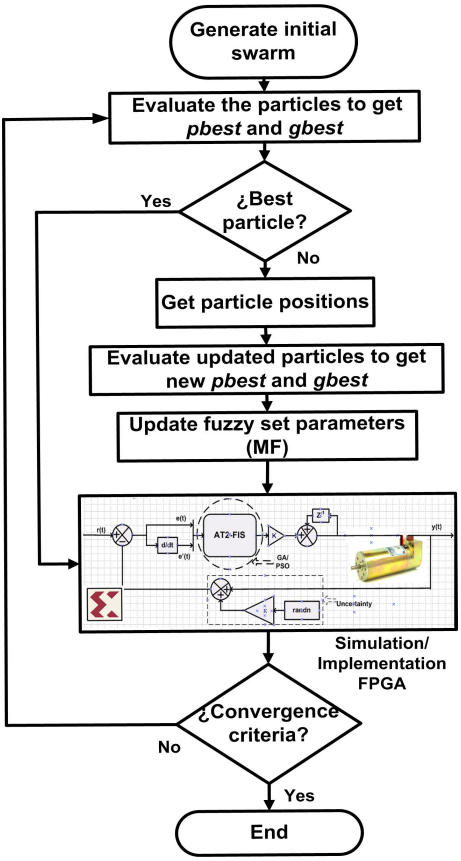


Fig. 10 T2-MFs optimization with PSO for the AT2-FLC for ReSDCM

The initial particles are created randomly respecting the ranges of the T2-MFs in Table 3. The particle swarm optimization fits the parameters of the T2-MFs in order to find the best AT2-FLC for ReSDCM using Equation 8, Equation 9 and Equation 10, which are evaluated in the simulation/implementation of the FPGA block of Figure 10, and for all experiments 10 particles are used.

6 Results of Average Approximation of the Interval Type-2 Fuzzy Logic Controller for FPGA and Their Optimization

To demonstrate the performance of the T2-MFs optimization for the AT2-FLC for ReSDCM in FPGA implementation, we considered two main experiments: T2-MFs with PSO for the AT2-FLC for ReSDCM using XSG, and T2-MFs with PSO for the AT2-FLC for ReSDCM in the FPGA device.

Several experiments were performed to find the best AT2-FLC optimized with PSO. The main idea is to achieve a comparison of the results obtained with PSO using XSG and PSO using FPGA device for the T2-MFs in an AT2-FLC for ReSDCM.

6.1 Results for T2-MFs Parameters with PSO for AT2-FLC Using XSG

Experiments were conducted for the T2-MFs with PSO using an AT2-FLC for ReSDCM in simulation environment using Simulink and XSG, the latter to simulate the AT2-FIS synthesizable VHDL code for FPGA.

Table 4 shows some results obtained for the T2-MFs with PSO, each experiment is an AT2-FLC with different T2-MFs parameters. Based on previous experience we changed the c_1 and c_2 parameters, which are used to calculate the velocity of each particle of the swarm, the number of iterations varies between 30 and 40, the calculated error is the average error, the time shown is the runtime of the optimization, and each of the experiments was carried out with 10 particles.

Table 4 Results for the T2-MFs with PSO for AT2-FLC using XSG

No.	Iteration Number	C_1	C_2	Average Error	Time(min)
1	30	0.3	0.2	0.2175	26.5080
2	35	0.3	0.2	0.1947	30.1500
3	30	0.3	0.3	0.1785	26.8491
4	30	0.3	0.3	0.1785	37.3229
5	30	0.3	0.2	0.1684	35.3005
6	30	0.3	0.15	0.1598	28.0082
7	30	0.3	0.2	0.1684	18.4350
8	40	0.3	0.25	0.1570	18.9340
9	40	0.3	0.25	0.2015	22.6246
10	32	0.3	0.25	0.0872	28.4346
11	30	0.3	0.25	0.2034	22.3819
12	30	0.5	0.3	0.2015	10.1622
13	30	0.4	0.3	0.1560	18.7199

Table 4 (*continued*)

14	30	0.25	0.2	0.2175	26.3258
15	30	0.3	0.25	0.0742	16.9798
16	30	0.3	0.25	0.1161	16.8035
17	30	0.29	0.25	0.0912	17.5950
18	40	0.305	0.25	0.0390	21.7936
19	30	0.28	0.25	0.0921	17.7113

For this set of experiments the best AT2-FLC was obtained the experiment No.18, because it has a lower average error.

Figure 11 shows the T2-MFs of the error input for experiment No. 18.

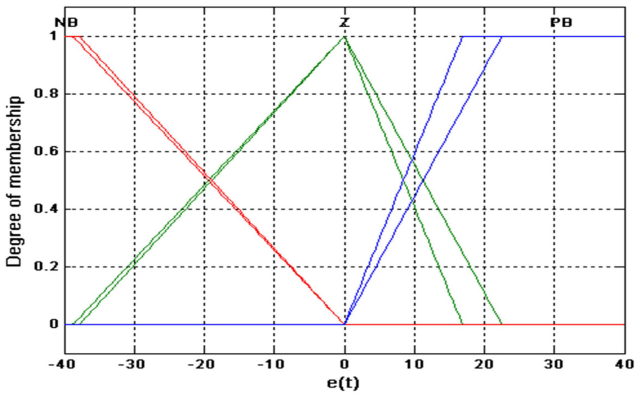


Fig. 11 T2-MFs for error input of the AT2-FIS for ReSDCM for experiment No. 18 using XSG

Figure 12 shows the T2-MFs of the change of error input for experiment No. 18.

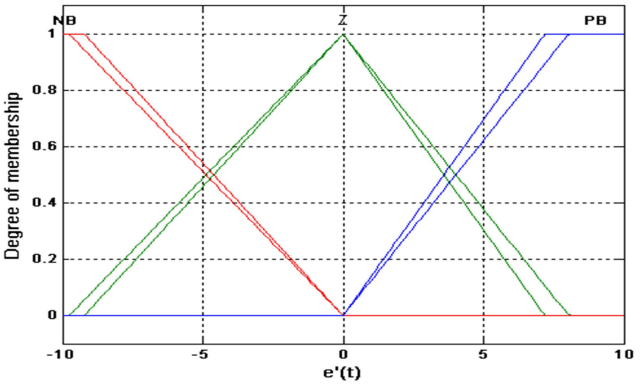


Fig. 12 T2-MFs for change of error input of the AT2-FIS for ReSDCM for experiment No. 18 using XSG

Figure 13 shows the T2-MFs of the output for experiment No. 18.

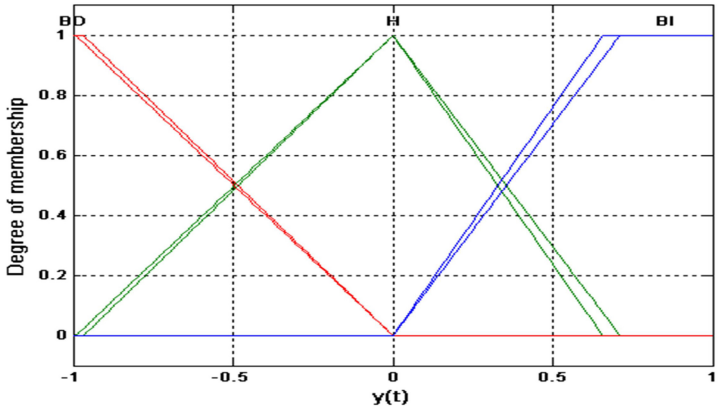


Fig. 13 T2-MFs for output of the AT2-FIS for ReSDCM for experiment No. 18 using XSG

Figure 14 shows the speed of the DC motor at 30 rpm for experiment No. 18.

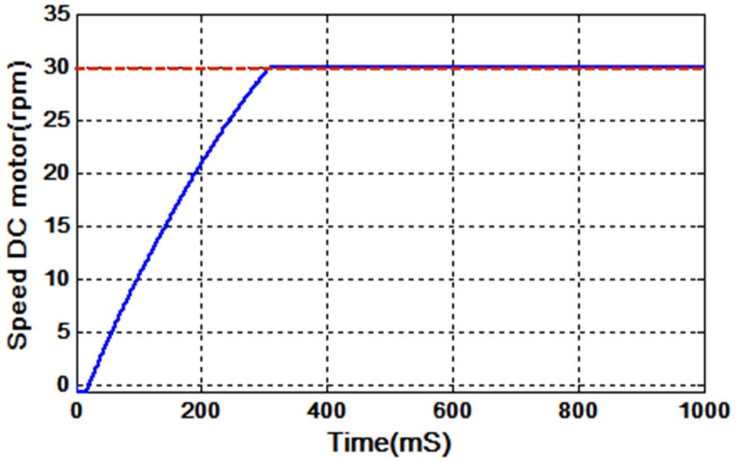


Fig. 14 Speed of the DC motor at 30 rpm for experiment No. 18 using XSG

Figure 15 shows the convergence error of the PSO for experiment No. 18.

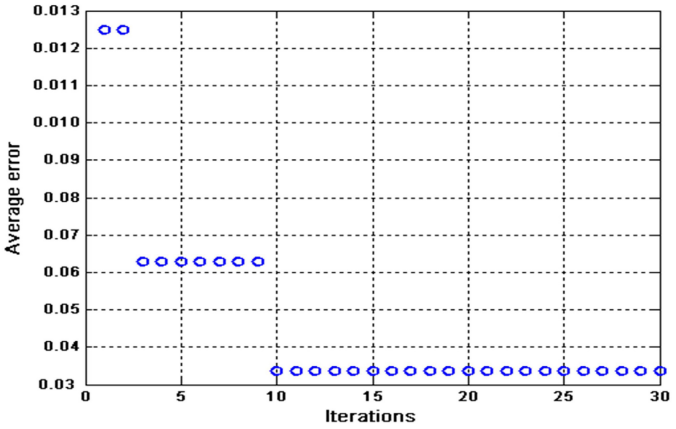


Fig. 15 Convergence error of the PSO for experiment No. 18 using XSG

The average error for the AT2-FLC in XSG with PSO is 0.39 and the runtime is 21.79 minutes.

Table 5 shows the comparison of results for the AT2-FLC (Experiment No.3) adding different levels of uncertainty.

Table 5 AT2-FLC in XSG optimized with PSO with some level of uncertainty

No.	Uncertainty level (x)	Average Error
1	0	0.0390
2	0.001	0.0366
3	0.005	0.0314
4	0.008	0.0314
5	0.05	0.0930
6	0.08	0.0924
7	0.1	0.1044
8	0.5	0.9946
9	0.8	1.1742

The idea of applying uncertainty to the AT2-FLC is to check its robustness of this controller. Figure 16 shows the speed of DC motor for AT2-FLC optimized with PSO for some level of uncertainty (x=0.001).

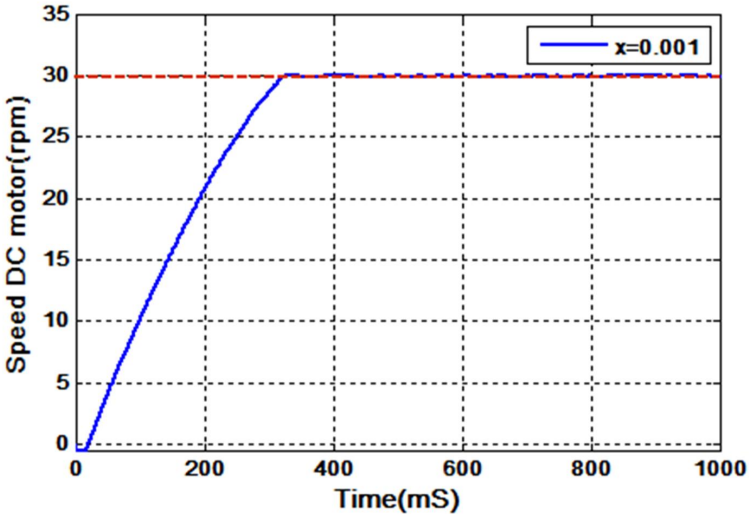


Fig. 16 AT2-FLC optimized with PSO at 30 rpm for some level of uncertainty

In the next section, the comparisons of results between XSG and FPGA device are shown.

6.2 Results for T2-MFs Parameters with PSO for AT2-FLC Using the FPGA Device

Different experiments were performed for the T2-MFs with PSO for the AT2-FLC using the FPGA device.

Table 6 shows the different results, each with different characteristics for the C_1 and C_2 constants and it shows the average error and runtime of the PSO.

Table 6 Results for the T2-MF with PSO for AT2-FLC using FPGA device

No.	Iteration Number	C_1	C_2	Average Error	Time(min)
1	30	0.19	0.19	0.6744	94.6352
2	30	0.15	0.15	1.1352	171.9780
3	30	0.2	0.2	0.5955	92.3928
4	30	0.2	0.2	1.8481	100.2600
5	30	0.21	0.21	0.8050	92.3712
6	30	0.25	0.2	1.1345	92.5114
7	30	0.25	0.25	1.5162	100.1839
8	30	0.2	0.19	0.6786	92.2988
9	30	0.2	0.19	1.1669	92.5059
10	30	0.1	0.1	1.0780	72.7945

Table 6 (*continued*)

11	30	0.1	0.1	1.0224	173.1366
12	30	0.2	0.3	1.5601	98.2220
13	30	0.1	0.09	1.3637	104.2218
14	30	0.1	0.09	0.9950	161.7849
15	30	0.1	0.09	1.1246	88.7282
16	30	0.1	0.08	0.9102	104.0216
17	30	0.09	0.09	0.6164	115.9914
18	30	0.09	0.09	1.3634	172.9863
19	30	0.09	0.08	1.2134	132.1412

For this set of experiments the best AT2-FLC was obtained for experiment No. 3, because it has a lower average error.

Figure 17 shows the T2-MFs of the error input for experiment No. 3.

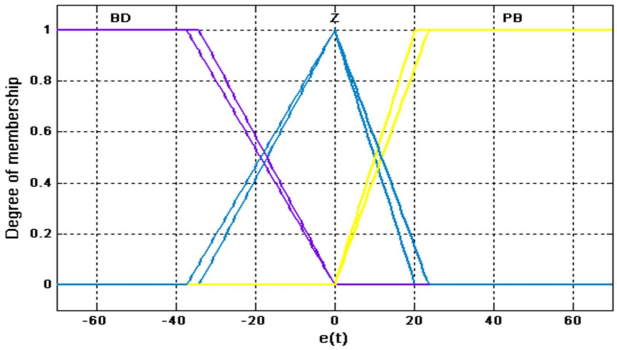


Fig. 17 T2-MFs for error input of the AT2-FIS for ReSDCM for experiment No. 3 using FPGA device

Figure 18 shows the T2-MFs of the change of error input for experiment No. 3.

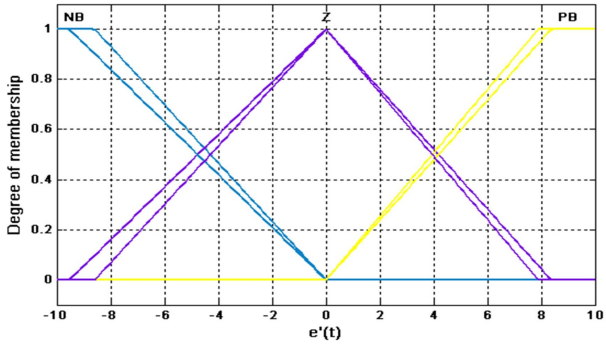


Fig. 18 T2-MFs for change of error input of the AT2-FIS for ReSDCM for experiment No. 3 using FPGA device

Figure 19 shows the T2-MFs of the output for experiment No. 3.

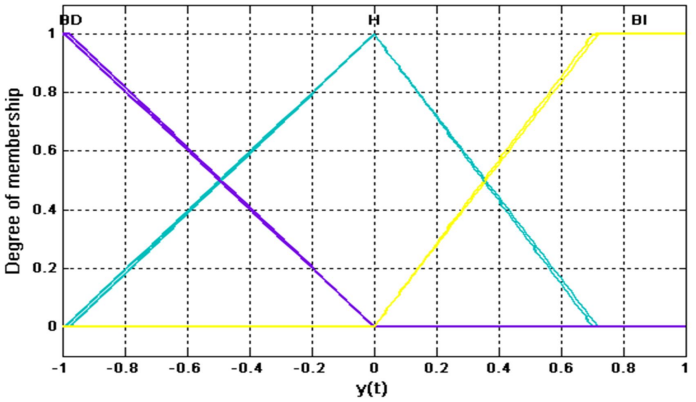


Fig. 19 T2-MFs for output of the AT2-FIS for ReSDCM for experiment No. 3 using the FPGA device

Figure 20 shows the speed of the DC motor (30 rpm) for the AT2-FLC of experiment No. 3.

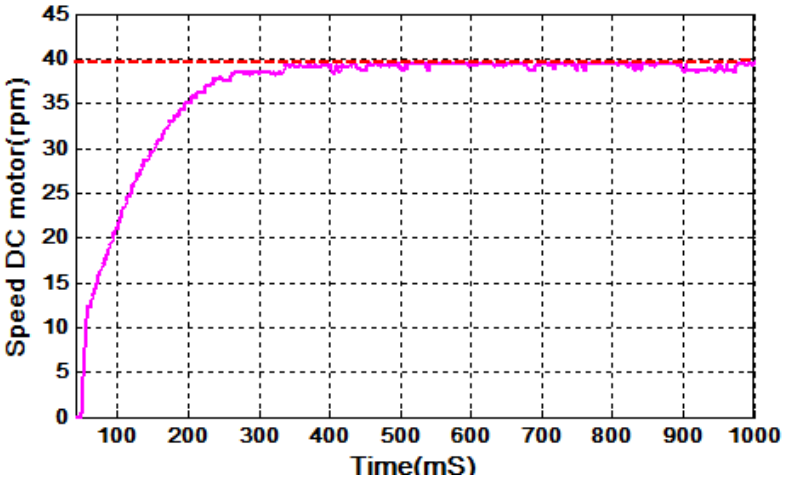


Fig. 20 Speed of the DC motor at 40 rpm for experiment No.3

Figure 21 shows the convergence error of the PSO for experiment No. 3.

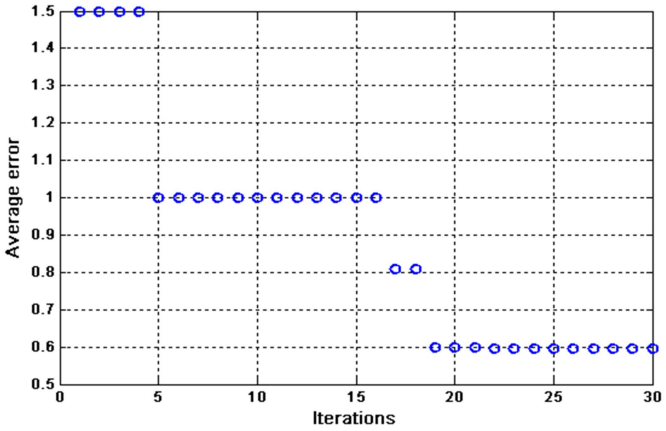


Fig. 21 Convergence error of the PSO for experiment No. 3 using FPGA device

Table 6 shows that the best AT2-FLC was that of experiment No.3 with 30 generations in 92.3928 minutes with 0.5955 of average error.

The best AT2- FLC optimized with PSO (Experiment No.3) was presented. The main objective is to apply uncertainty (Equation 7) to AT2-FLC for ReSDCM, in this case we are making comparisons for a desired speed of 40 rpm.

Table 7 shows the comparison of the AT2-FLC (Experiment No.3) adding different levels of uncertainty.

Table 7 Comparison with different levels of uncertainty of the best AT2-FLC in FPGA optimized with PSO

No.	Uncertainty level (x)	Average Error
1	0	0.5955
2	0.001	0.6940
3	0.005	0.7905
4	0.008	0.8386
5	0.05	0.7785
6	0.08	0.7972
7	0.1	0.9673
8	0.5	1.1724
9	0.8	1.3158
10	1	1.6142

The idea of applying uncertainty to the AT2-FLC is to check its robustness of this controller. Figure 22 shows the speed of DC motor for AT2-FLC optimized with PSO for different levels of uncertainty.

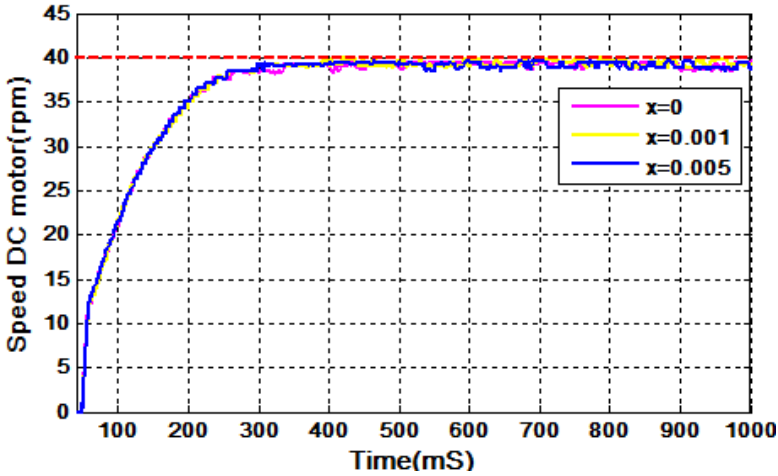


Fig. 22 AT2-FLC optimized with PSO at 40 rpm for different levels of uncertainty

Figure 23 shows a graphical comparison of the average errors of the AT2-FLC (in the FPGA device) optimization using the PSO in 19 different experiments.

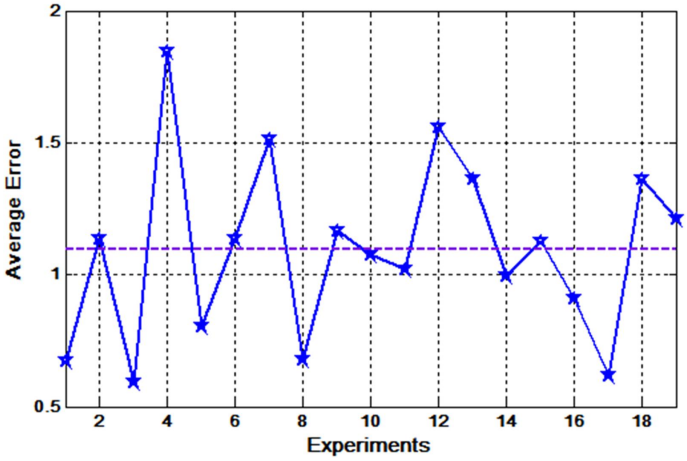


Fig. 23 Average errors for the PSO method for AT2-FLC

In Figure 23, we calculated the mean of average error for all experiments; in this case, the PSO method has an average error of 1.0948.

Figure 24 shows a graphical comparison of the runtime of the AT2-FLC (in FPGA device) optimization using the PSO in 19 different experiments.

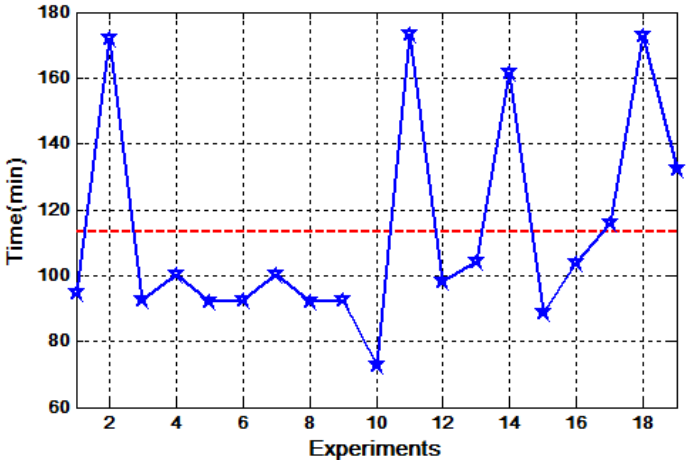


Fig. 24 Runtime for the PSO method for AT2-FLC

Of the runtime of the PSO obtained in Figure 24, we calculated the mean of the runtime; in this case, the PSO method has a runtime of 113.3244 min.

Now the results obtained for different codifications (number of bits) of the AT2-FLC in FPGA device are also shown. The idea is to compare the results obtained by changing the number of bits for encoding the AT2-FLC in VHDL code for FPGA device.

Experiments were conducted with different resolutions of the VHDL codification for AT2-FLC with PSO optimization, in this case we use 8, 10, 14, 16, 20 and 24 bits.

Experiments were conducted with different resolutions of the VHDL codification for AT2-FLC optimized with PSO, in this case we use 8, 10, 14, 16, 20 and 24 bits. Table 8 shows the results obtained of the AT2-FLC (in FPGA device) with PSO for ReSDCM at 30 rpm for different number of bits.

Table 8 Comparison of the best AT2-FLC optimized with PSO for different number of bits

Resolution (Bits)	Average Error
8	0.5955
10	0.3976
14	0.1917
16	0.1917
20	0.1917
24	0.1917

In Table 8 we can notice that after 14 bits, the average error does not change, this is because the AT2-FLC performed operations with floating point and it is likely that after a certain number of bits some data can not be considered.

Figure 25 shows the comparison of the speed of the DC motor for different numbers of bits.

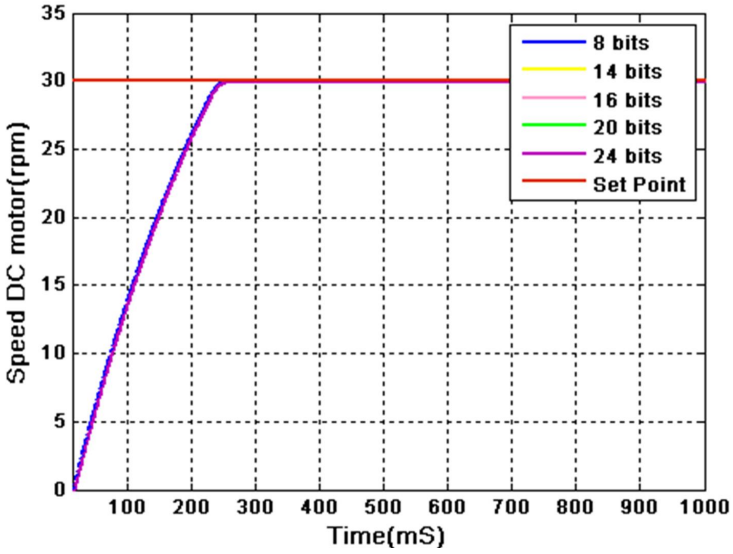


Fig. 25 Comparison of the speed DC motor for different number of bits

Figure 26(a) and Figure 26(b) show a zoom to observe the difference between speeds DC motor with different resolution for AT2-FLC at 30 rpm. In Figure 26(b), we notice the speed DC motor for 8 bits and 24 bits, other speeds are not appreciated because they have that the same average error as the speed of the DC motor for 24 bits.

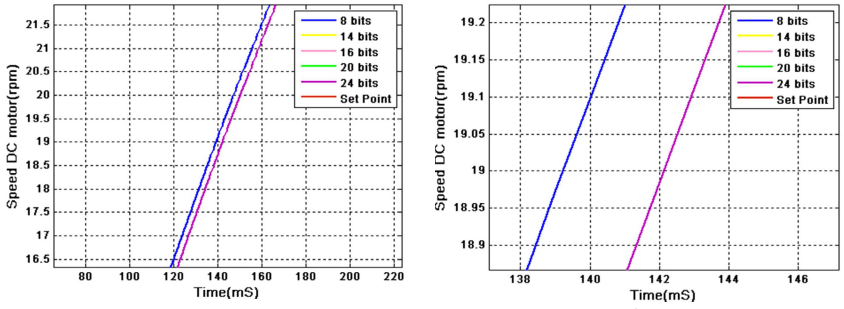


Fig. 26 Comparison of the speed DC motor for different number of bits

In Figure 26(b) we have a close-up to get a better view of the behavior of the speeds, we note that the speed of the DC motor for 8 bits has a lower time delay compared to the others, however the others speeds of the DC motor have a lower average.

7 Conclusions

In this paper, an average approximation of an interval type-2 fuzzy system was designed and the hardware implementation was proposed, in this case to be implemented into a FPGA.

The T2-MFs parameters were optimized with PSO, the optimization takes place outside the FPGA, because once the AT2-FLC for ReSDCM was optimized, it not needed to re-optimize it, unless this one fails in the system or change the initial conditions. The objective function of the PSO considers three characteristics: overshoot, undershoot and steady state error.

Our goal is to achieve an optimized AT2-FLC in a small runtime, and for this reason the fuzzy rules are not changed and we propose an optimization for T2-MFs where only some of these parameters are modified.

Due to the fact that our AT2-FLC has the feature that the user selects the number of bits which encode the controllers, various experiments were conducted demonstrating that an 8 bits parallel implementation of the algorithm is capable of provide real time operation for this hardware platform. However we make comparisons for different number of bits for VHDL, reaching the conclusion for our particular application, which is the regulation speed of the DC motor, the use of 14 bits is the best option because the error decreases in a 32.19 percentage compared with the 8 bits initially used in the implementation. For all experiments, it was considered the common goal of controlling the speed of the DC motor in a FPGA.

An AT2-FLC implementation based on a Xilinx Spartan 3AN FPGA was proposed. We have shown the device utilization for FPGA, these results are encouraging because they allows us to introduce more T2-MFs and fuzzy rules, in other words, a more complex AT2-FLC to obtain a better result, but this would increase the runtime.

References

[1] Castillo, O.: Interval Type-2 Fuzzy Logic for Control Applications. In: IEEE International Conference on Granular Computing, pp. 79–84 (2010)

[2] Castillo, O.: Interval Type-2 Mamdani Fuzzy Systems for Intelligent Control. In: Trillas, E., Bonissone, P.P., Magdalena, L., Kacprzyk, J. (eds.) Combining Experimentation and Theory. STUDFUZZ, vol. 271, pp. 163–178. Springer, Heidelberg (2011)

[3] Castillo, O.: Type-2 Fuzzy Logic in Intelligent Control Applications. STUDFUZZ, vol. 272. Springer, Heidelberg (2012)

[4] Castillo, O., Melin, P.: Optimization of Type-2 Fuzzy Systems Based on Bio-Inspired Methods: A Concise Review. Information Sciences 205(1), 1–19 (2012)

[5] Clerk, M.: Particle Swarm Optimization. ISTE Ltd. (2006)
[6] Gutierrez-Rios, J., Brox, P., Fernandez-Hernandez, F., Baturone, I., Sanchez-Solano, A.: Fuzzy Motion Adaptive Algorithm and its Hardware Implementation for Video de-interlacing. Applied Soft Computing 11, 3311–3320 (2011)
[7] Jantzen, J.: Tunning of Fuzzy PID Controllers, Technical University of Denmark, Department of Automation, pp. 1–22 (1998)
[8] Klir, G.J., Mark, J.W.: Uncertainty Based Information: Elements of Generalized Information Theory. Physica-Verlag (1999)
[9] Man, K.F., Tang, K.S., Kwong, S.: Genetic algorithms. Springer (2000)
[10] Maldonado, Y., Castillo, O., Melin, P.: Design of a Type-2 Fuzzy Controller and its Comparison with Type-1 Fuzzy and PID Controllers for Velocity Regulation in a DC Motor. In: Proceedings of the World Conference on Soft Computing, pp. 1–6 (2011)
[11] Mendel, J.M.: Uncertainty Rule Based Fuzzy Logic Systems: Introduction and New Directions. Prentice-Hall, Upper-Saddle River (2001)
[12] Mendel, J.M.: Type-2 Fuzzy Sets and Systems: an Overview. IEEE Computational Intelligence Magazine 2, 20–29 (2007)
[13] Montiel, O., Sepúlveda, R., Maldonado, Y., Castillo, O.: Design and Simulation of the Type-2 Fuzzification Stage: Using Active Membership Functions. In: Castillo, O., Pedrycz, W., Kacprzyk, J. (eds.) Evolutionary Design of Intelligent Systems. SCI, vol. 257, pp. 273–293. Springer, Heidelberg (2009)
[14] Sepulveda, R., Castillo, O., Melin, P., Montiel, O.: An Efficient Computational Method to Implement Type-2 Fuzzy Logic in Control Applications. In: Melin, P., Castillo, O., Ramírez, E.G., Kacprzyk, J., Pedrycz, W. (eds.) Anal. and Des. of Intel. Sys. using SC Tech. ASC, vol. 41, pp. 45–52. Springer, Heidelberg (2007)
[15] Sepúlveda, R., Montiel, O., Olivas, J., Castillo, O.: Methodology to Test and Validate a VHDL Inference Engine of a Type-2 FIS, through the Xilinx System Generator. In: Castillo, O., Pedrycz, W., Kacprzyk, J. (eds.) Evolutionary Design of Intelligent Systems. SCI, vol. 257, pp. 295–308. Springer, Heidelberg (2009)
[16] Zadeh, L.A.: Fuzzy Sets. Information and Control 8, 338–353 (1965)

Genetic Optimization of Modular Type-1 Fuzzy Controllers for Complex Control Problems

Leticia Cervantes and Oscar Castillo

Tijuana Institute of Technology,
Tijuana, Mexico
ocastillo@tectijuana.mx

Abstract. In this paper a method to design modular type-1 fuzzy controllers using genetic optimization is presented. The method is tested with a problem that requires five individual controllers. Simulation results with a genetic algorithm for optimizing the membership functions of the five individual controllers are presented. Simulation results show that the proposed modular control approach offers advantages over existing control methods.

1 Introduction

This paper focuses on the field of fuzzy logic and control area, these areas can work together to solve various control problems. The problem of water level control for a three tank system is illustrated. This control is carried out by controlling five valves whose outputs are the inputs to the three tanks. The main idea in this paper is to apply a genetic algorithm to optimize the membership functions of the five controllers. Each controller has to open and close one of the valves. To control each of the valves we have five type-1 fuzzy systems and each fuzzy system has to control one valve of the three tanks. After that, the simulation is carried out using type-1 fuzzy systems, and then genetic algorithms are used to optimize the five controllers. Finally results are presented and compared.

The rest of the paper is organized as follows: In section 2 some basic concepts to understand the work are presented, Section 3 shows a case study, problem description and results are presented and finally in Section 4 conclusion is shown.

2 Background and Basic Concepts

In this section some basic concepts needed for this work are presented.

2.1 Genetic Algorithm

Genetic algorithms (GAs) were proposed by John Holland in the 1960s and were developed by Holland and his students and colleagues at the University of

P. Melin, O. Castillo (Eds.): *Soft Computing Appli. in Optimization,* STUDFUZZ 294, pp. 125–154.
DOI: 10.1007/978-3-642-35323-9_6

Michigan in the 1960s and the 1970s [2][3]. In contrast with evolution strategies and evolutionary programming, Holland's original goal was not to design algorithms to solve specific problems, but rather to formally study the phenomenon of adaptation as it occurs in nature and to develop ways in which the mechanisms of natural adaptation might be imported into computer systems [15][19]. Holland's 1975 book Adaptation in Natural and Artificial Systems presented the genetic algorithm as an abstraction of biological evolution and gave a theoretical framework for adaptation under the GA [4][5]. A GA allows a population composed of many individuals to evolve under specified selection rules to a state that maximizes the "fitness" [17]. Holland's GA is a method for moving from one population of "chromosomes" (e.g., strings of ones and zeros, or "bits") to a new population by using a kind of "natural selection" together with the genetics inspired operators of crossover, mutation, and inversion [18]. Each chromosome consists of "genes" (e.g., bits), each gene being an instance of a particular "allele" (e.g., 0 or 1) [14][10]. The selection operator chooses those chromosomes in the population that will be allowed to reproduce, and on average the fitter chromosomes produce more offspring than the less fit ones [28]. Crossover exchanges subparts of two chromosomes, roughly mimicking biological recombination between two single chromosome ("haploid") organisms; mutation randomly changes the allele values of some locations in the chromosome; and inversion reverses the order of a contiguous section of the chromosome, thus rearranging the order in which genes are arrayed. (Here, as in most of the GA literature, "crossover" and "recombination" will mean the same thing.) [7][16]. Some of the advantages of a GA include: Optimizes with continuous or discrete variables, doesn't require derivative information, simultaneously searches from a wide sampling of the cost surface, deals with a large number of variables [13][29].

A typical algorithm might consist of the following:

1. Start with a randomly generated population of n l–bit chromosomes (candidate solutions to a problem).
2. Calculate the fitness $f(x)$ of each chromosome x in the population.
3. Repeat the following steps until n offspring have been created:
   - Select a pair of parent chromosomes from the current population, the probability of selection being an increasing function of fitness. Selection is done "with replacement," meaning that the same chromosome can be selected more than once to become a parent.
   - With probability P_c (the "crossover probability" or "crossover rate"), cross over the pair at a randomly chosen point (chosen with uniform probability) to form two offspring. If no crossover takes place, form two offspring that are exact copies of their respective parents. (Note that here the crossover rate is defined to be the probability that two parents will cross over in a single point. There are also "multipoint crossover" versions of the GA in which the crossover rate for a pair of parents is the number of points at which a crossover takes place.)
   - Mutate the two offspring at each locus with probability P_m (the mutation probability or mutation rate), and place the resulting chromosomes in the

new population. If n is odd, one new population member can be discarded at random.

- Replace the current population with the new population.

Go to step 2 [30][31].

2.2 Fuzzy Systems

The idea of fuzzy systems appeared very early in the literature of fuzzy sets; it was originated by Zadeh (1965). The concept of a fuzzy system is intimately related to that of a fuzzy set. En order to make our discussion self-contained, it will be helpful to begin with a brief summary of some of the basic definitions pertaining to such sets. Research on fuzzy systems seems to have developed in two main directions. The first is rather formal and considers fuzzy systems as a generalization of nondeterministic systems. These have been studied within the same conceptual framework as classical systems. This approach has given birth to a body of abstract results in such fields as minimal realization theory and formal automata theory, sometimes expressed in the setting of category theory. The system is considered over a given period during which inputs, outputs, and relations may change [28][13].

A system will be called fuzzy as soon as inputs or outputs are modeled as fuzzy sets or their interactions are represented by fuzzy relations. Usually, a system is also described in terms of state variables. In a fuzzy system a state can be a fuzzy set. However, the notion of a fuzzy state is quite ambiguous and needs to be clarified. Note that generally a fuzzy system is an approximate representation of a complex process that is not itself necessarily fuzzy [20][21]. According to Zadeh, the human ability to perceive complex phenomena stems from the use of names of fuzzy sets to summarize information [22]. The notion of probabilistic system corresponds to a different point of view: all the available information at any time is modeled by probability distributions, built from repeated experiments. A fuzzy system can be described either as a set of fuzzy logical rules or as a set of fuzzy equations [23][24]. Fuzzy logical rules must be understood as propositions associated with possibility distributions. For instance, “if last input is small, then if last output is large, then current output is medium”, where “small” is a fuzzy set on the universe of inputs, and “medium” and “large” are fuzzy sets on the universe of outputs [25][26] . Let u_t , y_t, and s_t denote respectively the input, output, and state of a system S at time t. U, Y, S are respectively the set of possible inputs, outputs, and states [27][32]. Such a system is said to be deterministic if it is characterized by state equations of the form:

$$s_{t+1} = \delta(u_t, s_t), \qquad y_t = \sigma(s_t), \qquad t \in \mathbb{N}. \tag{1}$$

s_0 is called the initial state; δ and Ω are functions from U X S and from S to S and Y, respectively. S is said to be nondeterministic if S_{t+1} and / or Y_t, are not uniquely determined by U_t and S_t [33][1]. Let S_{t+1} and Y_t be the sets of possible values of S_{t+1} and Y_t, respectively, given U_t, and S_t. S_{t+1} and Y_t, may be understood as

binary possibility distributions over *S* and *Y*, respectively. In some cases a fuzzy system is used to control complex problem to obtain better results [8][9][6].

2.3 Fuzzy Control Systems

Control systems theory, or what is called modern control systems theory today, can be traced back to the age of World War II, or even earlier, when the design, analysis, and synthesis of servomechanisms were essential in the manufacturing of electromechanical systems. The development of control systems theory has since gone through an evolutionary process, starting from some basic, simplistic, frequency-domain analysis for single-input single output (SISO) linear control systems, and generalized to a mathematically sophisticated modern theory of multi-input multi-output (MIMO) linear or nonlinear systems described by differential and/or difference equations.

It is believed that the advances of space technology in the 1950s completely changed the spirit and orientation of the classical control systems theory: the challenges posed by the high accuracy and extreme complexity of the space systems, such as space vehicles and structures, stimulated and promoted the existing control theory very strongly, developing it to such a high mathematical level that can use many new concepts like state-space and optimal controls. The theory is still rapidly growing today; it employs many advanced mathematics such as differential geometry, operation theory, and functional analysis, and connects to many theoretical and applied sciences like artificial intelligence, computer science, and various types of engineering. This modern control systems theory, referred to as *conventional* or *classical* control systems theory, has been extensively developed. The theory is now relatively complete for linear control systems, and has taken the lead in modern technology and industrial applications where control and automation are fundamental. Basically, the aim of fuzzy control systems theory is to extend the existing successful conventional control systems techniques and methods as much as possible, and to develop many new and special-purposed ones, for a much larger class of complex, complicated, and ill-modeled systems – fuzzy systems. This theory is developed for solving real-world problems [11].

Fuzzy controllers have been well accepted in control engineering practice. The major advantages in all these fuzzy-based control schemes are that the developed controllers can be employed to deal with increasingly complex systems to implement the controller without any precise knowledge of the structure of entire dynamic model. As a knowledge-based approach, the fuzzy controller usually depends on linguistics-based reasoning in design. However, even though a system is well defined mathematically, the fuzzy controller is still preferred by control engineers since it is relatively more understandable whereas expert knowledge can be incorporated conveniently. Recently, the fuzzy controller of nonlinear systems was studied by many authors and has also been extensively adopted in adaptive control of robot manipulators. It has been proven that adaptive fuzzy control is a powerful technique and being increasingly applied in the discipline of systems

control, especially when the controlled system has uncertainties and highly nonlinearities [12].

3 Case Study

In this Section the problem description is presented and results are shown.

3.1 Problem Description

In this work the case study considers the problem of water level control for a 3 tanks system where the 3 tanks include valves that are opened or closed, these valves must be well controlled to give the desired level of water in each of the three tanks. The end tanks have a valve that fills and in the middle of the 3 tanks there are two valves that control the water level between tanks 1 and 2, and tanks 2 and 3. The water tank 3 has a valve to output more water flow, the case study model is made in Simulink and has three inputs (tank 1, tank2 and tank3), and these inputs correspond to the existing water levels in tank 1, tank2 and tank3. The outputs of the model made in Simulink has five valves, which provide water (v1 and v2) valves that are interconnected tanks (v13 and v32) and finally the output valve is responsible for the drainage of the three tanks (v20). The problem is shown in Figure 1.

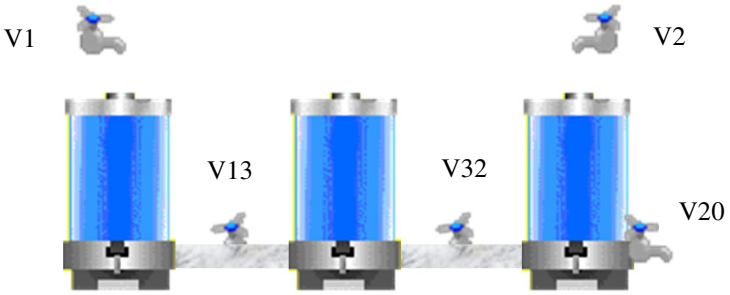


Fig. 1 Water control of 3 tanks

3.2 Type-1 Fuzzy System

For this case study it was necessary to use fuzzy systems to realize the simulation, each fuzzy system has one or two inputs depend on the valve. The Valves that are between 2 tanks are using 2 inputs (tank1 and tank2 or tank2 and tank3). The outputs are the valves, in total 5 fuzzy systems were used in this problem. The fuzzy systems are shown in Figures 2 to 6.

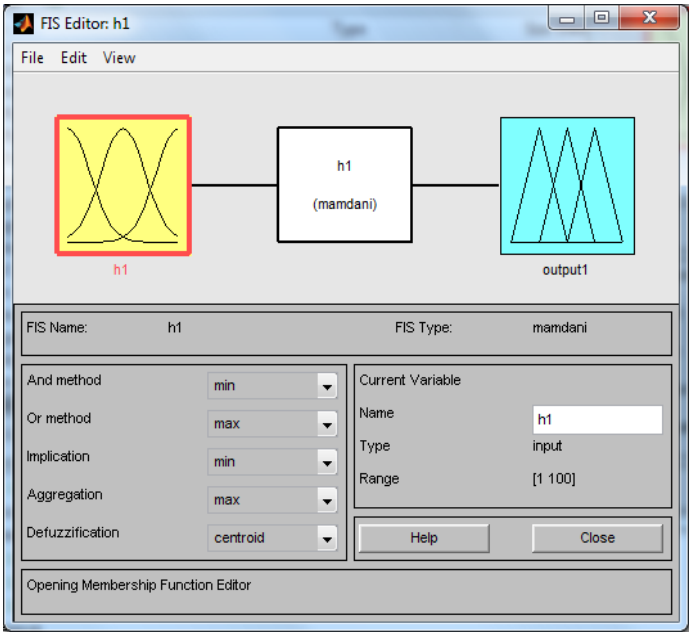


Fig. 2 Fuzzy system to control valve 1

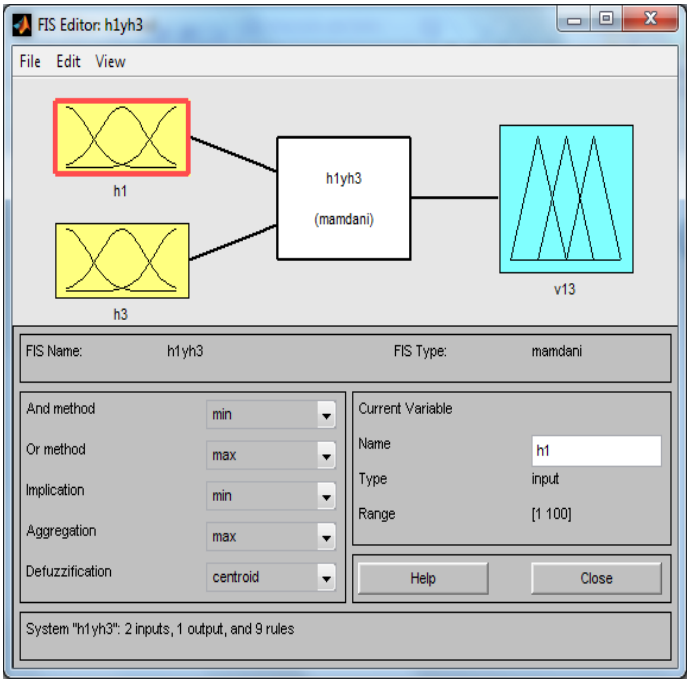


Fig. 3 Fuzzy system to control valve 13

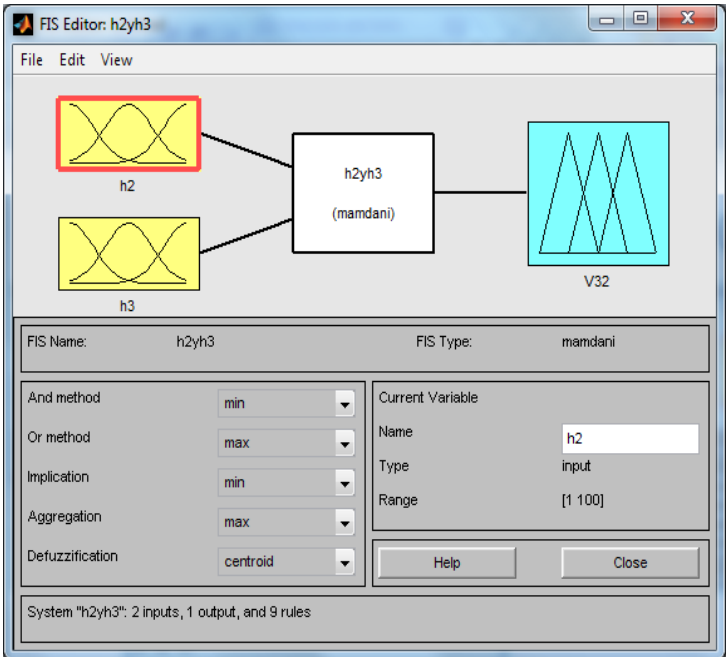


Fig. 4 Fuzzy system to control valve 32

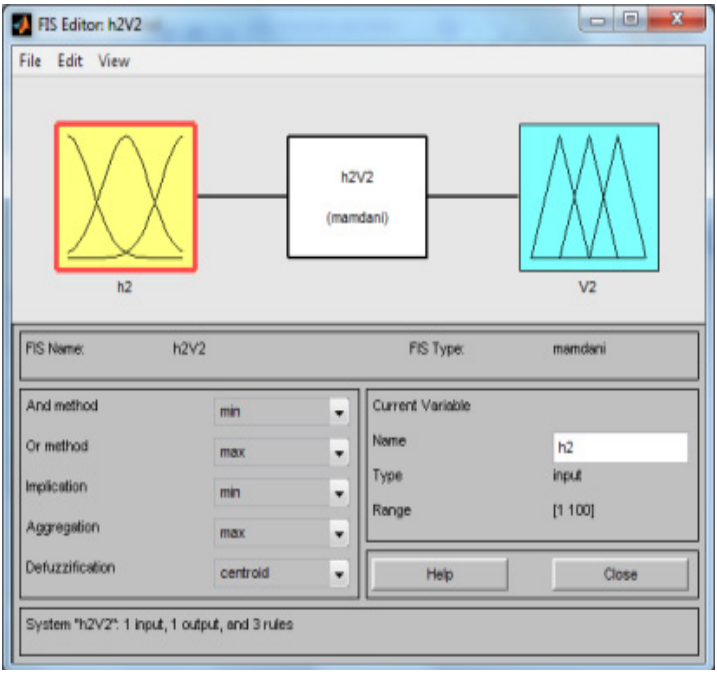


Fig. 5 Fuzzy system to control valve 2

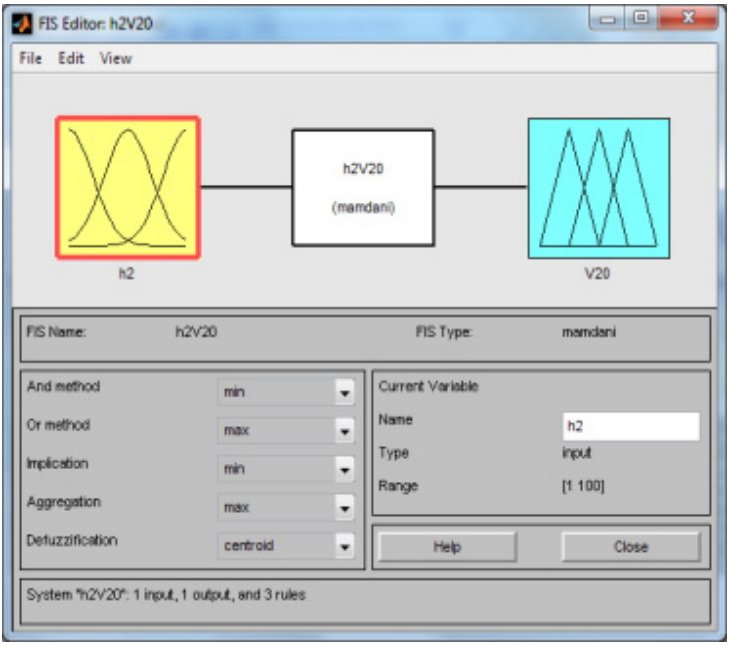


Fig. 6 Fuzzy system to control valve 20

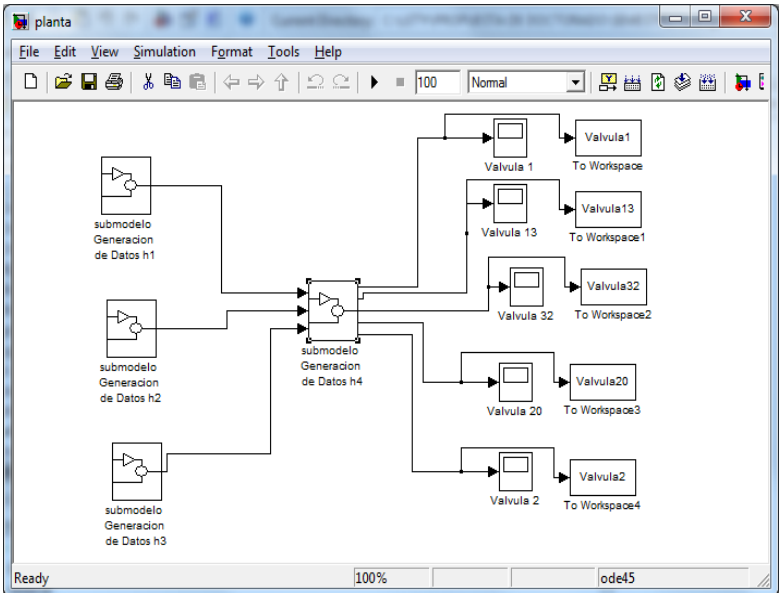


Fig. 7 Simulation plant

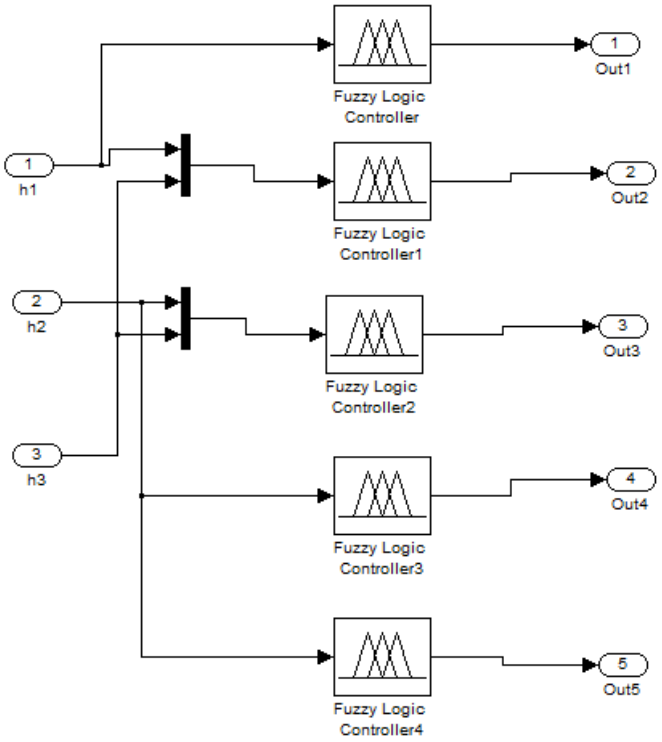


Fig. 8 Simulation plant showing inputs and outputs

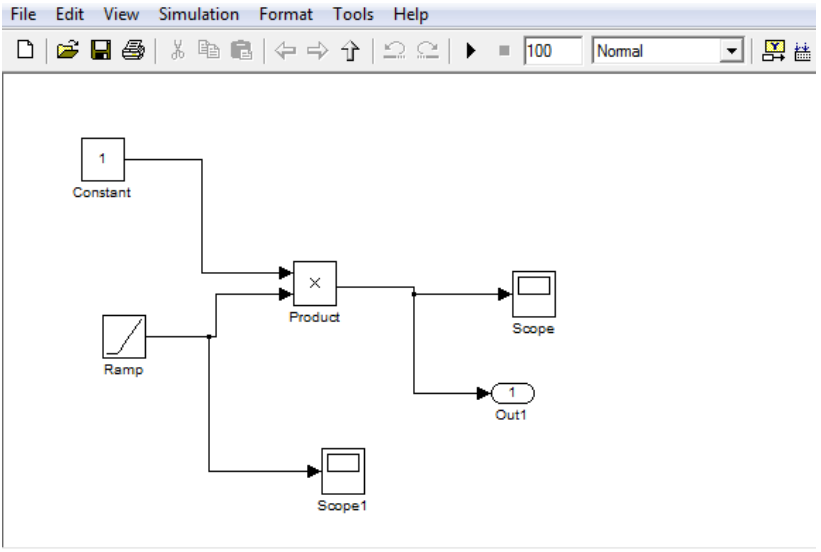


Fig. 9 Tank water simulation plant

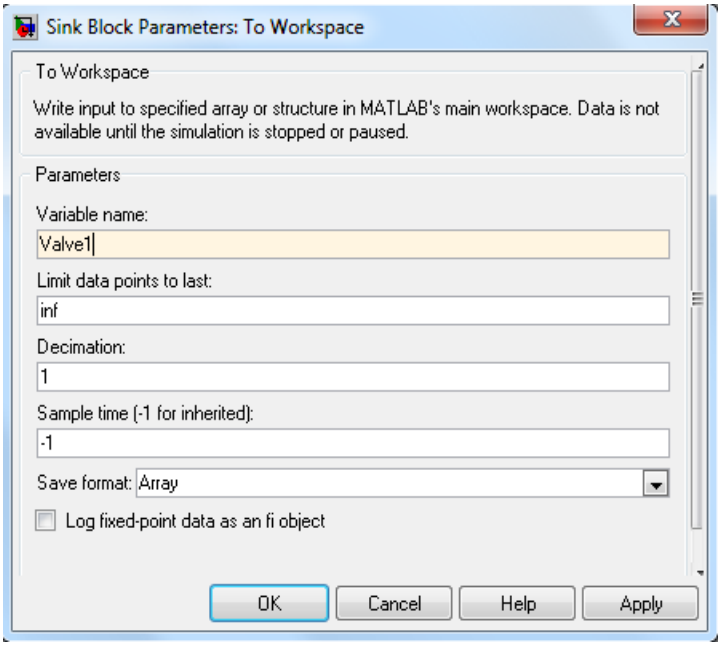


Fig. 10 Data block of the simulation plant

Having created the previous fuzzy systems, the simulation was performed using the Matlab language. The simulation plant is shown in Figures 7 to 10.

The simulation was carried out using the fuzzy systems shown before, the membership functions used in this case were triangular, Gaussian and trapezoidal, and the fuzzy systems with the different types of membership functions used in this case of study are shown in Figures 11 to 16.

All the valves in the inputs and outputs have 3 membership functions, all the membership functions in each input or output have the same position initially and this is because a genetic algorithm is applied to optimize each membership function.

When the genetic algorithm is used the membership functions start to move within the specified range. Later in section 3.3 the fuzzy system with genetic algorithm is presented where it shows new positions in all de membership functions. Figures 11 to 16 show the membership functions in the inputs and outputs of all fuzzy systems. The fuzzy systems that have one input are presented in Figures 11 to 13, and the fuzzy systems that have 2 inputs are presented in Figures 14 to 16.

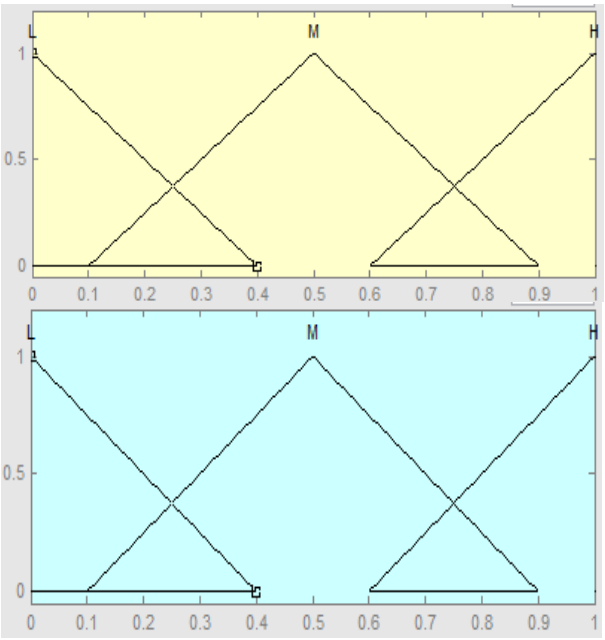


Fig. 11 Triangular membership functions use in valve 1, valve 2 and valve 20

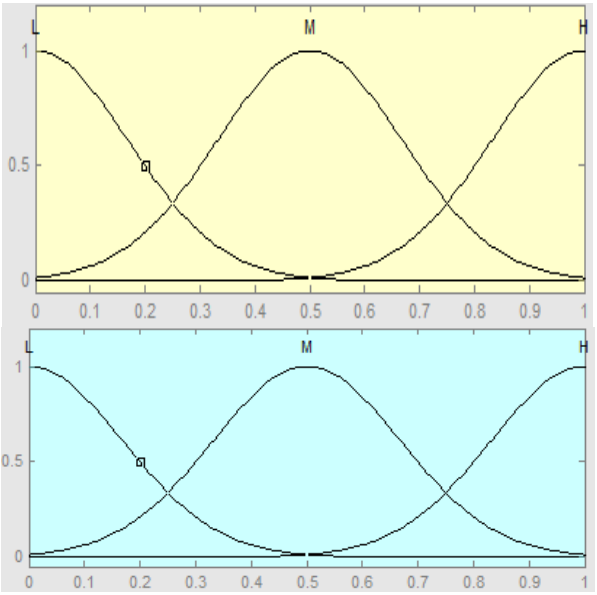


Fig. 12 Gaussian membership functions use in valve 1,valve 2 and valve 20

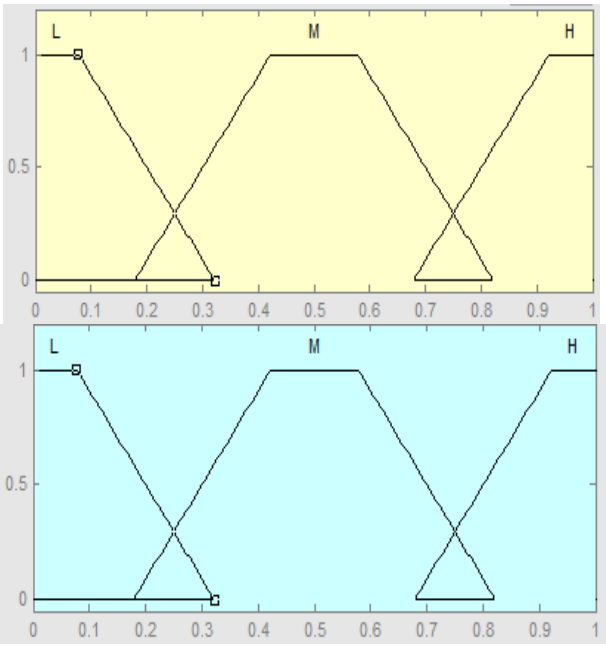


Fig. 13 Trapezoidal membership functions use in valve 1,valve 2 and valve 20

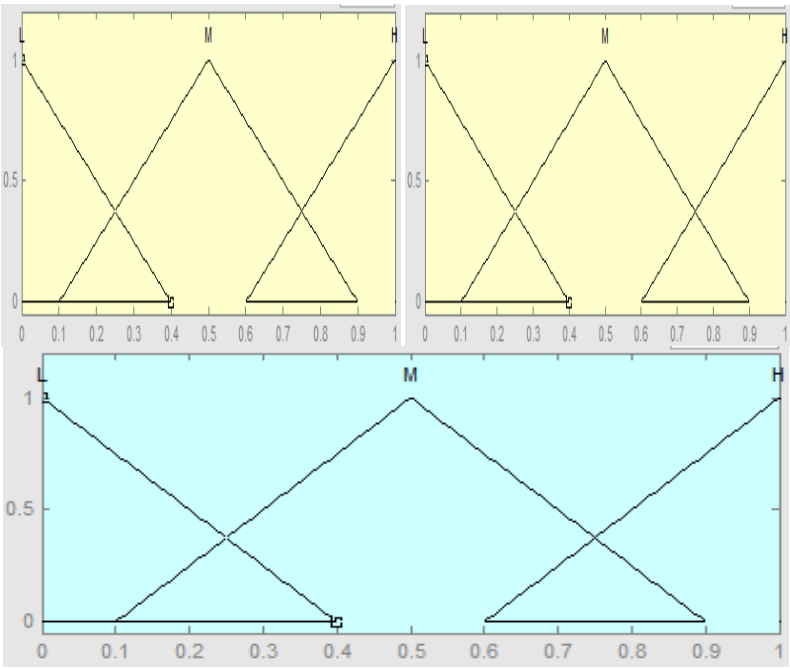


Fig. 14 Triangular membership functions use in valve 13 and valve 32

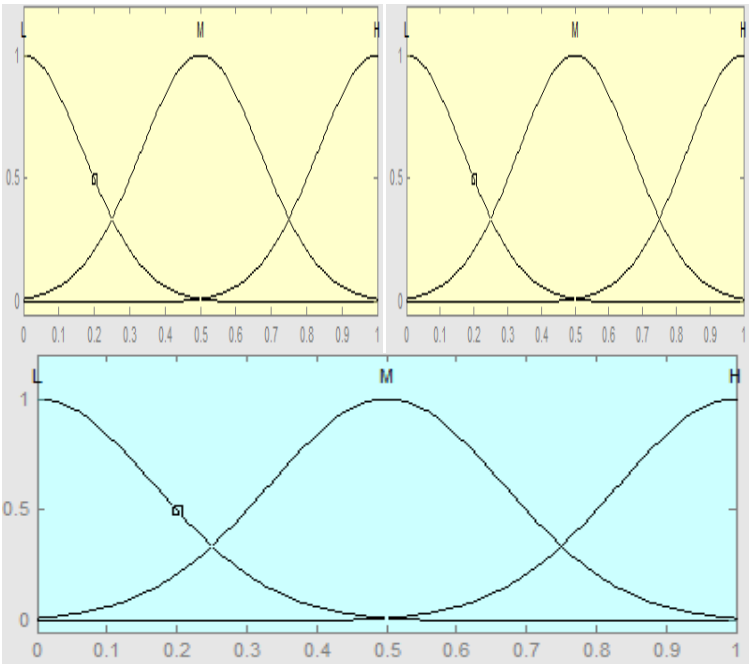


Fig. 15 Gaussian membership functions use in valve 13 and valve 32

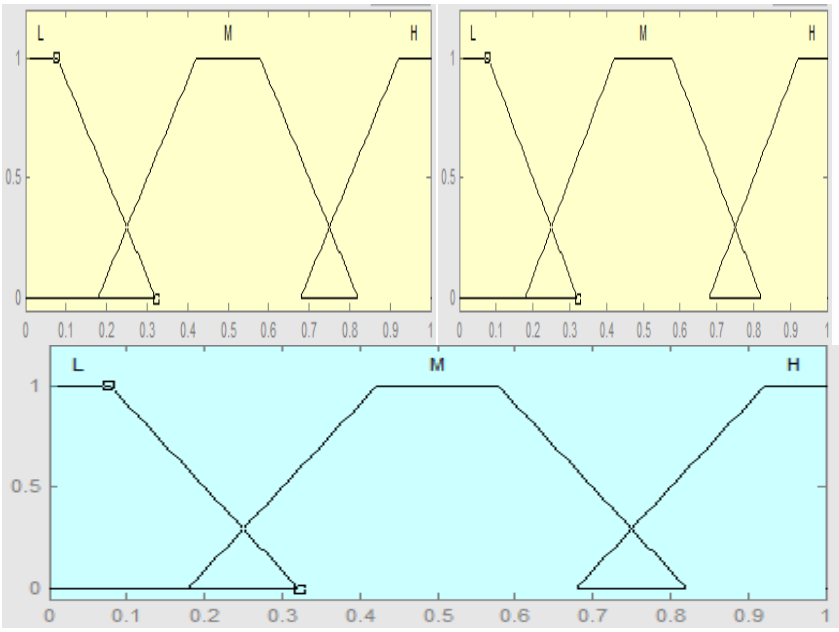


Fig. 16 Trapezoidal membership functions use in valve 13 and valve 32

Table 1 Results for the simulation plant using triangular membership functions

Using Triangular Membership Function	Error
valve 1	0.9246
valve 13	0.9278
valve 2	0.9278
valve 20	0.9279
valve 32	0.8341

Table 2 Results for the simulation plant using Gaussian membership functions

Using Gaussian Membership Function	Error
valve 1	0.898
valve 13	0.8994
valve 2	0.8994
valve 20	0.8995
valve 32	0.8463

Table 3 Results for the simulation plant using trapezoidal membership functions

Using Trapezoidal Membership Function	Error
valve 1	0.9522
valve 13	0.9551
valve 2	0.9504
valve 20	0.9551
valve 32	0.8233

The results with type-1 fuzzy systems are presented in Tables 1 to 3.

In the previous Tables the error of each valve was presented using different types of membership functions. The error when triangular membership functions were used is different as in trapezoidal and Gaussian. Is important to use different types of membership functions because the error can be vary and depends how complex is the problem to control sometimes the case of the study can be better using Gaussian membership function for the soft behavior or in other case can be better using another type, but is important to consider other types. In Figures 17 to 21 graphics are shown of each valve with the 3 types of membership functions used in this case.

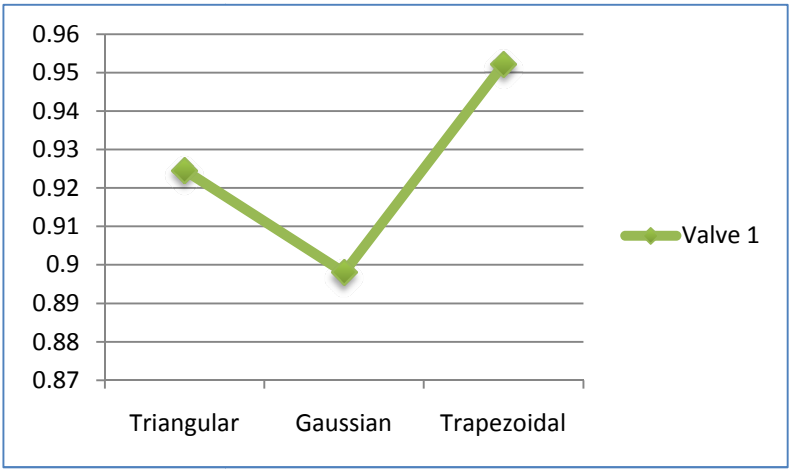


Fig. 17 Error of Valve 1 with 3 types of membership functions

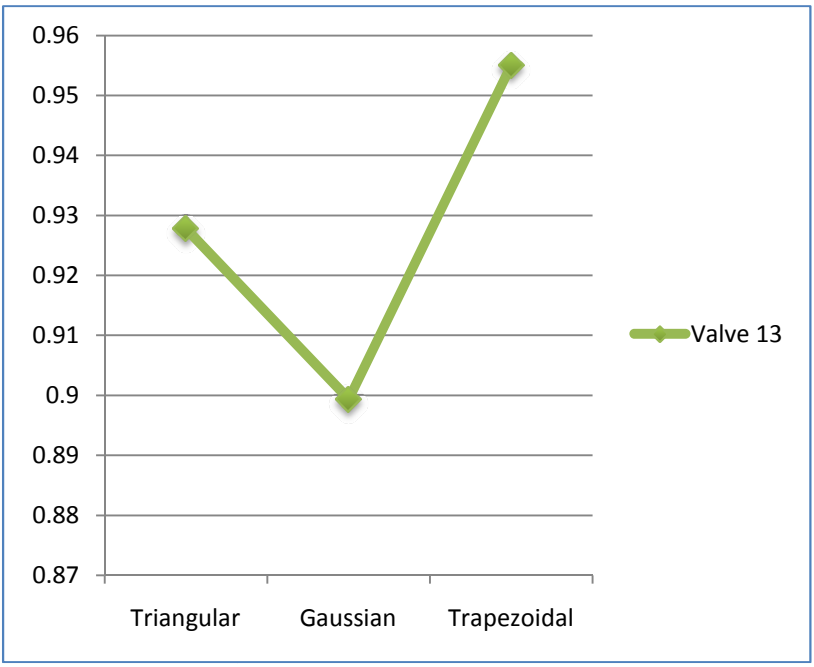


Fig. 18 Error of Valve 13 with 3 types of membership functions

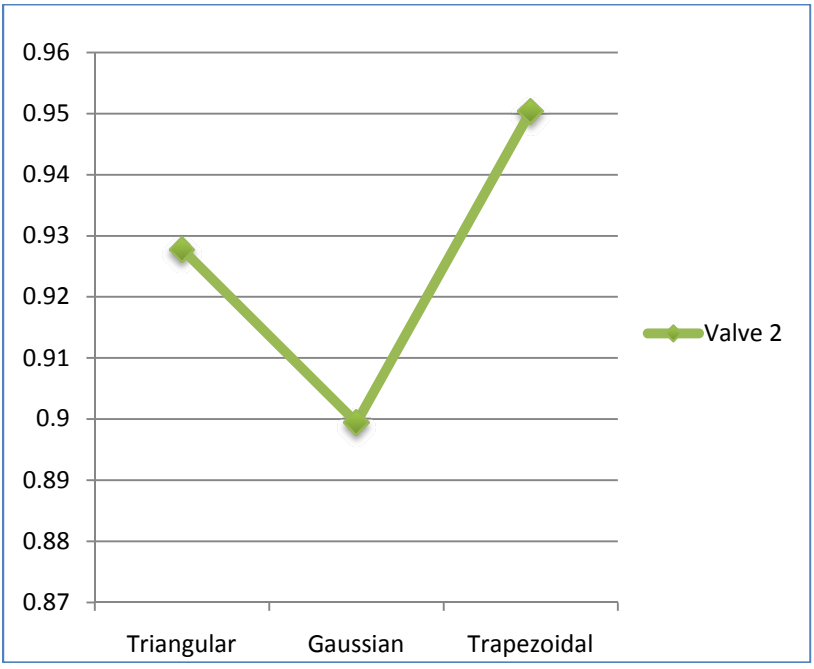


Fig. 19 Error of Valve 2 with 3 types of membership functions

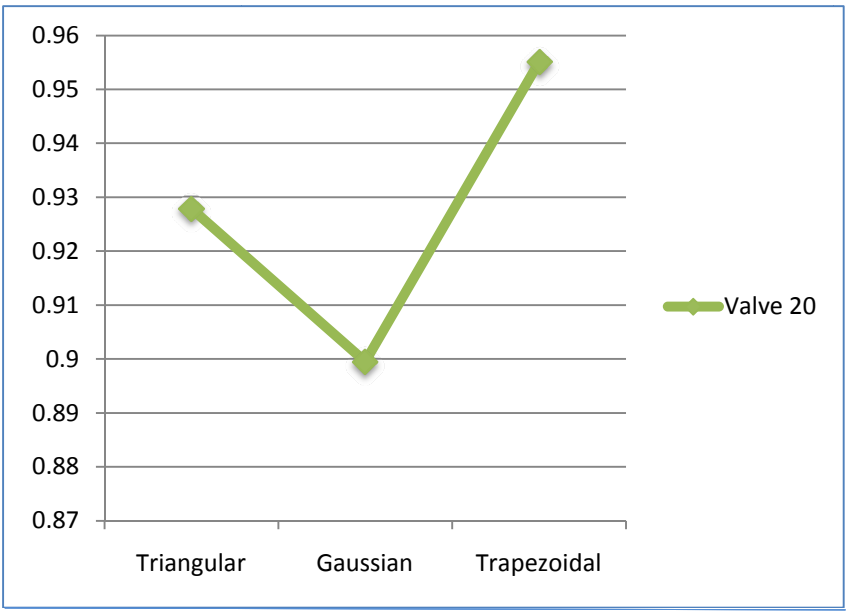


Fig. 20 Error of Valve 20 with 3 types of membership functions

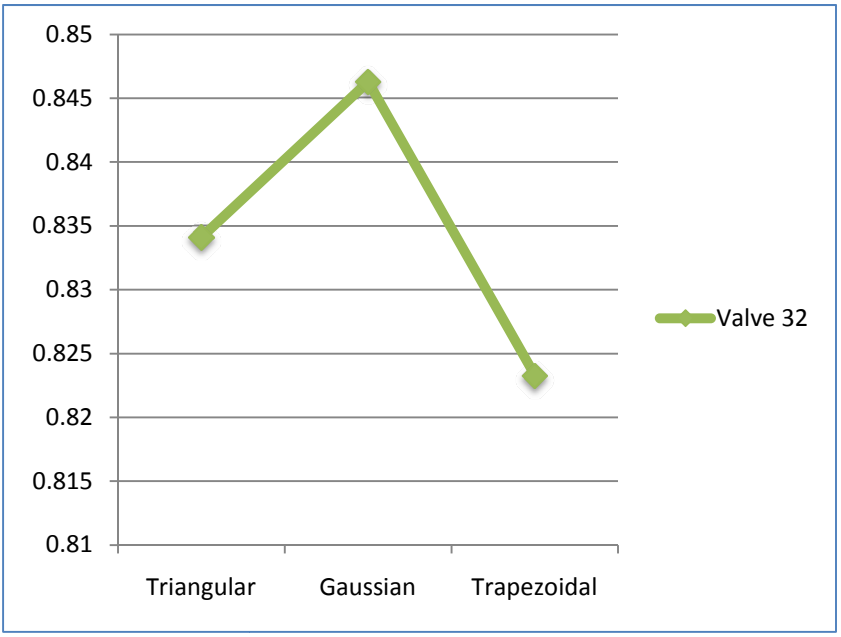


Fig. 21 Error of Valve 20 with 3 types of membership functions

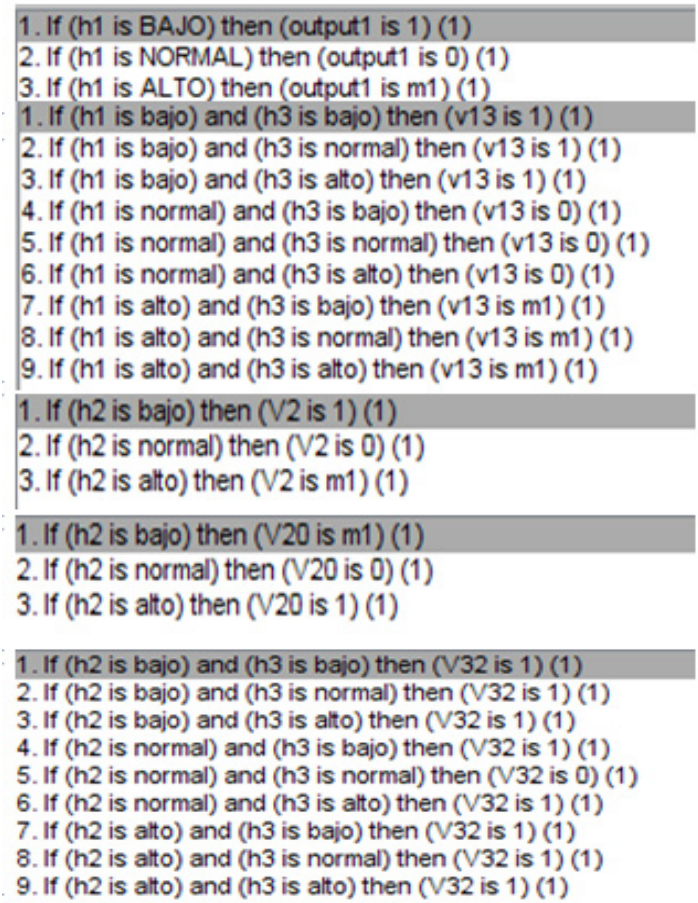


Fig. 22 Rules of the 5 type-1 fuzzy systems

The rules used to control in the case of the three tanks are shown in Figure 22.

The set of rules shown above rules are for the five fuzzy systems used to control the open and closed valves from the three tanks.

The first three rules are the controller number 1, the 9 following rules are controller 2, the third set of rules are the controller 3, the fourth set of 3 rules are the controller 4 and the last 9 rules are controller number 5.

The difference in the number of rules of each controller is because depending on the number of inputs, outputs and membership functions of fuzzy system will have a number of rules to be had. For example to control valve number one has only one input which is the tank 1, one output and has 3 membership functions therefore the number of rules are 3. The valves between 2 tanks need 2 inputs (tank1 and tank2 or tank 2 and tank3), these valves have one output and three membership functions therefore need 9 rules for fuzzy systems.

3.3 Genetic Algorithm

After obtaining the previous mentioned results, genetic algorithm optimization was performed. The genetic algorithm is used to optimize the membership functions of each fuzzy system (inputs and outputs).

In the genetic algorithm the membership functions of the 5 controllers were optimized.

In the algorithm the error of each controller is taken and finally the results of each controller were added, and the final result is divided between the number of controllers. The fitness function is shown in next equation:

$$f(y) = \frac{\left(\text{Error C1} = \sum_{i=1}^{n} \frac{\left| y^i_{REF_1} - y^i_{FS_1} \right|}{n} + \text{Error C2} = \sum_{i=1}^{n} \frac{\left| y^i_{REF_2} - y^i_{FS_2} \right|}{n} + \ldots \text{Error CN} = \sum_{i=1}^{n} \frac{\left| y^i_{REF_N} - y^i_{FS_N} \right|}{n} \right)}{N} \quad (2)$$

Where Y_{REF} is the reference, Y_{FS} is the output of the controller and **n** is the number of point used in comparison. Error C1 is the error of control 1 to N, and N in the number of the controllers.

The parameters used in the GA are shown in Figure 23.

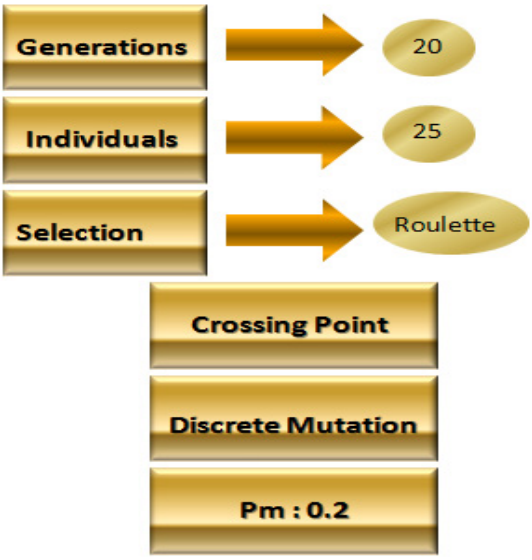


Fig. 23 Parameters of the genetic algorithm

After the use of the genetic algorithm the results obtained in the simulation are shown in Table 4.

Table 4 Results for the simulation plant using triangular membership functions and genetic algorithm

Error using triangular membership functions and genetic algorithm				
Valve 13	Valve 1	Valve 20	Valve 2	Valve 32
0.109	0.1146	0.0939	0.2077	0.218
0.131	0.1228	0.1329	0.1861	0
0.119	0.1275	0.111	0.239	0
0.115	0.1116	0.1092	0.2216	0
0.109	0.0908	0.1191	0.214	0
0.109	0.1132	0.0954	0.1922	0
0.117	0.1225	0.1003	0.1853	0
0.107	0.1102	0.1146	0.1938	0
0.105	0.0993	0.0851	0.2428	0
0.125	0.1196	0.113	0.1433	0
0.123	0.1191	0.1394	0.246	0
0.115	0.1114	0.091	0.1539	0
0.117	0.1231	0.101	0.1818	0
0.107	0.1444	0.0661	0.1366	0
0.117	0.1225	0.1003	0.1853	0

The above table shows a lower error in comparison with only using a type-1 fuzzy system. In the last table a genetic algorithm was used with triangular membership functions, the error is different in each valve even though the parameters are the same in all the tests. Some Graphics are shown in Figures 24 to 29 to present the behavior of each valve. In the last graphic the behavior of all valves is shown to observe all the behaviors.

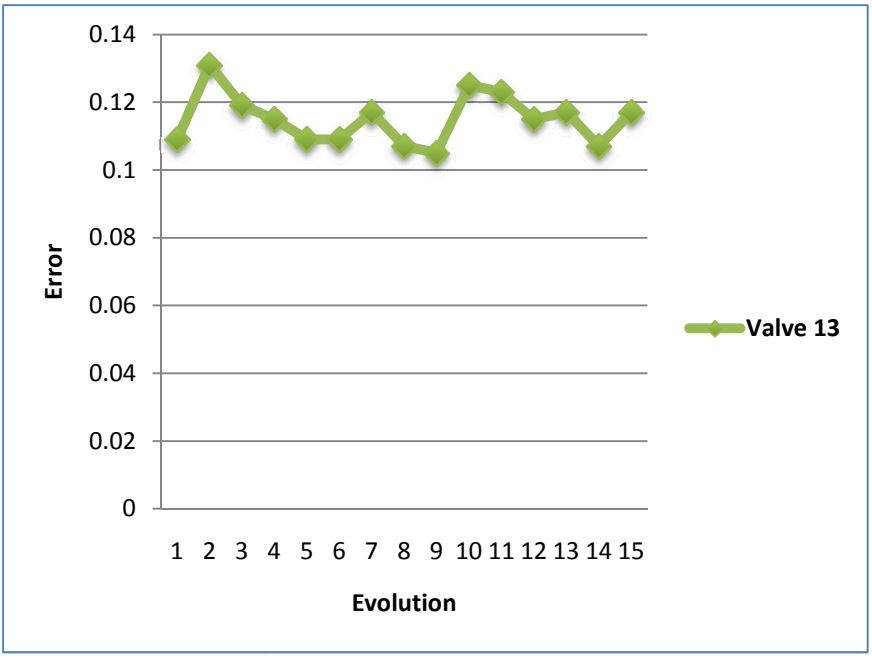


Fig. 24 Error of Valve 13 using a genetic algorithm

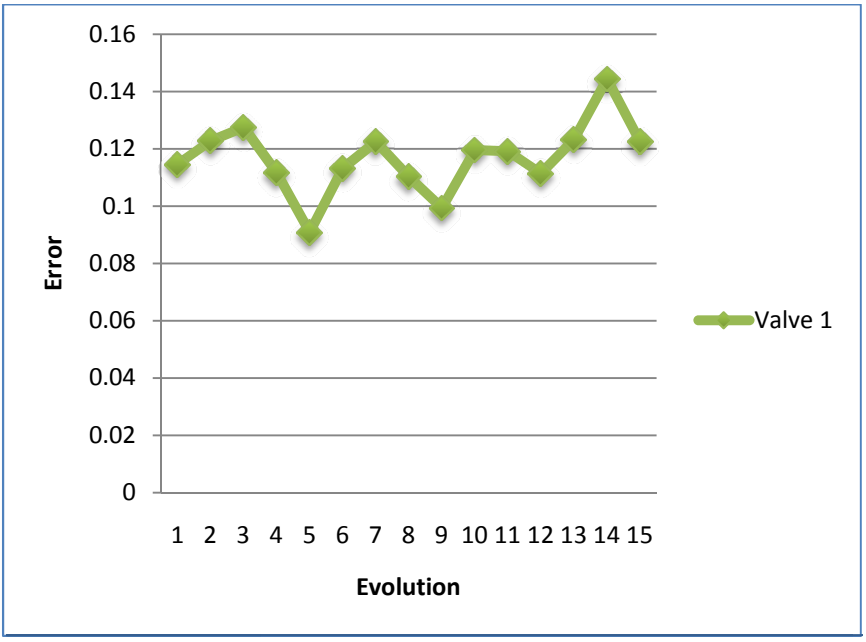


Fig. 25 Error of Valve 1 using a genetic algorithm

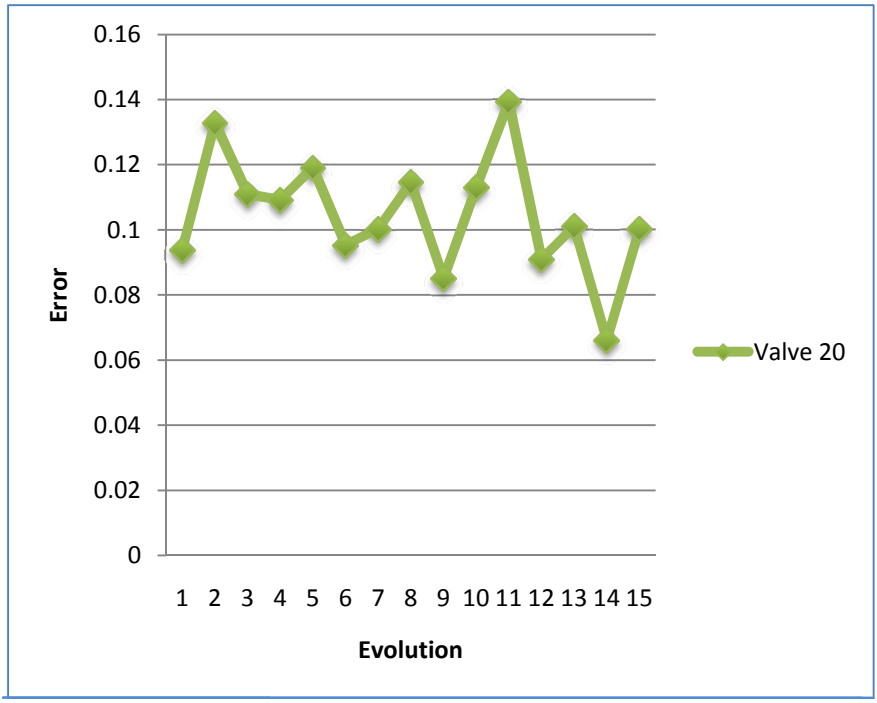


Fig. 26 Error of Valve 20 using a genetic algorithm

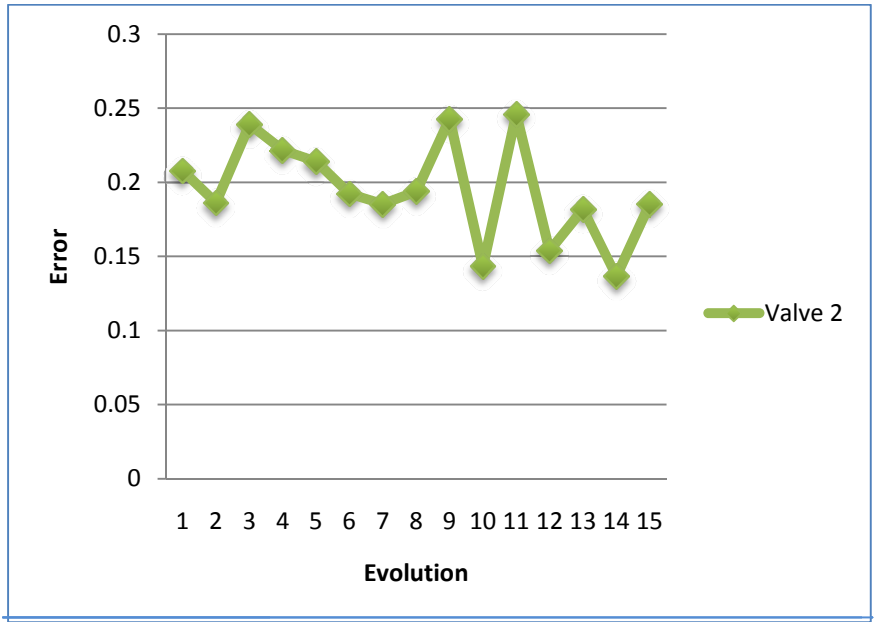


Fig. 27 Error of Valve 2 using a genetic algorithm

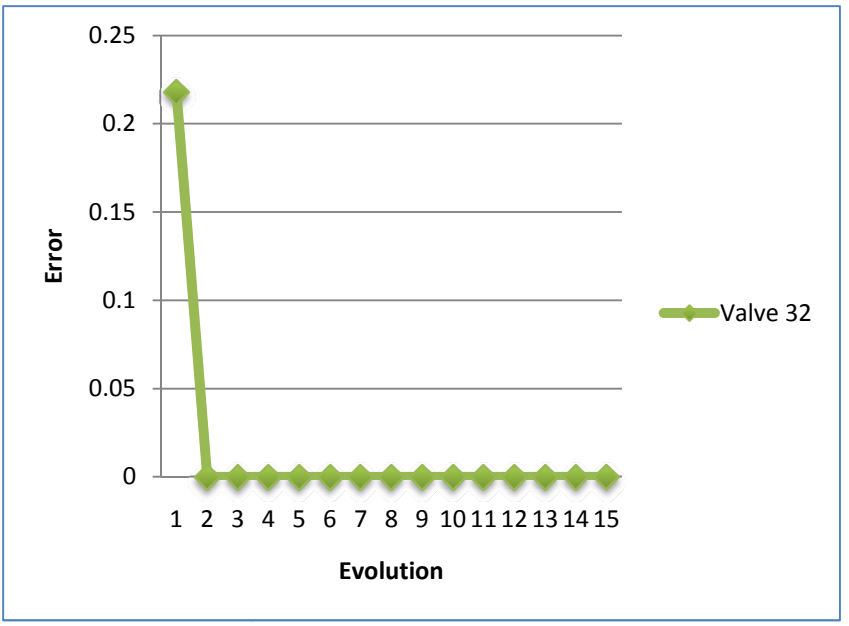


Fig. 28 Error of Valve 32 using a genetic algorithm

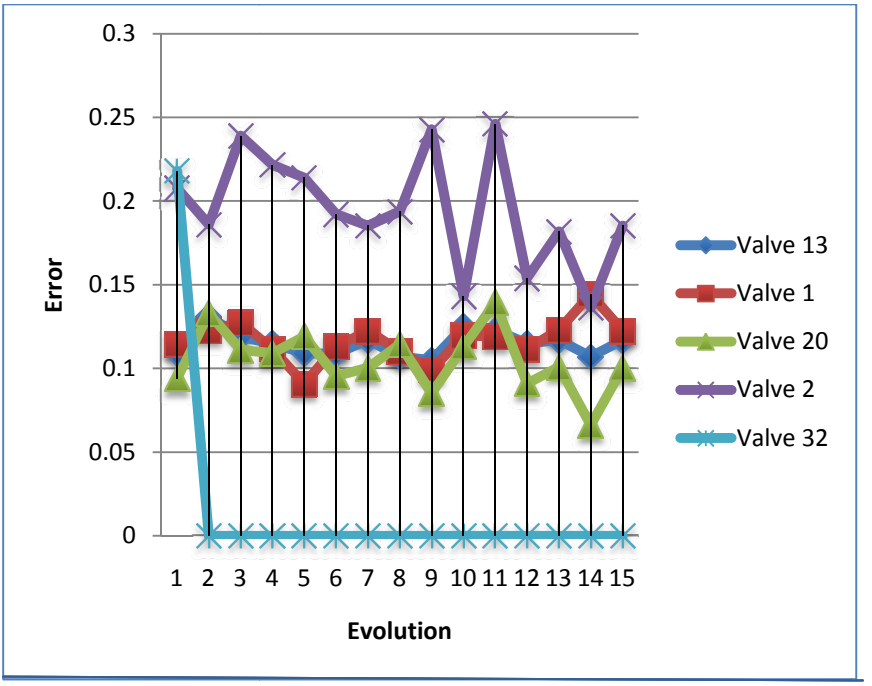


Fig. 29 Behavior of each valve using a genetic algorithm

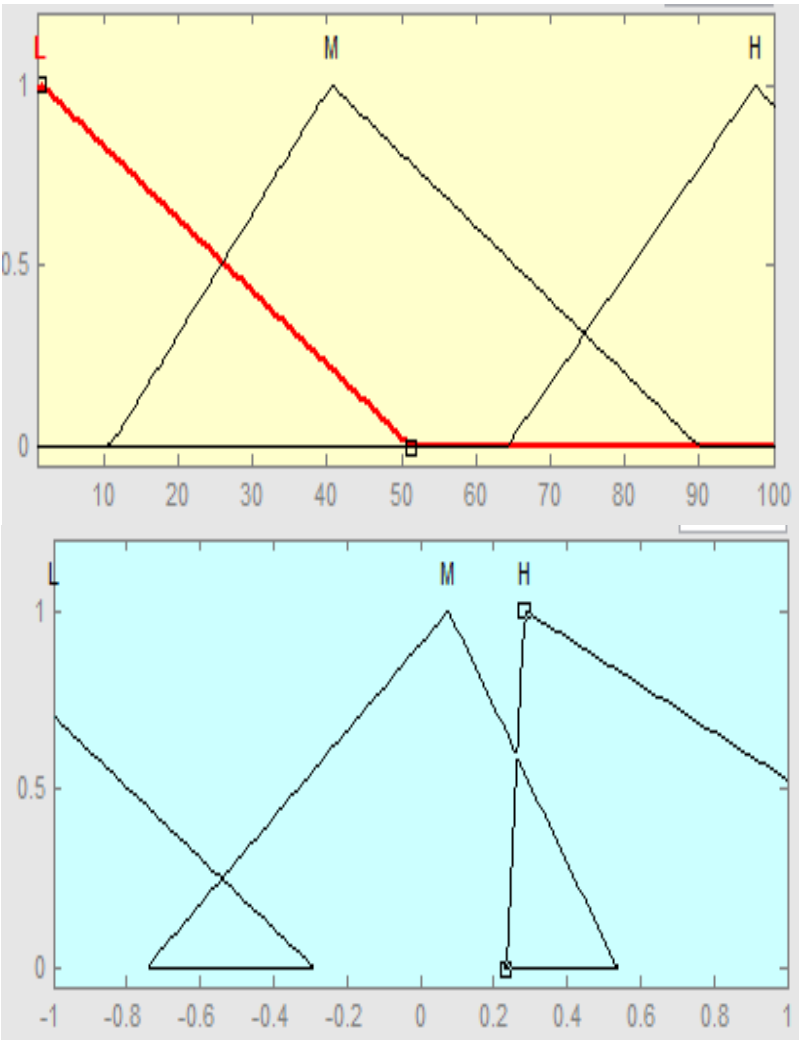


Fig. 30 Best Fuzzy system of valve 1 using genetic algorithm

Applying the genetic algorithm to a type-1 fuzzy system of each valve it was obtain the best fuzzy system of each valve as shown in Figures 30 to 34.

Last figure represents the best fuzzy system of valve 1 and its membership function of the input and the output. Yellow box is the input of the fuzzy system and the blue box is the output of the fuzzy system. In next fuzzy systems all the inputs of each are the yellow boxes and the outputs are the blue boxes.

All the fuzzy systems have 3 membership functions in the inputs and the outputs of each valve. When the genetic algorithm was implemented, more than 1 fuzzy systems were obtained, but in this case the best of the 15 evolutions is presented.

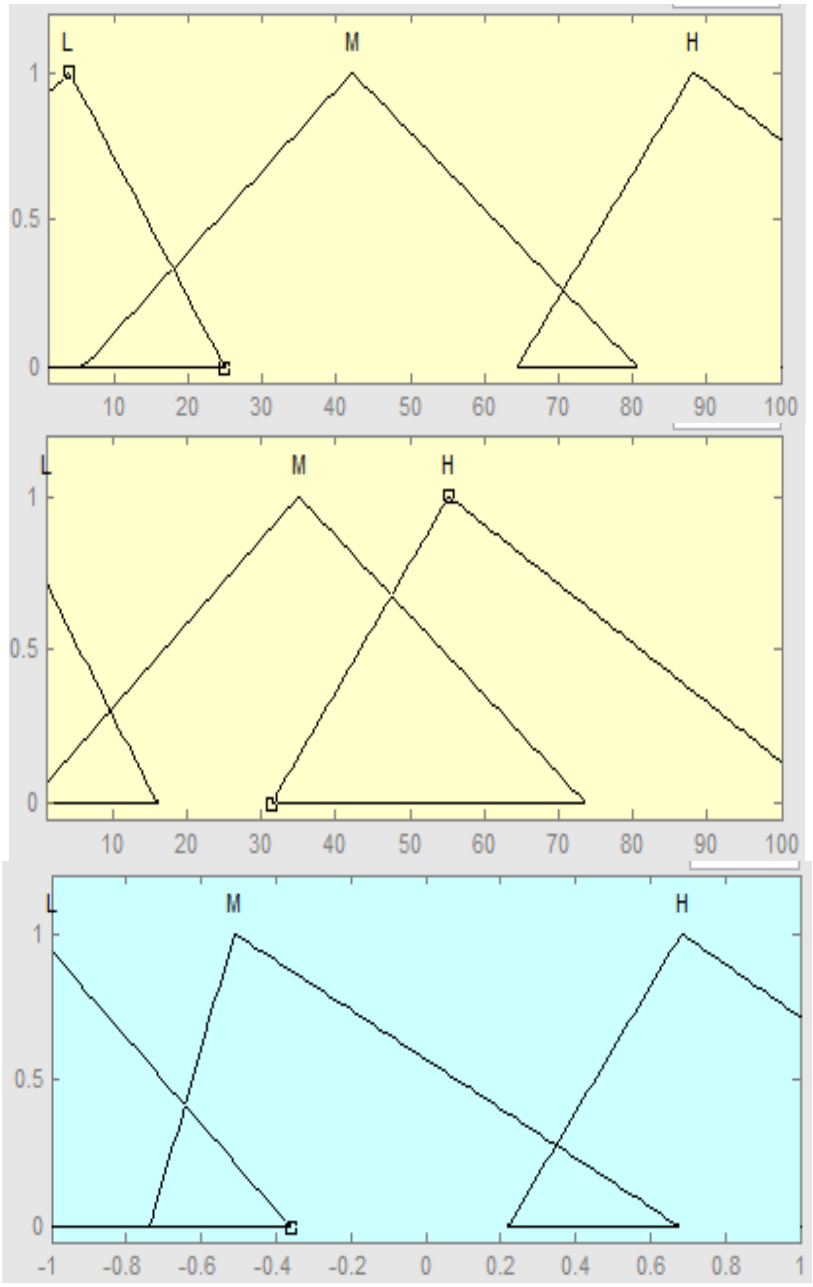


Fig. 31 Best Fuzzy system of valve 13 using genetic algorithm

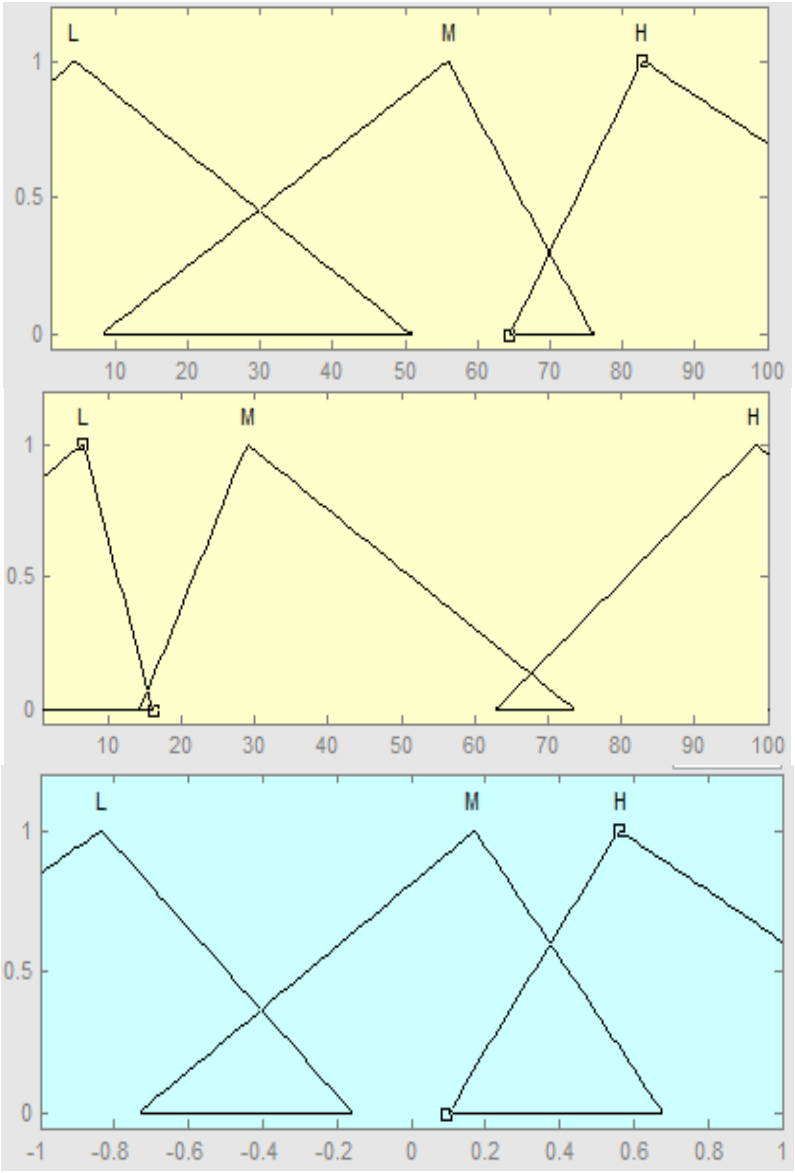


Fig. 32 Best Fuzzy system of valve 32 using genetic algorithm

Recall that this fuzzy system has two inputs because the valve 13 that is controlled is fed by two tanks (Tank 1 and Tank2).

This case is the same as that of the last fuzzy system, it needs two inputs to control the valve 32 because this valve is fed by two tanks (Tank 2 and Tank3). Valve 32 and valve 13 are the only ones needs two inputs, the reason is because as was explain those valves are between two tanks.

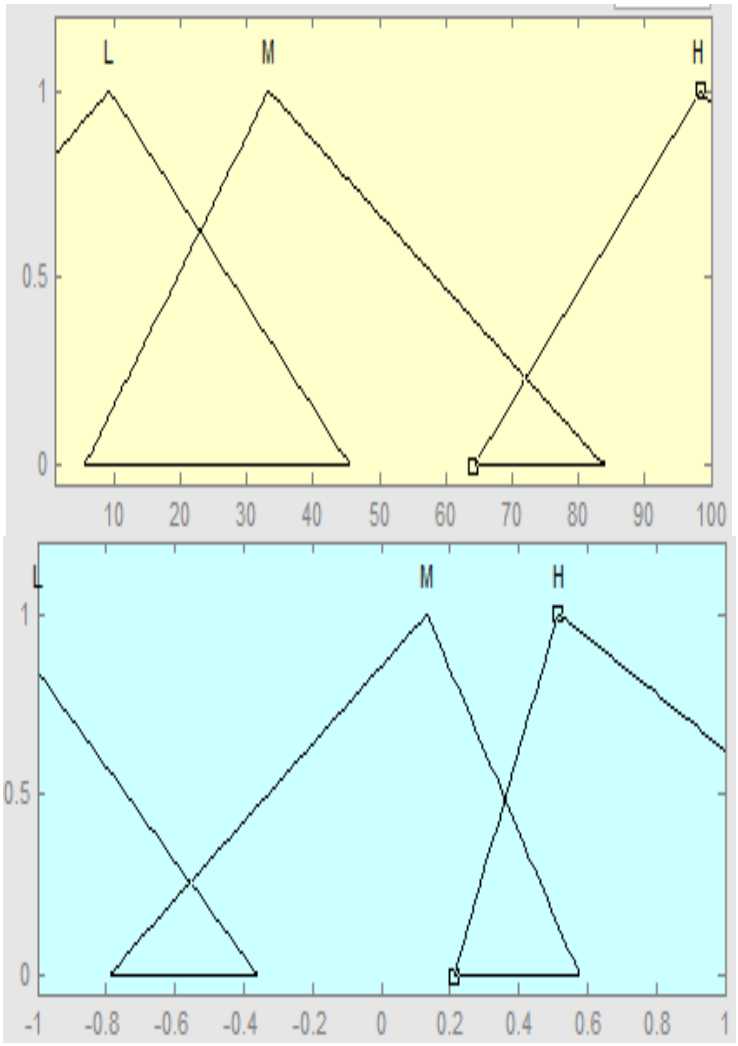


Fig. 33 Best Fuzzy system of valve 2 using genetic algorithm

Fuzzy systems have become a tool that can be useful to try and model the complex and nonlinear systems. And these fuzzy systems in this case study helps improve control valves. Membership functions can be varied to get more results. These fuzzy systems use three membership functions to establish the level of open or closed for the valves, the level of each membership function in the valves are open completely, half open and close.

The granulation of fuzzy systems may be increased and instead of using three membership functions it can be used 5 or another option, which could consider the valve as open medium, open, closed, half closed, fully closed. This depends on how you want to study the problem.

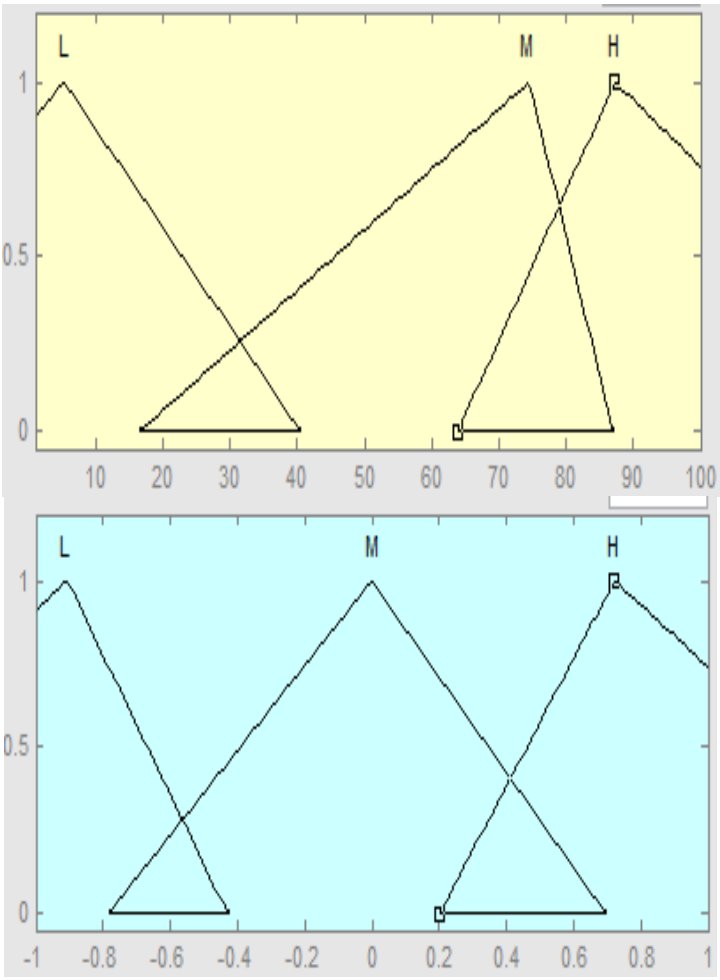


Fig. 34 Best Fuzzy system of valve 20 using genetic algorithm

4 Conclusions

A benchmark problem was used to test the proposed approach and based on the obtained results we can say that to achieve control of the present problem, a genetic algorithm is a good alternative to obtain a good fuzzy controller.

When a complex control problem is at hand, we start working on the case study, and once results are obtained with type-1 fuzzy systems it is a good choice to use a genetic algorithm for optimizing membership functions of the inputs and outputs of the controllers and to obtain better control, as was the case in this control problem. In the moment when genetic algorithm was used, results were better than with an initial type-1 fuzzy system, this is possible because in the moment

that genetic algorithm is applied, it moves the parameters of the membership functions and the system has more options to control de valves and the genetic algorithm is evaluated to obtain the best fuzzy system to control the open and close valves and this is why better results are obtained by optimizing the membership functions.

References

1. Castillo, O.: Type-2 Fuzzy Logic in Intelligent Control Applications. STUDFUZZ, vol. 272. Springer, Heidelberg (2012)
2. Castillo, O., Martinez-Marroquin, 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)
3. 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)
4. Castillo, O., Melin, P.: New fuzzy-fractal-genetic method for automated Mathematical Modelling and Simulation of Robotic Dynamic Systems. In: IEEE International Conference on Fuzzy Systems, vol. 2, pp. 1182–1187 (1998)
5. Castillo, O., Melin, P., Montiel, O., Sepúlveda, R.: Optimization of interval type-2 fuzzy logic controllers using evolutionary algorithms. Soft Comput. 15(6), 1145–1160 (2011)
6. Castillo, O., Kacprzyk, J., Pedrycz, W.: Soft Computing for Intelligent Control and Mobile Robotics. Springer (2011)
7. Cázarez, N., Aguilar, L., Castillo, O.: Fuzzy logic control with genetic membership function parameters optimization for the output regulation of a servomechanism with nonlinear backlash. Expert System Appl. 37(6), 4368–4378 (2010)
8. 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. SCI, vol. 318, pp. 213–224. Springer, Heidelberg (2010)
9. Cervantes, L., Castillo, O., Melin, P.: Intelligent Control of Nonlinear Dynamic Plants Using a Hierarchical Modular Approach and Type-2 Fuzzy Logic. In: Batyrshin, I., Sidorov, G. (eds.) MICAI 2011, Part II. LNCS, vol. 7095, pp. 1–12. Springer, Heidelberg (2011)
10. Coley, A.: An Introduction to Genetic Algorithms for Scientists and Engineers. World Scientific (1999)
11. Chen, G., Pham, T.: Introduction to Fuzzy Sets, Fuzzy Logic, and Fuzzy Control Systems (2001)
12. Dadios, E.: Fuzzy Logic-Controls, Concepts, Theories and Applications (2012)
13. Dubois, D., Prade, H.: Fuzzy sets and Systems: Theory and Applications (1980)
14. Gibbens, P., Boyle, D.: Introductory Flight Mechanics and Performance. University of Sydney, Australia (1999)
15. Haupt, R., Haupt, S.: Practical Genetic Algorithm. Wiley Interscience (2004)
16. Hidalgo, D., Melin, P., Castillo, O.: An optimization method for designing type-2 fuzzy inference systems based on the footprint of uncertainty using genetic algorithms. Expert Syst. Appl. 39(4), 4590–4598 (2012)

17. Martinez-Soto, R., Castillo, O., Aguilar, L.: 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)
18. Melin, P., Castillo, O.: A new method for adaptive model-based control of non-linear plants using type-2 fuzzy logic and neural networks. In: IEEE International Conference on Fuzzy Systems, vol. 1, pp. 420–425 (2003)
19. Mitchell, M.: An Introduction to Genetic Algorithms. Massachusetts Institute of Technology (1999)
20. Rachman, E., Jaam, J., Hasnah, A.: Non-linear simulation of controller for longitudinal control augmentation system of F-16 using numerical approach. Information Sciences Journal 164(1-4), 47–60 (2004)
21. Reiner, J., Balas, G., Garrard, W.: Flight control design using robust dynamic inversion and time- scale separation. Automatic Journal 32(11), 1493–1504 (1996)
22. Sanchez, E., Becerra, H., Velez, C.: Combining fuzzy, PID and regulation control for an autonomous mini-helicopter. Journal of Information Sciences 177(10), 1999–2022 (2007)
23. Sefer, K., Omer, C., Okyay, K.: Adaptive neuro-fuzzy inference system based autonomous flight control of unmanned air vehicles. Expert Systems with Applications Journal 37(2), 1229–1234 (2010)
24. Song, Y., Wang, H.: Design of Flight Control System for a Small Unmanned Tilt Rotor Aircraft. Chinese Journal of Aeronautics 22(3), 250–256 (2009)
25. Walker, D.J.: Multivariable control of the longitudinal and lateral dynamics of a fly by wire helicopter. Control Engineering Practice 11(7), 781–795 (2003)
26. Wu, D.: A Brief Tutorial on Interval Type-2 Fuzzy Sets and Systems (July 22, 2010)
27. Wu, D., Mendel, J.: On the Continuity of Type-1 and Interval Type-2 Fuzzy Logic Systems. IEEE T. Fuzzy Systems 19(1), 179–192 (2011)
28. Zadeh, L.: Fuzzy Sets and Fuzzy Information Granulation Theory. Beijing Normal University Press, Baijing (2000)
29. Zadeh, L.: Fuzzy Sets, Information and Control, vol. 8(3), pp. 338–353 (1965)
30. Zadeh, L.: Shadows of Fuzzy Sets. Probl. Peredachi Inf. 2(1), 37–44 (1966)
31. Zadeh, L.: Fuzzy Logic. Neural Networks and Soft Computing Commun. ACM 37(3), 77–84 (1994)
32. Zadeh, L.A.: Some reflections on soft computing, granular computing and their roles in the conception, design and utilization of information/intelligent systems. Soft Computting 2, 23–25 (1998)
33. 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)

Part III

Soft Computing in Pattern Recognition Applications

Multi-Objective Hierarchical Genetic Algorithm for Modular Granular Neural Network Optimization

Daniela Sánchez and Patricia Melin

Tijuana Institute of Technology,
Tijuana, Mexico
pmelin@tectijuana.mx

Abstract. In this paper we propose a multi-objective hierarchical genetic algorithm (MOHGA) for modular neural network optimization. A granular approach is used due to the fact that the dataset is divided into granules or sub modules. The main objective of this method is to know the optimal number of sub modules or granules, but also allow the optimization of the number of hidden layers, number of neurons per hidden layer, error goal and learning algorithms per module. The proposed MOHGA is based on the Micro genetic algorithm and was tested for a pattern recognition application. Simulation results show that the proposed modular neural network approach offers advantages over existing neural network models. Finally the modular neural networks are joined using type-2 fuzzy integration, which allows having a system with a better behavior and results.

1 Introduction

Hybrid intelligent systems are computational systems that integrate different intelligent techniques. Examples of these techniques are modular neural networks (MNN) and genetic algorithms (GA). Hybrid intelligent systems are now being used to support complex problem solving and decision making in a wide variety of tasks. Hybrid intelligent systems allow the representation and manipulation of different types and forms of data and knowledge, which may come from various sources. In this paper these techniques are combined using a granular approach. It was decided to apply the proposed method to pattern recognition to test the approach with complex problems.

Biometrics plays an important role in public security and information security domains. Using various physiological characteristics of the human, such as face, facial thermo grams, fingerprint, iris, retina, hand geometry etc., biometrics accurately identifies each individual and distinguishes one from another [1].

The recognition of people is of great importance, since it allows us to have a greater control about when a person has access to certain information, area, or simply to identify if the person is the one who claims to be.

The achieved results indicate that biometric techniques are much more precise and accurate than the traditional techniques. Other than precision, there have

P. Melin, O. Castillo (Eds.): *Soft Computing Appli. in Optimization,* STUDFUZZ 294, pp. 157–185.
DOI: 10.1007/978-3-642-35323-9_7

always been certain problems which remain associated with the existing traditional techniques. As an example consider possession and knowledge. Both can be shared, stolen, for-gotten, duplicated, misplaced or taken away. However the danger is minimized in case of biometric means [37].

There are many works that combine different techniques and they have demonstrated that the integration of different intelligent techniques provide good results, such as in [19][29][30][31][32][34][35][42].

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

2 Basic Concepts

In this section we present a brief overview of the basic concepts used in this research work.

2.1 Modular Neural Networks

An artificial neuron is a computational model inspired in the natural neurons. Natural neurons receive signals through synapses located on the dendrites or membrane of the neuron. When the signals received are strong enough (surpass a certain threshold), the neuron is activated and emits a signal though the axon. This signal might be sent to another synapse, and might activate other neurons. The complexity of real neurons is highly abstracted when modeling artificial neurons. These basically consist of inputs (like synapses), which are multiplied by weights (strength of the respective signals), and then computed by a mathematical function which determines the activation of the neuron. Another function (which may be the identity) computes the output of the artificial neuron (sometimes in dependence of a certain threshold). ANNs combine artificial neurons in order to process information [5].

Neural networks (NNs) can be used to extract patterns and detect trends that are too complex to be noticed by either humans or other computer techniques [24]. The modular neural networks (MNNs) are comprised of modules. The idea on which this kind of learning structure is based on the divide-and-conquer paradigm: the problem should be divided into smaller sub problems that are solved by experts (modules) and their partial solutions should be integrated to produce a final solution [4][26][43]. A module can be a sub-structure or a learning sub procedure of the whole network [3].

The results of the different applications involving Modular Neural Networks (MNNs) lead to the general evidence that the use of modular neural networks implies a significant learning improvement comparatively to a single NN and especially to the backpropagation NN. Each neural network works independently in its own domain. Each of the neural networks is build and trained for a specific task [28].

2.2 Type-2 Fuzzy Logic

Fuzzy logic is a useful tool for modeling complex systems and deriving useful fuzzy relations or rules [39]. However, it is often difficult for human experts to define the fuzzy sets and fuzzy rules used by these systems [47]. 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 [7] [25] [56].

The concept of a type-2 fuzzy set, was introduced by Zadeh (1975) as an extension of the concept of an ordinary fuzzy set (henceforth called a "type-1 fuzzy set"). A type-2 fuzzy set is characterized by a fuzzy membership function, i.e., the membership grade for each element of this set is a fuzzy set in [0,1], unlike a type-1 set where the membership grade is a crisp number in [0,1]. Such sets can be used in situations where there is uncertainty about the membership grades themselves, e.g., an uncertainty in the shape of the membership function or in some of its parameters. Consider the transition from ordinary sets to fuzzy sets. When we cannot determine the membership of an element in a set as 0 or 1, we use fuzzy sets of type-1. Similarly, when the situation is so fuzzy that we have trouble determining the membership grade even as a crisp number in [0,1], we use fuzzy sets of type-2 [20][21][22][42].

Uncertainty in the primary memberships of a type-2 fuzzy set, $\tilde{A}$, consists of a bounded region that we call the "footprint of uncertainty" (FOU). Mathematically, it is the union of all primary membership functions [9][10][33].

A type-2 fuzzy set $\tilde{A}$, is characterized by the membership function (see expression 1):

$$\tilde{A}= \{((x,u), \mu_{\tilde{A}}(x,u)) \mid \forall x \in X, \forall u \in J_x \subseteq [0,1] \} \quad (1)$$

where x means the input variable, u means a type-1 membership function, J_x means an interval $\subseteq [0,1]$, and $\mu_{\tilde{A}}$ means a type-2 membership function. Another expression (2) for A is,

$$\tilde{A}= \int_{x\in X}\int_{u\in J_X} \mu_{\tilde{A}}(x,u)/(x,u) \qquad J_x \subseteq [0,1] \quad (2)$$

The distinction between type-1 and type-2 is associated with the nature of the membership functions, which is not important when forming the rules. The structure of the rules remains exactly the same in the type-2 case, but now some or all of the sets involved are type-2.

Consider a type-2 FLS having r inputs $x_1 \in X_1, \ldots, x_r \in X_r$ and one output $y \in Y$. As in the type-1 case, we can assume that there are M rules; but, in the type-2 case the lth rule has the form

$$R^1 : \text{IF } x_1 \text{ is } \tilde{A}^1_1 \text{ and } \ldots x_p \text{ is } \tilde{A}^1_p \text{, THEN y is } Y^1 \qquad 1=1,\ldots,M$$

This rule represents a type-2 fuzzy relation between the input space $X_1 \times \ldots \times X_r$, and the output space, Y, of the type-2 fuzzy system.

If we considered two fuzzy sets (type-2) named $\tilde{A}_1$ and $\tilde{A}_2$ their union is another type-2 fuzzy set, just as the union of type-1 fuzzy sets A_1 and A_2 is another type-1 fuzzy set. More formally, we have the following expression (3)

$$\tilde{A}1 \cup \tilde{A}2 = \int_{x \in X} \mu_{\tilde{A}1 \cup \tilde{A}2}(x)/x \tag{3}$$

The intersection of $\tilde{A}_1$ and $\tilde{A}_2$ is another type-2 fuzzy set, just as the intersection of type-1 fuzzy sets A_1 and A_2 is another type-1 fuzzy set. More formally, we have the following expression (4)

$$\tilde{A}1 \cap \tilde{A}2 = \int_{x \in X} \mu_{\tilde{A}1 \cap \tilde{A}2}(x)/x \tag{4}$$

The complement of set $\tilde{A}$ is another type-2 fuzzy set, just as the complement of type-1fuzzy set A is another type-1 fuzzy set. More formally we have the following expression (5)

$$\tilde{A}' = \int_{x} \mu_{\tilde{A}'1}(x)/x \tag{5}$$

The basics of fuzzy logic do not change from type-1 to type-2 fuzzy sets, and in general will not change for type-n. A higher type number just indicates a higher degree of fuzziness [8].

2.3 Multi-Objective Hierarchical Genetic Algorithm

A Genetic algorithm (GA) is an optimization and search technique based on the principles of genetics and natural selection [18][36][44]. GAs are nondeterministic methods that employ crossover and mutation operators for deriving offspring. GAs work by maintaining a constant-sized population of candidate solutions known as individuals (chromosomes) [13][24][38].

Introduced in [45], a Hierarchical genetic algorithm (HGA) is a type of genetic algorithm. Its structure is more flexible than the conventional GA. The basic idea under hierarchical genetic algorithm is that for some complex systems, which cannot be easily represented, this type of GA can be a better choice. The complicated chromosomes may provide a good new way to solve the problem [46][48].

Multi-objective optimization (MO) seeks to optimize the components of a vector-valued cost function. Unlike single objective optimization, the solution to this problem is not a single point, but a family of points known as the Pareto-optimal set. Each point in this surface is optimal in the sense that no improvement can be achieved in one cost vector component that does not lead to degradation in at least one of the remaining components [15].

There are three general approaches to multi-objective optimization. The first is to combine the individual objective functions into a single composite function (Aggregating functions). The second is to use Population-based approaches and the third is to use Pareto-based approaches. A Pareto optimal set is a set of solutions that are non-dominated with respect to each other. Pareto optimal sets can be of varied sizes, but the size of the Pareto set increases with the increase in the number of objectives [2].

In this work the multi-objective genetic algorithm is based on a Micro genetic algorithm, proposed in [11][12]. Two main characteristics of this kind of genetic algorithm are that it works with a small population and has a re initialization process.

2.4 Granular Computing

Granular computing (GrC), as a general computing paradigm of problem solving, has been received much attentions recently, although its basic ideas and principles have been studied extensively in various research communities and application domains for a long time in explicit or implicit fashions. Zadeh [57] first introduced the notion of information granulation in 1979 and suggested that fuzzy set theory may find potential applications in this respect, which pioneers the explicit study of granular computing. With the concept of his information granulation, Zadeh further presented granular mathematics [58]. Pawlak proposed the rough set theory to deal with inexact information by using rough sets to approximate a crisp set in 1982 [41], and investigated the granularity of knowledge from the point of view of rough set theory [40]. Hobbes [23] presented a theory of granularity as the base of knowledge representation, abstraction, heuristic search, and reasoning in 1985. In his theory the problem world is represented as various grains and only interesting ones are abstracted to learn concepts.

The conceptualization of the world can be performed at different granularities and switched between granularities. In 1992, Giunchigalia and Walsh presented a theory of abstraction to improve the conceptualization of granularities [16]. Lin suggested the term “granular computing” to label this growing research field in 1997 [27]. Yao investigated the trinity model of granular computing from three perspectives: philosophy, methodology, and computation [54][17].

Granular computing is often defined as an umbrella term to cover many theories, methodologies, techniques, and tools that make use of granules in complex problem solving. Granular computing is a new term for the problem solving paradigm and may be viewed more on a philosophical rather than technical level [49][50][51][52].

Granular computing has begun to play important roles in bioinformatics, e-Business, security, machine learning, data mining, high-performance computing and wireless mobile computing in terms of efficiency, effectiveness, robustness and uncertainty [6][54][55].

A granule may be interpreted as one of the numerous small particles forming a larger unit. The philosophy of thinking in terms of levels of granularity, and its implementation in more concrete models, would result in disciplined procedures that help to avoid errors and to save time for solving a wide range of complex problems. At least three basic properties of granules are needed: internal properties reflecting the interaction of elements inside a granule, external properties revealing its interaction with other granules and, contextual properties showing the relative existence of a granule in a particular environment [53].

3 General Architecture of the Proposed Method

The proposed method combines modular neural networks (MNN) and fuzzy logic as response integrators. In particular, it can be used for pattern recognition. This proposed method is able to use some data sets, for example to use "N" biometric measures to identify someone and the data of each biometric measure would be divided into different numbers of sub modules. The general architecture of the proposed method is shown in Figure 1. For joining the different responses of each biometric measure fuzzy integration is used. The proposed method also performs the optimization of the modular neural networks (as number of layers, goal error, number of neurons, etc.) and the different parameters of the fuzzy integrator.

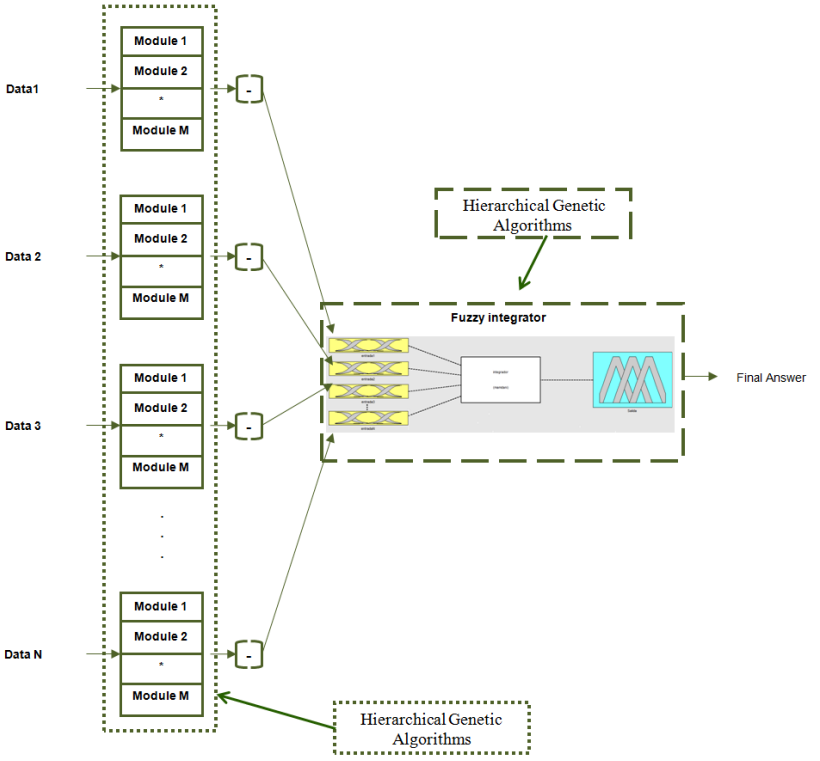


Fig. 1 The general architecture of the proposed method

3.1 General Architecture of the Proposed Method for the Modular Neural Network

The proposed method for MNN consists in changing the number of modules and the data per module, for example in the case of human recognition, it means that there will be different number of persons in each sub module. The number of sub

modules can be established by a genetic algorithm, but at this moment the number is established randomly. The architecture of the proposed method for the modular neural network is shown in Figure 2.

This method also chooses randomly which images will be used for training, but first the percentage of images for training is established (at this moment that percentage is defined randomly).

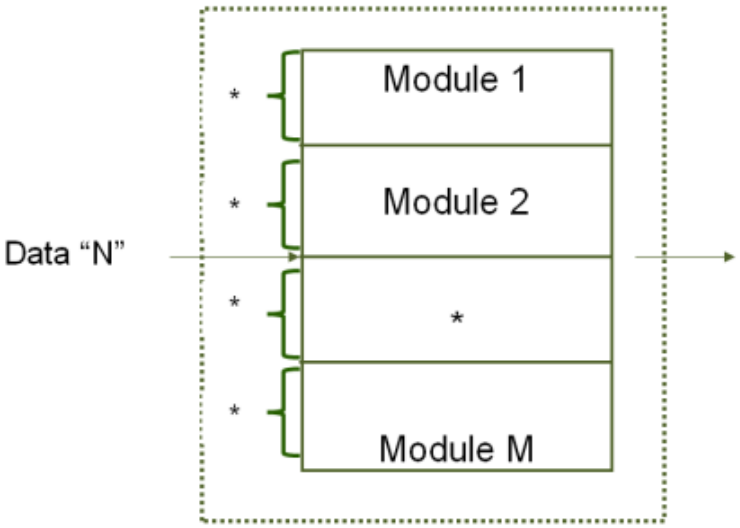


Fig. 2 The architecture of proposed method for the modular neural network

3.2 Description of the Multi-Objective Hierarchical Genetic Algorithm for MNN Optimization

With the purpose of knowing the optimal number of modules and the percentage of data for training, it is proposed the use of a genetic algorithm that allows the optimization of these parameters and others as the number of hidden layers, number of neurons per hidden layer, error goal and learning algorithms per module.

Figure 3 shows the chromosome, which was proposed for optimization of the neural networks.

The way in which the multi-objective hierarchical genetic algorithm works is illustrated in Figure 4 and described in more detail below.

First, a random population is generated. This random population is divided in two parts: a non-replaceable and replaceable portion. The non-replaceable portion never changes during the evolution, this helps to provide diversity. The replaceable portion experiences changes after certain condition is satisfied, this condition is called nominal convergence.

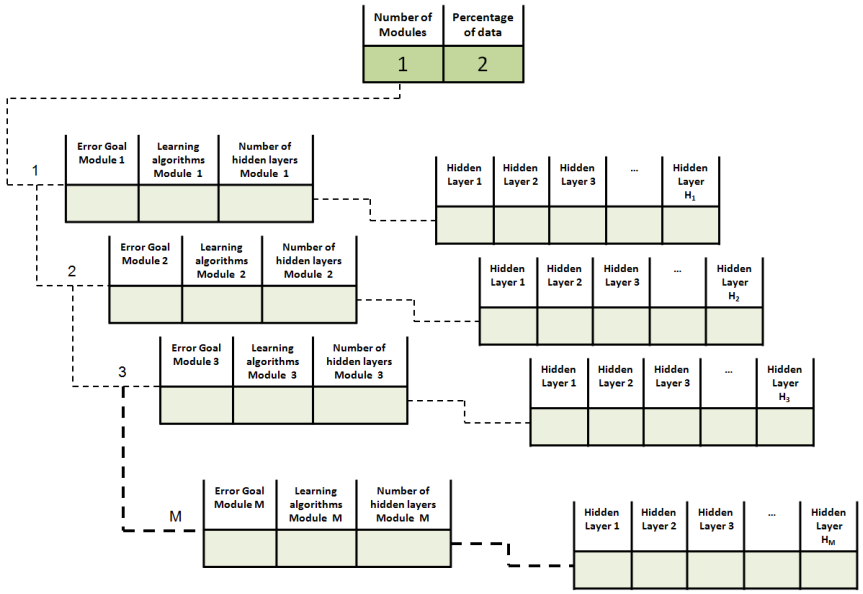


Fig. 3 The chromosome of the multi-objective hierarchical genetic algorithm for the MNN

The working population at the beginning is taken (with a certain probability) from both portions of the main population. During each cycle, the MOHGA uses conventional genetic operators.

The external memory is initially empty, in each cycle the non-dominated vectors found are saved in that memory, logically a comparison is performed between the new vectors found and vectors already stored.

The MOHGA has two kinds of convergence. The first is the usually used (for example when it has the maximum number of cycle or generations, or when the value desired of one objective function is obtained). The second is called Nominal Convergence, in this case is established each 5 generations, here two non dominated vectors are taken of the external memory and these are compared with two vectors of the Replaceable portion, if the two vectors taken of the replaceable portion are dominated by the others, those vector are replaceable for the two vectors of the external memory, then the working population is reinitialized.

3.3 Objective Functions

In order to not only get the network that provides us with the lowest error of recognition another objective function is set, and so not only obtain the best network with the lowest error of recognition , but also obtain a modular neural network that uses the lowest percentage of data for the training phase. The objective functions are defined below:

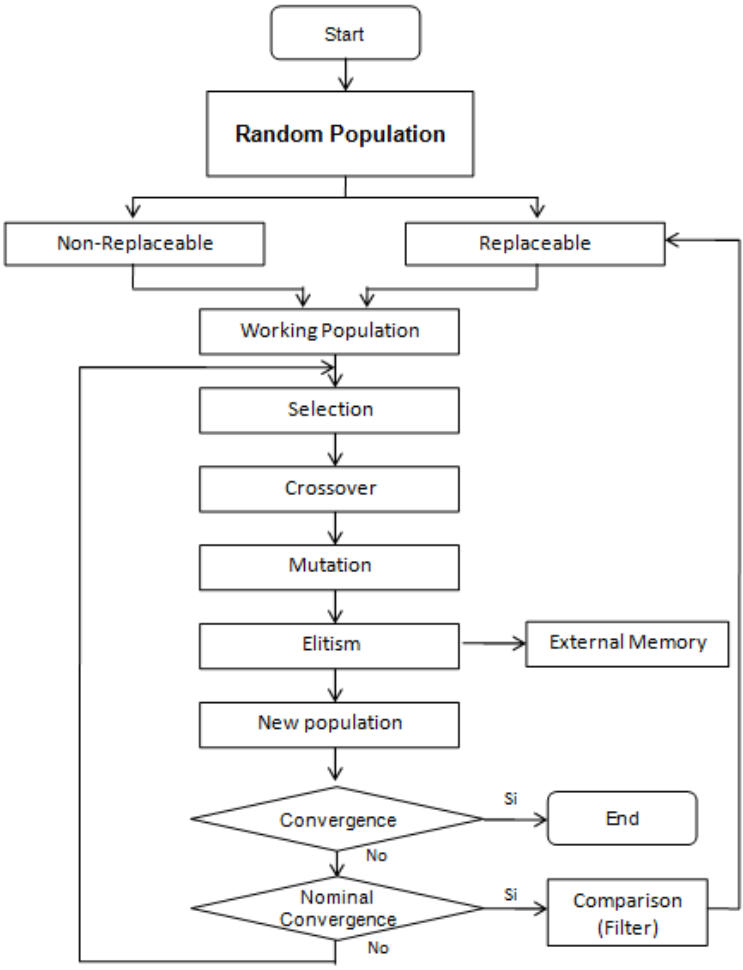


Fig. 4 Diagram that illustrates the way in which the multi-objective hierarchical genetic algorithm works

$$\text{Min } f_1 = \sum_{i=1}^{m}\left(\left(\sum_{j=1}^{n_m} X_j\right)/n_m\right) \tag{2}$$

$$\text{Min } f_2 = \textbf{percentage of data} \tag{3}$$

3.4 Type-2 Fuzzy Integration

The proposed method uses type-2 fuzzy logic for combining the response of each modular neural network. Four non-optimized fuzzy integrator were used to

perform tests in Figure 5 an example is shown. In the method, the number of inputs of each fuzzy integrator depends on how many modular neural networks will be needed, in this work 2 inputs are needed, because this method is tested for human recognition based on ear and voice biometrics.

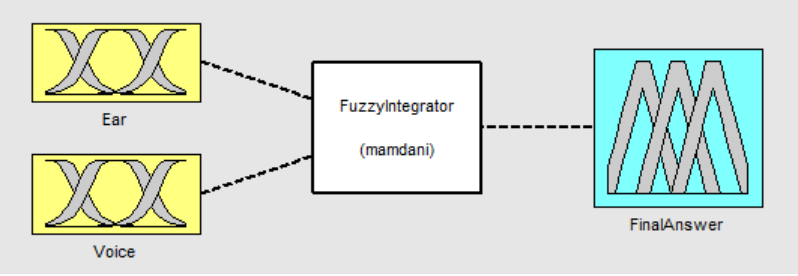


Fig. 5 Example of fuzzy integrator

In Figure 6, the fuzzy integrator #1 is presented, this fuzzy integrator uses trapezoidal membership functions, in this case, 3 membership functions in each input and output are used. The rules for this fuzzy integrator are shown in Figure 8.

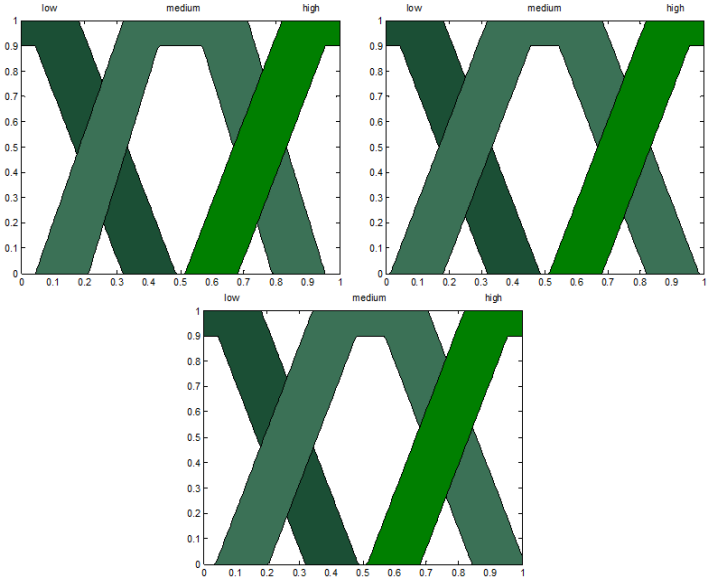


Fig. 6 Fuzzy Integrator #1

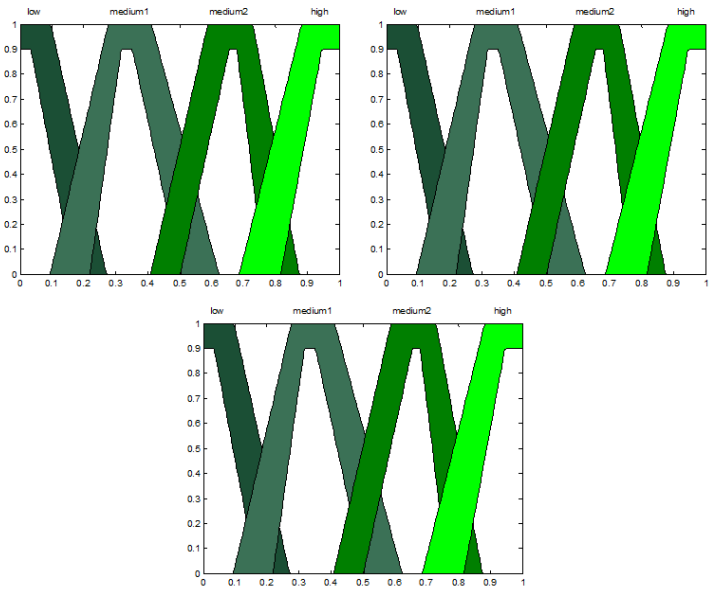


Fig. 7 Fuzzy Integrator #2

```
1. If (input1 is low) and (input2 is low) then (output1 is low) (1)
2. If (input1 is low) and (input2 is medium) then (output1 is medium) (1)
3. If (input1 is low) and (input2 is high) then (output1 is high) (1)
4. If (input1 is medium) and (input2 is low) then (output1 is low) (1)
5. If (input1 is medium) and (input2 is medium) then (output1 is medium) (1)
6. If (input1 is medium) and (input2 is high) then (output1 is high) (1)
7. If (input1 is high) and (input2 is low) then (output1 is low) (1)
8. If (input1 is high) and (input2 is medium) then (output1 is medium) (1)
9. If (input1 is high) and (input2 is high) then (output1 is high) (1)
```

Fig. 8 Rules of the fuzzy Integrator #1

```
1. If (input1 is low) and (input2 is low) then (output1 is low) (1)
2. If (input1 is low) and (input2 is medium1) then (output1 is medium1) (1)
3. If (input1 is low) and (input2 is medium2) then (output1 is medium2) (1)
4. If (input1 is low) and (input2 is high) then (output1 is high) (1)
5. If (input1 is medium1) and (input2 is low) then (output1 is low) (1)
6. If (input1 is medium1) and (input2 is medium1) then (output1 is medium1) (1)
7. If (input1 is medium1) and (input2 is medium2) then (output1 is medium2) (1)
8. If (input1 is medium1) and (input2 is high) then (output1 is high) (1)
9. If (input1 is medium2) and (input2 is low) then (output1 is low) (1)
10. If (input1 is medium2) and (input2 is medium1) then (output1 is medium1) (1)
11. If (input1 is medium2) and (input2 is medium2) then (output1 is medium2) (1)
12. If (input1 is medium2) and (input2 is high) then (output1 is high) (1)
13. If (input1 is high) and (input2 is low) then (output1 is low) (1)
14. If (input1 is high) and (input2 is medium1) then (output1 is medium1) (1)
15. If (input1 is high) and (input2 is medium2) then (output1 is medium2) (1)
16. If (input1 is high) and (input2 is high) then (output1 is high) (1)
```

Fig. 9 Rules of the fuzzy Integrator #2 to #4

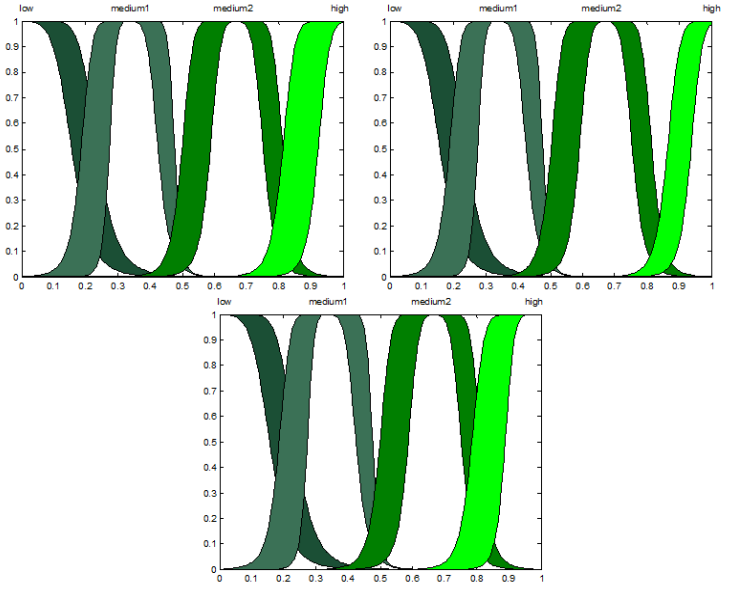


Fig. 10 Fuzzy Integrator #3

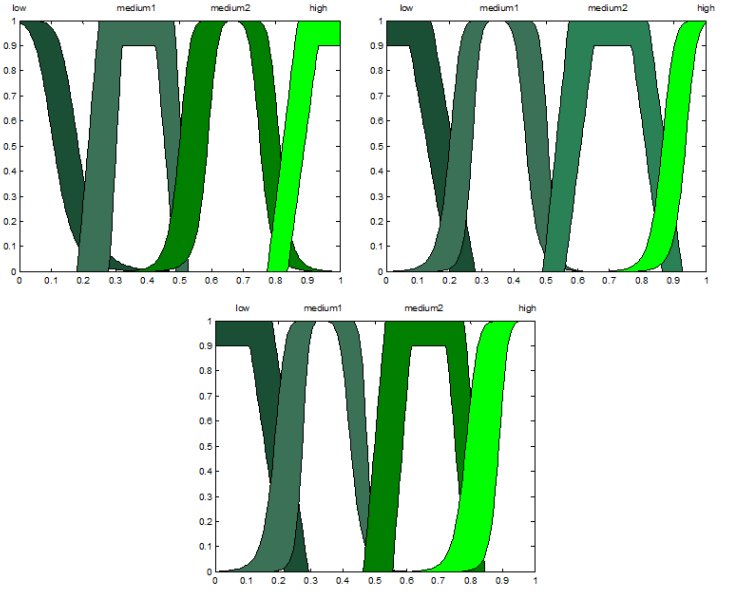


Fig. 11 Fuzzy Integrator #4

In Figure 7, the fuzzy integrator #2 is presented, this fuzzy integrator uses trapezoidal membership functions, in this case 4 membership functions in each input and output are used. The rules for the fuzzy integrator #2 to #4 are shown in Figure 9.

In Figure 10, the fuzzy integrator #3 is presented, this fuzzy integrator uses *gBell* membership functions, in this case 4 membership functions in each input and output are used.

In Figure 11, the fuzzy integrator #4 is presented, this fuzzy integrator uses trapezoidal and *gBell* membership functions, in this case 4 membership functions in each input and the output are used.

3.5 Databases

The databases used in this work are described below in more detail.

3.5.1 Ear Database

We used a database of the University of Science and Technology of Beijing [14]. The database consists of 77 people, which contain 4 images per person (one ear), the image dimensions are 300 x 400 pixels, the format is BMP. A sample of ear images is shown in Figure 12.

Fig. 12 Sample of ear database

The persons are students and teachers from the department of Information Engineering. Two images with angle variation and one with illumination variation are used. Figure 13 shows an example of the pre-processing applied to each image in the ear.

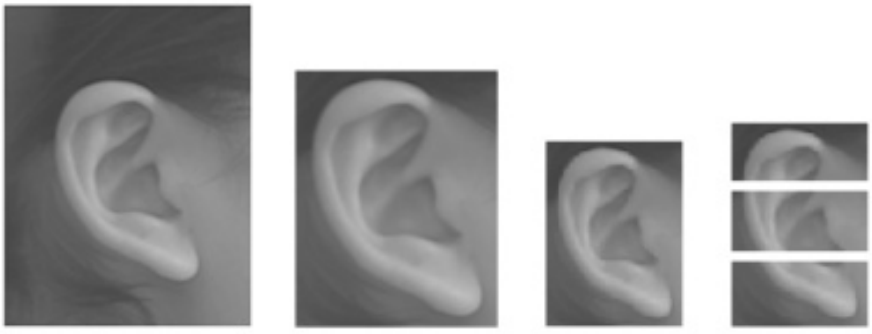


Fig. 13 Sample pre-processing done to the images of ear

3.5.2 Voice Database

In the case of voice, the database consists of 10 voice samples (of 77 persons), WAV format. The persons are students from the Tijuana Institute of Technology. The word that they said in Spanish was "ACCESAR". To preprocess the voice the Mel Frequency Cepstral Coefficients were used.

4 Experimental Results

In this section the results obtained in this work are presented. It was decided to use the database already described above. For the integration of responses the winner takes all method was used.

4.1 Non Optimized

In this test the images percentage and the images, which would be used for training, were established randomly. The non optimized results of the modular neural network are shown below.

4.1.1 Non Optimized Results of Ear

In this section, the non optimized results of ear are shown. Two tests are presented, in the first test, 3 modules are used, and in the second test, the number of modules is random.

4.1.1.1 Non Optimized Results of Ear with 3 Modules

In this test, it is established that 3 was the number of modules, the variables that were established randomly were the percentage of images used and the images, which would be used for training.

The best 5 results for the ear are shown in Table 1. In this test, it can be noticed that when the number of data per module is varied the rate of recognition varies.

It can be noticed that in the training # 4, that when the images 2, 3, and 4 are used a rate of recognition of 100% is obtained.

4.1.1.2 Non Optimized Results of Ear with Different Number of Modules

In this test, the number of modules, the percentage of images and the images, which would be used for training were established randomly. The 6 best results for the ear with different number of modules are shown in Table 2. In this test, it can notice that when the number of data per module and the number of sub modules are varied the rate of recognition varies.

It can be noticed that in the training # 1, that when the images 1, 2, and 3 are used a rate of recognition of 100% is obtained.

Table 1 The best results for the ear (Non Optimized)

Training	Images for training	Persons per module	Recognition Rate
T1O1	(1,3 and 4)	Module # 1 (1 to 6) Module # 2 (7 to 14) Module # 3 (15 to 77)	67.53% (52/77)
T1O2	(2 and 4)	Module # 1 (1 to 38) Module # 2 (39 to 70) Module # 3 (71 to 77)	77.92% (120/154)
T1O3	(1 and3)	Module # 1 (1 to 9) Module # 2 (10 to 44) Module # 3 (45 to 77)	83.11% (128/154)
T1O4	(2, 3 and 4)	Module # 1 (1 to 40) Module # 2 (41 to 50) Module # 3 (51 to 77)	100% (77/77)
T1O5	(2 and 3)	Module # 1 (1 to 23) Module # 2 (24 to 47) Module # 3 (48 to 77)	93.50% (144/154)

Table 2 The best results for the ear (Non Optimized)

Training	Images for training	Persons per module	Recognition Rate
T2O1	(1,2 and 3)	Mod. 1 (1 to 2) Mod. 2 (3 to 11) Mod. 3 (12 to 25) Mod. 4 (26 to 36) Mod. 5 (37 to 43) Mod. 6 (44 to 58) Mod. 7 (59 to 62) Mod. 8 (63 to 77)	100% (77/77)
T2O2	(2 and 3)	Mod. 1 (1 to 5) Mod. 2 (6 to 13) Mod. 3 (14 to 49) Mod. 4 (50 to 52) Mod. 5 (53 to 77)	89.61% (138/154)
T2O3	(2 and 4)	Mod. 1 (1 to 11) Mod. 2 (12 to 51) Mod. 3 (52 to 77)	81.16% (125/154)
T2O4	(2 and 4)	Mod. 1 (1 to 18) Mod. 2 (19 to 30) Mod. 3 (31 to 45) Mod. 4 (46 to 51) Mod. 5 (52 to 59) Mod. 6 (60 to 77)	83.11% (128/154)
T2O5	(3 and 4)	Mod. 1 (1 to 6) Mod. 2 (7 to 14) Mod. 3 (15 to 18) Mod. 4 (19 to 26) Mod. 5 (27 to 42) Mod. 6 (43 to 51) Mod. 7 (52 to 58) Mod. 8 (59 to 65) Mod. 9 (66 to 77)	87.01% (134/154)

4.1.2 Non Optimized Results of Voice

In this section, the non optimized results of ear are shown. Two tests are presented, in the first test, 3 modules are used, and in the second test, the number of modules is random.

4.1.2.1 Non Optimized Results of Voice with 3 Modules

In this test, it is established that 3 was the number of modules, the variables that were established randomly were the percentage of images used and the images, which would be used for training. The 6 best results for the ear with 3 modules are shown in Table 3.

It can be noticed that in the training # 3, that when the voices 1, 3, 5, 7, 8 and 10 are used, a rate of recognition of 96.75% using 8 sub modules is obtained.

Table 3 The best results for voice (Non Optimized)

Training	Voices for training	Persons per module	Recognition Rate
T1V1	53% (1,3,6,7 and 9)	Module # 1 (1 to 22) Module # 2 (23 to 57) Module # 3 (58 to 77)	278/385 72.20%
T1V2	48% (1,2,5,6 and 7)	Module # 1 (1 to 39) Module # 2 (40 to 68) Module # 3 (69 to 77)	260/385 67.53%
T1V3	35% (2,5,8 and 9)	Module # 1 (1 to 36) Module # 2 (37 to 68) Module # 3 (69 to 77)	401/462 86.79%
T1V4	46% (3,5,6,7 and 10)	Module # 1 (1 to 40) Module # 2 (41 to 67) Module # 3 (68 to 77)	347/385 90.12%
T1V5	59% (1,3,5,7,8 and 10)	Module # 1 (1 to 7) Module # 2 (8 to 39) Module # 3 (40 to 77)	298/308 96.75%

4.1.2.2 Non Optimized Results of Voice with Different Number of Modules

In this test, the number of modules, the percentage of images and the images, which would be used for training were established randomly.

The best 5 results for the voice are shown in Table 4. In this test, we can notice that when the number of data per module is varied the rate of recognition varies.

It can be noticed that in the training # 3, that when the voices 1, 3, 5, 7, 8 and 10 are used, a rate of recognition of 96.75% using 8 sub modules is obtained.

Table 4 The best results for voice (Non Optimized)

Training	Voices for training	Persons per module	Recognition Rate
T2V1	53% (4,5,7,9,10)	Mod. # 1 (1 a 14) Mod. # 2 (15 a 26) Mod. # 3 (27 a 38) Mod. # 4 (39 a 41) Mod. # 5 (42 a 45) Mod. # 6 (46 a 62) Mod. # 7 (63 a 77)	360/385 93.50%
T2V2	81% (1,2,3,4,5,6,8,10)	Mod. # 1 (1 a 11) Mod. # 2 (12 a 40) Mod. # 3 (41 a 77)	112/154 72.72%
T2V3	9% (4)	Mod. # 1 (1 a 16) Mod. # 2 (17 a 32) Mod. # 3 (33 a 45) Mod. # 4 (46 a 64) Mod. # 5 (65 a 77)	534/693 77.05%
T2V4	71% (1,2,3,4,5,6,8)	Mod. # 1 (1 a 15) Mod. # 2 (16 a 36) Mod. # 3 (37 a 53) Mod. # 4 (54 a 58) Mod. # 5 (59 a 63) Mod. # 6 (64 a 77)	201/231 87.01%
T2V5	65% (1,3,4,7,8,9,10)	Mod. # 1 (1 a 3) Mod. # 2 (4 a 17) Mod. # 3 (18 a 19) Mod. # 4 (20 a 28) Mod. # 5 (29 a 33) Mod. # 6 (34 a 53) Mod. # 7 (54 a 60) Mod. # 8 (61 a 77)	226/231 97.83%

4.2 Optimized Results

These tests make use of the multi-objective hierarchical genetic algorithm, this MOHGA allows the optimization of parameters of the modular neural network, such as number of sub modules, percentage of data for training, number of hidden layers, number of neurons per hidden layer, error goal and learning algorithms per module. The solutions that have a recognition rate greater than 97% are taken, and of the resulting set, the solution with lower percentage of data is the best for us.

4.2.1 Optimized Results of the Ear

The main parameters used in this evolution are shown in Table 5 and the Pareto optimal set found for the ear are shown in Figure 14.

Table 5 Main parameters of the MOHGA

Memory Size	Non-Replaceable Memory	Replaceable Memory	Working Memory	Pareto Optimal	Duration
50	25	25	5	7	12:48:08

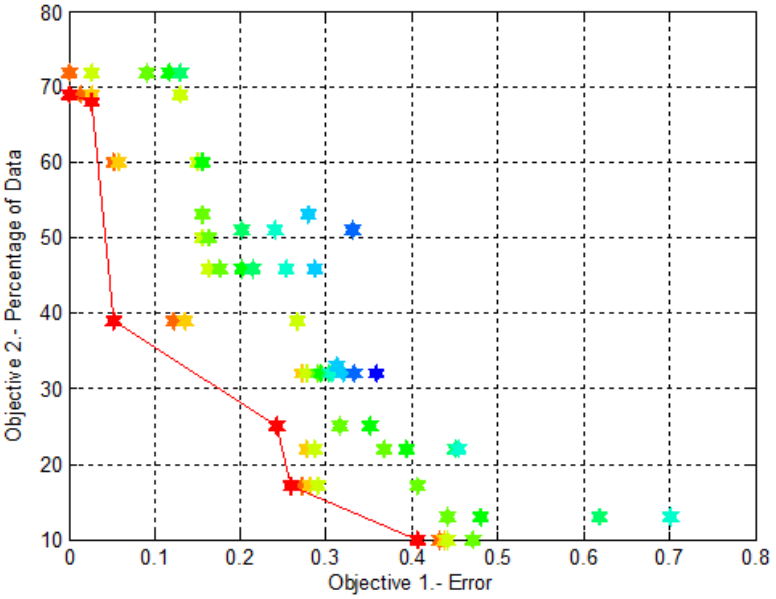


Fig. 14 Pareto optimal set for the evolution of the ear

The solutions found in the Pareto optimal set are shown in Table 6.

Table 6 The best results for the ear (Pareto optimal set)

Solution	Num. of Modules	% Of data	Total rec.	Error
SO1	5	69%	100%	0
SO2	6	68%	97.40%	0.0260
SO3	5	39%	94.80%	0.0519
SO4	5	25%	75.75%	0.2424
SO5	9	17%	74.02%	0.2597
SO6	9	17%	74.02%	0.2597
SO7	5	10%	59.30%	0.4069

The different architectures found by the proposed MOHGA are shown in Table 7.

Table 7 The best result of the ear (Optimized)

Solu-tion	% and images	Num. Hidden layers and Num. of neurons	Persons per module	Rec. Rate	Error
SO1	69% (2,3 and 4)	*3(173,135,44)* *2(153,120)* *4(72,184,96,116)* *2(197,166)* *3(164,22,94)*	Module # 1 (1 to 6) Module # 2 (7 to 13) Module # 3 (14 to 27) Module # 4 (28 to 53) Module # 5(54 to 77)	77/77 100%	0
SO2	68% (1,2 and 3)	*2(41,23)* *3(129,181,30)* *5(82,93,68,140,33)* *4(109,113,131,178)* *1(27)* *1(90)*	Module # 1 (1 a 19) Module # 2 (20 a 21) Module # 3 (22 a 42) Module # 4 (43 a 60) Module # 5 (61 a 68) Module # 6 (69 a 77)	75/77 97.40%	0.0260
SO3	39% (2 and 3)	*4(137,117,163,30)* *4(198,94,151,100)* *4(59,198,102,133)* *3(170,140,173)* *3(113,140,56)*	Module # 1 (1 a 18) Module # 2 (19 a 39) Module # 3 (40 a 57) Module # 4 (58 a 64) Module # 5 (65 a 77)	146/154 94.80%	0.0519
SO4	25% (1)	*3(61,133,146)* *1(114)* *4(82,169,30,123)* *2(64,184)* *2(129,150)*	Module # 1 (1 a 17) Module # 2 (18 a 32) Module # 3 (33 a 48) Module # 4 (49 a 60) Module # 5 (61 a 77)	175/231 75.75%	0.2424
SO5	17% (1)	*2(76,169)* *1(59)* *5(135,103,176,146,198)* *3(175,133,77)* *4(128,177,123,167)* *2(167,111)* *2(180,171)* *4(148,148,22,58)* *4(37,82,86,109)*	Module # 1 (1 a 12) Module # 2 (13 a 20) Module # 3 (21 a 25) Module # 4 (26 a 29) Module # 5 (30 a 33) Module # 6 (34 a 48) Module # 7 (49 a 58) Module # 8 (59 a 62) Module # 9 (63 a 77)	171/231 74.02%	0.2597
SO6	17% (4)	*2(76,169)* *1(59)* *5(135,103,176,146,198)* *3(175,133,77)* *2(128,177)* *4(167,111,88,81)* *2(180,171)* *3(148,148,22)* *2(37,82)*	Module # 1 (1 a 10) Module # 2 (11 a 18) Module # 3 (19 a 23) Module # 4 (24 a 26) Module # 5 (27 a 35) Module # 6 (36 a 41) Module # 7 (42 a 57) Module # 8 (58 a 62) Module # 9 (63 a 77)	171/231 74.02%	0.2597

Table 7 (*continued*)

SO7	10% (4)	*2(146,190)* *2(80,60)* *3(129,170,72)* *3(46,36,181)* *1(112)*	Module # 1 (1 a 25) Module # 2 (26 a 42) Module # 3 (43 a 65) Module # 4 (66 a 68) Module # 5 (69 a 77)	137/231 59.30%	0.4069

4.2.2 Optimized Results of the Voice

The main parameters used in this evolution are shown in Table 8 and the Pareto Optimal set found for the voice are shown in Figure 15.

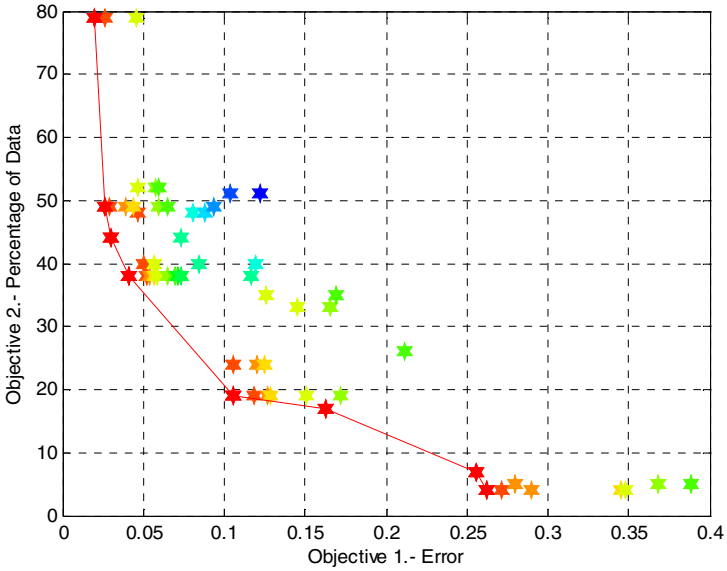


Fig. 15 Pareto optimal set of the evolution of voice

Table 8 Main parameters of the MOHGA

Memory Size	Non-Replaceable Memory	Replaceable Memory	Working Memory	Pareto Optimal	Duration
50	25	25	5	9	01:51:12

The solutions found in the Pareto optimal set are shown in Table 9.

Table 9 The best results for voice (Pareto Optimal)

Solution	Num. of Modules	% Of data	Total rec.	Error
SV1	5	79%	98.05%	0.0195
SV2	9	49%	97.40%	0.0260
SV3	9	44%	96.96%	0.0303
SV4	5	38%	95.88%	0.0411
SV5	10	19%	89.44%	0.1055
SV6	10	19%	89.44%	0.1055
SV7	7	17%	83.76%	0.1623
SV8	7	7%	74.45%	0.2554
SV9	6	4%	73.73%	0.2626

The different architectures found by the proposed MOHGA are shown in Table 10.

Table 10 The best result of the voice (Optimized)

Solu-tion	% and images	Num. Hidden layers and Num. of neurons	Persons per module	Rec. Rate	Error
SV1	79% (1,2,4,5, 6,8,9 and 10)	*3(171,47,23)* *1(196)* *4(131,197,60,38)* *3(149,102,124)* *2(154,93)*	Module # 1 (1 a 19) Module # 2 (20 a 46) Module # 3 (47 a 59) Module # 4 (60 a 73) Module # 5 (74 a 77)	151/154 98.05%	0.0195
SV2	49% (1,3,8,9 and 10)	*4 (57,144,128,83)* *4 (156,189,158,193)* *5(123,105,169,110,105)* *1(89)* *3(78,143,62)* *2(101,38)* *4(22,60,91,173)* *4(81,128,139,118)* *4(145,28,187,32)*	Module # 1(1 a 14) Module # 2(15 a 35) Module # 3(36 a 46) Module # 4(47 a 50) Module # 5(51 a 53) Module # 6(54 a 55) Module # 7(56 a 64) Module # 8(65 a 72) Module # 9(73 a 77)	375/385 97.40%	0.0260
SV3	44% (2,5,9 and 10)	*4 (36,178,109,162)* *4 (165,106,128,83)* *3 (83,94,118)* *2 (139,60)* *3 (68,195,127)* *4 (68,61,103,181)* *4 (61,137,59,187)* *3 (46,87,76)* *5 (82,47,189,32,129)*	Module # 1(1 a 8) Module # 2(9 a 19) Module # 3(20 a 26) Module # 4(27 a 34) Module # 5(35 a 43) Module # 6(44 a 54) Module # 7(55 a 57) Module # 8(58 a 66) Module # 9(67 a 77)	448/462 96.96%	0.0303

Table 10 (*continued*)

SV4	38% (2,3,8 and 10)	*1 (92)*	Module # 1(1 a 16)	443/462 95.88%	0.0411
		3 (183,52,119)	Module # 2(17 a 36)		
		1 (21)	Module # 3(37 a 38)		
		3 (192,92,93)	Module # 4(39 a 59)		
		3 (184,147,61)	Module # 5(60 a 77)		
SV5	19% (1 and 9)	*3(139,117,143)*	Module # 1(1 a 9)	551/616 89.44%	0.1055
		5(112,23,178,55,59)	Module # 2(10 a 11)		
		3(104,130,81)	Module # 3(12 a 13)		
		2(130,191)	Module # 4(14 a 17)		
		3(194,80,132)	Module # 5(18 a 35)		
		3(138,87,141)	Module # 6(36 a 43)		
		3(106,113,162)	Module # 7(44 a 60)		
		3(176,145,108)	Module # 8(61 a 67)		
		4(132,67,69,116)	Module # 9(68 a 74)		
		3(98,132,179)	Module # 10(75 a 77)		
SV6	19% (2 and 8)	*2(36,178)*	Module # 1(1 a 3)	551/616 89.44%	0.1055
		1(165)	Module # 2(4 a 10)		
		3(83,94,118)	Module # 3(11 a 26)		
		2(139,60)	Module # 4(27 a 29)		
		4(68,195,127,186)	Module # 5(30 a 34)		
		2(68,61)	Module # 6(35 a 48)		
		4(61,137,59,187)	Module # 7(49 a 59)		
		2(46,87)	Module # 8(60 a 71)		
		2(82,47)	Module # 9(72 a 73)		
		4(109,81,68,61)	Module # 10(74 a 77)		
SV7	17% (1 and 5)	*4(117,141,110,127)*	Module # 1(1 a 8)	516/616 83.76%	0.1623
		3(171,149,189)	Module # 2(9 a 15)		
		3(44,48,62)	Module # 3(16 a 34)		
		3(148,158,158)	Module # 4(35 a 44)		
		5(109,130,134,129,150)	Module # 5(45 a 55)		
		2(54,176)	Module # 6(56 a 58)		
		3(95,77,152)	Module # 7(59 a 77)		
SV8	7% (2)	*4(107,74,187,179)*	Module # 1(1 a 20)	516/693 74.45%	0.2554
		4(39,79,134,133)	Module # 2(21 a 38)		
		4(68,177,98,52)	Module # 3(39 a 47)		
		2(97,127)	Module # 4(48 a 50)		
		1(90)	Module # 5(51 a 53)		
		4(180,59,175,182)	Module # 6(54 a 68)		
		4(149,158,93,199)	Module # 7(69 a 77)		
SV9	4% (7)	*1(136)*	Module # 1(1 a 17)	511/693 73.73%	0.2626
		2(87,88)	Module # 2(18 a 19)		
		5(105,73,144,67,87)	Module # 3(20 a 44)		
		1(183)	Module # 4(45 a 47)		
		3(37,124,97)	Module # 5(48 a 67)		
		2(47,82)	Module # 6(68 a 77)		

4.3 Comparison among Non-optimized and Optimized Results

In this section, a comparison among the different results is performed. The best results of each test are presented in order to better analyze the results.

4.3.1 Comparison among Non-optimized and Optimized Results of Ear

The best results of ear in the different tests don't vary, a recognition rate of 100% is obtained in each test, these results, using 3 images for the training phase, it means, only using 1 image for the testing phase. The best results of each test are presented in Table 11 to 13.

Table 11 The best results for the ear (Non Optimized)

Training	Images for training	Persons per module	Recognition Rate
T1O4	(2, 3 and 4)	Module # 1 (1 to 40) Module # 2 (41 to 50) Module # 3 (51 to 77)	100% (77/77)

Table 12 The best results for the ear (Non Optimized)

Training	Images for training	Persons per module	Recognition Rate
T2O1	(1,2 and 3)	Mod. 1 (1 to 2) Mod. 2 (3 to 11) Mod. 3 (12 to 25) Mod. 4 (26 to 36) Mod. 5 (37 to 43) Mod. 6 (44 to 58) Mod. 7 (59 to 62) Mod. 8 (63 to 77)	100% (77/77)

Table 13 The best result of the ear (Optimized)

Num. of Mod.	% and images	Num. Hidden layers and Num. of neurons	Persons per module	Rec. Rate	Error
SO5	69% (2,3 and 4)	*3(173,135,44)* *2(153,120)* *4(72,184,96,116)* *2(197,166)* *3(164,22,94)*	Module # 1 (1 to 6) Module # 2 (7 to 13) Module # 3 (14 to 27) Module # 4 (28 to 53) Module # 5(54 to 77)	77/77 100%	0

4.3.2 Comparison among Non-optimized and Optimized Results of Voice

The best results of each test are presented in Table 14 to 16. The best result for us is presented in Table 16, this is the best result for us, because a recognition rate of

97.40% is obtained using only 5 samples of voice, less images than the best result presented in Table 15.

Table 14 The best results for voice (Non Optimized)

Training	Voices for training	Persons per module	Recognition Rate
T1V5	59% (1,3,5,7,8 and 10)	Module # 1 (1 to 7) Module # 2 (8 to 39) Module # 3 (40 to 77)	298/308 96.75%

Table 15 The best results for voice (Non Optimized)

Training	Voices for training	Persons per module	Recognition Rate
T2V5	65% (1,3,4,7,8,9,10)	Mod. # 1 (1 a 3) Mod. # 2 (4 a 17) Mod. # 3 (18 a 19) Mod. # 4 (20 a 28) Mod. # 5 (29 a 33) Mod. # 6 (34 a 53) Mod. # 7 (54 a 60) Mod. # 8 (61 a 77)	226/231 97.83%

Table 16 The best result of the voice (Optimized)

Num. of Mod.	% and voices	Num. Hidden layers and Num. of neurons	Persons per module	Rec. Rate	Error
SV9	49% (1,3,8,9 and 10)	*4 (57,144,128,83)* *4 (156,189,158,193)* *5(123,105,169,110, 105)* *1(89)* *3(78,143,62)* *2(101,38)* *4(22,60,91,173)* *4(81,128,139,118)* *4(145,28,187,32)*	Module # 1(1 to 14) Module # 2(15 to 35) Module # 3(36 to 46) Module # 4(47 to 50) Module # 5(51 to 53) Module # 6(54 to 55) Module # 7(56 to 64) Module # 8(65 to 72) Module # 9(73 to 77)	375/385 97.40%	0.0260

4.4 Fuzzy Integration

Seven cases were established for combining different training of ear and voice, for non-optimized and optimized results. In column 2 and 3 are shown how the trainings were combined each other. The different results obtained with the different fuzzy integrator already described are shown in Table 17.

Table 17 Comparison among the Fuzzy Integrator #1 to #4

Case	Ear	Voice	Fuzzy integrator #1	Fuzzy integrator #2	Fuzzy integrator #3	Fuzzy integrator #4
1	T2O4 83.11%	SV4 95.88%	446/462 97.61%	445/462 96.32%	446/462 96.53%	445/462 96.32%
2	SO1 100%	T2V5 97.835	229/231 99.13%	230/231 99.56%	230/231 99.56%	230/231 99.56%
3	SO7 59.30%	T2V2 67.53%	299/385 77.66%	305/385 79.22%	304/385 78.96%	307/385 79.74%
4	T2O3 81.16%	SV9 73.73%	620/693 89.46%	622/693 89.75%	621/693 89.61%	620/693 89.46%
5	T1O1 67.53%	SV2 97.40%	374/385 97.14%	373/385 96.88%	373/385 96.88%	373/385 96.88%
6	T1O5 93.50%	T1V4 90.12%	374/385 97.14%	373/385 96.88%	374/385 97.14%	374/385 97.14%
7	SO7 97.40	SV8 74.45%	661/693 95.38%	671/693 96.82%	669/693 96.53%	671/693 96.82%

Different results are obtained with the fuzzy integrators, these results are considered good results because the fuzzy integrator provides good results even when the modular neural networks (ear and voice) are not the best trainings o evolutions. These results can be improved when a hierarchical genetic algorithm will be developed.

5 Conclusions

A new method for combining modular neural networks with a granular approach was proposed. The main goal of this work was providing the modular neural networks with the following characteristics: allow changing the number of modules, data per module, and percentage of data for training, all of that with the goal of obtaining a better rate of recognition.

A multi-objective hierarchical genetic algorithm was developed for optimization of some parameters of this model of modular neural networks, those parameters are the number of modules, percentage of data for training, goal error per module, number of hidden layers per module and their respective neurons. This MOHGA is able to obtain the best modular neural network with the lowest error of recognition and that uses the lowest percentage of data for the training phase.

In this work when the tests with the ear are compared, a significant difference does not exist, because the database has few images per person, but when a comparison is performed among the non-optimized and optimized results in the case of the voice, a better recognition rate is obtained using less data in the training phase.

Finally, fuzzy integrators were used for combining the responses of the modular neural networks. In this work, four non-optimized fuzzy integrators were proposed to perform this part of the proposed method. Each fuzzy integrator has different parameters such as number of rules, number, type and parameters of membership functions, because the behavior of each integrator wanted to be observed. Good results were obtained, even when the results of both modular neural networks (ear and voice) were not the best. The results in the integration phase can be improved if a hierarchical genetic algorithm is developed (as we presented in the general architecture).

References

1. Abiyev, R., Altunkaya, K.: Personal Iris Recognition Using Neural Network. Near East University, Department of Computer Engineering, Lefkosa, North Cyprus (April 2008)
2. Abraham, A., Jain, L., Goldberg, R.: Evolutionary Multiobjective Optimization, 1st edn. Springer (2005); Softcover reprint of hardcover
3. Auda, G., Kamel, M.S.: Modular Neural Networks a Survey. Int. J. Neural Syst. 9(2), 129–151 (1999)
4. Azamm, F.: Biologically Inspired Modular Neural Networks. PhD thesis, Virginia Polytechnic Institute and State University, Blacksburg, Virginia (2000)
5. Bajpai, S., Jain, K., Jain, N.: Artificial Neural Networks. International Journal of Soft Computing and Engineering (IJSCE) 1(NCAI 2011), 2231–2307 (2011) ISSN: 2231-2307
6. Bargiela, A., Pedrycz, W.: The roots of granular computing. In: IEEE International Conference on Granular Computing (GrC), pp. 806–809 (2006)
7. Castillo, O., Melin, P.: Soft Computing for Control of Non-Linear Dynamical Systems. Springer, Heidelberg (2001)
8. Castillo, O., Melin, P.: Type-2 Fuzzy Logic Theory and Applications, pp. 29–43. Springer, Berlin (2008)
9. Castro, J.R., Castillo, O., Melin, P.: An Interval Type-2 Fuzzy Logic Toolbox for Control Applications. In: FUZZ-IEEE 2007, pp. 1–6 (2007)
10. Castro, J.R., Castillo, O., Melin, P., Rodriguez-Diaz, A.: Building Fuzzy Inference Systems with a New Interval Type-2 Fuzzy Logic Toolbox. Transactions on Computational Science 1, 104–114 (2008)
11. Coello Coello, C.A., Lamont, G.B., van Veldhuizen, D.A.: Evolutionary Algorithms for Solving Multi-Objective Problems, 2nd edn. Springer (2007)
12. Coello Coello, C.A., Toscano Pulido, G.: A Micro-Genetic Algorithm for Multiobjective Optimization. In: EMO, pp. 126–140 (2001)
13. Coley, A.: An Introduction to Genetic Algorithms for Scientists and Engineers. Wspc (Har/Dskt edition) (1999)
14. Database Ear Recognition Laboratory from the University of Science & Technology Beijing (USTB). Found on the Web page: `http://www.ustb.edu.cn/resb/en/index.htm` (accessed September 21, 2009)
15. Fonseca, C.M., Fleming, P.J.: Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization. In: ICGA 1993, pp. 416–423 (1993)

16. Giunchglia, F., Walsh, T.: A theory of abstraction. Artificial Intelligence 56, 323–390 (1992)
17. Han, J., Dong, J.: Perspectives of Granular Computing in Software Engineering. In: GrC 2007, pp. 66–71 (2007)
18. Haupt, R., Haupt, S.: Practical Genetic Algorithms, 2nd edn., pp. 42–43. Wiley-Interscience (2004)
19. 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. Inf. Sci. 179(13), 2123–2145 (2009)
20. 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: IJCNN 2008, pp. 738–745 (2008)
21. 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 Computing for Hybrid Intelligent Systems, 89–114 (2008)
22. Hidalgo, D., Melin, P., Licea, G., Castillo, O.: Optimization of Type-2 Fuzzy Integration in Modular Neural Networks Using an Evolutionary Method with Applications in Multimodal Biometry. In: Aguirre, A.H., Borja, R.M., Garciá, C.A.R. (eds.) MICAI 2009. LNCS, vol. 5845, pp. 454–465. Springer, Heidelberg (2009)
23. Hobbs, J.: Granularity. In: Proc. of IJCAI, pp. 432–435 (1985)
24. Huang, J., Wechsler, H.: Eye Location Using Genetic Algorithm. In: Second International Conference on Audio and Video-Based Biometric Person Authentication, pp. 130–135 (1999)
25. Jang, J., Sun, C., Mizutani, E.: Neuro-Fuzzy and Soft Computing. Prentice Hall, New Jersey (1997)
26. Khan, A., Bandopadhyaya, T., Sharma, S.: Classification of Stocks Using Self Organizing Map. International Journal of Soft Computing Applications 4, 19–24 (2009)
27. Lin, T.Y., Granular computing, announcement of the BISC Special Interest Group on Granular Computing (1997)
28. 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 (2005)
29. Melin, P., Kacprzyk, J., Pedrycz, W. (eds.): Bio-inspired Hybrid Intelligent Systems for Image Analysis and Pattern Recognition. SCI, vol. 256. Springer, Heidelberg (2009)
30. 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)
31. Melin, P., Mendoza, O., Castillo, O.: Face Recognition with an Improved Interval Type-2 Fuzzy Logic Sugeno Integral and Modular Neural Networks. IEEE Transactions on Systems, Man, and Cybernetics, Part A 41(5), 1001–1012 (2011)
32. Melin, P., Sánchez, D., Castillo, O.: Genetic optimization of modular neural networks with fuzzy response integration for human recognition. Information Sciences 197, 1–19 (2012)
33. Mendel, J.: Uncertain Rule-Based Fuzzy Logic Systems: Introduction and New Directions. Prentice-Hall, Upper Saddle River (2001)
34. Mendoza, O., Melin, P., Castillo, O.: Interval type-2 fuzzy logic and modular neural networks for face recognition applications. Appl. Soft Comput. 9(4), 1377–1387 (2009)

35. 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)
36. Mitchell, M.: An Introduction to Genetic Algorithms, 3rd edn. A Bradford Book (1998)
37. Moreno, B., Sanchez, A., Velez, J.F.: On the Use of Outer Ear Images for Personal Identification in Security Applications. In: IEEE 33rd Annual International Carnahan Conference on Security Technology, pp. 469–476 (1999)
38. Nawa, N., Takeshi, F., Hashiyama, T., Uchikawa, Y.: A study on the discovery of relevant fuzzy rules using pseudobacterial genetic algorithm. IEEE Transactions on Industrial Electronics 46(6), 1080–1089 (1999)
39. 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)
40. Pawlak, Z.: Granularity of knowledge, indiscernibility and rough sets. In: Proceedings of IEEE International Conference on Fuzzy Systems, pp. 106–110 (1998)
41. Pawlak, Z.: Rough Sets. International Journal of Computer and Information Sciences 11, 341–356 (1982)
42. 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: Melin, P., Kacprzyk, J., Pedrycz, W. (eds.) Soft Computing for Recognition Based on Biometrics. SCI, vol. 312, pp. 85–102. Springer, Heidelberg (2010)
43. Santos, J.M., Alexandre, L.A., Marques de Sá, J.: Modular Neural Network Task Decomposition Via Entropic Clustering. In: ISDA (1), pp. 62–67 (2006)
44. Segovia, J., Szczepaniak, P.S., Niedzwiedzinski, M.: E-Commerce and Intelligent Methods, 1st edn., p. 181. Physica-Verlag (2002)
45. Tang, K.S., Man, K.F., Kwong, S., Liu, Z.F.: Minimal Fuzzy Memberships and Rule Using Hierarchical Genetic Algorithms. IEEE Trans. Ind. Electron. 45(1), 162–169 (1998)
46. Wang, C., Soh, Y.C., Wang, H., Wang, H.: A Hierarchical Genetic Algorithm for Path Planning in a Static Environment with Obstacles. In: IEEE CCECE 2002 Canadian Conference on Electrical and Computer Engineering, vol. 3, pp. 1652–1657 (2002)
47. Wang, W., Bridges, S.: Genetic Algorithm Optimization of Membership Functions for Mining Fuzzy Association Rules. Department of Computer Science Mississippi State University (March 2, 2000)
48. Worapradya, K., Pratishthananda, S.: Fuzzy supervisory PI controller using hierarchical genetic algorithms. In: 5th Asian Control Conference, vol. 3, pp. 1523–1528 (2004)
49. Yao, J.T.: A ten-year review of granular computing. In: Proceedings of the 3rd IEEE International Conference on Granular Computing (GrC), pp. 734–739 (2007)
50. Yao, J.T.: Information granulation and granular relationships. In: Proceedings of 2005 IEEE Conference on Granular Computing (GrC), pp. 326–329 (2005)
51. Yao, Y.Y.: A Partition Model of Granular Computing. In: Peters, J.F., Skowron, A., Grzymała-Busse, J.W., Kostek, B.z., Świniarski, R.W., Szczuka, M.S. (eds.) Transactions on Rough Sets I. LNCS, vol. 3100, pp. 232–253. Springer, Heidelberg (2004)
52. Yao, Y.Y.: Granular computing: basic issues and possible solutions. In: Proceedings of the 5th Joint Conferences on Information Sciences, pp. 186–189 (2000)

53. Yao, Y.Y.: On Modeling Data Mining with Granular Computing. In: 25th International Computer Software and Applications Conference (COMPSAC), pp. 638–649 (2001)
54. Yao, Y.Y.: Perspectives of granular computing. In: IEEE International Conference on Granular Computing (GrC), pp. 85–90 (2005)
55. Yu, F., Pedrycz, W.: The design of fuzzy information granules: Tradeoffs between specificity and experimental evidence. Applied Soft Computing 9(1), 264–273 (2009)
56. Zadeh, L.A.: Fuzzy Sets. Journal of Information and Control 8, 338–353 (1965); Jang, J., Sun, C., Mizutani, E.: Neuro-Fuzzy and Soft Computing. Prentice Hall, New Jersey (1997)
57. Zadeh, L.A.: Fuzzy sets and information granularity. In: Gupta, M., Ragade, R., Yager, R. (eds.) Advances in Fuzzy Set Theory amd Applications, pp. 3–18. North-Holland Publishing Co. (1979)
58. Zadeh, L.A.: Towards a theory of fuzzy information granulation and its centrality in human reasoning and fuzzy logic. Fuzzy Sets and Systems 19, 111–127 (1997)

Type-2 Fuzzy Weight Adjustment for Backpropagation in Prediction Time Series and Pattern Recognition

Fernando Gaxiola, Patricia Melin, and Fevrier Valdez

Tijuana Institute of Technology, Tijuana México
fergaor_29@hotmail.com, pmelin@tectijuana.mx,
fevrier@tectijuana.edu.mx

Abstract. In this paper a genetic algorithm is used to optimize the three neural networks in an ensemble model. Genetic algorithms are also used to optimize the two type-2 fuzzy systems that work in the backpropagation learning method with type-2 fuzzy weight adjustment. The mathematical analysis of the proposed learning method architecture and the adaptation of 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 an ensemble neural network of three neural networks and average integration to obtain the final result is presented. The proposed approach is applied to a case of time series prediction and to pattern recognition.

1 Introduction

This paper is focused on the optimization of a neural network ensemble with type-2 fuzzy weights. The optimization is performed in the number of neurons in the hidden layer and in the type-2 fuzzy inference systems used in the hidden and output layer to obtain the type-2 fuzzy weights of each neural network forming the ensemble.

The proposed approach is applied to time series prediction for the Mackey-Glass series. The objective is obtaining the minimum prediction error for the data of the time series. The approach is also applied to a problem of pattern recognition.

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

This neural network is based on supervised learning, where the network operates by having both the correct input and output, and the network adjusts its weights to try in minimize the error of the calculated output.

The research is based in working with the weights of a neural network in a different way to the traditional approach, which is important because this affects the performance of the learning process of the neural network.

P. Melin, O. Castillo (Eds.): *Soft Computing Appli. in Optimization,* STUDFUZZ 294, pp. 187–213.
DOI: 10.1007/978-3-642-35323-9_8

This conclusion is based on the use of neural networks of this type, where some research works have shown that the training of neural networks for the same problem initialized with different weights or its adjustment in a different way, but at the end 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 optimization of the ensemble neural network with type-2 fuzzy weights proposed in this paper. Section 5 presents the simulation results for the proposed method. Finally, in section 6, 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][22].

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][31].

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 function f applied to the sum, resulting on the final value of the output, also called y_i [28], obtaining the following equation:

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

Where f may be a nonlinear function with a binary output + -1, a linear function f (z) = z, or as sigmoidal logistic function:

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

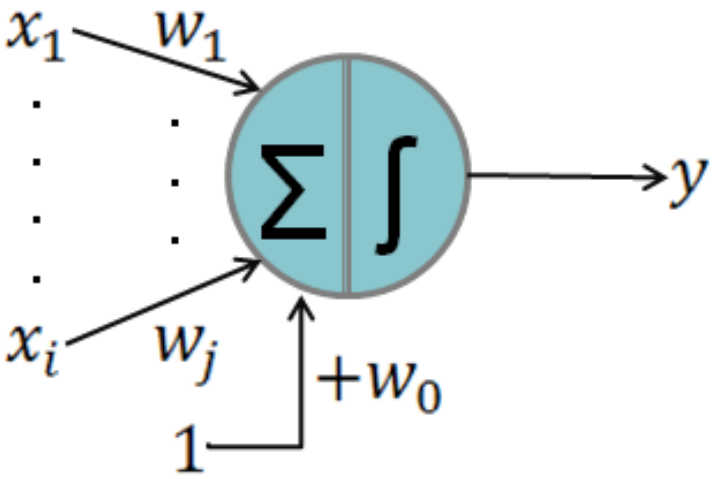


Fig. 1 Schematics of an artificial neuron

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.

There exists many works about adjustment or managing of weights but only the most important and relevant for this research will be considered here [4] [10] [35]:

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 at a higher learning rate that lead to great change in the weights [28] [14].

Adaptive Learning Rate.- Works by calculating the initial output of the network and the initial error. Later for each epoch new weights and bias are calculated using the current learning rate, and new outputs and errors are calculated.

To perform the new calculations, if the new error is greater than previous error more than a reason predefined, the new weights and bias are discarded and the learning rate decreases (multiplied for a decrement constant), otherwise the weights and biases remain. Moreover if the new error is less than the previous error, the learning rate increases (multiplied by an increment constant).

Conjugate Gradient Algorithm.- This is a search for weight adjustments along conjugate directions. Versions of the conjugate gradient algorithm differ in the way in which a constant βk is calculated.

- Fletcher-Reeves update [12]: the constant β_k is calculated using the equation:

$$\beta_k = \frac{\mathbf{g}_k^T \mathbf{g}_k}{\mathbf{g}_{k-1}^T \mathbf{g}_{k-1}} \tag{3}$$

  That is the reason of the norm for the square of the current gradient at the norm of the square of the previous gradient.

- Polak-Ribiere updated [12]: the constant β_k is calculated using the equation:

$$\beta_k = \frac{\Delta \mathbf{g}_{k-1}^T \mathbf{g}_k}{\mathbf{g}_{k-1}^T \mathbf{g}_{k-1}} \tag{4}$$

  This is the internal product of the previous changes in the gradient with the current gradient divided for the square of the norm of the previous gradient.

- Powell-Beale Restart [3] [29]: the restart is performed if it has very little orthogonality between the current gradient and the previous gradient. This is tested with the following inequality:

$$\left| \mathbf{g}_{k-1}^T \mathbf{g}_k \right| \geq 0.2 \left\| \mathbf{g}_k \right\|^2 \tag{5}$$

  If the condition is validated, the search address is restarting at the negative of the gradient.

- Scaled Conjugate Gradient [25]: this method decreases the search time line, combines the approach of reliability model region with the approach of the conjugate gradient.

Gedeon T. [13], performed the weights adjustment with a discrete selection following the Hebbian paradigm: the force of the connection w_{ij} is proportional at the correlation of the activity of the neurons I and j.

Monirul and Murase [26], used a temporal frozen of the weights when the output does not change in a few epochs of successive trainings.

Meltser at al. [24], performed a weights adjustment of the network through BFGS Quasi-Newton method (Broyden-Fletcher-Goldfarn-Shanno), which is a convergent quadratic method that used the Quasi-Newton method to calculate an approximation of the Hebbian matrix.

Barbouinis et al. [2], performed the weights updating using the identification of recursive error prediction (RPE), which allows that the vector of estimated weights is continuously found in each epoch using recursive calculates.

Yeung et al. [36], proposed a new training objective function for a network with radial basis functions, which is used to adjust the weights.

Kamarthi and Pittner [20], focused in obtaining a weight prediction of the network at a future epoch using extrapolation.

Neville et al. [27], worked with sigma-pi networks which are transformed for performing a second task of assignation for which they were initially trained, which scales the first assignment.

Casasent et al. [4], presented a new classificatory neural network for pattern recognition (PQNN) that used weights with complex values and the non linear square law.

De Wilde [9], work performed assuming a non zero diagonal in the weight matrix instead of the zero diagonal that the most researchers assume for the neural networks completely connected.

Yam et al. [35], developed an algorithm to find the initial optimal weights of feedforward neural networks based on the Cauchy inequality and a linear algebraic method.

Draghici [10], Calculates a range of weights for a category of given problems and ensures that the network has the capacity to solve the given problems with integer weights in that range.

Ishibuchi et al. [17], 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. [18], 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 inputs of the fuzzy neurons, as the scalar product of the weights for the input vector.

Recent works on type-2 fuzzy logic have been developed in time series prediction, like that of Castro et al. [7], and other researchers [1][21].

Recent research on genetic algorithm optimization have been developed in neural networks and fuzzy logic, like that of Sanchez et al. [32], and other researchers [30][34].

3 Proposed Method and Problem Description

The focus of this work is to generalize the backpropagation algorithm using type-2 fuzzy sets to allow the neural network to handle data with uncertainty. At the same time, it will be necessary to optimize type-2 fuzzy sets for the corresponding applications and this will require a method to automatically vary the footprint of uncertainty (FOU) of the membership functions.

The initial weight selection will be done differently to the traditional random initialization of weights performed with the backpropagation algorithm (Fig. 2); the proposed method will work with type-2 fuzzy weights, 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 (Fig. 3) [26].

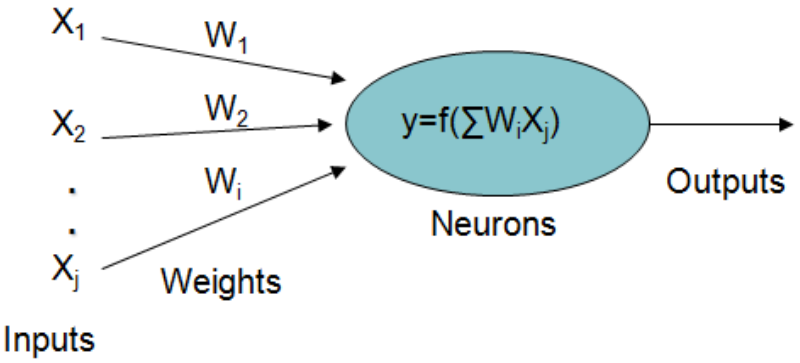


Fig. 2 Scheme of current management of numerical weights (type-0) for the inputs of each neuron

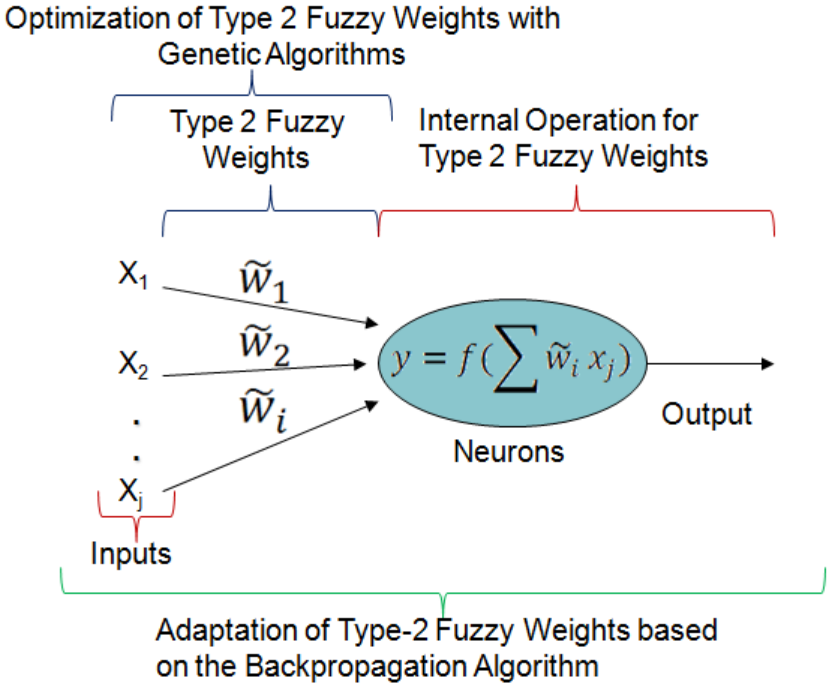


Fig. 3 Schematic of the proposed management of type-2 fuzzy weights for the inputs of each neuron

We considered modifying the current methods of adjusting weights that allow convergence to the correct weights for the problem. We developed a method for adjusting weights to achieve the desired result, searching for the optimal way to work with type-2 fuzzy weights [19].

We used a genetic algorithm for obtaining the optimal type-2 fuzzy weights of the neural network; because in the literature it can be found that it has been very difficult and exhaustive to manually find optimal values for a problem [16].

To define the activation function f (-) to use, the linear and sigmoidal functions were tested, because these functions have been used in similar approaches.

4 Optimization of the Ensemble Neural Network Architecture with Type-2 Fuzzy Weights

The proposed ensemble neural network architecture with type-2 fuzzy weights (see Fig. 4) is described as follows:
Layer 0: Inputs.

$$x = [x_1, x_2, \cdots, x_n] \tag{6}$$

Layer 1: Interval type-2 fuzzy weights for the hidden layer of each neural network.

$$\widetilde{w} = \left[\underline{w}, \overline{w}\right] \tag{7}$$

Where $\left[\underline{w}, \overline{w}\right]$ are the weights of the consequents of each rule of the type-2 fuzzy system with inputs (current fuzzy weight, change of weight) and output (new fuzzy weight).

Layer 2: Hidden neuron with interval type-2 fuzzy weights.

$$Net = \sum_{i=1}^{n} x_i \widetilde{w_i} \tag{8}$$

Layer 3: Output neuron with interval type-2 fuzzy weights.

$$Out = \sum_{i=1}^{n} y_i \widetilde{w_i} \tag{9}$$

Layer 4: Obtain a single output of each one of the three neural networks.

Layer 5: Obtain a final output with the average integration.

The first experiment was performed in time-series prediction, specifically for the Mackey-Glass time series (for τ=17).

We considered three neural networks in the ensemble: the first network with 25 neurons in the hidden layer and 1 neuron in the output layer; the second network with 28 neurons in the hidden layer and 1 neuron in the output layer; and the third network with 38 neurons in the hidden layer and 1 neuron in the output layer (see Fig. 4). This ensemble neural network handles type-2 fuzzy weights in each one of its hidden layers and output layer. In each hidden layer and output of each network we are working with a type-2 fuzzy inference system to obtain new weights in each epoch of the network [5][23][15][33].

The combination of responses of the ensemble neural network is performed by average integration.

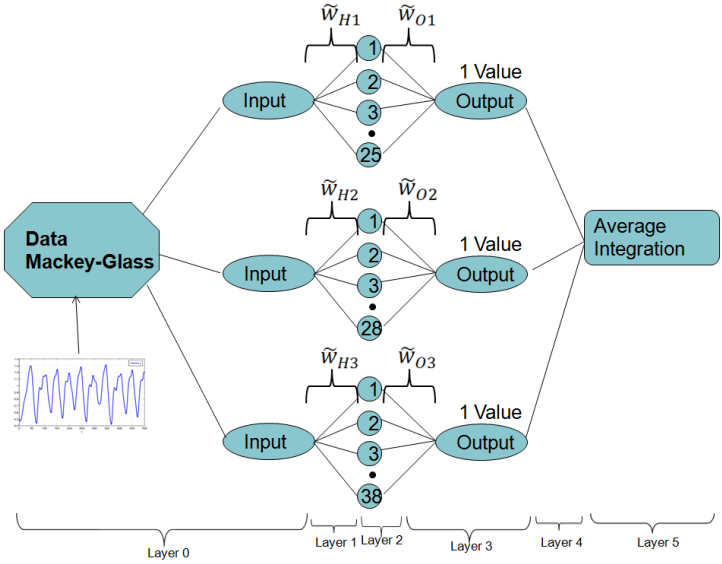


Fig. 4 Ensemble neural network architecture with type-2 fuzzy weights

We used 2 similar type-2 fuzzy systems in each neural network.

The first type-2 fuzzy system consists of two inputs: the weight in the current epoch and the change of the weight for the next epoch, and one output: the new weight for the next epoch (see Fig. 5).

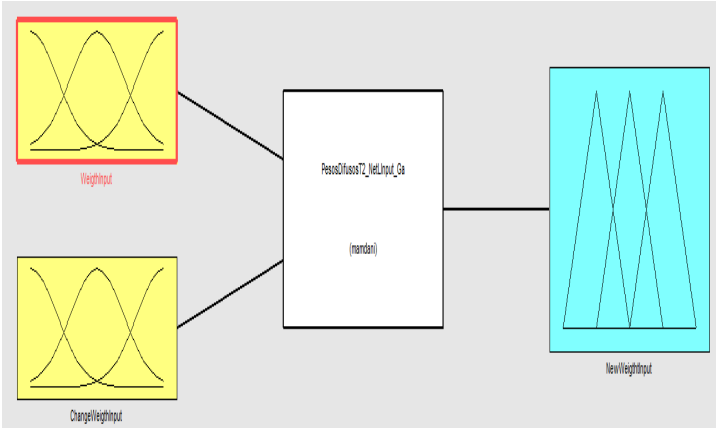


Fig. 5 Structure of the used type-2 fuzzy inference system in the hidden layer

The input of the current weight consists of two triangular membership functions with range of -1 to 1. The input of change of the weight consists of two triangular membership functions with range of -0.1 to 0.1. The output of the new weight consists of two triangular membership functions with range of -1 to 1 (see Fig. 6).

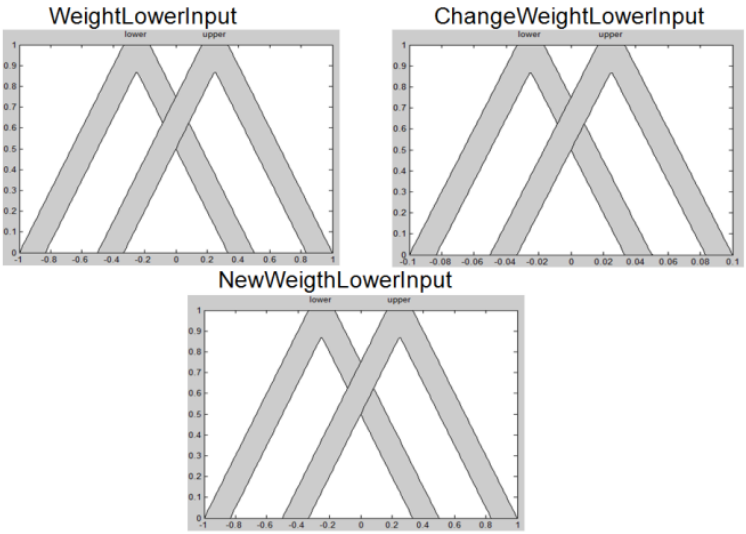


Fig. 6 Inputs and outputs of the type-2 fuzzy inference system for the hidden layer

We used six rules for the type-2 fuzzy inference system of the hidden layer, the four combinations of two membership functions and we added two rules when the change of weight is null (see Fig. 7).

1. If (WeigthInput is lower) and (ChangeWeigthInput is lower) then (NewWeigthtInput is lower) (1)
2. If (WeigthInput is lower) and (ChangeWeigthInput is upper) then (NewWeigthtInput is lower) (1)
3. If (WeigthInput is upper) and (ChangeWeigthInput is lower) then (NewWeigthtInput is upper) (1)
4. If (WeigthInput is upper) and (ChangeWeigthInput is upper) then (NewWeigthtInput is upper) (1)
5. If (WeigthInput is lower) then (NewWeigthtInput is lower) (1)
6. If (WeigthInput is upper) then (NewWeigthtInput is upper) (1)

Fig. 7 Rules of the type-2 fuzzy inference systems used in the hidden layer

The second type-2 fuzzy system consists of two inputs: the weight in the current epoch and the change of the weight for the next epoch, and one output: the new weight for the next epoch (see Fig. 8).

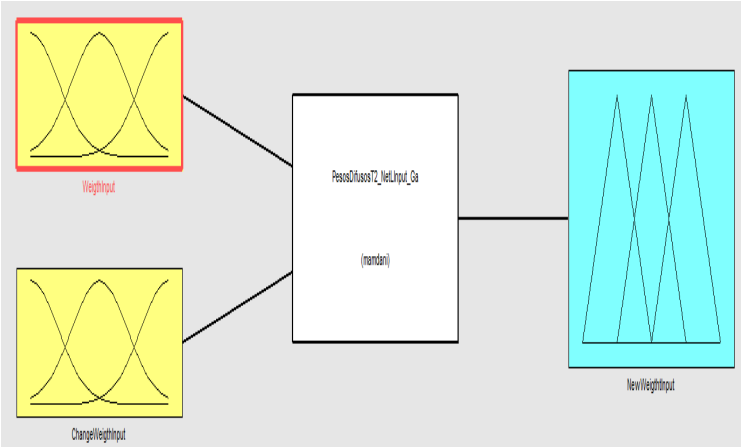


Fig. 8 Structure of the used type-2 fu zzy inference system in the output layer

The input of the current weight consists of two triangular membership functions with range of -0.01 to 0.01. The input of change of the weight consists of two triangular membership functions with range of -0.1 to 0.1. The output of the new weight consists of two triangular membership functions with range of -0.01 to 0.01 (see Fig. 9).

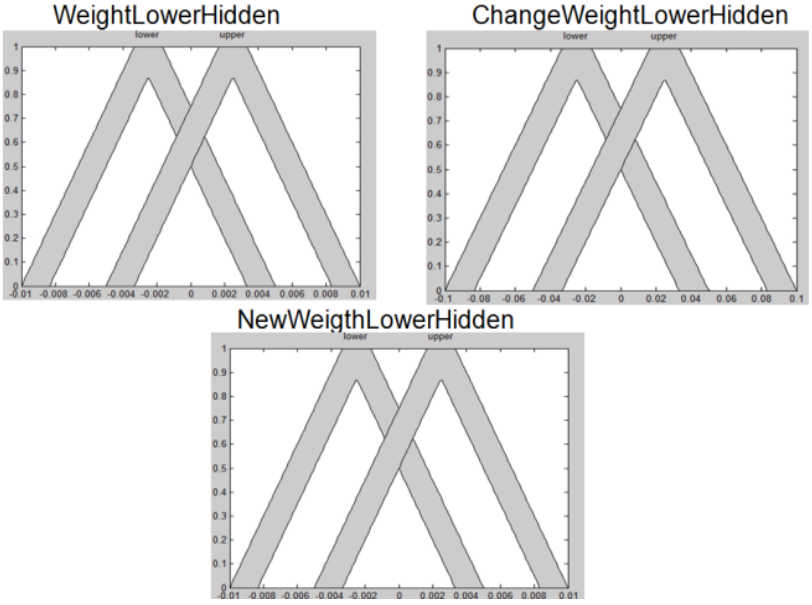


Fig. 9 Inputs and outputs of the type-2 fuzzy inference system for the output layer

We used six rules for the type-2 fuzzy inference system for the output layer, the four combination of two membership functions and we added two rules for the case when the change of weight is null (see Fig. 10).

1. If (WeigthInput is lower) and (ChangeWeigthInput is lower) then (NewWeigthtInput is lower) (1)
2. If (WeigthInput is lower) and (ChangeWeigthInput is upper) then (NewWeigthtInput is lower) (1)
3. If (WeigthInput is upper) and (ChangeWeigthInput is lower) then (NewWeigthtInput is upper) (1)
4. If (WeigthInput is upper) and (ChangeWeigthInput is upper) then (NewWeigthtInput is upper) (1)
5. If (WeigthInput is lower) then (NewWeigthtInput is lower) (1)
6. If (WeigthInput is upper) then (NewWeigthtInput is upper) (1)

Fig. 10 Rules of the type-2 fuzzy inference systems used in the output layer

The optimization was performed for the numbers of neurons in the hidden layer of each neural network, and for the weights of the hidden layer and output layer, in Fig. 11 this is described:

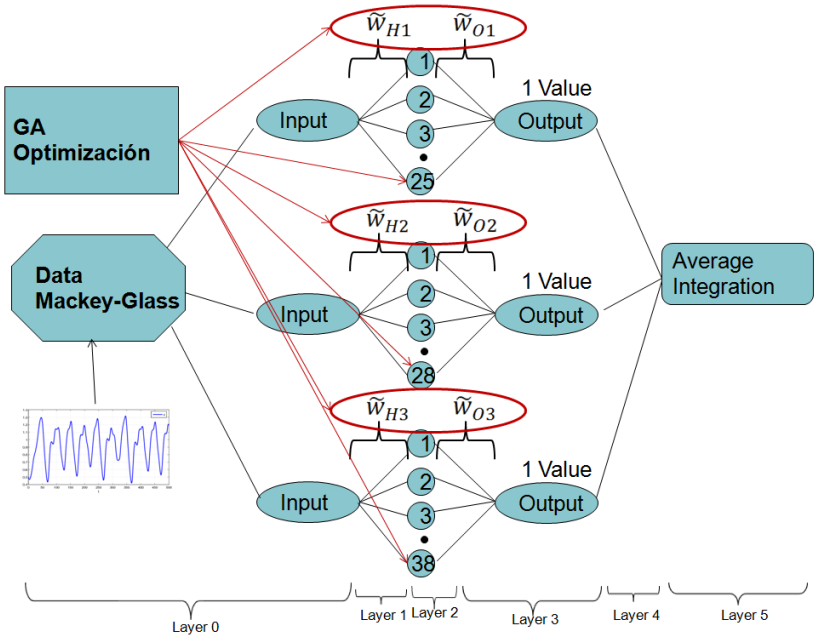


Fig. 11 Proposed optimization of the ensemble neural network architecture with type-2 fuzzy weights

The second experiment was performed in a pattern recognition application, specifically for the human iris biometric measure.

We used a database of human Iris from the Institute of Automation of the Chinese Academy of Sciences (CASIA) (see Fig. 12). It consists of 9 images per person, for a total of 10 individuals, giving a total of 90 images. The image dimensions are 320 x 280, JPEG format.

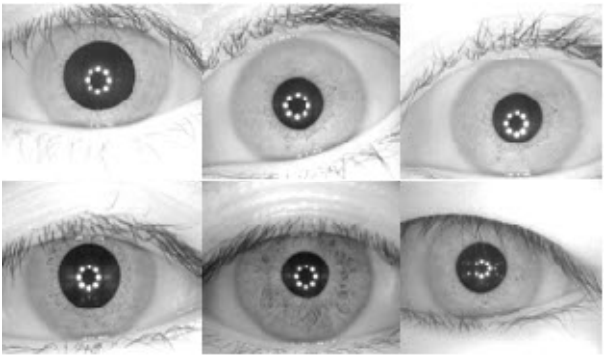


Fig. 12 Examples of the human iris images from the CASIA database

The images of the human iris introduced to the two neural networks were preprocessed as follows:

- Obtain the coordinates and radius of the iris and pupil.
- Making the cut in the Iris.
- Resize the cut of the Iris to 21-21 pixels.
- Convert images from vector to matrix.
- Normalize the images.

Obtain Coordinates of the Center and Radius of the Iris-Pupil: To obtain the coordinates of the center and radius of the iris and pupil of images in the CASIA database, we used a method that involves applying a series of filters and mathematical calculations to achieve the desired gain.

First, we apply edge detection with the Canny method (see Fig. 13 (a)), then the process continues using a gamma adjustment of the image (see Fig 13 (b)), to the resulting image obtained above no maximum suppression is applied (see Fig. 13 (c)), and subsequently we applied to the image a threshold method (see Fig. 13 (d)).

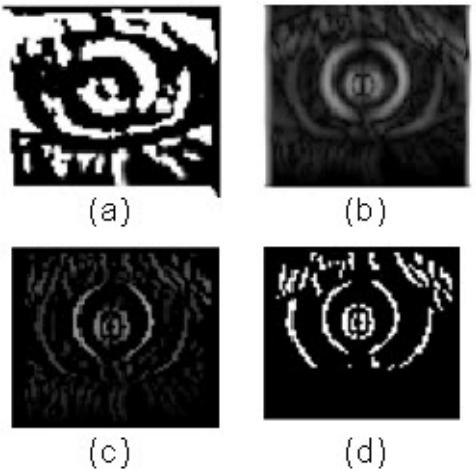


Fig. 13 (a) Edge detection with Canny's method (b) Image Adjust Gamma (c) No Maxima Suppression (d) Threshold

Finally, we apply the Hough transform to find the maximum in the Hough space and, therefore, the circle parameters (row and column at the center of the iris and the radius).

To obtain the coordinates of the center and radius of the pupil, the same process indicated above is used, but now taking into account at the end of the center coordinates and radius of the iris to identify the pupil.

Cut out the Iris: After obtaining the coordinates of the Iris, the upper right and lower left points are calculated to make the cut (see Fig. 14).

RowUpLeft = RowIris – RadiusIris;
RowLowRight = (RowIris + RadiusIris) - RowUpLeft;
ColUpLeft = ColumnIris - RadiusIris;
ColLowRight = (ColumnIris + RadiusIris) - ColUpLeft;

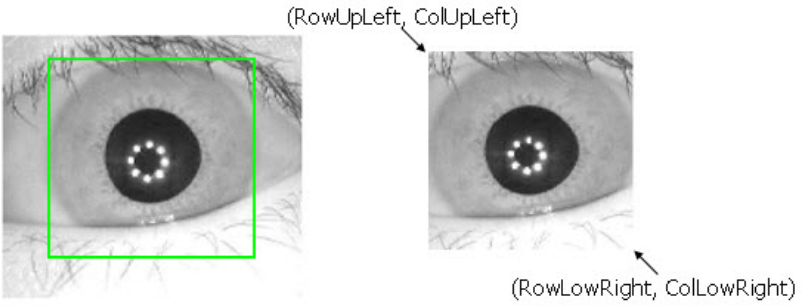


Fig. 14 Cut of iris

The proposed architecture with two neural networks with type-2 fuzzy weights consists of 120 neurons in the hidden layer and 10 neurons in the output layer, the inputs are the preprocessed iris images with a total of 10 persons (60 for training – 60 for test in total) (see Fig. 15). The inputs vary in ±5 percent between the two networks.

We considered two neural networks managing type-2 fuzzy weights in each hidden layer and output layer. In each hidden layer and output layer a type-2 fuzzy inference system was used to obtain the new weights in each epoch of the network.

The two neural networks used the learning method that updates weight and bias values according to the resilient backpropagation algorithm. The update weights are adapted for manage type-2 fuzzy weights.

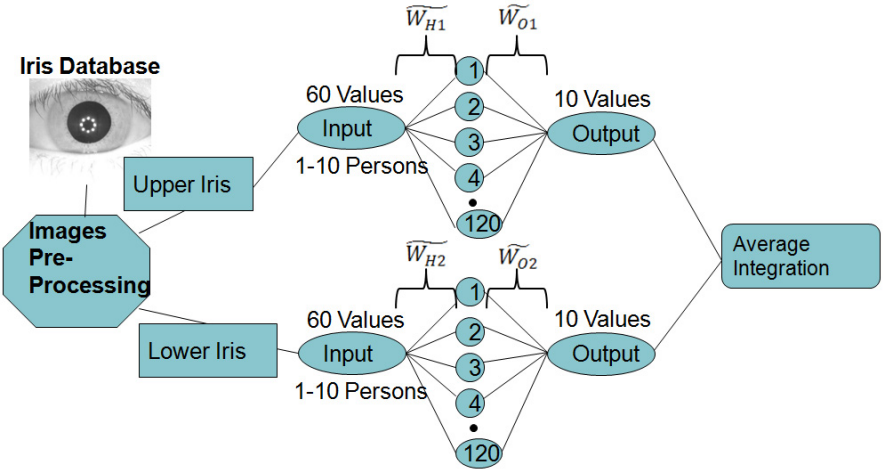


Fig. 15 Proposed neural networks with type-2 fuzzy weights architecture for pattern recognition of human iris biometric measure

We used four type-2 fuzzy inference systems to obtain the new weights, one for the hidden layer in the one network and the second network, and one for the output layer in the first network and the second network. The four type-2 fuzzy inference system consists of two inputs (actual weight and change of weight) and one output (new weight) (see Fig. 16).

The integration of the two networks is realized with average integration.

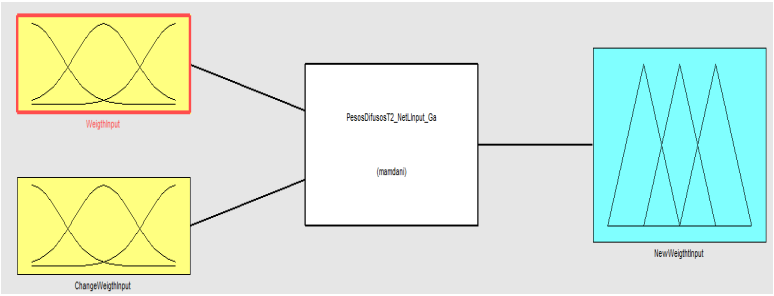


Fig. 16 Structure for the four type-2 fuzzy integration system

The input of the current weight for the type-2 inference system for the hidden layer in the first neural network consists of two triangular membership functions with range of -3500 to 1000. The input of change of the weight consists of two triangular membership functions with range of -60 to 60. The output of the new weight consists of two triangular membership functions with range of -3500 to 1000 (see Fig. 17).

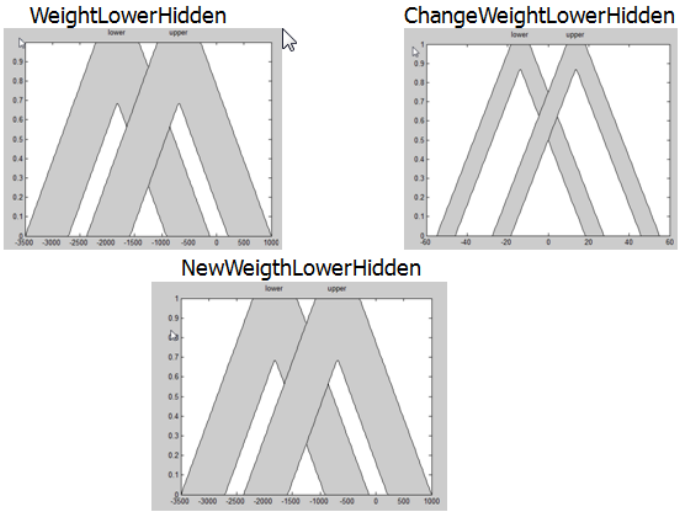


Fig. 17 Inputs and Output for the type-2 fuzzy inference system for the hidden layer in the first network

The input of the current weight for the type-2 inference system for the output layer in the first neural network consists of two triangular membership functions with range of -2 to 2. The input of change of the weight consists of two triangular membership functions with range of -1 to 1. The output of the new weight consists of two triangular membership functions with a range of -2 to 2 (see Fig. 18).

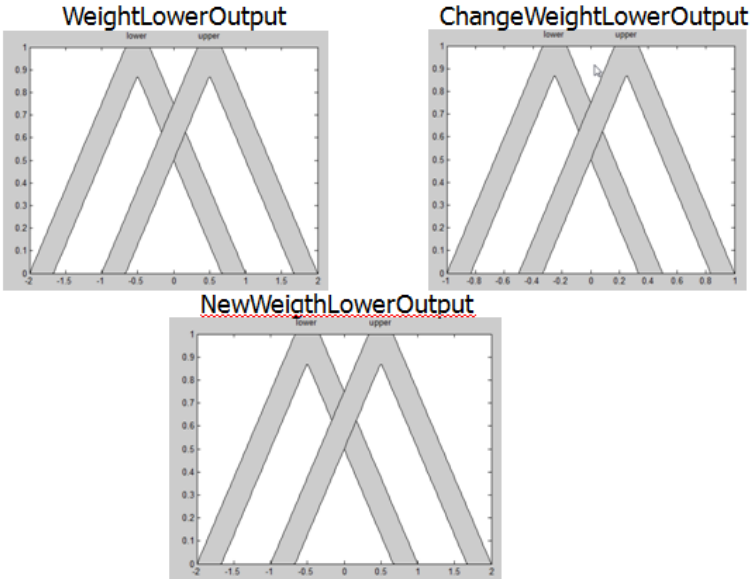


Fig. 18 Inputs and Output for the type-2 fuzzy inference system for the output layer in the first network

We used six rules for the type-2 fuzzy inference system for the hidden and output layer in the first neural network, the four combinations of two membership functions and we added two rules when the change of weight is null (see Fig. 19).

1. If (WeigthLowerInput is lower) and (DelthaLowerInput is lower) then (NewWeigthtLowerInput is lower) (1)
2. If (WeigthLowerInput is lower) and (DelthaLowerInput is upper) then (NewWeigthtLowerInput is upper) (1)
3. If (WeigthLowerInput is upper) and (DelthaLowerInput is lower) then (NewWeigthtLowerInput is lower) (1)
4. If (WeigthLowerInput is upper) and (DelthaLowerInput is upper) then (NewWeigthtLowerInput is upper) (1)
5. If (WeigthLowerInput is lower) then (NewWeigthtLowerInput is lower) (1)
6. If (WeigthLowerInput is upper) then (NewWeigthtLowerInput is lower) (1)

Fig. 19 Rules for the four type-2 fuzzy inference system

The input of the current weight for the type-2 inference system for the hidden layer in the second neural network consists of two triangular membership functions with range of -2 to 2. The input of change of the weight consists of two triangular membership functions with a range of -1 to 1. The output of the new weight consists of two triangular membership functions with a range of -2 to 2 (see Fig. 20).

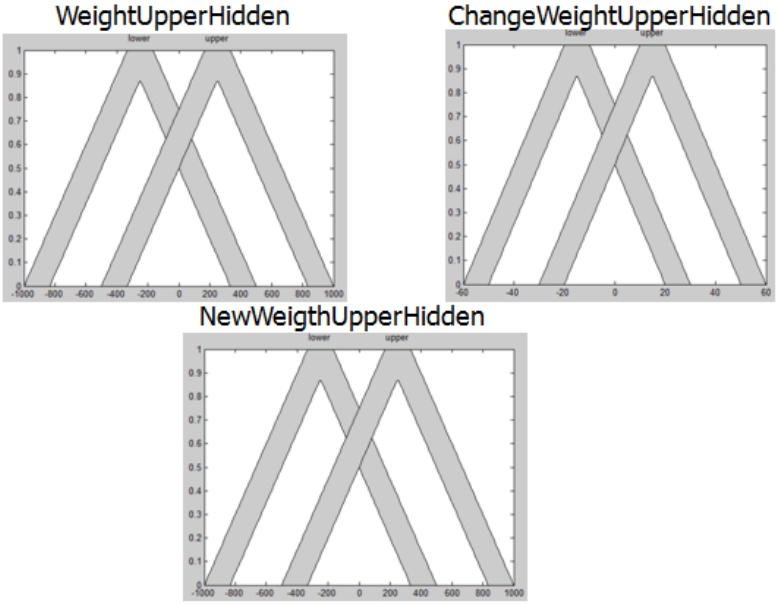


Fig. 20 Inputs and Output for the type-2 fuzzy inference system for the hidden layer in the second network

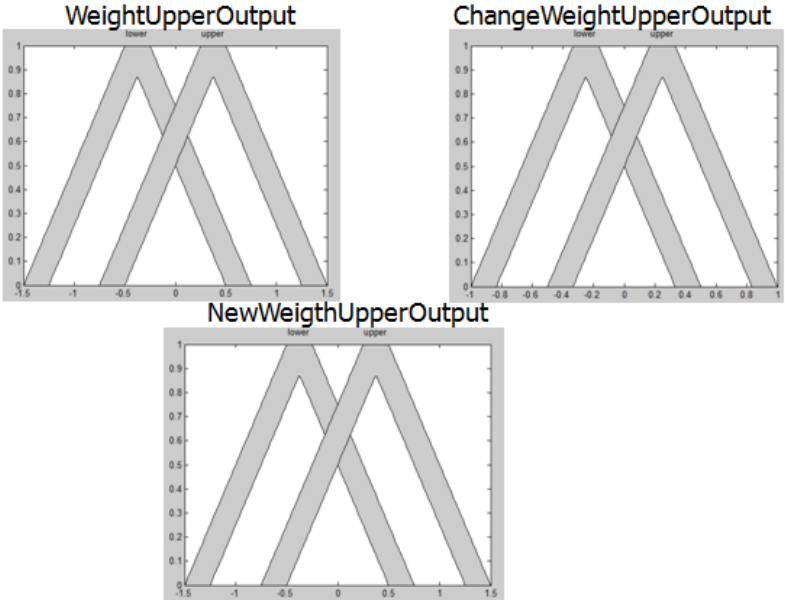


Fig. 21 Inputs and Output for the type-2 fuzzy inference system for the output layer in the second network

The input of the current weight for the type-2 inference system for the hidden layer in the second neural network consists of two triangular membership functions with range of -1.5 to 1.5. The input of change of the weight consists of two triangular membership functions with a range of -1 to 1. The output of the new weight consists of two triangular membership functions with a range of -1.5 to 1.5 (see Fig. 21).

We used six rules for the type-2 fuzzy inference system for the output layer, for the four combinations of two membership functions and we added two rules for when the change of weight is null (see Fig. 22).

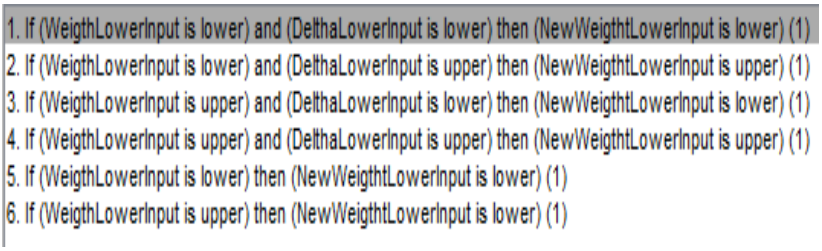


Fig. 22 Rules for the type-2 fuzzy inference system used in hidden and output layer for the first and second neural network

5 Simulation Results

The obtained results for the first experiment without optimizing the neural network and type-2 fuzzy systems are shown on Table 1 and Fig. 23, which means that all parameters of the neural network and type-2 fuzzy systems are established empirically. The best prediction error is of 0.0788.

Table 1 Results for the ensemble neural network for series Mackey-Glass

No.	Epoch	Network error	Time	Prediction error
E1	100	0.000000001	00:01:09	0.0788
E2	100	0.000000001	00:02:11	0.0905
E3	100	0.000000001	00:02:12	0.0879
E4	100	0.000000001	00:01:14	0.0822
E5	100	0.000000001	00:01:13	0.0924
E6	100	0.000000001	00:02:13	0.0925
E7	100	0.000000001	00:01:08	0.0822
E8	100	0.000000001	00:01:09	0.0924
E9	100	0.000000001	00:01:07	0.0826
E10	100	0.000000001	00:01:07	0.0879

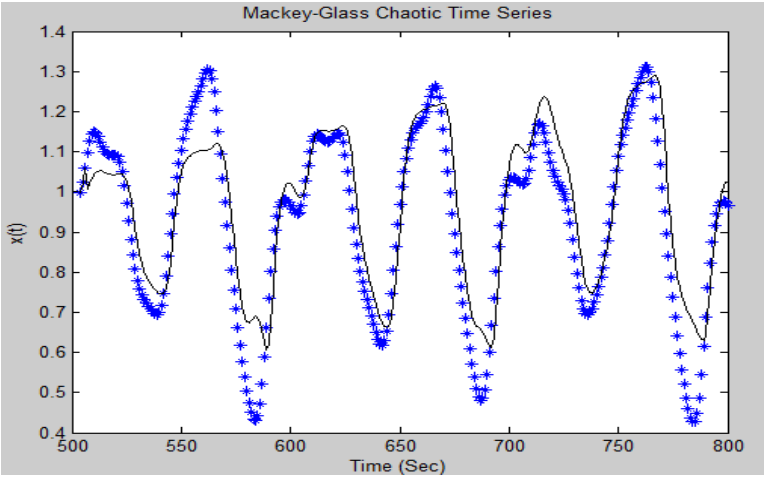


Fig. 23 Plot of real data against prediction data of the Mackey-Glass time series for the ensemble neural network with type-2 fuzzy weights

The population of the genetic algorithm (GA) consists of forty individuals to perform the search. The individuals used in the GA are of binary type and with a size of 81 gens. The estimated number of generations used for the GA to obtain a good optimization is of 20 generations, in which the GA performed operations to change the gens of the individuals and obtain different results in each generation.

To assign a fitness to the individuals of the GA we used ranking and stochastic universal sampling selection to find the individuals to which the evolutionary operations are applied to obtain new individuals for the next generation.

The evolutionary operations consist of single point crossover, and 0.0086 of mutation that are applied in the selected individuals for evolve in a new individual.

The parameters used to optimize the ensemble neural network are described in Table 2**:**

Table 2 Parameters of the genetic algorithm used for optimization of the ensemble neural network

Individuals	40
Gens	81 (binary)
Generations	20
Assign Fitness	Ranking
Selection	Stochastic Universal Sampling
Crossover	Single-Point
Mutation	0.0086
Individuals	40

The individual for this genetic algorithm is binary with a size of 81 gens.

Each individual considered in the genetic algorithm describes the membership functions for the inputs and outputs of the type-2 fuzzy inference system used in the hidden and output layer of the neural network. Also, each individual describes the numbers of neurons in the hidden layer for each neural network of the ensemble.

The objective function used obtains the fitness for each individual (ObjVal) in the genetic algorithm as the sum of the prediction error (errProm) divided by total of data of Mackey-Glass time series (Total) used in this experiment for test (297 in this experiment).

$$ObjVal = \frac{errProm}{Total} \tag{10}$$

The ensemble neural network architecture obtained with the genetic algorithm consists of the following: the first network with 27 neurons in the hidden layer and 1 neuron in the output layer; the second network with 31 neurons in the hidden layer and 1 neuron in the output layer; and the third network with 30 neurons in the hidden layer and 1 neuron in the output layer (see Fig. 24).

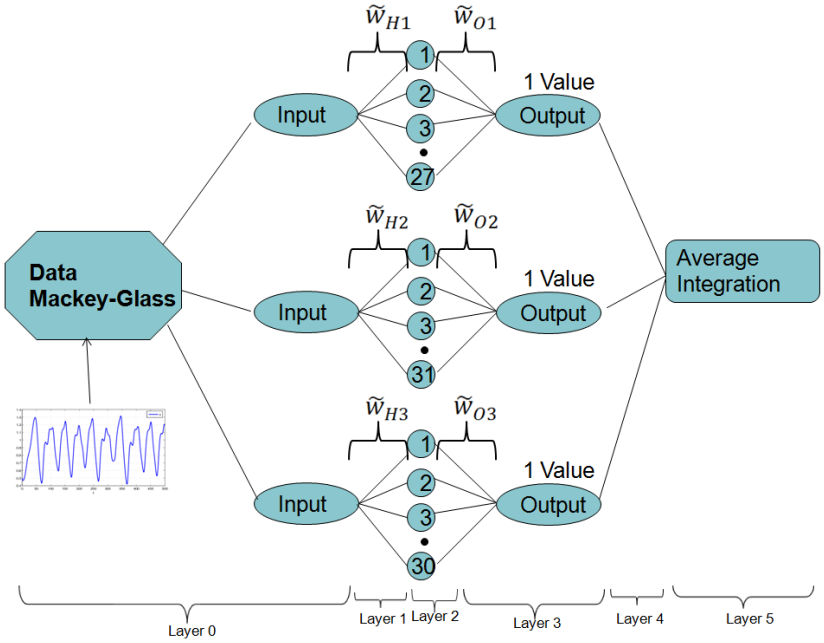


Fig. 24 Ensemble neural network architecture with type-2 fuzzy weights obtained with the genetic algorithm

The obtained results of the GA optimizing the ensemble neural network are shown on Table 3 and Fig. 25. The best error is of 0.0518 optimizing the numbers of neurons and type-2 fuzzy systems.

Table 3 Results for the optimized ensemble neural network for the Mackey-Glass time series

No.	Prediction error
E1	0.0518
E2	0.0611
E3	0.0787
E4	0.0715
E5	0.0655
E6	0.0614
E7	0.0724
E8	0.0712
E9	0.0724

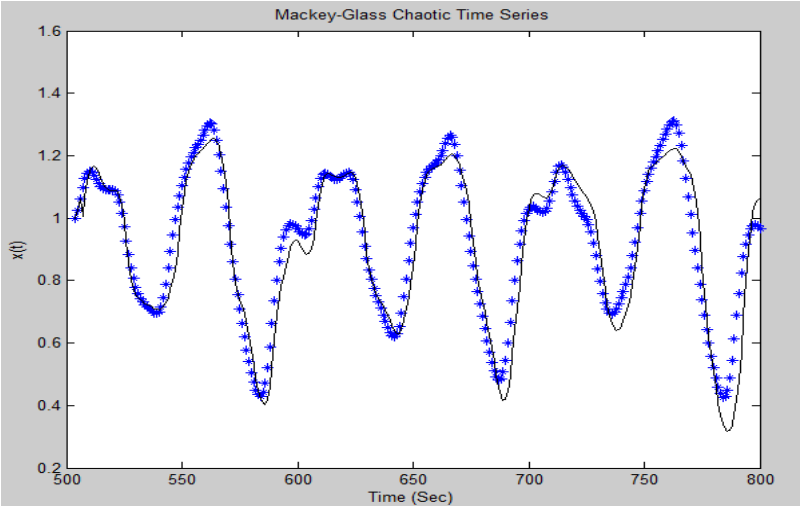


Fig. 25 Plot of real data against prediction data of the Mackey-Glass time series for the ensemble neural network with optimized type-2 fuzzy weights

We obtained 2 similar type-2 fuzzy systems in each neural network.

The first type-2 fuzzy system obtained consist of two inputs: the weight in the current epoch and the change of the weight for the next epoch, and one output: the new weight for the next epoch (see Fig. 26).

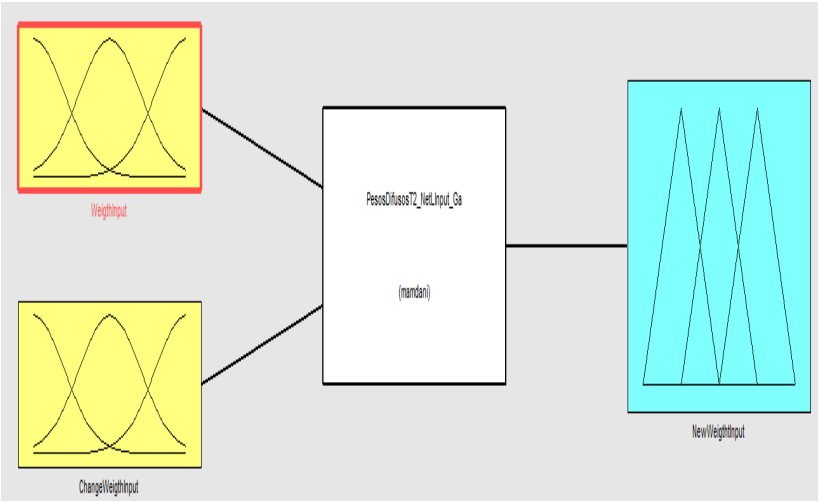


Fig. 26 Structure of the type-2 fuzzy inference system obtained in the hidden layer

The input of the current weight consists of two triangular membership functions with range of -1 to 1. The input of change of the weight consists of two triangular membership functions with range of -0.1 to 0.1. The output of the new weight consists of two triangular membership functions with range of -1 to 1 (see Fig. 27).

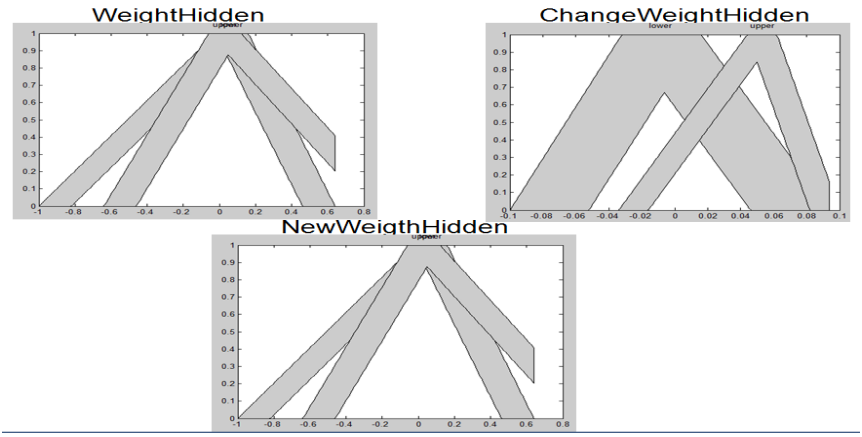


Fig. 27 Inputs and outputs of the type-2 fuzzy inference system for the hidden layer

We used six rules for the type-2 fuzzy inference system for the hidden layer, the four combination of two membership functions and we added two rules when the change of weight is null (see Fig. 28).

1. If (WeigthInput is lower) and (ChangeWeigthInput is lower) then (NewWeigthtInput is lower) (1)
2. If (WeigthInput is lower) and (ChangeWeigthInput is upper) then (NewWeigthtInput is lower) (1)
3. If (WeigthInput is upper) and (ChangeWeigthInput is lower) then (NewWeigthtInput is upper) (1)
4. If (WeigthInput is upper) and (ChangeWeigthInput is upper) then (NewWeigthtInput is upper) (1)
5. If (WeigthInput is lower) then (NewWeigthtInput is lower) (1)
6. If (WeigthInput is upper) then (NewWeigthtInput is upper) (1)

Fig. 28 Structure of the used type-2 fuzzy inference system in the hidden layer

The second type-2 fuzzy system obtained consist of two inputs: the weight in the current epoch and the change of the weight for the next epoch, and one output: the new weight for the next epoch (see Fig. 29).

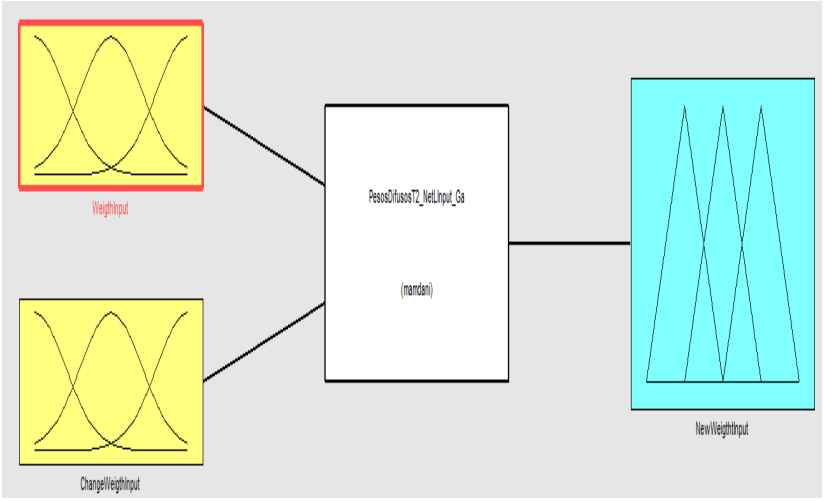


Fig. 29 Structure of the type-2 fuzzy inference system obtained in the output layer

The input of the current weight consists of two triangular membership functions with range of -0.01 to 0.01. The input of change of the weight consists of two triangular membership functions with range of -0.1 to 0.1. The output of the new weight consists of two triangular membership functions with range of -0.01 to 0.01 (see Fig. 30).

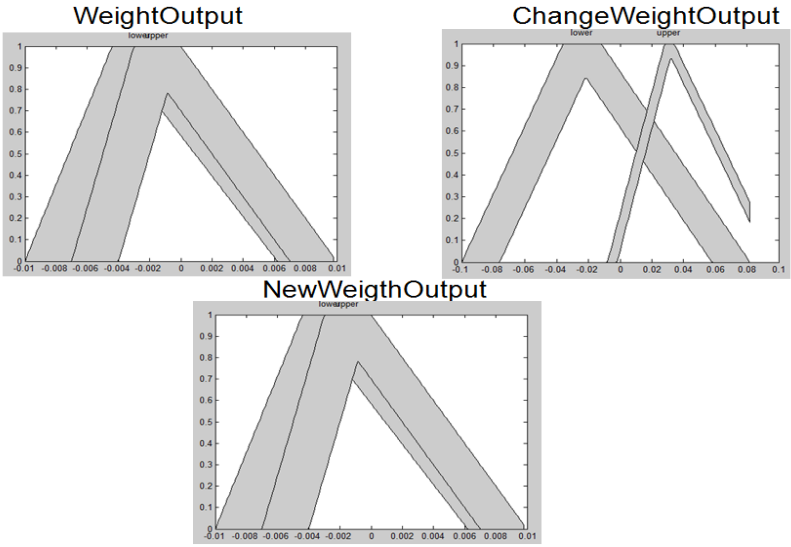


Fig. 30 Inputs and outputs of the type-2 fuzzy inference system for the output layer

We used six rules for the type-2 fuzzy inference system for the output layer, the four combination of two membership functions and we added two rules when the change of weight is null (see Fig. 31).

1. If (WeigthInput is lower) and (ChangeWeigthInput is lower) then (NewWeigthtInput is lower) (1)
2. If (WeigthInput is lower) and (ChangeWeigthInput is upper) then (NewWeigthtInput is lower) (1)
3. If (WeigthInput is upper) and (ChangeWeigthInput is lower) then (NewWeigthtInput is upper) (1)
4. If (WeigthInput is upper) and (ChangeWeigthInput is upper) then (NewWeigthtInput is upper) (1)
5. If (WeigthInput is lower) then (NewWeigthtInput is lower) (1)
6. If (WeigthInput is upper) then (NewWeigthtInput is upper) (1)

Fig. 31 Rules of the type-2 fuzzy inference systems used in the output layer

The obtained results for the second experiment 5 tests were performed with the proposed modular neural network under the same conditions and the same database of the iris; in Table 4 we show the obtained results:

Table 4 Parameters of the genetic algorithm used for optimization the ensemble neural network

Experiment	Epoch	Error	Time	Total Recognition
T1	12	0.01	72 min.	76.66 % (23/30)
T2	12	0.01	72 min.	70 % (21/30)
T3	12	0.01	72 min.	73.33 % (23/30)
T4	12	0.01	72 min.	83.33 % (25/30)
T5	12	0.01	71 min.	70 % (21/30)

The best result is a total recognition of 25 out of 30 images of iris of 10 persons; giving a recognition rate of 83.33 %.

The architecture neural network works with 12 epoch of iteration and 0.01 error of network and training algorithm of that updates weights and bias values according to the resilient backpropagation algorithm (trainrp), with this parameters the time of execution for the neural networks was of 72 minutes

The average of the 5 tests is a percentage of recognition of 74.66.

6 Conclusions

In the first experiment an ensemble neural network learning method with type-2 fuzzy weights was optimized with a genetic algorithm. The result with the ensemble neural network with type-2 fuzzy weights optimized for the Mackey-Glass time series is a prediction error of 0.0518. The architecture for the optimized ensemble neural network is: the first network with 30 neurons in the hidden layer and 1 neuron in the output layer; the second network with 29 neurons in the hidden layer and 1 neuron in the output layer; and the third network with 26 neurons in the hidden layer and 1 neuron in the output layer.

The result of prediction error of 0.0518 for the Mackey-Glass time series is good considering that the number of GA generations was relatively small.

In the second experiment we used two neural networks with type-2 fuzzy weight. The result with the neural network with type-2 fuzzy weights for the human iris biometrics measure is a percentage of recognition of 83.33 %. The architecture for the two neural networks: the first network with 120 neurons in the hidden layer and 10 neuron in the output layer, and the second network with 120 neurons in the hidden layer and 10 neuron in the output layer.

The results obtained in these experiments showed that the type-2 fuzzy weights worked good in the two areas of research: prediction time series and pattern recognition.

References

1. Abiyev, R.H.: A Type-2 Fuzzy Wavelet Neural Network for Time Series Prediction. In: García-Pedrajas, N., Herrera, F., Fyfe, C., Benítez, J.M., Ali, M. (eds.) IEA/AIE 2010, Part III. LNCS, vol. 6098, pp. 518–527. Springer, Heidelberg (2010)

2. Barbounis, T.G., Theocharis, J.B.: Locally Recurrent Neural Networks for Wind Speed Prediction using Spatial Correlation. Information Sciences 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 Networks 8(6), 989–998 (1995)
5. Castillo, O., Melin, P.: A review on the design and optimization of interval type-2 fuzzy controllers. Applied Soft Computing 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. Information Sciences 179(13), 2175–2193 (2009)
7. Castro, J.R., 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: Castillo, O., Kacprzyk, J., Pedrycz, W. (eds.) Soft Computing for Intell. Control and Mob. Robot. SCI, vol. 318, pp. 269–285. Springer, Heidelberg (2010)
8. Cazorla, M., Escolano, F.: Two Bayesian Methods for Junction Detection. IEEE Transaction on Image Processing 12(3), 317–327 (2003)
9. De Wilde, O.: The Magnitude of the Diagonal Elements in Neural Networks. Neural Networks 10(3), 499–504 (1997)
10. Draghici, S.: On the Capabilities of Neural Networks using Limited Precision Weights. Neural Networks 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. Computer Journal 7, 149–154 (1964)
13. Gedeon, T.: Additive Neural Networks and Periodic Patterns. Neural Networks 12(4-5), 617–626 (1999)
14. Hagan, M.T., Demuth, H.B., Beale, M.H.: Neural Network Design, p. 736. PWS Publishing, Boston (1996)
15. 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)
16. Haupt, R., Haupt, S.: Practical Genetic Algorithms, p. 272. John Wiley and Sons, Inc., Hoboken (2004)
17. 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)
18. 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. 1650–1655 (1993)
19. 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 (1997)
20. Kamarthi, S., Pittner, S.: Accelerating Neural Network Training using Weight Extrapolations. Neural Networks 12(9), 1285–1299 (1999)

21. Karnik, N., Mendel, J.: Applications of Type-2 Fuzzy Logic Systems to Forecasting of Time-Series. Information Sciences 120(1-4), 89–111 (1999)
22. Martinez, G., Melin, P., Bravo, D., Gonzalez, F., Gonzalez, M.: Modular Neural Networks and Fuzzy Sugeno Integral for Face and Fingerprint Recognition. In: Abraham, A., de Baets, B., Köppen, M., Nickolay, B. (eds.) Applied Soft Computing Technologies: The Challenge of Complexity. ASC, vol. 34, pp. 603–618. Springer, Heidelberg (2006)
23. Melin, P.: Modular Neural Networks and Type-2 Fuzzy Systems for Pattern Recognition, pp. 1–204. Springer (2012)
24. Meltser, M., Shoham, M., Manevitz, L.: Approximating Functions by Neural Networks: A Constructive Solution in the Uniform Norm. Neural Networks 9(6), 965–978 (1996)
25. Moller, M.F.: A Scaled Conjugate Gradient Algorithm for Fast Supervised Learning. Neural Networks 6, 525–533 (1993)
26. Monirul Islam, M.D., Murase, K.: A New Algorithm to Design Compact Two-Hidden-Layer Artificial Neural Networks. Neural Networks 14(9), 1265–1278 (2001)
27. Neville, R.S., Eldridge, S.: Transformations of Sigma–Pi Nets: Obtaining Reflected Functions by Reflecting Weight Matrices. Neural Networks 15(3), 375–393 (2002)
28. Phansalkar, V.V., Sastry, P.S.: Analysis of the Back-Propagation Algorithm with Momentum. IEEE Transactions on Neural Networks 5(3), 505–506 (1994)
29. Powell, M.J.D.: Restart Procedures for the Conjugate Gradient Method. Mathematical Programming 12, 241–254 (1977)
30. Pulido, M., Melin, P., Castillo, O.: Genetic Optimization of Ensemble Neural Networks for Complex Time Series Prediction. IJCNN, 202–206 (2011)
31. 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: Castillo, O., Melin, P., Kacprzyk, J., Pedrycz, W. (eds.) Soft Computing for Hybrid Intel. Systems. SCI, vol. 154, pp. 171–186. Springer, Heidelberg (2008)
32. 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: Melin, P., Kacprzyk, J., Pedrycz, W. (eds.) Soft Comp. for Recogn. Based on Biometrics. SCI, vol. 312, pp. 85–102. Springer, Heidelberg (2010)
33. Sepúlveda, R., Castillo, O., Melin, P., Montiel, O.: An Efficient Computational Method to Implement Type-2 Fuzzy Logic in Control Applications. In: Melin, P., Castillo, O., Ramírez, E.G., Kacprzyk, J., Pedrycz, W. (eds.) Anal. and Des. of Intel. Sys. using SC Tech. ASC, vol. 41, pp. 45–52. Springer, Heidelberg (2007)
34. Valdez, F., Melin, P., Parra, H.: Parallel Genetic Algorithms for Optimization of Modular Neural Networks in Pattern Recognition. In: IJCNN, pp. 314–319 (2011)
35. Yam, J., Chow, T.: A Weight Initialization Method for Improving Training Speed in Feedforward Neural Network. Neurocomputing 30(1-4), 219–232 (2000)
36. Yeung, D., Chan, P., Ng, W.: Radial Basis Function Network Learning using Localized Generalization Error Bound. Information Sciences 179(19), 3199–3217 (2009)

Brain Computer Interface Development Based on Recurrent Neural Networks and ANFIS Systems

Emanuel Morales-Flores[1], Juan Manuel Ramírez-Cortés[1], Pilar Gómez-Gil[2], and Vicente Alarcón-Aquino[3]

[1] Department of Electronics, National Institute of Astrophysics, Optics and Electronics, Tonantzintla, Puebla, Mexico
[2] Department of Computer Science, National Institute of Astrophysics, Optics and Electronics, Tonantzintla, Puebla, Mexico
[3] Department of Electronics, University of the Americas, Cholula, Puebla, Mexico

Abstract. Brain Computer Interfaces (BCI) is the generic denomination of systems aiming to establish communication between a human being and an automated system, based on the electric brain signals detected through a variety of modalities. Among these, electroencephalographic signals (EEG) have received considerable attention due to several factors arising on practical scenarios, such as noninvasiveness, portability, and relative cost, without lost on accuracy and generalization. In this chapter we discuss the characteristics of a typical phenomenon associated to motor imagery and mental tasks experiments, known as event related synchronization and desynchronization (ERD/ERS), as well as its energy distribution in the time-frequency space. The typical behavior of ERD/ERS phenomenon has led proposal of different approaches oriented to the solution of the identification problem. In this work, an architecture based on adaptive neuro-fuzzy inference systems (ANFIS) assembled to a recurrent neural network, applied to the problem of mental tasks temporal classification, is presented. The electroencephalographic signals (EEG) are pre-processed through band-pass filtering in order to separate the set of energy signals in alpha and beta bands. The energy in each band is represented by fuzzy sets obtained through an ANFIS system, and the temporal sequence corresponding to the combination to be detected, associated to the specific mental task, is entered into a recurrent neural network. Experimentation using EEG signals corresponding to mental tasks exercises, obtained from a database available to the international community for research purposes, is reported. Two recurrent neural networks are used for comparison purposes: Elman network, and a fully connected recurrent neural network (FCRNN) trained by RTRL-EKF (real time recurrent learning – extended Kalman filter). A classification rate of 88.12 % in average was obtained through the FCRNN during the generalization stage.

P. Melin, O. Castillo (Eds.): *Soft Computing Appli. in Optimization,* STUDFUZZ 294, pp. 215–236.
DOI: 10.1007/978-3-642-35323-9_9

1 Introduction

Brain Computer Interfaces are systems aiming to translate the electrical brain signals generated by a human being as a results of some thoughts, in commands able to perform some control actions in computerized mechanisms. In other words BCIs measure brain activity, process it, and produce control signals that reflect the user's intent. Brain activity produces several physical phenomena which can be measured using a variety of sensing equipment. Among these phenomena, which can be of significant relevance for BCI development, are electrical potentials and hemodynamic measurements. Electrical potential measurements include action and field potentials which can be sensed through invasive methods, such as electro-corticography, and non-invasive, such as electroencephalography and magneto-encephalography techniques. Hemodynamic measurements include functional magnetic resonance imaging (fMRI), positron emission tomography (PET), and functional near-infrared brain monitoring (fNIRS). Among these, electroencephalographic signals (EEG) have received considerable attention due to several factors arising on practical scenarios, such as noninvasiveness, cost effectiveness, portability, ease of acquisition, and time resolution, which are ideal attributes for the development of practical brain computer interface applications. There are three main stages which can be distinguished in a BCI system: detection of the neural signals from the brain, an algorithm for decoding these signals, and a methodology for mapping decoded signals into some predefined activities. The general scheme of a BCI is shown in Fig. 1.

In recent years, there has been a growing interest in the research community on signal processing techniques oriented to solve the multiple challenges involved in BCI applications [1-3]. An important motivation to develop BCI systems, among

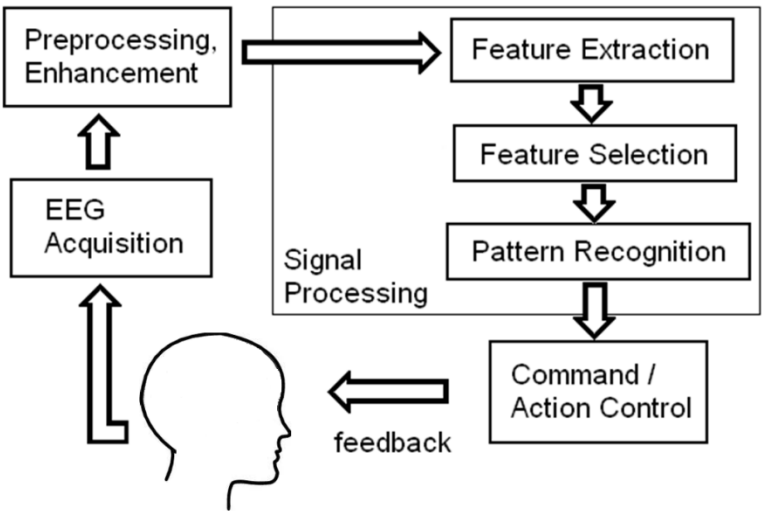


Fig. 1 General scheme of a Brain Computer Interface system

some others, would be to allow an individual with motor disabilities to have control over specialized devices such as computers, speech synthesizers, assistive appliances or neural prostheses.

A dramatic relevance arises when thinking about patients with severe motor disabilities such as locked-in syndrome, which can be caused by amyotrophic lateral sclerosis, high-level spinal cord injury or brain stem stroke. BCIs would increase an individual's independence, leading to an improved quality of life and reduced social costs. Electroencephalography (EEG) refers to recording electrical activity from the scalp with electrodes. A BCI based on EEG analyzes ongoing electric brain activity for brain patterns that originate from specific brain areas. To get consistent recordings from specific regions of the head, scientists rely on a standard system for accurately placing electrodes, which is called the International 10–20 System [4], generally used in clinical EEG recording and EEG research as well as BCI field. The name 10–20 indicates that the most commonly used electrodes are positioned 10, 20, 20, 20, 20, and 10% of the total naison-inion distance. Fig. 2 shows the electrode positions and denominations used in the international 10-20 system.

Measuring brain activity effectively is a critical step for brain–computer communication. However, measuring activity is not enough, because a BCI can only detect and classify specific patterns of activity in the ongoing brain signals that are associated with specific events. What the BCI user makes to produce these patterns is determined by the neurological mechanisms or processes that BCI system employs.

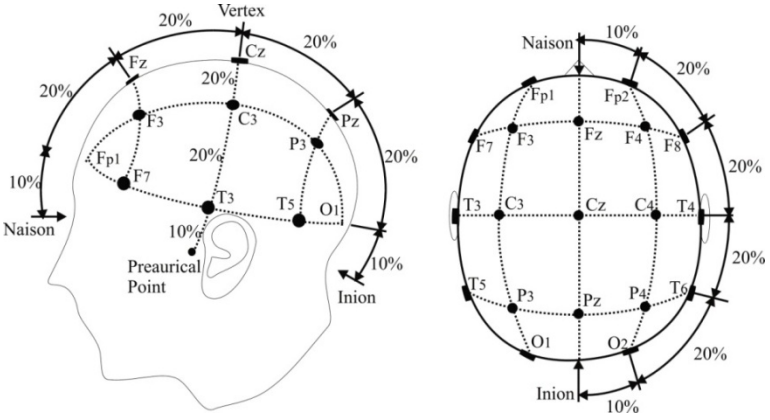


Fig. 2 EEG electrodes international 10-20 system

Current research on BCI systems distinguishes seven main categories according to the neurological mechanisms or processes involved: sensorimotor activity [5,6], P300 [7,8], visual evoked potentials [9,10], slow cortical potentials [11], activity of neural cell and response to mental tasks [12], as well as multiple neuro-mechanisms, which use a combination of two or more of the previous (see [2] for

a review). Each category constitutes a paradigm which can be used for developing BCI systems in practical scenarios. P300 evoked potentials occur with latency around 300 milliseconds in response to target stimuli that occur unexpectedly. In a P300 controlled experiment, subjects are usually instructed to respond in a specific way to some stimuli, which can be auditory, visual, or somatosensory. P300 signals come from the central-parietal region of the brain and can be found more or less throughout the EEG on a number of channels. The P300 is an important signature of cognitive processes such as attention and working memory and an important clue in the field of neurology to study mental disorders and other psychological dysfunctions [8]. Another neurological mechanism widely studied for developing BCI systems is motor imagery (MI), which is obtained from the sensory motor brain activity. In general, two types of patterns are usually present in this mechanism: event related potentials (ERP), detected as energy changes in α (8-13 Hz), and β (14-20 Hz) bands generated when a voluntary movement is performed, and movement related potentials (MRP), which are low frequency patterns that initially appear between 1–1.5 s before the corresponding movement. In the first case, the event related potentials consist, in general terms, in decrements or increments of the energy on the ongoing EEG signal at certain frequency bands, which are described in the literature as the ERD/ERS phenomenon (Event Related Desynchronization and Synchronization) [13,14]. A crucial issue is to successfully estimate and translate the ERD/ERS phenomenon into a meaningful feature vector which can be used as input to some pattern recognition scheme. The analysis should be able to capture the spectral dynamic of the signal contained in the temporal evolution of the involved spectral bands. Several feature extraction techniques have been used for that purposes, such as: amplitude values of EEG [15], band power [16], power spectral density [17,18], auto-regressive (AR) and adaptive auto-regressive models (AAR) [19], windowed Fourier analysis, cross correlation, and some others. As these ERPs are locked in time but not in phase and they are highly non-stationary [20], the detection of these patterns turns into a difficult task in which some approaches oriented to follow the time evolution of the signals, such as time series prediction, and recurrent neural networks, could provide adequate results.

Another neurological mechanism in which ERD/ERS phenomenon is also present is the neural activity obtained in response to mental tasks. Mental task-based BCI systems have captured the attention of the research community, in part due to their independence of additional interfaces such as the screen of alphanumeric characters used in VEP, or the arrows and symbols used in motor imagery experiments, as well as the relative flexibility of the user to carry out some mental tasks at his /her own will. Several feature extraction methods for mental task-based BCI design have been reported, most of them based on parametric, such as autoregressive or adaptive models [21], non-parametric models based on several schemes of spectral analysis such as Wavelet transform or Stockwell transform [22,23], or fuzzy sets [24]. In this sense, it has been shown that information contained in spectral bands α (8-13 Hz), β (14-20 Hz), γ (24-37 Hz), or even in higher frequencies [25], can be used to detect neural activity directly related to specific mental tasks. Time-frequency analysis can be carried

out using different approaches such as Wavelet analysis [22], filter bank [26], empirical mode decomposition [27], and others. Those approaches reflect only the estimated power across a range of frequencies. In a number of reported works, non-linear classifiers such as neural network and support vector machine algorithms are used [28]. Recently, there have been several studies oriented to capture temporal behavior through predictive schemes and recurrent neural networks with good results, which encourage further research in that direction [29-30].

To achieve the goal of translating brain activity into commands for computers there are two main approximations: regression and classification algorithms. Using classification algorithms is the most popular approach to identify patterns of brain activity. Most brain patterns used to control BCI are related to time variations of EEG in specific frequency bands. The time course of EEG signals has to be taken into account during feature extraction and one alternative is using a dynamical classifier. To obtain temporal information it is necessary to extract features from several time segments in order to build a temporal sequence. In this work we present a temporal classification approach on a two-state mental task experiment applying, for comparison purposes, two recurrent neural networks: Elman and Fully Connected Recurrent Neural Network (FCRNN). The proposed scheme performs the feature extraction based on an Adaptive Neuro-fuzzy Inference System (ANFIS), previous to the temporal classification stage.

The rest of the chapter is organized as follows: Section 2 describes theory related to ANFIS. Section 3 presents mathematical background associated to recurrent neural networks. Section 4 describes the proposed methodology on temporal classification of the mental task experiment. Section 5 presents and analyzes the obtained results. Section 6 presents some concluding remarks, perspectives, and future direction of this research oriented to the implementation of a BCI system.

2 Adaptive Neuro-Fuzzy Inference System (ANFIS)

Adaptive Neuro Fuzzy Inference Systems (ANFIS) combine the learning capabilities of neural networks with the approximate reasoning of fuzzy inference algorithms. Embedding a fuzzy inference system in the structure of a neural network has the benefit of using known training methods to find the parameters of a fuzzy system. Specifically, ANFIS uses a hybrid learning algorithm to identify the membership function parameters of Takagi-Sugeno type fuzzy inference systems. The task of the learning algorithm for this architecture is to tune all the modifiable parameters defining the fuzzy partitions and making the ANFIS output match the training data. In this work, the ANFIS model included in the MATLAB toolbox has been used for experimentation purposes. A combination of least-squares and backpropagation gradient descent methods is used for training the FIS membership function parameters to model a given set of input/output data through a multilayer neural network. ANFIS systems have been recently used for optimization, modeling, prediction, and signal detection, among others [31,32]. The ANFIS architecture (type-3 ANFIS) is shown in Fig. 3.

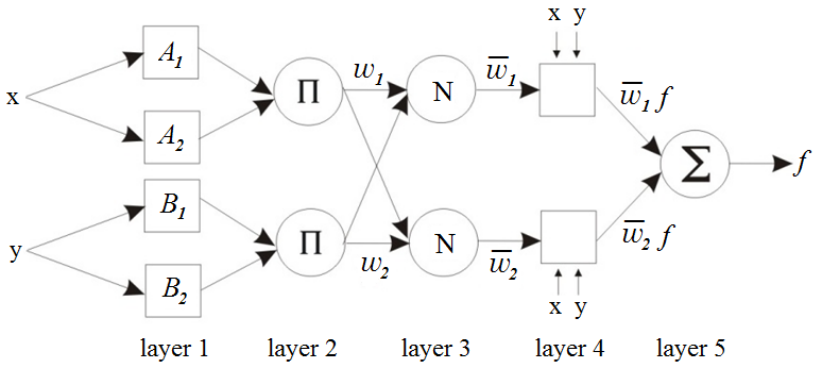


Fig. 3 ANFIS architecture

In this figure x and y are inputs to the node i in layer 1. A_i and B_i are linguistic labels e.g. (small, medium, large, etc.). In other words, the output of each node is the membership function of A_i and B_i, and specifies the degree to which the given x or y satisfies the quantifier A_i and B_i respectively. The output of each node in this layer is described as follows:

$$O_i^1 = \mu_{A_i}(x)$$

Every node in layer 2 is a circle node labeled which multiplies the incoming signals and sends the product out.

$$\omega_i = \mu_{A_i}(x) \times \mu_{B_i}(x)$$

In layer 3 each node is a circle node labeled N. The ith node calculates the ratio of the ith rule's firing strength to the sum of all rules' firing strengths:

$$\bar{\omega} = \frac{\omega_i}{\omega_1 + \omega_2}, i = 1, 2$$

Every node in layer 4 is a square node that performs the following function:

$$O_i^4 = \bar{\omega}_i f_i = \bar{\omega}_i (p_i x + q_i y + r_i) \quad ,$$

where $\{p_i, q_i, r_i\}$ is the parameter set.

The single node in the 5 layer is a circle node labeled Σ that computes the overall output as the summation of all incoming signals

$$O_i^1 = \sum_i \bar{\omega}_i f_i = \frac{\sum_i \omega_i f_i}{\sum_i \omega_i}$$

The architecture presented is functionally equivalent to a type-3 fuzzy inference system. For detailed information see reference [33].

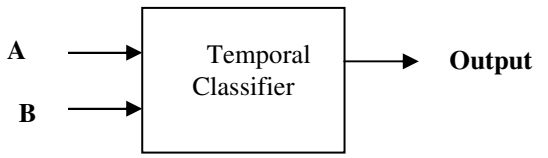
3 Neural Network Classifiers

Nowadays, artificial neural networks are a popular tool to tackle complex classification problems. Specifically, the ability of recurrent neural networks (RNN) to model nonlinear dynamical systems has been widely proved [34]. Therefore, it is fairly common to use RNN for several kinds of temporal information processing, as in prediction, control systems and temporal classification systems [35].

Next, we present a brief description of the problem of temporal classification and the solution applied in this research using two architectures of RNN to build the temporal classifier required for mental task-based BCI systems.

3.1 Temporal Classification

Temporal classification refers to the assignation of a class, based on features obtained in different time periods. Such features are represented as vectors forming a temporal sequence of components. Temporal classification is a difficult task because, in order to obtain the correct class, it is mandatory to consider not only the values of the features but also the order in which they appear in a specific time period. The definition of the size of time that must be considered in order to get the right classification is also a challenge. Fig. 4 illustrates a simple temporal classification problem. Suppose that we want to identify if the sequence {1,2} is sensed in input A when the sequence {2,1} is sensed in input B. If so, the expected classification outcome is "yes", otherwise it is "No". The table therein Fig. 4 illustrates the desired outputs of such classifier in the first 10 time periods.



time	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
Input A	1	1	2	2	1	2	1	1	2	1
Input B	1	2	1	2	2	2	1	2	1	2
Output	No	No	Yes	No	No	No	No	No	Yes	No

Fig. 4 A simple temporal classification problem

In this example, the classifier must be able to "remember" the last two inputs, in order to identify the sequences correctly. Looking this table, it is fairly easy to figure out that the sequences defining the involved classes have a size of two.

However, this is not the case for more complicated problems as the one presented in this research, in which a human mental state has to be identified by a sequence of features occurring in a EEG. For such cases the classifier would have to automatically model a dynamics memorizing the feature sequences using the right size of past events. In other words, time has to be implicitly represented in the model. In this research a temporal classifier is used as the last component of the system classifying mental tasks (see Fig. 6). The classifier has to find out if the involved mental task occurs or it does not, that is, it works as a binary classifier.

3.2 Adaptive Temporal Classifiers

The building of a classifier able to label sequences requires several steps. The most important decisions to resolve during its design are: the definition of the structure of a feature vector representing the information of the sequence, the mathematical model used for the classifier and the training strategy used in such model. Section 4 describes how the structure of the feature vector for the classifier of mental tasks was built in this research. With respect to the mathematical model of the classifier, we chose to use RNN for two reasons: first, RNN are able to build internal representations involving time and second, most recurrent neural architectures are able to model chaos [36]. This last reason refers to the fact that the dynamics in an EEG is chaotic, according to several authors (for example see [37]).

Regarding to the selection of a right RNN and training algorithm, there are many choices when they are used for building temporal classifiers. The most versatile models are the ones proposed by Jordan [38], Elman [39], Werbos [40] and Williams and Zipser [41]. Other works have used more sophisticated structures, for example [42].

For the results presented here, we built and tested the performance of two classifiers using two types of recurrent neural networks: a Simple Recurrent Network (SRN), also known as "Elman network" [39] and a fully connected recurrent neural network (FCRNN) with external inputs, similar to the one described in [40,47]. SRN was trained using the algorithm "Back Propagation through time" (BPTT) [40] and FCRNN was trained with the algorithm "Real Time Recurrent Learning – Extended Kalman filter" (RTRL-EKF) [43,44] using the implementation proposed in [48]. These architectures and algorithms are briefly described next.

3.3 Simple Recurrent Network (SRN or Elman Network)

Time can be represented in several ways in recurrent neural networks. In a SRN, time is implicitly represented using a context layer. This model was introduced by Elman [39], which in spite of being rather simple, is able to memorize previous states of a sequence. SRN architecture has 4 layers: an input layer, a hidden layer, an output layer and a context layer. (see Fig. 5). The representation of past events is achieved because nodes in the context layer memorize the outputs of nodes in

hidden layer coming from a previous time. This context layer is able to create a map of some temporal properties of the system.

In general, the state-space model of a RNN can be described by the following equations [44]:

$$\mathbf{x}_{n+1} = \mathbf{a}(\mathbf{x}_n, \mathbf{u}_n) \tag{1}$$

$$\mathbf{y}_n = \mathbf{B}\mathbf{x}_\mathbf{n} \tag{2}$$

where:

$\mathbf{y}_n$	represents the output of the system (all neurons in the network),
$\mathbf{u}_n = \{u_n, u_{n-1} \ldots u_{n-q+1}\}$	is a vector of the exogenous inputs in different steps,
$\mathbf{x}_\mathbf{n}$	is the output of a bank of q unit-time delays, q being the number of nodes in the input layer.
$\mathbf{a}(\cdot,\cdot)$	is a nonlinear function characterizing the hidden layer.
B	is the matrix of synaptic weights characterizing the output layer.

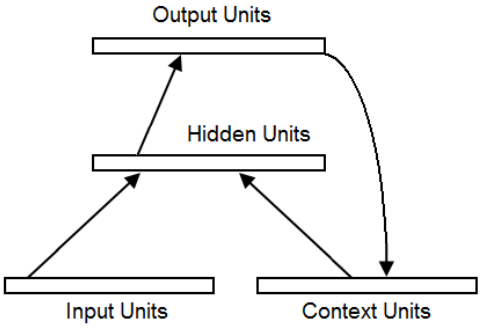


Fig. 5 The Simple Recurrent Network [39]

Notice that in this model, the hidden layer is non-linear (equation 1) and the output layer is linear (equation 2). A SRN is a special case of this model, where the connection weights and the output layer may also be non-linear. In the results reported here, a hyperbolic tangent sigmoid transfer function was used for the nodes in hidden layer ('tansig' Matlab function) and a logarithmic sigmoid transfer function ('logsig' Matlab function) was used for the output layers.

SRN may be trained in different ways. For the experiments reported in this chapter, we used a gradient descent back propagation algorithm with adaptive learning rate. Function 'calcgbtt', provided by the neural network toolbox of Matlab V6.0. was used as the gradient function, which calculates the bias and

weight performance gradients using the back-propagation through time algorithm (BPTT) [40]. BPTT is a supervised learning algorithm originally proposed by Werbos [46] and independently discover by Rumelhart and collaborators [47], that attempts to minimize the output error of the network obtained over a period of time. This error is calculated as:

$$\mathbf{E} = \sum_{t=1}^{T} (\mathbf{D}_{t,\mathbf{n}} - \mathbf{y}_{t,\mathbf{n}})^2 \tag{3}$$

where $\mathbf{D}_{t,\mathbf{n}}$ is the desired output of the neurons in the network where an output is required at time t, and T is the size of the sequence being used to train the network. The core of back-propagation is a efficient method for the calculation of derivatives that allow to minimize the error described in equation 3. BPTT constructs a feed-forward network with identical behavior over a particular time interval that the involved RNN. The main drawback of BPTT is that it requires to use the complete training sequence for each training epoch in order to calculate the gradient. For a detailed explanation of BPTT see [44].

3.4 Fully-Connected Recurrent Neural Network

A fully-connected recurrent neural network with one input layer, one hidden layer and one output layer was also used in this work to build a temporal classifier. The term "fully connected" means that all neurons in the network are connected each other. The input layer is formed by neurons receiving an external input; the output layer is formed by nodes whose outputs are considered the output of the system; the training sequence contains the desired values for such outputs (corresponding class). As occurring with other layered neural network architectures, the number of neurons in the hidden layer depends upon the complexity of the problem and the appropriate number of them requires to be defined by experimentation.

As we explained before, there are several algorithms to train recurrent neural networks. BPTT has been very popular during many years, but currently it is known that very useful algorithms for training recurrent neural networks are based on Kalman Filtering (KF) [48]. KF is a common method to estimate unknown variables of a system based on the observations of measurements across time. KF is based on the idea that the involved dynamical system of the problem is hidden and can only be observed or measured through some time series (sequences). In KF the dependency among two consecutive states, measurements and the state process is assumed linear [49]. Therefore, an Extended Kalman Filtering (EKF) is required when nonlinear systems are involved, as in the case of recurrent neural networks. In EKF, a linearization around the current working point is applied before that standard KF is performed. EKF has been widely studied and applied using different strategies to train RNN, for example in [49-51]. It also has been combined with other algorithms, for example with “back-truncated propagation through time” [51] and with RTRL [43].

For the experiments presented in this research, we used a combination of the RTRL and KF proposed by [43]. The training algorithm RTRL contains two main steps (see [44,52]): gradient calculation and weights adjustments. RTRL is used to calculate the derivatives of the gradients and EKF is used for modifying the weights. According to [44], the state-space model of this network, when training, is defined by two models:

1) The system model, described by:

$$\mathbf{w}_{n+1} = \mathbf{w}_n + \omega_n \quad , \tag{4}$$

where:

$\mathbf{w}_n$ is the weight (state) vector

ω_n is a white Gaussian noise.

2) The measurement model, described by:

$$\mathbf{d}_n = \mathbf{b}(\mathbf{w}_n, \mathbf{v}_n, \mathbf{u}_n) + \mathbf{v}_n \quad , \tag{5}$$

where:

$\mathbf{d}_n$ is the desirable response o the system, playing the role of the "observable",

$\mathbf{v}_n$ represents the recurrent node activities inside the network,

$\mathbf{u}_n$ denotes the input signal to the network and

$\mathbf{v}_n$ is a vector denoting measurement noise corrupting $\mathbf{d}_n$.

EKF allows the estimation of the value of the correction in the state space model, updating weights as follows:

$$\hat{\mathbf{w}}_n = \hat{\mathbf{w}}_{n-1} + \mathbf{G}_n \boldsymbol{\alpha}_n \tag{6}$$

$$\boldsymbol{\alpha}_n = \mathbf{d}_n - \mathbf{b}(\hat{\mathbf{w}}_{n-1}, \mathbf{v}_n, \mathbf{u}_n) \quad , \tag{7}$$

where:

$\mathbf{G}_n$ is the Kalman gain, calculated using:

$$\mathbf{G}_n = \mathbf{P}_{n-1}\mathbf{B}_n^T[\mathbf{B}_n\mathbf{P}_{n-1}\mathbf{B}_n^T + \mathbf{Q}_{v,n}]^{-1} \tag{7}$$

$$\mathbf{P}_n = \mathbf{P}_{n-1} - \mathbf{G}_n\mathbf{B}_n\mathbf{P}_{n-1} + \mathbf{Q}_{\omega,n} \tag{8}$$

$\mathbf{B}_n$ is the Jacobian of the partial derivatives with respect to the state, that is, the weights, which is calculated using RTRL algorithm.

$\mathbf{Q}_{\omega,n}$ is the covariance matrix of the dynamic noise ω_n,

$\mathbf{P}_n$ is the prediction error covariance matrix, and

$\mathbf{Q}_{v,n}$ is the covariance matrix of the measurement noise $\mathbf{v}_n$.

The calculation of partial derivatives $\mathbf{B}_n$ is defined as:

$$\mathbf{B}_n = \begin{bmatrix} \frac{\partial y_1}{\partial w_1} & \frac{\partial y_1}{\partial w_2} & \cdots \frac{\partial y_1}{\partial w_m} \\ \frac{\partial y_2}{\partial w_1} & \frac{\partial y_2}{\partial w_2} & \cdots \frac{\partial y_2}{\partial w_m} \\ \cdot & & \\ \frac{\partial y_p}{\partial w_1} & \frac{\partial y_p}{\partial w_2} & \cdots \frac{\partial y_q}{\partial w_m} \end{bmatrix}, \tag{9}$$

where q is the total number of neurons in the networks and m is the total number of weights. Using RTRL, derivatives in $\mathbf{B}_n$ are calculated as [53]:

$$\frac{\partial y_i(n+1)}{\partial w_{kl}} = \sigma'(x_i(n)) \left[\sum_{j=1}^{m} w_{ij} \frac{\partial y_j(n)}{\partial w_{kl}} + \delta_{ik} z_l(n) \right] \tag{10}$$

$\sigma'(\cdot)$ is the derivative of the neuron transfer function $\sigma(\cdot)$; $x_i(n) = \sum_{j=1}^{m} w_{ij} z_j(n)$ is the input to each neuron, δ_{ik} is the Kronocker delta. For further details, see [44,52,53].

For the experiments showed here we used an implementation of RTRL-EKF created by [52], which is itself based on the Matlab functions created by [45]. A very good description of the data structures used in such software is given by [48]. In that reference, the interested lector can find a very good algorithm to implement RTRL-EKF using GPA architecture.

4 Proposed Methodology

A block diagram of the proposed scheme is represented in Fig. 6. The algorithm is described as follows: preprocessing of the EEG signals obtained from P4 electrode includes a blind source separation through Independent Component Analysis (ICA) in order to remove eye blink and other artifacts. The signal is then filtered in order to obtain the *alpha* and *beta* bands, and the power signal for each band is computed. The power signal in each band is partitioned into 5 windows with a 50 % overlapping as a feature reduction process. The signal is passed through an

ANFIS system in order to obtain a representation in fuzzy sets corresponding to the evolution in time of the estimated power across both spectral bands alpha and beta. Temporal sequences corresponding to the combination of energy bands for each mental task are input into a recurrent neural network, which is trained to deliver a classification decision on the corresponding mental task.

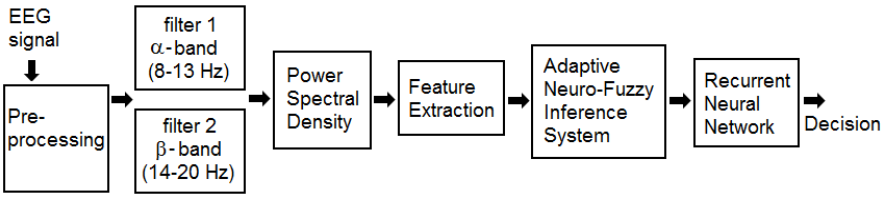


Fig. 6 Block diagram of the proposed architecture for mental tasks classification

Preprocessing EEG data in order to eliminate the artifacts added during the recording sessions is an essential task to facilitate accurate classification. The most corruptive of the artifacts is due to eye blinks because it produces a high amplitude signal called electrooculogram (EOG) that can be many times greater than the EEG signals of interest.

The use of ICA for blind source separation of EEG data is based on an assumption that EEG data recorded from multiple scalp sensors are linear sums of temporally independent components arising from spatially fixed, distinct or overlapping brain networks [54]. The goal of ICA is to recover statistically independent sources given only sensor observations that are unknown linear mixtures of the unobserved independent source signals. ICA reduces the statistical dependencies of the signals, attempting to make the signals as independent as possible which make ICA capable of separating artifact components from EEG data since they are usually independent of each other [55].

As mentioned before, $x_i(t)$ are assumed to be the result of linear combinations of the independent sources, as expressed in:

$$x_i(t) = a_{i1}s_i(t) + a_{i2}s_2(t) + \cdots + a_{in}s_n(t)$$

Or in matrix form:

$$\mathbf{x} = \mathbf{A}\mathbf{s}$$

where:

A is a matrix containing mixing parameters and
s is the source signals.

The goal of ICA is to calculate the original source signals from the mixture by estimating a de-mixing matrix U that gives:

$$\hat{\mathbf{s}} = \mathbf{U}\mathbf{x}$$

Both the mixing matrix **A** and the matrix containing the sources **S** are unknown. The non mixing matrix **U** is found by optimizing a cost function. Several different cost functions can be used for performing ICA, e.g. kurtosis, negentropy, etc., therefore, different methods exist to estimate U. For that purpose the source signals are assumed to be non-Gaussian and statistically independent. The requirement of non-Gaussianity stems from the fact that ICA relies on higher order statistics to separate the variables, and higher order statistics of Gaussian signals are zero. In this way, ICA is applied to EEG signal from P4 electrode in order to remove eye blink artifacts. For additional information see [54]. The result of preprocessing EEG data is shown in Fig. 7.

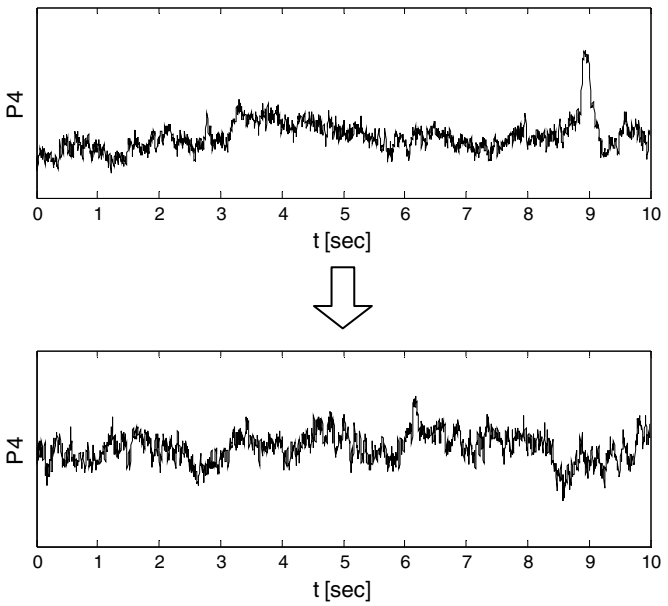


Fig. 7 EEG data before and after preprocessing

Elliptic filters of five order were used in order to obtain the *alpha* and *beta* bands. After filtering EEG data, the power for each band is computed squaring the amplitude of samples; then, the power signal in each band is partitioned into 5 windows with a 50 % overlapping as a feature reduction process. The signal is passed through an ANFIS system in order to obtain a representation in fuzzy sets corresponding to the evolution in time of the estimated power across both spectral bands alpha and beta. Temporal sequences corresponding to the combination of energy bands for each mental task are input into a recurrent neural network, which is trained to deliver a classification decision on the corresponding mental task.

5 Experimental Results

EEG data were obtained previously by Keirn and Aunon [56] and are available on line for research purposes. Ten trials for each mental task resulted in a total of 20 patterns. Details of the procedure followed to detect the signals can be consulted in the cited reference. A brief description is as follows: an Electro-Cap elastic electrode cap was used to record data from positions C3, C4, P3, P4, O1, and O2 defined by the 10-20 system of electrode placement. In the original data set, there were seven subjects performing five different mental tasks and one subject performing two different mental tasks. Signals were recorded for ten seconds during the task at a sampling frequency of 250 Hertz, and each task was repeated five times per session. Subjects attended two sessions recorded on different weeks, resulting in a total of ten trials for each task. The two mental tasks are described as follows. In the task described as mental letter composing, the subjects were instructed to mentally compose a letter to a friend or relative without vocalizing. The second mental task described as visual counting, was constructed by asking the subjects to imagine numbers being written sequentially on a blackboard, with the previous number erased before the next number was written. Experiments were executed using MATLAB version 7.6 in a personal computer with a 2.0 GHz AMD Turion processor and 3GB RAM. Figure 8 shows an example of the normalized power signal corresponding to alpha and beta bands for each mental task.

According to the proposed procedure previously described, feature extraction is performed on the power signals by a window-averaging with a 50% window overlap. Fig. 9 shows an example of the feature vectors obtained through the

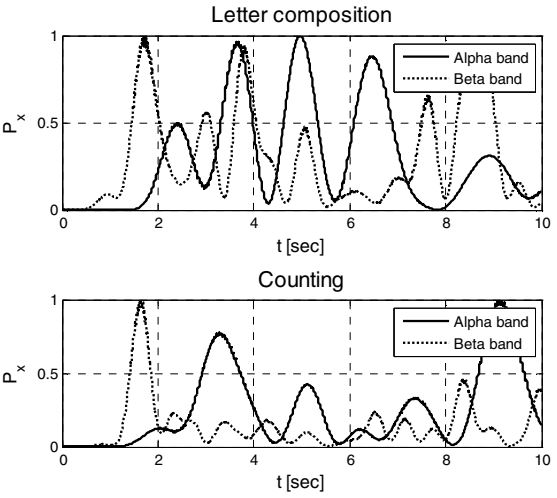


Fig. 8 Alpha and beta band power for letter composition and counting task

described procedure, corresponding to the referred mental tasks. As Fig. 9 illustrates, the power representation of alpha and beta bands presents variations associated to temporal evolution of power bands following each mental task. Since the power in bands shows variations for each subject and trial, we propose the use of an adaptive system allowing the assignment of membership functions in an automatic way in order to represent the configuration of bands through fuzzy sets, translating each experiment into a simple sequence that preserve the temporal evolution of the performed mental task. Fig. 10 shows an example of the state assignment corresponding to the case of letter composition task.

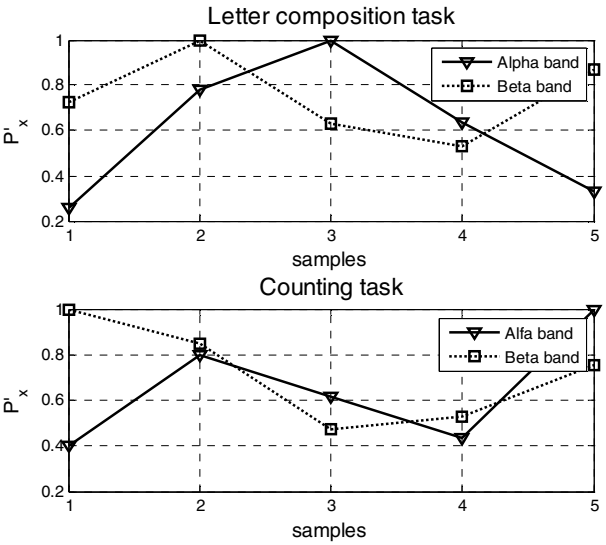


Fig. 9 Result of feature extraction process for two different mental tasks

The feature extraction process is then applied to each trial in the mental tasks database, obtaining some sequences representing the state transitions of power band configurations and corresponding to each mental task. The ANFIS system was trained with the features extracted over all trials, considering an input representation with eight membership functions. Fig. 11 shows an example of the results obtained from the ANFIS training for the two mental tasks. Temporal classification of the obtained feature vectors representing each mental task was performed using a recurrent neural network. In this paper we compare the performance of two models previously described: a simple recurrent neural network or Elman network and a Full Connected Recurrent Neural Network FCRNN. In both cases, the architecture of the recurrent neural networks was: 1 node in the input layer, 10 nodes in the hidden layer and 1 node in the output layer. The architecture was determined by experimentation, with the best results obtained using the described configuration.

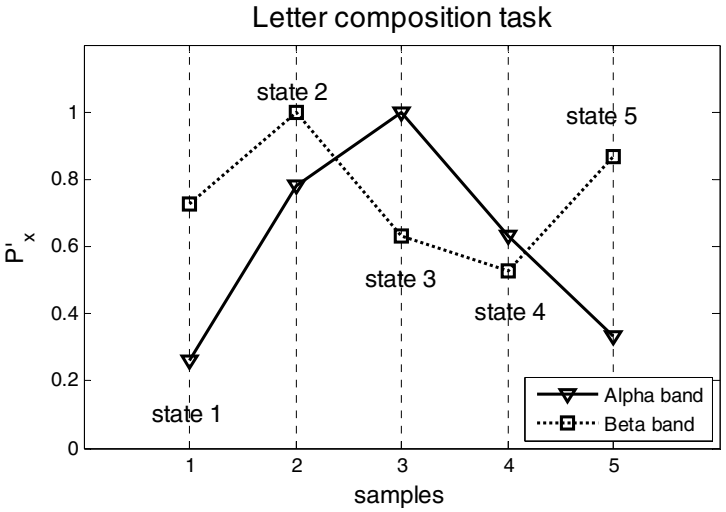


Fig. 10 State assignment for letter composition task

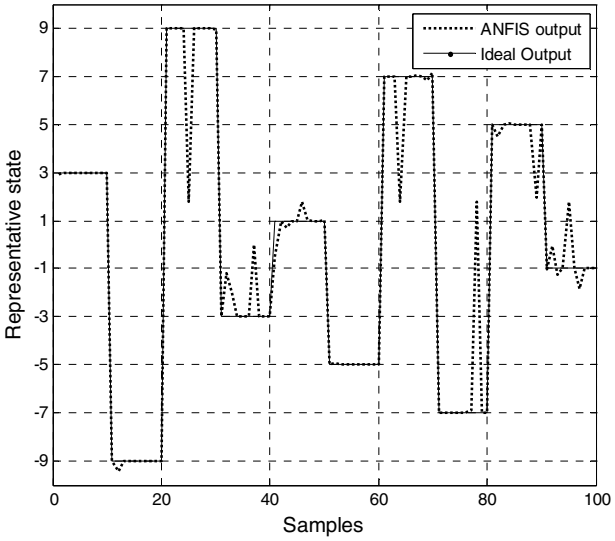


Fig. 11 Result of ANFIS training

Temporal classification results are reported based on a leave-one-out (LOO-CV) cross-validation. LOO-CV is typically used in the analysis of small datasets, where the relatively high variance of the estimator is offset by the stability resulting from the greater size of the training partition than is possible using conventional k-fold cross-validation [57].

Ten trials for each mental task result in a total of 20 patterns. The dataset was partitioned in 5 folds with 4 trials each one. LOO-CV was performed using four folds for training and the remaining one for testing. Table 1 summarizes the temporal classification results obtained in average from both, training and testing cases, with the two recurrent neural networks previously described.

Table 1 Results on temporal classification; training and testing

RNN	Training 500 epochs			Testing	
	MSE	Performance	time	MSE	Performance
Elman	0.0328	91.75%	3'49''	0.0401	90.16%
FCRNN	0.0121	94.61%	1' 12''	0.0528	88.12%.

6 Conclusions

In this chapter, an architecture based on adaptive neuro-fuzzy inference systems (ANFIS) assembled to recurrent neural networks, applied to the problem of mental tasks temporal classification, has been presented. Information on power signal obtained from Alpha and Beta bands constituted a good descriptor with an adequate separability, providing a good balance between complexity and classification rate. The feature vectors representing each mental task following a fuzzy-set paradigm, provided a good description about the temporal evolution of the power signal. A classification rate in training of 94.61 % in average was obtained through the FCRNN, with an 88.12 % of classification using leave-on-out cross validation in the testing stage. A comparison with the Elman Network indicates a better performance of the FCRNN during the training stage, with a slightly better performance of the Elman network on generalization. In both cases, an architecture of the neural network with 10 nodes in the hidden layer provided the better results. Further experimentation oriented to the construction of a database for BCI applications is currently in progress.

Acknowledgments. The first author acknowledges the financial support from the Mexican National Council for Science and Technology (CONACYT), scholarship No. 224304. This research has been partially supported by CONACYT Grant No. CB-2010-155250.

References

[1] Brunner, P., Bianchi, L., Guger, C., Cincotti, F., Schalk, G.: Current trends in hardware and software for brain–computer interfaces (BCIs). Journal of Neural Engineering 8, 025001 (2011)

[2] Bashashati, M., Fatourechi, R., Ward, K., Birch, G.E.: A survey of signal processing algorithms in brain-computer interfaces based on electrical brain signals. Journal of Neural Engineering 4(2), R32–R57 (2007)

[3] Berger, T.W., Chapin, J.K., Gerhardt, G.A., McFarland, D.J., Principe, J.C., Soussou, W.V., Taylor, D.M., Tresco, P.A.: WTEC Panel Report on International Assessment of Research and Development in Brain-Computer Interfaces. World Technology Evaluation Center, Inc. (2007), http://www.wtec.org/bci/BCI-finalreport-26Aug2008-lowres.pdf

[4] Hosni, S.M., Gadallah, M.E., Bahgat, S.F., AbdelWahab, M.S.: Classification of EEG signals using different feature extraction techniques for mental-task BCI. In: 2007 International Conference on Computer Engineering Systems, pp. 220–226 (2007)

[5] Neuper, C., Scherer, R., Wriessnegger, S., Pfurtscheller, G.: Motor imagery and action observation: Modulation of sensorimotor brain rhythms during mental control of a brain–computer interface. Clinical Neurophysiology 120(2), 239–247 (2009)

[6] Solis-Escalante, T., Muller-Putz, G., Brunner, C., Kaiser, V., Pfurtscheller, G.: Analysis of sensorimotor rhythms for the implementation of a brain switch for healthy subjects. Biomedical Signal Processing and Control 5(1), 15–20 (2010)

[7] McFarland, D.J., Sarnacki, W.A., Townsend, G., Vaughan, T., Wolpaw, J.R.: The P-300-based brain–computer interface (BCI): Effects of stimulus rate. Clinical Neurophysiology 122(4), 731–737 (2011)

[8] Ramirez-Cortes, J.M., Alarcon-Aquino, V., Rosas-Cholula, G., Gomez-Gil, P., Escamilla-Ambrosio, J.: Anfis-Based P300 Rhythm Detection Using Wavelet Feature Extraction on Blind Source Separated EEG Signals. In: Ao, S., Amouzegar, M., Rieger, B.B. (eds.) Intelligent Automation and Systems Engineering, ch. 27. LNEE, vol. 103, pp. 353–365. Springer, New York (2011)

[9] Shyu, K.K., Lee, P.L., Liu, Y.J., Sie, J.J.: Dual-frequency steady-state visual evoked potential for brain computer interface. Neuroscience Letters 483(1), 28–31 (2010)

[10] Horki, P., Solis-Escalante, T., Neuper, C., Muller-Putz, G.R.: Hybrid Motor Imagery and Steady-state Visual Evoked Potential Based BCI for Artificial Arm Control. In: Proceedings of the First Tools for Brain Computer Interaction Workshop, Graz, Austria, p. 46 (2010)

[11] Wang, H., Li, C.S., Li, Y.G.: Brain-computer interface design based on slow cortical potentials using Matlab/Simulink. In: Proceedings of the International Conference on Mechatronics and Automation, Changchun, China, pp. 1044–1048 (2009)

[12] Khare, V., Santhosh, J., Anand, S., Bhatia, M.: Performance comparison of three artificial neural network methods for classification of electroencephalograph signals of five mental tasks. J. Biomedical Science and Engineering 3, 200–205 (2010)

[13] Pfurtscheller, G.: Spatiotemporal ERD/ERS patterns during voluntary movement and motor imagery. Supplements to Clinical Neurophysiology 53, 196–198 (2000)

[14] Chiappa, S., Bengio, S.: HMM and IOHMM modeling of EEG rhythms for asynchronous BCI systems. In: European Symposium on Artificial Neural Networks, ESANN (2004)

[15] Millan, J.R., Mouriño, J.: Asynchronous BCI and local neural classifiers: an overview of the adaptive brain interface project. IEEE Transactions on Neural Systems and Rehabilitation Engineering 11, 159–161 (2003)

[16] Pfurtscheller, G., Neuper, C., Schlogl, A., Lugger, K.: Separability of EEG signals recorded during right and left motor imagery using adaptive autoregressive parameters. IEEE Trans. Rehabil. Eng. 6, 316–325 (1998)

[17] Pfurtscheller, G., Neuper, C., Flotzinger, D., Pregenzer, M.: EEG-based discrimination between imagination of right and left hand movement. Electroenceph. Clin. Neurophysiology 103, 642–651 (1997)

[18] Wang, T., He, B.: An efficient rhythmic component expression and weighting synthesis strategy for classifying motor imagery EEG in a brain–computer interface. J. Neural Eng. 1, 1–7 (2004)

[19] Wang, T., Denga, J., He, B.: Classifying EEG-based motor imagery tasks by means of time-frequency synthesized spatial patterns. Clinical Neurophysiology 115, 2744–2753 (2004)

[20] Durka, P.: Matching Pursuit and Unification in EEG Analysis. Artech House, Inc., Norwood (2007)

[21] Wang, J., Xu, G., Wang, L., Zhang, H.: Feature extraction of brain-computer interface based on improved multivariate adaptive autoregressive models. In: Proceedings of the 3rd International Conference on Biomedical Engineering and Informatics (BMEI), Yantai, China, pp. 895–898 (2010)

[22] Kołodziej, M., Majkowski, A., Rak, R.J.: A New Method of EEG Classification for BCI with Feature Extraction Based on Higher Order Statistics of Wavelet Components and Selection with Genetic Algorithms. In: Dobnikar, A., Lotrič, U., Šter, B. (eds.) ICANNGA 2011, Part I. LNCS, vol. 6593, pp. 280–289. Springer, Heidelberg (2011)

[23] Vijean, V., Hariharan, M., Saidatul, A., Yaacob, S.: Mental tasks classifications using S-transform for BCI applications. In: Proceedings of the IEEE Conference on Sustainable Utilization and Development in Engineering and Technology, Semenyih, Malaysia, pp. 69–73 (2011)

[24] Lotte, F.: The use of fuzzy inference systems for classification in EEG-based brain-computer interfaces. In: Proceedings of the 3rd International Brain-Computer Interfaces Workshop and Training Course, Graz, Austria (2006)

[25] Zhang, L., He, W., He, C., Wang, P.: Improving mental task classification by adding high frequency band information. Journal of Medical Systems 34(1), 51–60 (2010)

[26] Palaniappan, R.: Utilizing Gamma band to improve mental task based brain-computer interface design. IEEE Transactions on Neural Systems and Rehabilitation Engineering 14(3), 299–303 (2006)

[27] Park, C., Looney, D., Kidmose, P., Ungstrup, M., Mandic, D.P.: Time-frequency analysis of EEG asymmetry using bivariate Empirical Mode Decomposition. IEEE Transactions on Neural Systems and Rehabilitation Engineering 19(4), 366–373 (2011)

[28] Kousarrizi, M.R.N., Ghanbari, A.A., Teshnehlab, M., Shorehdeli, M.A., Gharaviri, A.: Feature extraction and classification of EEG signals using Wavelet Transform, SVM and artificial neural networks for brain computer interfaces. In: Proceedings of the International Joint Conference on Bioinformatics, Systems Biology and Intelligent Computing, Shanghai, China, pp. 352–355 (2009)

[29] Forney, E.M., Anderson, C.W.: Classification of EEG during imagined mental tasks by forecasting with Elman recurrent neural networks. In: Proceedings of the International Joint Conference on Neural Networks, San Jose, California, USA, pp. 2749–2755 (2011)

[30] Coyle, D., McGinnity, T.M., Prasad, G.: Improving the separability of multiple EEG features for a BCI by neural-time-series-prediction-preprocessing. Biomedical Signal Processing and Control 5(3), 196–204 (2010)
[31] Chang, F., Chang, Y.: Adaptive neuro-fuzzy inference system for prediction of water level in reservoir. Advances in Water Resources 29(1), 1–10 (2006)
[32] Subasi, A.: Application of adaptive neuro-fuzzy inference system for epileptic seizure detection using wavelet feature extraction. Computers in Biology and Medicine 37(2), 227–244 (2007)
[33] Jang, J.S.R.: ANFIS: adaptive-network-based fuzzy inference system. IEEE Transactions on Systems Man and Cybernetics 23(3), 665–685 (1993)
[34] Mandic, D., Chambers, J.: Recurrent neural networks for prediction. John Wiley & Sons, Chinchester (2001)
[35] Fuchs, E., Gruber, C., Reitmaier, T., Sick, B.: Processing short-term and long-term information with a combination of polynomial approximation techniques and time-delay neural networks. IEEE Transactions on Neural Networks 20(9), 1450–1462 (2009)
[36] Gomez-Gil, P.: Long term prediction, chaos and artificial neural networks. Where is the meeting point? Engineering Letters 15(1), 1–5 (2007)
[37] Skarda, C., Freeman, W.: How brains make chaos in order to make sense of the world. Behavioral and Brain Sciences 10, 161–195 (1987)
[38] Jordan, M.: Serial order: a parallel distributed processing approach. Technical Report TR-8604. UC San Diego Institute for Cognitive Science, San Diego (1986)
[39] Elman, J.: Finding structure in time. Cognitive Science 14, 179–211 (1990)
[40] Werbos, P.: Backpropagation through time: what it does and how to do it. Proceedings IEEE 74(10), 1550–1560 (1990)
[41] Williams, R., Zipser, D.: A learning algorithm for continually running fully recurrent neural networks. Neural Computation 1, 270–280 (1989)
[42] Graves, A., Fernandez, S., Gomez, F., Schmidhuber, J.: Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In: Proceedings of the 23rd International Conference on Machine Learning, pp. 369–376. ACM, Pittsburgh (2006), doi:10.1145/1143844.1143891
[43] Williams, R.: Some observations on the use of the extended Kalman Filter as a recurrent network learning algorithm. Technical Report NU-CCS-92-1, North-eastern University, Boston, MA (1992)
[44] Haykin, S.: Neural Networks, 2nd edn. Prentice Hall, Upper Saddle River (1999)
[45] Cernansky, M.: Matlab functions for training recurrent neural networks RTRL-EKF (2008), `http://www2.fiit.stuba.sk/~cernans/main/download.html` (accessed January 2009)
[46] Werbos, P.: Beyond regression: new tools for prediction and analysis of the behavioral sciences. PhD Thesis, Cambridge, MA (1974)
[47] Rumelhart, D., Hinton, E., Williams, R.: Learning internal representations by error propagation. In: Parallel Distributed Processing: Explorations in the Microstructure of Cognition, vol. I. Bradford Books, Cambridge (1986)
[48] Čerňanský, M.: Training Recurrent Neural Network Using Multistream Extended Kalman Filter on Multicore Processor and Cuda Enabled Graphic Processor Unit. In: Alippi, C., Polycarpou, M., Panayiotou, C., Ellinas, G. (eds.) ICANN 2009, Part I. LNCS, vol. 5768, pp. 381–390. Springer, Heidelberg (2009)

[49] Ralaivola, L., d'Alché-Buc, F.: Nonlinear Time Series Filtering, Smoothing and Learning using the Kernel Kalman Filter. Technical Report, Universite Pierre et Marie Curie, Paris France (2004)
[50] Alanis, A., Sanchez, E., Loukianov, A.: Discrete-time adaptive backstepping nonlinear control via high-order neural networks. IEEE Transactions on Neural Networks 18(4), 1185–1195 (2007)
[51] Prokhorov, D.: Toyota prius hev neurocontrol and diagnostics. Neural Networks 21, 458–465 (2008)
[52] García-Pedrero, A.: Arquitectura neuronal apoyada en señales reconstruidas con wavelets para predicción de series de tiempo caóticas, M. Sc. Thesis, INAOE, Tonantzintla, Puebla (2009) (in spanish)
[53] Doka, K.: Handbook of brain theory and neural networks, 2nd edn. MIT Press, Cambridge (2002)
[54] Kachenoura, A., Albera, L., Senhadji, L., Comon, P.: ICA: A Potential Tool for BCI Systems. IEEE Signal Processing Magazine, 57–68 (January 2008)
[55] Keralapura, M., Pourfathi, M., Sirkeci-Mergen, B.: Impact of Contrast Functions in Fast-ICA on Twin ECG Separation. IAENG International Journal of Computer Science 38(1), 38–47 (2011)
[56] Keirn, Z.A., Aunon, J.I.: A new mode of communication between man and his surroundings. IEEE Trans. Biomed. Eng. 37(12), 1209–1214 (1990)
[57] Cawley, G.C.: Leave-One-Out Cross-Validation Based Model Selection Criteria for Weighted LS-SVMs. In: Proceedings of the International Joint Conference on Neural Networks, Vancouver, Canada, pp. 1661–1668 (2006)

Part IV

Soft Computing: Theory and New Models

An Analysis of the Relationship between the Size of the Clusters and the Principle of Justifiable Granularity in Clustering Algorithms

Mauricio A. Sanchez[1], Oscar Castillo[2], and Juan R. Castro[1]

[1] Universidad Autonoma de Baja California, Tijuana, Mexico
{mauricio.sanchez,jrcastro}@uabc.edu.mx
[2] Instituto Tecnologico de Tijuana, Tijuana, Mexico
ocastillo@hafsa.mx

Abstract. The initial process for the granulation of information is the clustering of data, once the relationships between this data have been found these become clusters, each cluster represents a coarse granule, whereas each data point represents a fine granule. All clustering algorithms find these relationships by different means, yet the notion of the principle of justifiable granularity is not considered by any of them, since it is a recent idea in the area of Granular Computing. This paper describes a first approach in the analysis of the relationship between the size of the clusters found and their intrinsic implementation of the principle of justifiable granularity. An analysis is done with two datasets, simplefit and iris, and two clustering algorithms, subtractive and granular gravitational.

Keywords: justifiable granularity, clustering algorithm, subtractive, granular gravitational.

1 Introduction

Granular computing is an area which has been gaining support since its initial conception[1],[2],[3],[4],[5],[6]. Focusing on how information is treated and represented, it describes how information should efficiently relate to each other, defining the size of each granule and confining the cardinality of data into a meaningful information granule[3]. This area has expanded into different interpretations, since it is more of a theory in general than a defined methodology of treating information, yet they all share the same objective, to obtain meaningful granules. Information granules can also be represented in a number of forms, fuzzy logic[7][8], rough sets[9][10], etc[11][12][13][14].

P. Melin, O. Castillo (Eds.): *Soft Computing Appli. in Optimization,* STUDFUZZ 294, pp. 239–263.
DOI: 10.1007/978-3-642-35323-9_10

The process of obtaining information granules is first preceded by the action of finding relations between all data; this process is usually done by a clustering algorithm. Clustering algorithms are defined as algorithms which find relationships between data, there are multiple methodologies in which such relationships are found, there are categorized as centroid based[15], density based[16], hierarchical based[17], among others[18][19]. Each one of these obtaining similar results, yet at the same time, finding different results, this is, there is a difference in performance on each type of algorithm. The end result of such algorithms are usually cluster centers as well as areas of influence, in the specifics of centroid based clustering algorithms, they find cluster centers between the universe of data and radial areas of influence, which can be easily mapped into fuzzy Gaussian membership functions[20]. This paper focuses on this type of clustering algorithms.

Although clustering algorithms obtain acceptable results in the relationships found[21], they do so in a manner that does not take into account if they adhere to the basic theory of granular computing or not, since many algorithms precede the existence of the area of granular computing. Yet clustering algorithms and granular computing are intertwined in such a way that you cannot remove one from the other, because finding relationships in data is essential to obtaining information granules.

One step in the correct direction of uniting clustering algorithms and granular computing is the implementation of the principle of justifiable granularity[22]; this principle is a first attempt to describe, in more detail, how an information granule is in fact meaningful, and not redundant or too specific. Most, if not all, clustering algorithms do not take into account if they create meaningful granules or not, they only concentrate on the end result, and not if the chosen granules are optimized and/or meaningful.

This paper is organized into multiple chapters which introduce concepts on clustering algorithms which then briefly describes two such algorithms, subtractive and granular gravitational. Afterwards, a description and discussion on the concept of the principle of justifiable granularity, and finally a discussion is given into how both clustering algorithms, which were previously described, intrinsically implement such principle.

2 Clustering Algorithms

Clustering algorithms have the main objective of finding hidden relationships between the data inside of a specific information universe. In the following sections two clustering algorithms will be described, only until how granules are found, since some continue onto the process of creating a fuzzy system from the clusters which are found, this is to focus on the discussion and analysis on how the intrinsically impellent the principle of justifiable granularity.

2.1 Subtractive Algorithm

This algorithm is density based, which means that its end results are calculated by analyzing the density of data points inside a given radius, this is done iteratively point by point until an objective function is within a specified tolerance[23].

The following, describes the main calculations done by this algorithm:

1. A measure of the potential (1) of each data point is first calculated, taking into account the value of the given radius (2).

$$P_i = \sum_{j=1}^{n} e^{-\alpha \|x_i - x_j\|^2} \tag{1}$$

$$\alpha = \frac{4}{r_a^2} \tag{2}$$

2. The highest potential is selected (2) and accordingly reduces the potential on the rest of the point, calculated with the support of the given radius (4)

$$P_i \Leftarrow P_i - P_{1^e}^{*\,-\beta \|x_i - x_{\hat{1}}\|^2} \tag{3}$$

$$\beta = \frac{4}{r_b^2} \tag{4}$$

3. This is repeated until the finalization condition (5) is met

$$P_k^* < \varepsilon P_1^* \tag{5}$$

4. All sigmas, or radial area of influence, are then calculated (6)

$$sigmas_i = \frac{r\big(\max(X) - \min(X)\big)}{\sqrt{8}} \tag{6}$$

Due to the nature of this algorithm, and the need to know the cardinality of each found cluster in respect to the information universe, as to apply the principle of justifiable granularity, a manner to find such cardinality (7) was implemented into the algorithm, this was done by calculating the distances between each found cluster and all data points, and the closest data points to each clusters were added to their respected cardinality (8), where the distance is calculated with the Euclidean distance measurement (9).

$$card\{c_i \in X_j\} \tag{7}$$

$$X_j = \text{data points closer to } \mathrm{c_i} \tag{8}$$

$$d_i = \sqrt{c_i - x_j}; \quad i! = j \tag{9}$$

2.2 Granular Gravitational Algorithm

This is a hybrid centroid-density based algorithm, meaning all the calculations are done based on point density and point distance, this is considering that Newton´s Law of Universal Gravitation[24] is utilized to carry out the main cluster calculations[25].

The following describes the main calculations done by this algorithm:

1. All gravitational forces (10) in the system are calculated

$$F_{ij} = G\frac{m_i m_j}{d(x_i, x_j)^2} \tag{10}$$

2. The sum force (11) for all data points is then sorted in descending order

$$sumF = \sum_{n=0}^{i} F_j \tag{11}$$

3. All points with strong gravitational force and within a given radius are joined (12), joining the point with lesser mass to the point with more mass

$$x_i \cup x_j\,; \quad m_i \cup m_j \tag{12}$$

4. This is repeated until a balance in the gravitational forces in the system is achieved
5. All sigmas are then calculated based on the strongest force exerted by any found cluster unto the rest of the information universe

3 Principle of Justifiable Granularity

This principle is "concerned with the formation of a meaningful information granule based on available experimental evidence"[22], meaning that finding clusters and assigning sigmas that give acceptable results is no longer relevant, and the correct size and cardinality of each cluster is.

3.1 Basic Theory

The principle of justifiable granularity is concerned with obtaining the adequate size of each information granule which was found for that specific information universe. In this theory, there are two main rules that must be followed:

1. The numeric evidence of a specific information granule must be as high as possible, this means that the cardinality of information within a granule must be high
2. The information granule should be as specific as possible, meaning that vagueness is reduced, relying only on a very strict cardinality

This completely contradicts itself, as shown in Fig.1, since in one side you must elevate the cardinality yet on the other side you must reduce it. This is not a problem in the final implementation, since a balance is found which conforms to both rules.

Fig. 1 Visual difference between both rules: a) High numerical evidence with low specificity, and b) Low numerical evidence with high specificity

Considering that the size of the granule is to be found, this size is separated into two segments, segment **a** and segment **b**, as shown in Fig. 2, in other words, the left side of the granule and the right side of the granule, since the Median of the data points, *Med(D)*, may not always be in the center and the data points may not necessarily be normalized in a symmetric position, the lengths of both **a** and **b** will most of the time not be the same.

Fig. 2 Granule showing the difference between the lengths of a and b, both starting at the Median of the data points

The cardinality of each information granule (1) is important, since it is the basis for finding the lengths required to obtain a meaningful granule.

$$card\{x_k \in \Omega\} \tag{13}$$

This cardinality is separated into two distinct sets, which conform to the specific calculation of the length for both a and b. Whereas the cardinality for calculating b (14) would only be defined by the data points larger than Med(D).

$$card\{x_K \in \Omega, x_k > Med(D)\} \tag{14}$$

Considering the contradicting nature of the two rules of the principle of justifiable granularity, to obtain the best balance between both, an optimization must be done to find the best possible solution. This is obtained by maximizing an objective function (15), with reference to a user criterion α (16) which controls the balance between both rules. Varying this user criterion affects how specific, or general, a granule can become.

$$V(b^*) = max_{b>Med(D)}[\mathrm{V(b)}] \tag{15}$$

$$\alpha \in [0, \alpha_{max}] \tag{16}$$

This optimization is done with an integration of the probability density function from *Med(D)* to all prototypes of b, multiplied by a scaling factor which integrates the user criterion for specificity (17).

$$V(b) = e^{(-\alpha|b-Med(D)|)} \int_{Med(D)}^{b} p(x)\,\mathrm{d}x \tag{17}$$

The behavior of *V(b)* is shown in Fig. 3, these plots are of an arbitrary example to show how *V(b)* behaves with different values of *α,* which in this case range from 0.1 to 0.5 in increments of 0.1, the smallest value of *α* being the topmost plot.

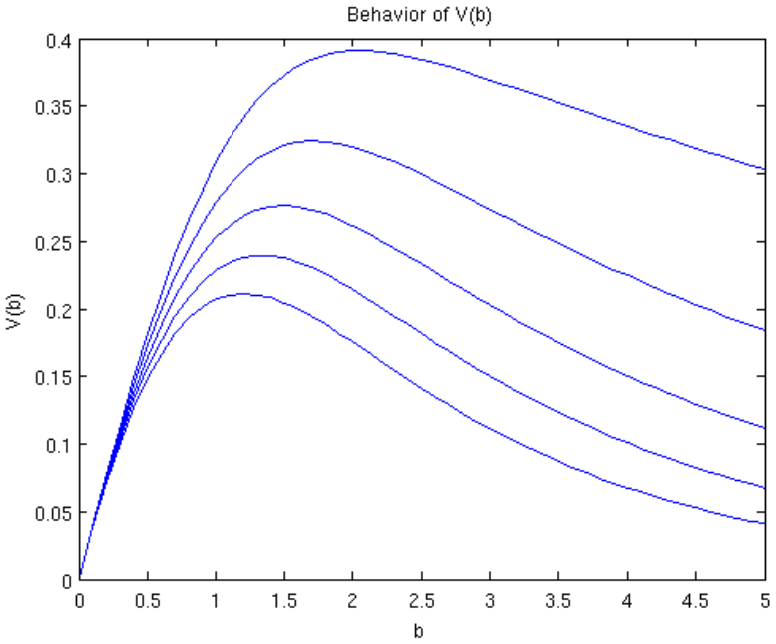


Fig. 3 Behavior of *V(b)* in respect to b, with different values of *α*

How this behavior of *V(b)* affects the final length of each granule is shown in Fig. 4, in this example, the value of *α* equals zero and is shown where *Med(D)* is located, this is, where both plots of the optimization *V(*)* start, or *V(a|b) = 0*; in this special case of *α=0*, the maximization of *V(*)* is achieved until b_{max} or a_{min} is reached, this is, the farthest point in the cardinality of the granule. Since this condition suggests that all numerical evidence must be utilized and the granule must be as non-specific as possible.

Continuing with the direct effect of *V(a)* and *V(b)* in regards to the final length of the granule, Fig. 5 shows how an *α=2,* which suggests granule length which is not too specific yet with not too much numerical evidence. This figure identifies where the optimization´s maximum is located and how it directly affects the length of each side of the granule.

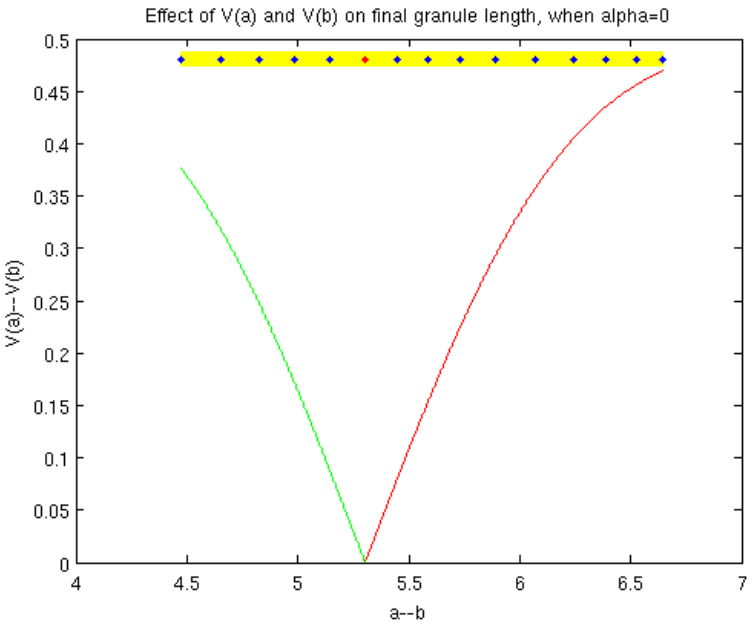


Fig. 4 Effect of the behavior of *V(a)* and *V(b)* on the final length of the granule, when the value of α is equal to zero

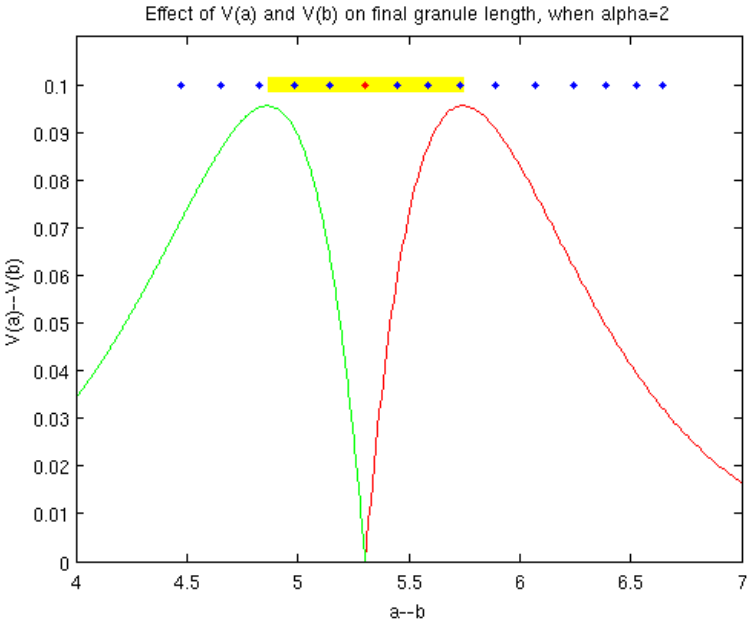


Fig. 5 Effect of *V(a)* and *V(b)* on the final length of the granule, when α=*2*

3.2 The Specificity Criterion

As briefly described in the previous section, this α criterion controls how specific or general the information granule will become, while still maintaining a balance between both rules, which, in nature, contradict each other.

As a simple implementation it is easy to start the value at 0 to represent the most general possible granule, yet it is somewhat trivial to know the maximum value α can take as to have a very specific granule.

A form to find an approximate optimum maximum value for the criterion is best described in Fig. 6, where its behavior is of an inverse natural logarithm plot. α_{max}, is an approximate value at best, since after a threshold is surpassed, no matter the increment of value, how it affects the scaling factor of the length of the information granule is negligent.

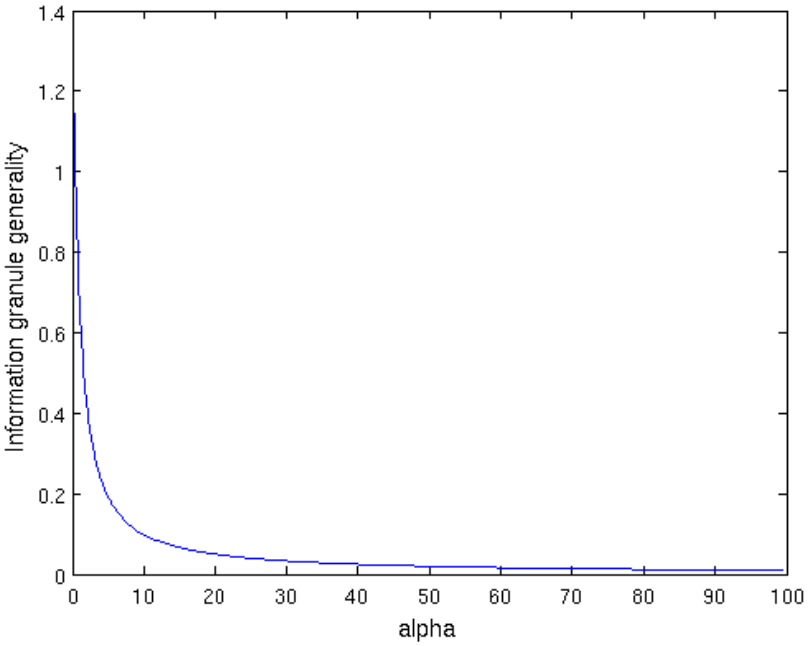


Fig. 6 Behavior of α in relation to the generality of the information granule

This behavior changes with each granule, as it is influenced by the values in each granule's cardinality. In Fig.7, an example of two different granules and the behavior here described is shown, where two distinct behaviors are seen. The above graph shows a very pronounced descent as the length is affected by the value of α, while the second graph shows two distinct behaviors that differ from the first one, the first difference is the speed of descent of the length in relation to the value of α, which is slower to reach a lower value in comparison with the previous graph; and the second noticeable difference, is that the initial values in the topmost

part of each plot descend at different moment, where in the first graph, the descent is immediate as the values of α start to increase, while the value of the second graph starts in a stable plateau before starting to descend, this plateau is reflected in the length as no change to the maximum coverage or generality of the granule, meaning that while the value of α changes, the length does not.

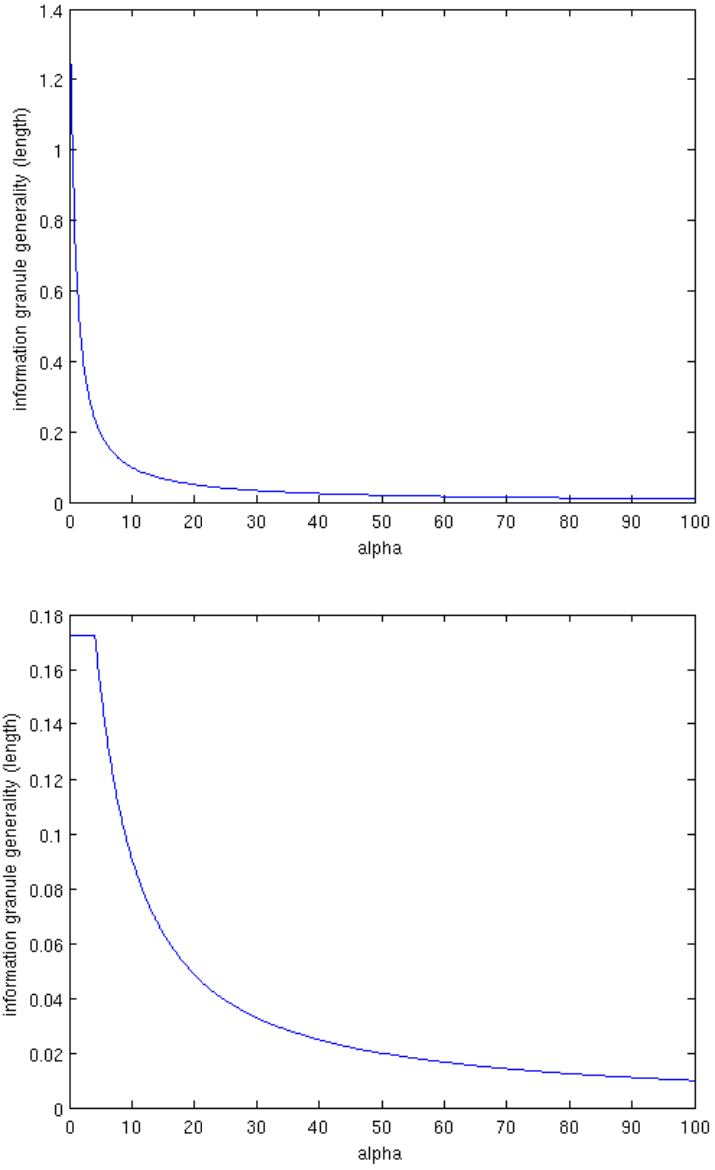


Fig. 7 Behavior of two different granules which shows the relation between α and the length of the granule

The approximate value of α_{max} is somewhere in the area where the behavior plot stops to descend fast and starts to stabilize, as shown in Fig. 8.

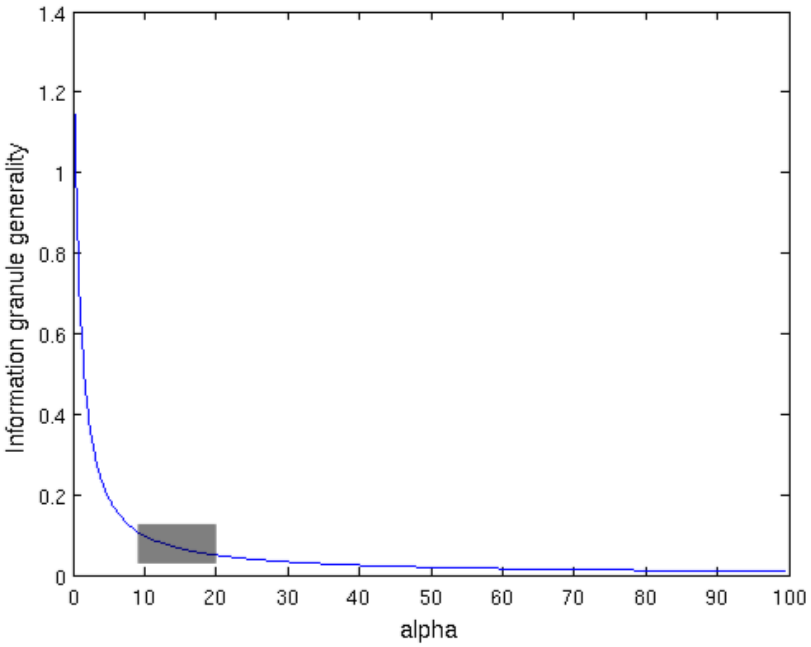


Fig. 8 Interval where the approximate value of α_{max} is found

A heuristic is offered which can find this approximate value of α_{max}, which is described as the natural logarithm of the cardinality of the chosen side (18) to find its length divided by the length of the closest point (x_1) to *Med(D)* (19).

$$\partial = count(card\{c_i \in \Omega > Med(D)\}) \tag{18}$$

$$\alpha_{max} = \frac{\log(\partial)}{|Med(D) - x_1|} \tag{19}$$

On other terms of application, and considering that choosing an α close to 0 will scale the information granule to a non specific granule, with a length near 0, the middle point between [0, α_{max}] or $\alpha_{max}/2$, will not necessarily impose a length of exactly half the size, since the behavior, as already shown, is non-linear, and it will greatly vary from the numerical evidence contained within each specific information granule. In Fig. 9, a clear example is shown of the difference between α=0 and α=α_{max}.

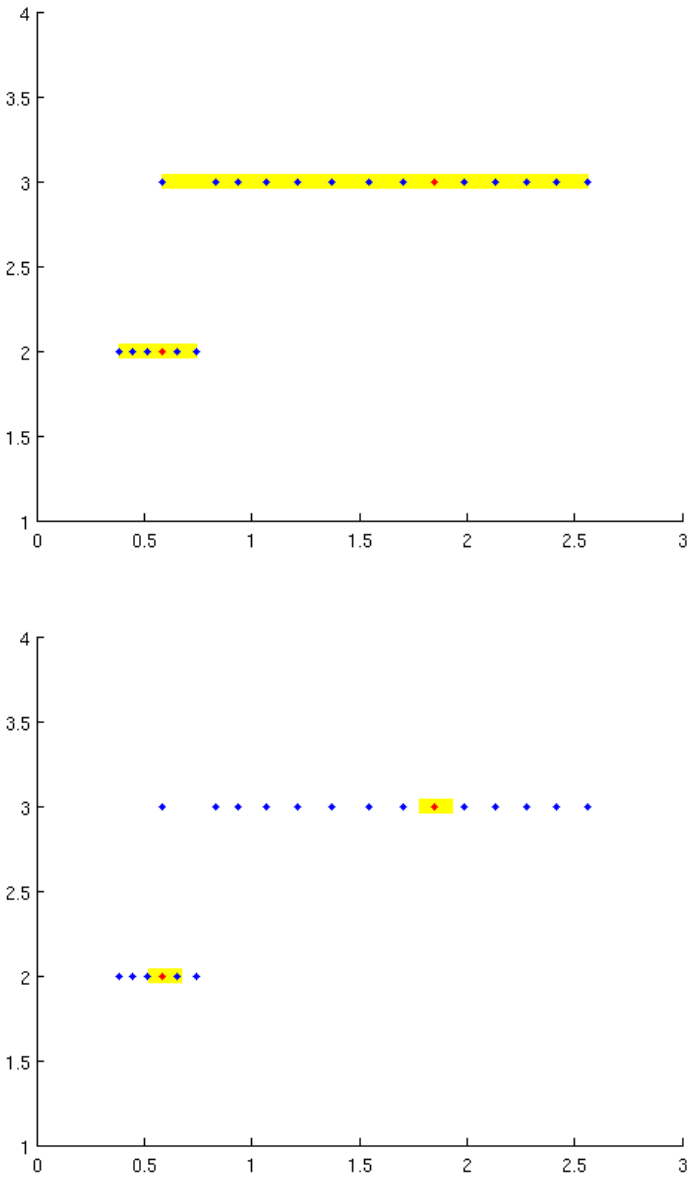


Fig. 9 Visual difference between utilizing zero and α_{max}. The upper graph shows a usage of α=0, and the lower graph shows the result of using $\alpha=\alpha_{max}$.

An apparent difficulty with utilizing this criterion is that for each information granule, you must specify two values for α, and considering that most datasets are defined by multiple inputs, each granule for each input must have two values. To better understand this, suppose a dataset with 4 inputs where 10 granules are found

after processing all information, to apply the principle of justifiable granularity one must consider that for every input there exists 10 granules, meaning 40 granules in total, and since each granule must specify two values for α, the user must input a total of 80 different values for α, which is not a very user friendly number in itself.

Considering this difficulty in applying this principle, there are multiple ways to address a solution. First, all values of α are set to the same value, although this would definitely reduce everything to a single number, it will purposely defeat the objective of applying the principle of justifiable granularity, this is considering that α_{max} varies from granule to granule, meaning this method is not viable unless the value is ultimately set to zero. Another method, similar to the previous one, is to use a scaling factor, since a heuristic has been proposed in this paper, it could be used to scale from its maximum value down to zero, $\alpha = \alpha_{max} \; x \; \alpha_{scale}$ where $0 \leq \alpha_{scale} \leq 1$, and apply it homogeneously to all values of α, this can be visualized in Fig. 10.

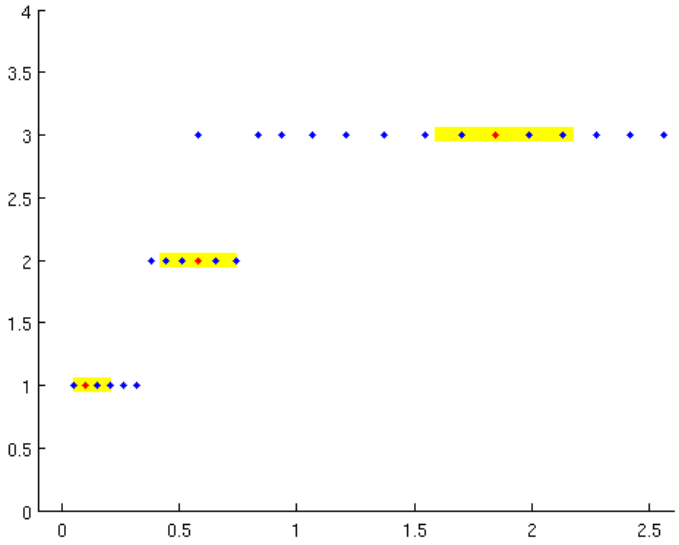


Fig. 10 Example of applying a global scaling to all lengths (a and b) for all granules, with a scaling factor set to 0.25

Another possible solution could be to apply a scaling factor, but instead of globally, apply it locally to each input, this could be used in cases where certain inputs variables can be relaxed in the values which it can receive or on the other hand, only accept values high where precision is required, an example is shown in Fig. 11 where two distinct inputs with two granules each implement this solution; this type of solutions could be seen in a control application, where inputs are from various sensors where some require higher precision and other lower precision.

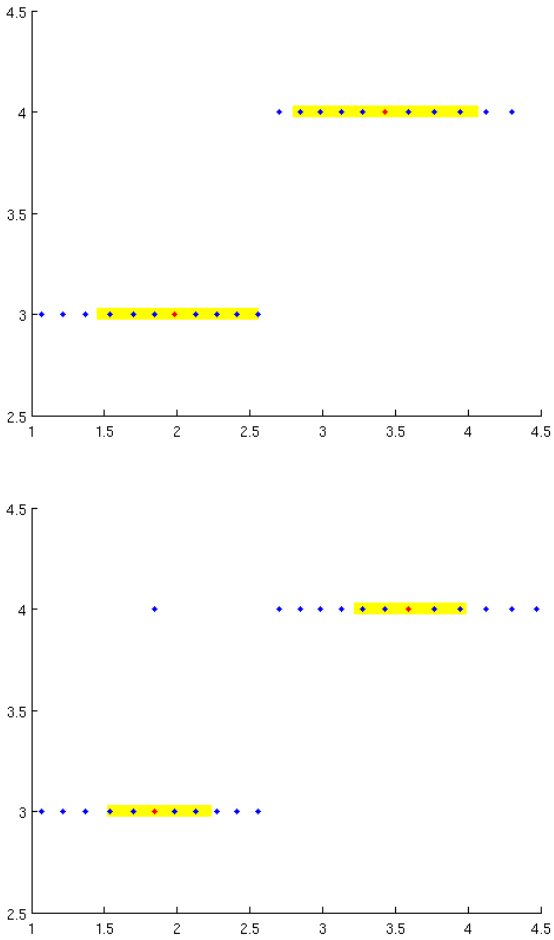


Fig. 11 Example of a hypothetical set of granules from two distinct inputs, where each input shares a global alpha in both lengths (a and b), the first input having a scaling value of 0.1 and the second input having a global scaling value of 0.2

These possible solutions are a few of many ideas which could address the problem of having to give an unrealistic amount of α values to the system.

4 Algorithm Analysis

The analysis which will be shown is of a comparison on the variable level, which is how the principle of justifiable granularity is applied. Comparing the length of each sigma in respect to the lengths found by the principle of justifiable granularity, considering the user criterion which varies the end results, they will be of different values, only to demonstrate how they vary with each specific cardinality.

Another consideration in the results is that both lengths a and b should have different values for their respective α, yet the same value was chosen to demonstrate another difference in how the lengths are different even if on the same granule, yet diversified depending on the cardinality of the left or right side of the *Med(D)*.

Originally, the value of *Med(D)* should be calculated depending on the data contained within each information granule, but since the comparison is done using existing clustering algorithms, they already obtain cluster centers, which in this case, represent the value of *Med(D)*, and for the sake of a fair comparison, without altering the results given by these algorithms, *Med(D)* will not be calculated and will be obtained directly from these algorithms, taken from their found centers.

The sigmas which are obtained by these algorithms will be compared to the calculated lengths by the principle of justifiable granularity.

For a more equal analysis of both algorithms, their results which will be analyzed will be done using their obtained clusters when they identify 100% the datasets.

4.1 Simplefit Dataset

This dataset is a very simple data fitting benchmark with one input and one output, this dataset can be seen in Fig. 12.

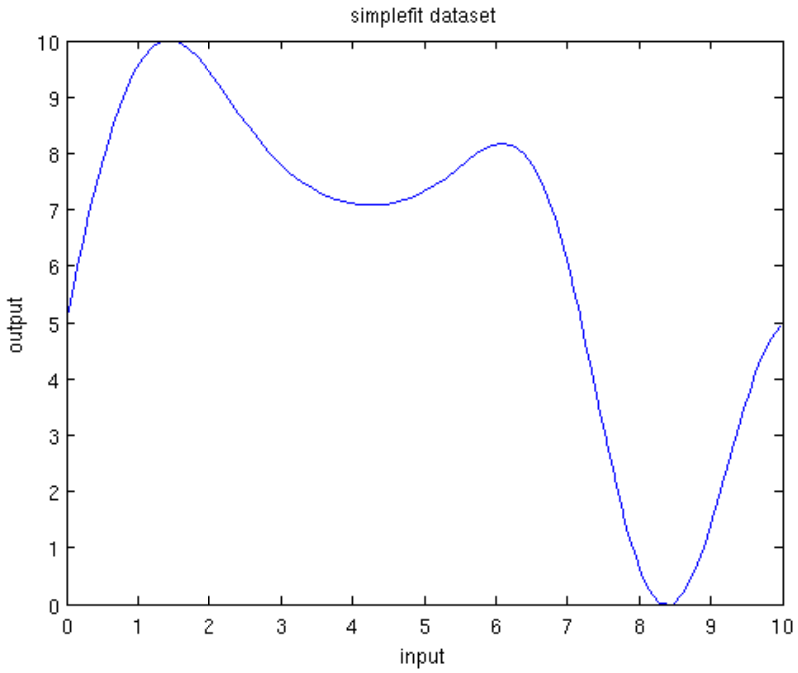


Fig. 12 Visual representation of the simplefit dataset with one input and one output

Both algorithms which are being analyzed will be compared utilizing $\alpha=0$ and $\alpha=2$ on both lengths of the granule, a and b. Fig. 13 shows the subtractive algorithm´s sigmas, represented by straight lines, against the lengths found by the principle of justifiable in this case, the granule is none-specific and completely generalized, which has $\alpha=0$, yet in comparison with the sigmas from the subtractive algorithm, half reach far beyond the cardinality of each granule.

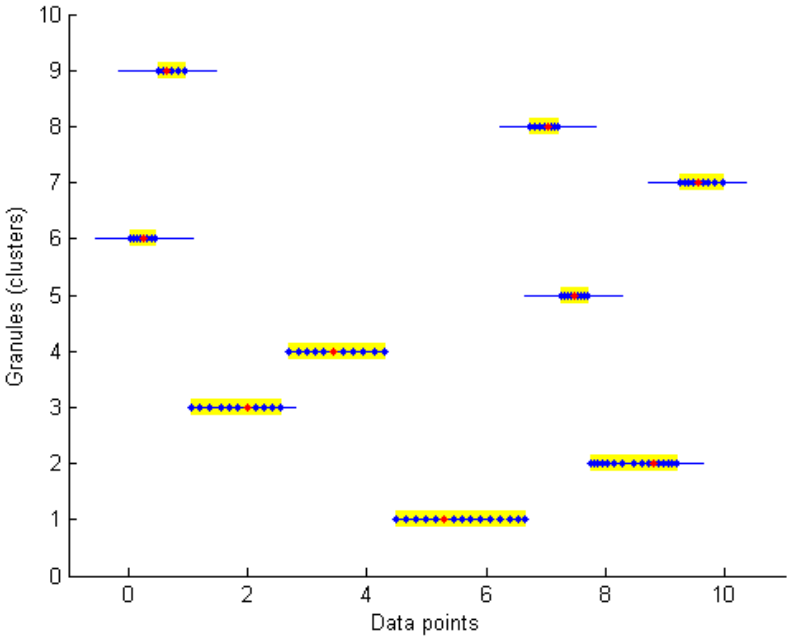


Fig. 13 Results for subtractive algorithm of the simplefit dataset under, comparing the sigmas (lines) found by the algorithm against the lengths suggested by the principle of justifiable granularity (shaded area), with $\alpha=0$

In Fig. 14, the same granules are shown but with $\alpha=2$, where it can be noted that all lengths vary widely since each granule's cardinality is different in nature. This example shows that the original length found by the subtractive algorithm cannot compare with the varying values of α, since such algorithm does not consider how specific a granule can be.

The granular gravitational algorithm, shown in Fig. 15, shows a very different behavior in its sigmas, compared to the subtractive algorithm´s sigmas, since they adapt much more to the cardinality of each granule, but in comparison with the lengths given by the principle of justifiable granularity, with $\alpha=0$, they are still very generalized and reach beyond the its cardinality in the same way the subtractive algorithm does.

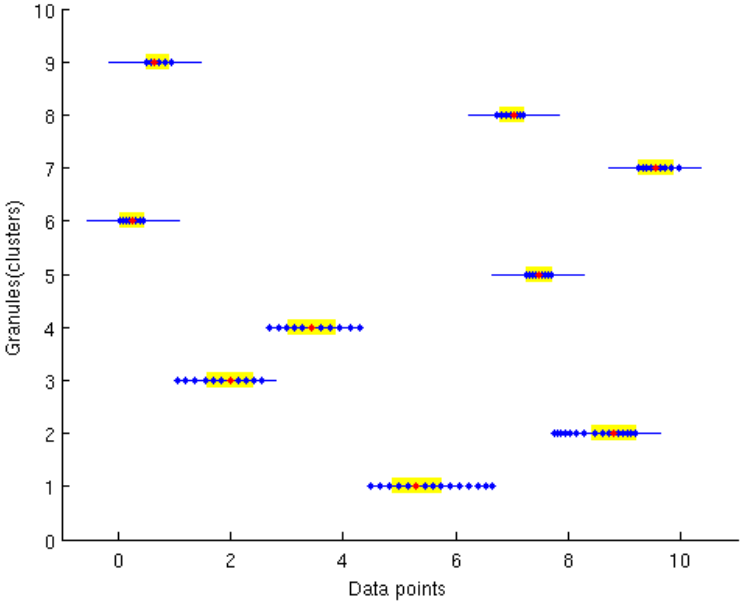


Fig. 14 Results for subtractive algorithm of the simplefit dataset under, comparing the sigmas (lines) found by the algorithm against the lengths suggested by the principle of justifiable granularity (shaded area), with α=*2*.

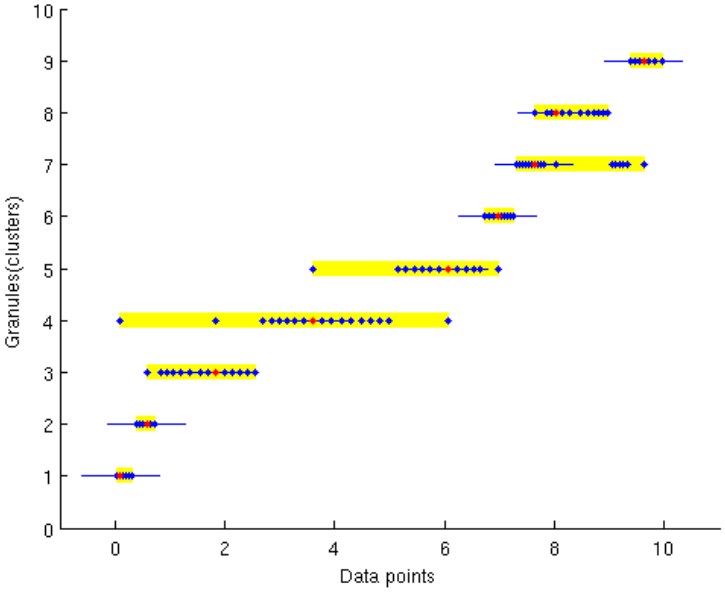


Fig. 15 Results for granular gravitational algorithm of the simplefit dataset, comparing the sigmas (lines) found by the algorithm against the lengths suggested by the principle of justifiable granularity (shaded area), utilizing α=*0*

In Fig. 16, the value set by the specificity criterion was set to 2, which much how the subtractive algorithm with the same value for α compares against the application of a variables specificity criterion, it cannot compare since the granular gravitational algorithm also does not contemplate how specific a granule can be.

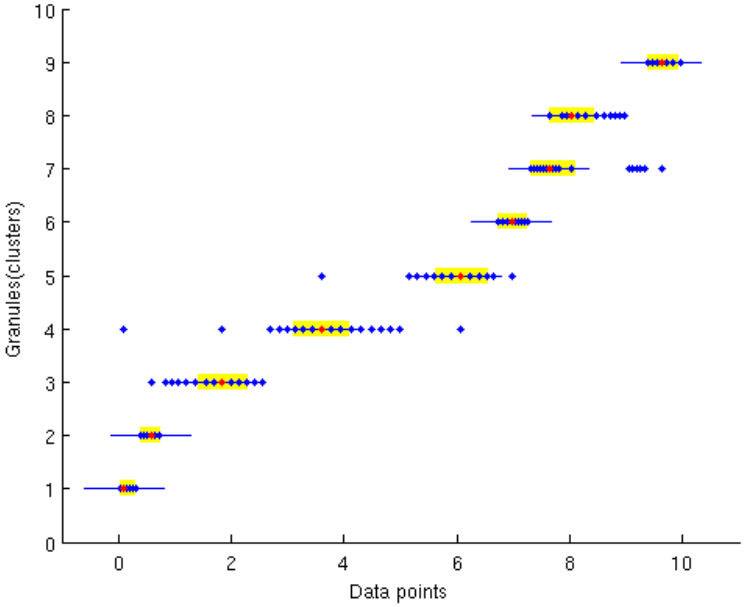


Fig. 16 Results for granular gravitational algorithm of the simplefit dataset, comparing the sigmas (lines) found by the algorithm against the lengths suggested by the principle of justifiable granularity (shaded area), utilizing $\alpha=2$

4.2 Iris Dataset

The iris dataset is a benchmark dataset in remarks to clustering and classification, with four inputs and three classes, and a non-linear solution, this makes for a very interesting dataset to test.

First to be analyzed, is the comparison for the subtractive algorithm, specifically for the input variable of the petal length and sepal width, comparing the sigmas obtained by the clustering algorithm against the lengths obtained by applying the principle of justifiable granularity with an $\alpha=0$ *and* $\alpha=5$. As shown in Fig. 17, the calculated sigmas by the subtractive algorithm still overreach, in most cases, far beyond the cardinality of each cluster, yet in some cases some data points are not inside that area of influence. For the lengths found by the principle of justifiable granularity, since the criterions which was chosen is zero, this shows the maximum length which should be used in order to generalize the granule, in some cases with this example, the lengths are almost identical, yet in others they are either too short or too long, giving mixed results.

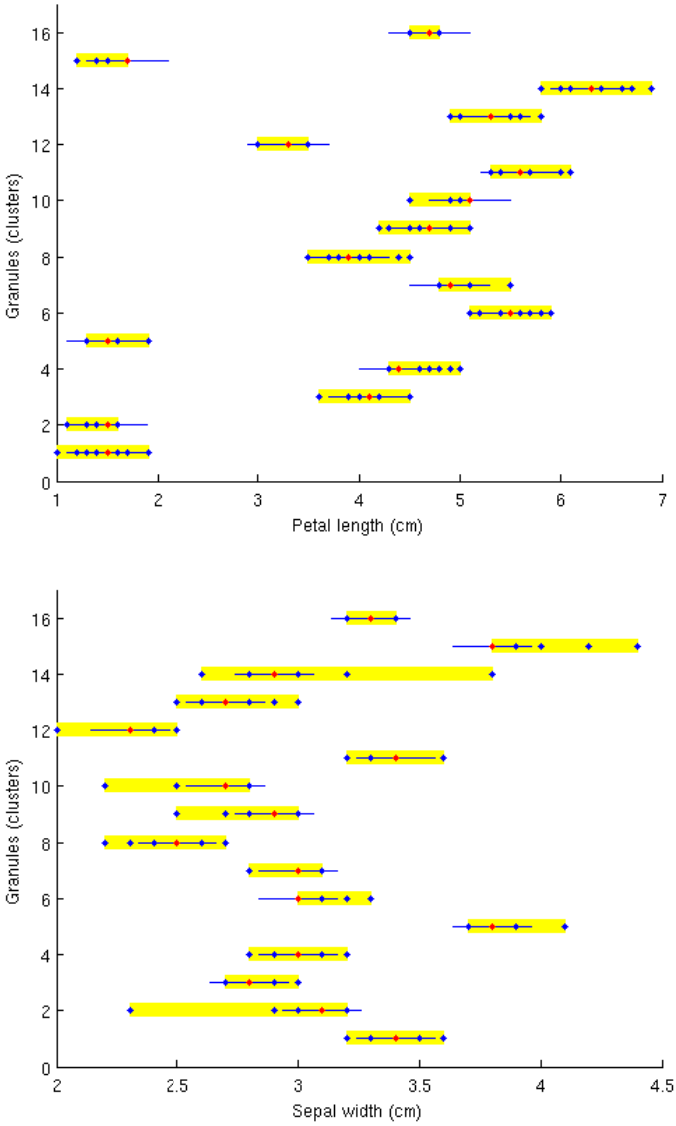


Fig. 17 Clusters found by the subtractive algorithm for the variables ´petal length´ and ´sepal width´, comparing the sigmas found by the algorithm (straight lines), against the lengths found by the principle of justifiable granularity (shaded areas), with a global $\alpha=0$

The same example of granules and inputs is now shown in Fig. 18, but with an α of 5, to show how the lengths of the granules adapt to each granules cardinality in respect to the user criterion. The general behavior with the selected specificity criterion is to have lengths which are shorter, hence having granules which rely more on having a higher precision in such inputs in comparison with having a more generalized granule.

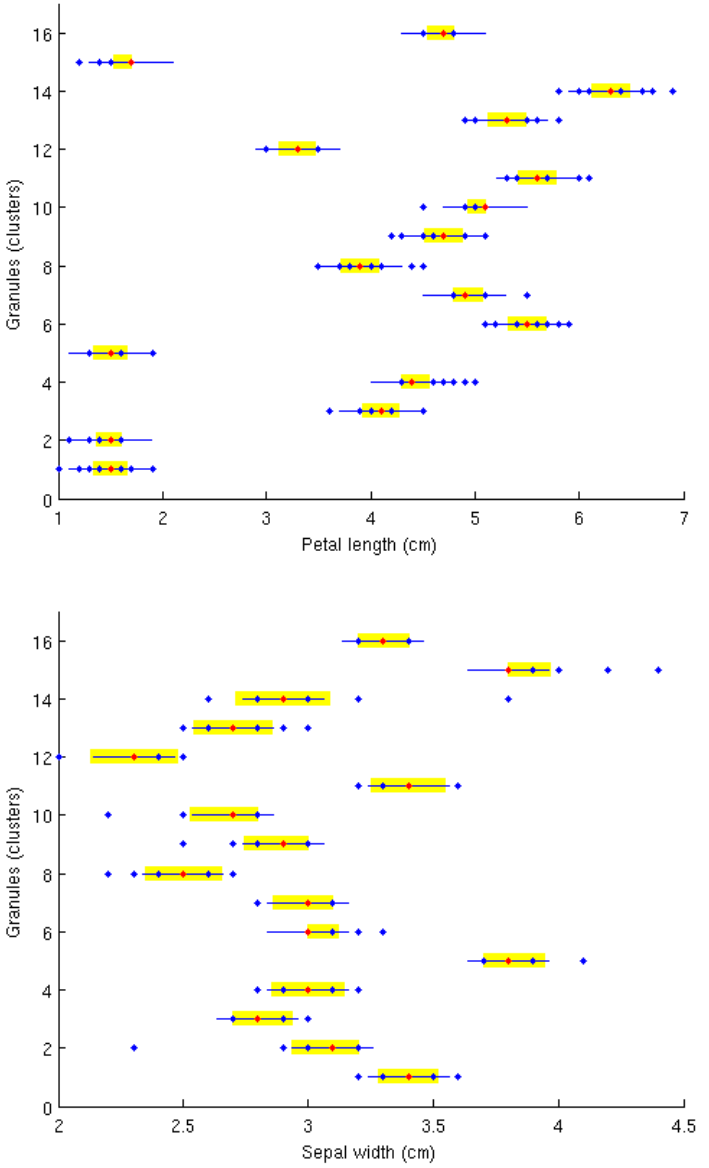


Fig. 18 Clusters found by the subtractive algorithm for the variables ´petal length´ and ´sepal width´, comparing the sigmas found by the algorithm (straight lines), against the lengths found by the principle of justifiable granularity (shaded areas), with a global $\alpha=5$

As for the granular gravitational algorithm´s results, the same variable was chosen with the same value for the specificity criterion. As shown in Fig. 19, the sigmas found by the clustering algorithm adapt more to the cardinality, yet this adaptation is not very noticeable. Comparing with the lengths found by applying

the principle of justifiable granularity, with an α of 0, the length of the granules are similar to those of the granular gravitational algorithm, yet differ in the sense that the lengths of a and b are shorter in general, this is more notable on the lower graph.

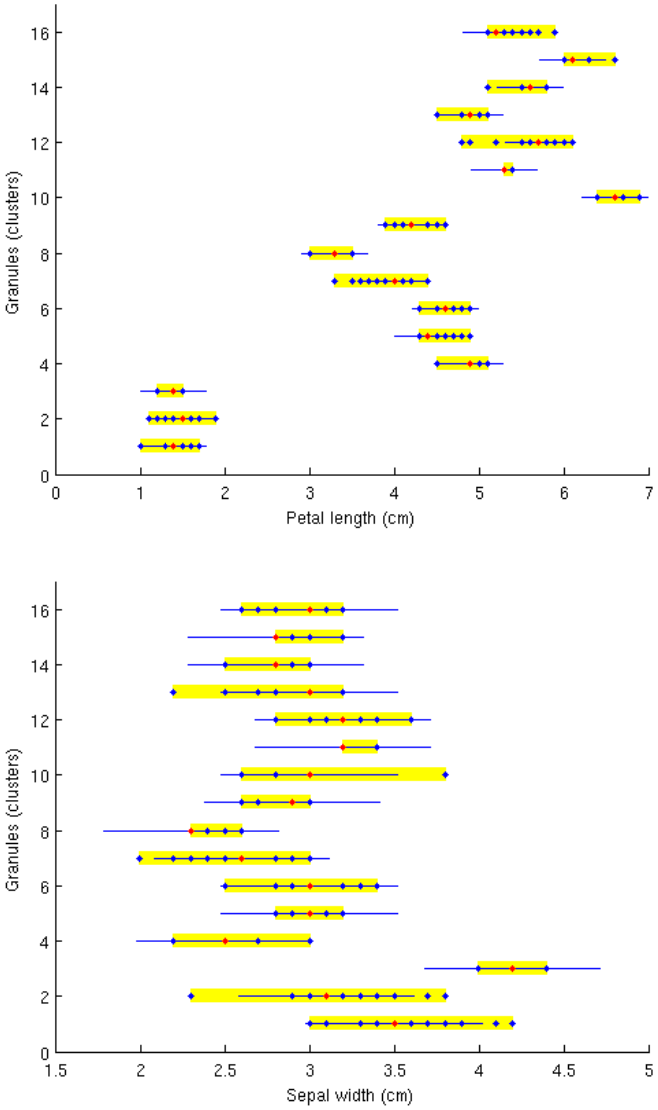


Fig. 19 Clusters found by the granular gravitational algorithm for the variable ´petal length´ and ´sepal width´, comparing the sigmas found by the algorithm (straight lines), against the lengths found by the principle of justifiable granularity (shaded areas) , with a global $\alpha=0$

In the next example, the same inputs and granules as before are used, but with a different value for the user criterion for specificity, in this case, 5 was used to equally compare with the previous algorithm (subtractive). As shown in Fig. 20, the same result as with the subtractive algorithm is obtained in the sense that it cannot compare since both the granular gravitational and the subtractive algorithm do not consider how specific or generalized a granule can be.

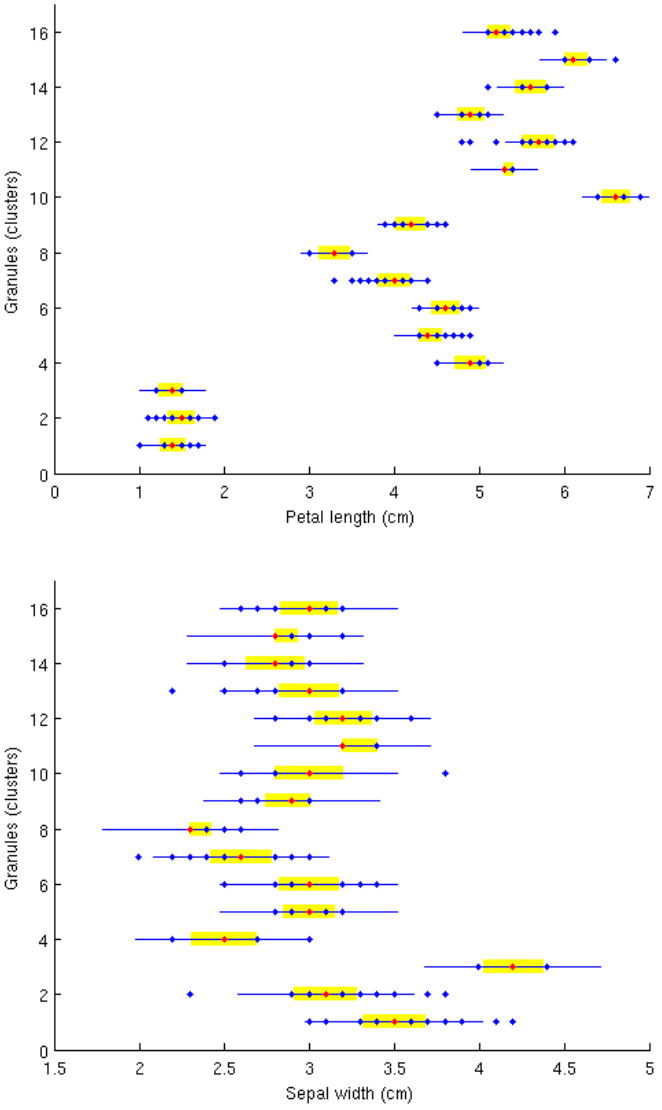


Fig. 20 Clusters found by the granular gravitational algorithm for the variable ´petal length´ and ´sepal width´, comparing the sigmas found by the algorithm (straight lines), against the lengths found by the principle of justifiable granularity (shaded areas) , with a global $\alpha=5$

In the next section, a much more detailed analysis regarding the cardinality, sigmas and results obtained by applying of the principle of justifiable granularity will be given.

4.3 Discussion

First of all, comparing both algorithm´s general performance, they obtain exactly the same number of clusters when they 100% identify these datasets, meaning their performance is very similar. The main difference in their results is that the subtractive algorithm´s sigmas are constraint to the same length of each cluster on the variable level, and the granular gravitational algorithm adapts its sigmas to global cardinality, meaning that the clusters at the variable level will not always be optimal.

Discussing now the intrinsic application of the principle of justifiable granularity in both algorithms, we first analyze the results obtained with the simplefit dataset. Since this comparison was done with a specificity criterion of 0 and 2. With zero, a full coverage of the lengths were expected, the subtractive algorithm had mixed results in this case, since some of its clusters were perfectly represented yet other clusters were very over-represented, having their length reach far beyond the cardinality; and the granular gravitational, in this case, did adapt more the cardinality of each cluster, in some cases ignoring isolated data points inside its own cardinality, yet in other cases its reach went far beyond its data limits. With the criterion set to 2, the change of coverage by each granule is very noticeable, but in this specific case, there is no direct comparison available with both algorithms, since they do not take into consideration how specific or not a granule can be.

Analyzing these results, we can assume that, in general, the subtractive algorithm obtains sigmas that are more specific in nature and the granular gravitational algorithm obtains sigmas that are less specific. Even with these differences, they both have very mixed results in the specificity length of their sigmas. This is considering that both algorithms generally give lengths of granules that are too long in direct comparison with the most generalized length given by applying the principle of justifiable granularity.

Directing attention to the iris dataset, these results show a different facet of the justifiability of granules, since the criterion which was chosen is 0 and 5, comparing the most generalized granule size, $\alpha=0$, and a granule size which is not very specific nor very general, $\alpha=5$, in fact, since α_{max} was not considered in this case, it is unknown if the value of 5 is the exact balance between specificity and numerical evidence of each granule, and considering that the cardinality of each granule affects the non-linear behavior of α_{max}, this value was simply chosen to demonstrate a higher value of specificity as to show how it affects the length of the granule.

Comparing both algorithms results, for the variables of the petal length and sepal width, in contrast to the length obtained after applying the principle of justifiable granularity, we can see that in both cases, the lengths of the sigmas obtained by the algorithms are similar in size, yet in some cases the granular gravitational sigmas adapt more to the cardinality of each cluster. With an α of zero, the

subtractive algorithm as well as the granular gravitational algorithm adjust similarly to the lengths given by applying the principle of justifiable granularity, but only in one of the inputs, *petal length* to be precise, yet the other input, *sepal width*, does not behave in this same manner, this could be explained by the form in how the cardinality is calculated, where both algorithms calculate them globally and not by input. The calculated length with a criterion of 5, in both cases, are similar in size for most clusters, and considering that both algorithms have a similar performance, we can assume that in this case, this similarity in obtained lengths by applying the principle of justifiable granularity is to be expected.

Reducing this discussion to fewer words, both algorithms have a very similar general performance with these datasets. The subtractive algorithm finds sigmas of the same length on the variable level and has mixed results with the specificity of each found cluster, while the granular gravitational algorithm finds sigmas that adapt more to the cardinality of each granule, but since this is done globally on each data point, it also has mixed results in respect to the cardinality of each cluster on the variable level, and its sigmas tend to be less specific that the sigmas found by the subtractive algorithm. And as already stated, both algorithms do not adapt nor consider how specific a granule should be.

5 Conclusion

5.1 Conclusions

In general, this analysis has given some insight into how, and possibly a why, these clustering algorithms obtain such acceptable results, when compared to the most generalized length given after applying the principle of justifiable granularity. Another finding is that they both obtain sigmas which are normally too generalized in respect to the cardinality of each information granule, resulting in the question of how would the final evaluation of identifications of the datasets would be affected if the lengths of each information granule was reduced according to the principle of justifiable granularity.

Both algorithms have mixed results when it comes to the intrinsic implementation of the principle of justifiable granulation, yet in general, they both find sigmas that are less specific that the most generalized possible length which can be found by applying this principle.

Considering the non-linear behavior of the specificity criterion, and how there is a fuzzy interval where its optimum value is found, an approximate heuristic has been proposed which finds such value, as well as multiple suggestions as to how generalize this value to limit the amount of user criterion values which must be given to each system of granules.

5.2 Future Work

Regarding the principle of justifiable granularity, a global specificity criterion could make it much easier to implement this theory, since having to choose this

value for two lengths, for every granule, and for every variable, is a non-realistic implementation.

Having obtained acceptable results with the application of the principle of justifiable granularity, integrating this into current clustering algorithms would greatly aid in the advancement of the general theory of granular computing, since the granules would now be meaningful and more specific to the needs of the problem to be solved.

Since a key component of the process of adjusting the lengths by applying the principle of justifiable granularity is a meaningful cardinality which properly reflects each input, a modification to current clustering algorithms to reflect this could greatly aid the end result of this principle.

More clustering algorithms can be tested in the same way the subtractive and granular gravitational algorithms were, as to measure more algorithms and asses their performance and creation of meaningful information granules.

Acknowledgement. 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: 314258.

References

1. Pedrycz, W.: Granular Computing - The Emerging Paradigm. Journal of Uncertain Systems 1, 38–61 (2007)
2. Castillo, O., Melin, P., Pedrycz, W.: Design of interval type-2 fuzzy mod-els through optimal granularity allocation. Applied Soft Computing 11, 5590–5601 (2011)
3. Bargiela, A., Pedrycz, W.: Toward a Theory of Granular Computing for Human-Centered Information Processing. IEEE Transactions on Fuzzy Systems 16, 320–330 (2008)
4. Han, J.H.J.: Approximation spaces in granular computing (2009)
5. Dai, J.D.J.: Algebraic method for granular computing (2008)
6. Yao, Y.: A Partition Model of Granular Computing. In: Peters, J.F., Skowron, A., Grzymała-Busse, J.W., Kostek, B.z., Świniarski, R.W., Szczuka, M.S. (eds.) Transactions on Rough Sets I. LNCS, vol. 3100, pp. 232–253. Springer, Heidelberg (2004)
7. Zadeh, L.A.: Fuzzy Sets. Information and Control 8, 338–353 (1965)
8. Zhang, Y.Z.Y., Zhu, X.Z.X., Huang, Z.H.Z.: Fuzzy Sets Based Granular Logics for Granular Computing (2009)
9. Pawlak, Z.: Rough sets. International Journal of Computer & Information Sciences 11, 341–356 (1982)
10. Sai, Y.S.Y., Nie, P.N.P., Chu, D.C.D.: A model of granular computing based on rough set theory (2005)
11. Yao, Y., Zhong, N.: Granular computing using information tables. In: Mining Rough Sets and Granular Computing 1997, pp. 1–23 (2002)
12. Zhang, L.Z.L., Zhang, B.Z.B.: Quotient space based multi-granular computing (2005)
13. Chen, Z., Lin, T.Y., Xie, G.: Matrix theory for Binary Granular Computing. IEEE (2011)
14. Chen, G.C.G., Zhong, N.Z.N., Yao, Y.Y.Y.: A hypergraph model of granular computing (2008)

15. Linda, O., Manic, M.: General Type-2 Fuzzy C-Means Algorithm for Uncertain Fuzzy Clustering. IEEE Transactions on Fuzzy Systems, 1 (2012)
16. Ester, M., Kriegel, H., Sander, J., Xu, X.: A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. Computer, 226–231 (1996)
17. Sibson, R.: SLINK: An optimally efficient algorithm for the single-link cluster method. The Computer Journal 16, 30–34 (1973)
18. Maroosi, A., Amiri, B.: A new clustering algorithm based on hybrid global optimization based on a dynamical systems approach algorithm. Expert Systems with Applications 37, 5645–5652 (2010)
19. Liu, X., Fu, H.: An Effective Clustering Algorithm With Ant Colony. Journal of Computers 5, 598–605 (2010)
20. Nock, R., Nielsen, F.: On weighting clustering. IEEE Transactions on Pattern Analysis and Machine Intelligence 28, 1223–1235 (2006)
21. Zaït, M., Messatfa, H.: A comparative study of clustering methods. Future Generation Computer Systems 13, 149–159 (1997)
22. Pedrycz, W.: The Principle of Justifiable Granularity and an Optimization of Information Granularity Allocation as Fundamentals of Granular Computing. Journal of Information Processing Systems 7, 397–412 (2011)
23. Chiu, S.L.: Fuzzy model identification based on cluster estimation. Journal of Intelligent and Fuzzy Systems 2, 267–278 (1994)
24. Newton, I.: Philosophiae Naturalis Principia Mathematica (1687)
25. Sanchez, M.A., Castillo, O., Castro, J.R., Rodríguez-Díaz, A.: Fuzzy granular gravitational clustering algorithm. In: North American Fuzzy Information Processing Society 2012 (2012)

Type-2 Fuzzy Logic Grammars in Language Evolution

Juan Paulo Alvarado-Magaña[1], Antonio Rodríguez-Díaz[1], Juan R. Castro[1], and Oscar Castillo[2]

[1] Autonomous University of Baja California,
Faculty of Chemical Sciences and Engineering, Tijuana, Baja California, Mexico
[2] Tijuana Institute of Technology, Tijuana, Baja California, Mexico
ardiaz@uabc.edu.mx

Abstract. This paper proposes a new approach to simulating language evolution; it expands on the original work done by Lee and Zadeh on Fuzzy Grammars and introduces a Type-2 Fuzzy Grammar. Ants in an Ant Colony Optimization algorithm are given the ability of embedding a message on the pheromone using a Type-2 Fuzzy Grammar. These ants are able to gradually adopt a foreign language by adjusting the grades of membership of their grammar. Results that show the effect of uncertainty in a language are given.

Keywords: ACO, Language Evolution, Type-2 Fuzzy Grammar.

1 Introduction

How humans developed language can be explain from two different perspectives. Some researchers believe that human linguistic abilities are innate ([7], [22], [23], [24]), this is what is called the Nativist point of view, which sustains that language is rooted in the brain's biology, in other words, the brain has an organ dedicated to language. The second point of view is known as Non-nativist, which claims that language is a byproduct of general intellectual abilities ([9], [26], [27]), this point of view doesn't assume the existence of certain characteristics in human biology but claims that language is an emergent response to evolutionary pressure applied to human ancestors.

Having these two points of view has created the Nativist versus Non-nativist divide. One example that supports Nativist is the existence of a critical period where children can learn a language; if a child is exposed to a language before the age of six he is able to learn it fluently regardless of intellectual and environmental circumstances, while an adult requires a greater amount of effort to learn a foreign language [21].

The origin of the Nicaragua Sign Language is another example of language nativism. The first school for deaf children in Nicaragua was opened in 1977, before then deaf people lived isolated with their immediate family and communicated using signs specific to their home. It wasn't until they opened these schools that

P. Melin, O. Castillo (Eds.): *Soft Computing Appli. in Optimization,* STUDFUZZ 294, pp. 265–286.
DOI: 10.1007/978-3-642-35323-9_11

children had the opportunity to socialize with others with similar abilities. Through this interaction they took each other's dialects and formed a more comprehensive vocabulary by which all children could communicate. Researchers noted that younger children could conjugate verbs in ways that older children could not and they also found that younger children were introducing new structures to the grammar as they were learning it ([17], [18], and [25]).

In [12] researchers found that in order to enhance a child's ability to learn adults adjust their language level when talking to them by using simpler grammatical forms and vocabulary, which suggest that humans are instinctively good at teaching language, which is a Non-nativist method of transmitting language.

To close the Nativist vs. Non-nativist division simulations of language evolution can be applied. As explained in [21] simulations should explore three different aspects: the Nativist vs. Non-Nativist perspectives, syntax evolution and finally the evolution of communication (usage of words as symbols).

There have been many attempts to simulate this phenomenon which have shown good results in formal languages ([2], [3], [13], [14], [15]); most common are those that use genetic algorithms ([3], [13], [15]) and neural networks ([14]), which will be further explain in later sections.

This paper explores grammar evolution and language acquisition using a Type-2 Fuzzy Grammar, which is an extension of Lee and Zadeh's original work in [19] on Fuzzy Grammars. A modified Ant Colony Optimization algorithm is used to simulate the social interactions required in a communicating society and results that support the validity of this approach are given, as well as a detailed explanation of how the simulation operates.

This paper has four sections organized as follows: section two is a collection of previous work. Section three is an explanation of the simulation itself; it includes a formal definition for Fuzzy Grammars as well as the modified ACO algorithm. Results are presented in section four and finally conclusions are shared in section five.

2 Previous Work

2.1 Emergent Vocal System in an Agent Society

This is an example of a Non-nativist simulation in which a language changes due to the constant interaction between participating agents. In [2], de Boer simulates a sound system organization using imitation games. To achieve this he proposes an experiment in which agents are added or removed from a population, each having a device that synthesizes sounds similar to the human voice and another device that receives and decodes sounds in real time.

During a game, an agent randomly generates a sound that is added to its lexicon, a second agent perceives this sound and tries to decode it and reproduce it, if the first agent finds that the imitation is sufficiently similar it gives positive feedback to the second agent. If however the sound is very different, the second agent tries to modify it to better match the original.

Using this method de Boer shows that a system of shared sounds can emerge through adaptive imitation games.

2.2 Development of Shared Symbols

Another case in which the simulation takes a Non-nativist approach is as follows. Hutchins and Hazelhurst in [14] tried to weaken the assumption that a lexicon used by agents must be provided externally by the researcher. They started with agents with no innate knowledge of a lexicon and prepared them with a finite set of situations they may experience. In order to identify them, each agent has a network that represents a situation as an activation of nodes (the network's connection weights are initialized at random).

Agents take turns performing roles as emitters and receivers. Both are presented with the same situation, and a pattern is formed within each network, the emitter then transmits its pattern to the receiver, and since both perceive the same situation the receiver can use the emitter's pattern as an additional input in order to adjust its own network.

After a certain number of iterations agents converge on a shared vocabulary, which means that when agents perceive the same situation the patterns produced by the network are equal for all agents.

2.3 The Bioprogram Hypothesis and the Baldwin Effect

Using a multi-agent system Briscoe simulated the propagation of language using genetic algorithms [3], this is an example of a simulation taking the Nativist point of view in consideration. Agents are created with a grammar capable of analyzing a sequence of categories or statements. The grammar is partly innate and contains categories with some of the ways they can be combined.

One agent generates a sequence and another tries to interpret it using its internal grammar, if a derivation tree can be generated then the interaction is successful.

The genetic part of the algorithm comes into play when agents are selected for reproduction. Each agent has a degree of fitness that depends on its success in interactions, the expressibility of the language and the amount of memory used to make derivations.

When an agent is unsuccessful in its interaction it can modify its grammar. The fittest agents are chosen for reproduction, the o spring's grammar is formed by both parents' grammar.

This method proves that through genetic assimilation one grammar will eventually dominate over an entire population.

2.4 Grammar Emergence in Communication Agents

Ikegami and Hashimoto demonstrated that a grammar can increase complexity, and therefore be more expressive, using an evolutionary method [13].

In this method, agents have an internal grammar they use to generate a string of ones and zeros that is then transmitted to other agents. Agents who receive the string try to use their internal grammar to replicate it, each agent keeps track of how many steps it needs to derive the string from its grammatical rules.

For selection, agents that interpret long chains in fewer steps randomly change their grammatical rules and are allowed to reproduce. Agents that fail to derive the string are removed from the population and since only the most successful agents are kept in the population this could be seen as another example of a Nativist simulation.

Through this process a regular grammar eventually transforms into a context-free grammar. It is known that a context-free grammar can generate more words than a regular grammar.

2.5 Evolution of Communication Agents in the Predator-Prey Pursuit game

Jim and Giles in [15] use the Predator-Prey pursuit game as a case study. During a game, agents communicate with each other by writing a string of ones and zeros to a message board. Once all messages have been posted, each agent reads all the strings and concatenates them into a single input that is passed to a finite state machine to determine the next move.

In order to evolve the language each predator is encoded in a chromosome. The initial generation usually doesn't capture the prey, but as generations advance the lengths of the strings grow and improve agent performance.

Jim and Giles find that there must be a minimum size language to solve such problems.

2.6 Comparison with Previous Work

As noted in [19], Fuzzy Grammars are a midway point between the precision of formal languages and the ambiguity of natural languages but the literature studied has presented simulations that only make use of traditional grammars. This is an opportunity to offer a new perspective to researchers by making use of Fuzzy Grammars.

Fuzzy Grammars provide much flexibility when defining a language. Production rules can be strengthen or weaken as necessary by raising or lowering their grades of membership in the grammar, making it easy to integrate new experimental rules without disturbing the established ones and also maintain seldom used rules without completely eliminating them. Rules with a high degree of membership are those that are endemic to the language, while those with a low degree of membership could be either new additions to the language or even part of the language that is going out of use.

Fuzzy Grammars could also model the acquisition of a foreign language by an individual. As the individual learns a new language he can add fuzzy production rules to his existing grammar and raise the grades of membership as he becomes

more skilled in the language. This characteristic can be further explored by using Type-2 Fuzzy Grammars.

Type-2 Fuzzy Grammars provide a degree of uncertainty that in this case will model an individual's mastery level of a new language. The production rules of an individual with little experience in a foreign language will have a high degree of uncertainty, while those individuals that are fluent in a language will have almost no uncertainty.

The proposed method is a new approach to the study of language evolution; it differs from those discussed earlier in that it doesn't use traditional crisp grammars but instead opts to use Type-2 Fuzzy Grammars. It also differs from previous experiments [1] in that Fuzzy Grammars are extended into a Type-2 Fuzzy Grammar.

3 Simulation

3.1 Ant Colony Optimization

To test the performance of Fuzzy Grammars a simulation in which communication is essential for success is needed; as such Dorigo's Ant Colony Optimization [10], also known as ACO, was selected as a case study.

ACO is a meta-heuristic inspired by the foraging habits of ants where social interaction is one of the most important aspects of ant survival; this interaction is modeled by Dorigo in the Ant Colony Optimization algorithm.

In ACO, individual ants leave a pheromone deposit to mark a solution in a problem space, doing so allows other ants to follow the pheromone trace and arrive at similar solutions. Pheromone intensity is either reinforced as more ants visit the same solution or it evaporates as bad solutions get discarded. Eventually the highest concentration is found around the best solutions. This experiment takes advantage of this feature by providing ants with a Fuzzy Grammar and allowing them to embed a message on the pheromone that other ants can understand.

The classic ACO algorithm uses a homogeneous colony to find a solution, which means that the colony is composed by only one type of ant (all ants are the same). The algorithm is extended in this experiment by including more than one group of ants who are segregated by a different Fuzzy Grammar; this allows multiple groups of ants to work on the same problem space to find multiple solutions.

During the simulation the colony will attempt to minimize De Jong's function, which has one global minimum.

If multiple groups of ants are placed on the same problem space to find De Jong's function's global minimum, then only one group will arrive at the solution because that group's pheromone will dominate the area in which the solution is located while other groups will reach solutions close to the global minimum, but since they can't follow the trace of the group at the solution due to differences in the Fuzzy Grammar, they will never reach the solution (figure 1). If however, all groups are able to assimilate each other's language through the use of Fuzzy Grammars, then all groups will eventually reach the global minimum.

The experiments show how two or more groups of ants are able to find the same solution to a problem by acquiring each other's language through the use of Fuzzy Grammars.

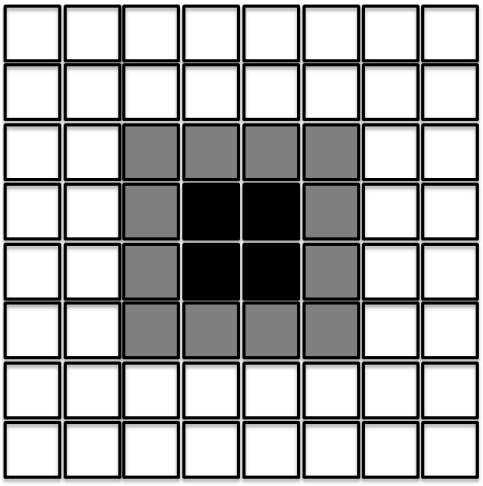


Fig. 1 The black pheromone dominates where the solution is located, the ants with the gray pheromone can't reach it

3.2 Type-2 Fuzzy Grammar

In 1975 Zadeh introduced a concept called Type-2 Fuzzy sets [30], which is an extension that permits the inclusion of uncertainty about the membership functions of traditional fuzzy sets. Ever since then there have been many contributions that use this concept, for example in Fuzzy Logic [5][20] and machine learning [4].

Lee and Zadeh's original definition for Fuzzy Grammars [19] is extended into a Type-2 Fuzzy Grammar by implementing the concept of a Type-2 fuzzy set as follows:

Definition 1. *A Type-2 Fuzzy Grammar is a quadruple G = (V_N; V_T; P'; S) in which V_T is a set of terminals, V_N is a set of non-terminals, P' is a set of fuzzy productions and S ε V_T is the set of starting variables*

The elements of *P'* are all productions in the form

$$\mu(\alpha \rightarrow \beta) = \omega', \omega' = [\underline{\omega}, \overline{\omega}] \tag{1}$$

Where $\alpha \rightarrow \beta$ expresses a re-writing rule, and are in $(V_N \cup V_T)$ and ω' is the grade of membership given in an interval [0, 1] of β given α. A fuzzy production where ω = [0, 0] is assumed to not be in *P'*.

A string of terminals *x* is said to be in the fuzzy language *L(G)* if and only if *x* is derivable from the starting variable *S*. The grade of membership of *x* in *L(G)* is given by

$$\mu_G(x) = [\underline{\mu}_G(x), \overline{\mu}_G(x)] \\ = [supmin(\underline{\mu}(S \to \alpha_1), \underline{\mu}(\alpha_1 \to \alpha_2), ..., \underline{\mu}(\alpha_n \to x)), \\ supmin(\overline{\mu}(S \to \alpha_1), \overline{\mu}(\alpha_1 \to \alpha_2), ..., \overline{\mu}(\alpha_n \to x))] \quad (2)$$

The uncertainty of a string of terminals *x* is given by $\Delta(x)$

$$\Delta(x) = \overline{\mu}_G(x) - \underline{\mu}_G(x) \quad (3)$$

The following is a sample of a Type-2 Fuzzy:

Let *G* be grammar $G = (V_N; V_T; P'; S)$ where:

$V_N = \{A, B, C\}$
$V_T = \{a, b, c\}$
$S = \{A\}$

The productions in *P'* are:

$\mu(A \to AB) = [0.75, 0.8]$
$\mu(B \to BC) = [0.6, 0.7]$
$\mu(A \to a) = [0.8, 0.9]$
$\mu(B \to b) = [0.7, 0.9]$
$\mu(C \to c) = [0.85, 0.95]$

The derivation of the string of terminals *"abc"* is as follows:

1. After applying the rule *"A → AB"* the resulting string is *"AB"* and $\mu(A \to AB) = [0.75, 0.8]$
2. After applying the rule *"B → BC"* the resulting string is *"ABC"* and $\mu(B \to BC) = [0.6, 0.7]$
3. After applying the rule *"A → a"* the resulting string is *"aBC"* and $\mu(A \to a) = [0.8, 0.9]$
4. After applying the rule *"B → b"* the resulting string is *"abC"* and $\mu(B \to b) = [0.7, 0.9]$
5. After applying the rule *"C → c"* the resulting string is *"abc"* and $\mu(C \to c) = [0.85, 0.95]$

Thus according to 2 $\mu_G(abc)$ is:

$\mu_G(abc) = [supmin(0.75, 0.6, 0.8, 0.7, 0.85), supmin(0.8, 0.7, 0.9, 0.9, 0.95)]$
$= [0.6, 0.7]$

And according to 3 $\Delta(abc)$ is:

$\Delta(abc) = 0.7 - 0.6 = 0.1$

3.3 The Algorithm

The simulation has two groups of ten ants (twenty in total, at the beginning ants share the same Fuzzy Grammar with members of their group (these grammars are given in Chomsky Normal Form). Both groups will attempt to minimize De Jong's function which is as follows:

$$f(x) = \sum_{i=1}^{n} x_i^2 \tag{4}$$

In this case ants will find the point in which equation 4 evaluates to zero in a two dimensional space, so $n = 2$.

The problem space is discretized and represented as a grid in order to allow ants to take steps in controlled increments (figure 2). In other words, ants walk across a plane such that each coordinate pair is used as input in De Jong's function. At the beginning of the simulation each ant is placed randomly on the grid to begin the search.

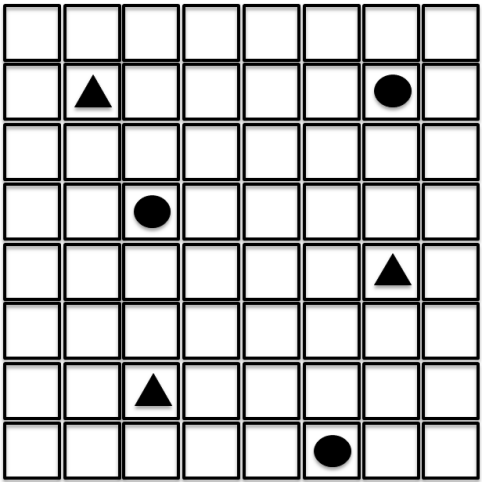


Fig. 2 Two groups of ants in a grid

As ants explore the problem space they will evaluate De Jong's function looking for the global minimum, finding it means they have found a food source and thanks to ACO other ants will be able to follow the pheromone trace to it.

When an ant leaves a pheromone deposit it also leaves a message generated by its Fuzzy Grammar. Each point in the problem space is a deposit that holds pheromones. Since many ants can pass over the same deposit, different pheromone levels are tracked. Before an ant can decide on a new position the dominant pheromone is determined, the pheromone with the highest intensity is the dominant one. Ants of either group will try to follow the dominant pheromone but only if

they understand the message embedded in it, in other words, if they were able to parse it using their Fuzzy Grammar.

As shown in Figure 3, ants will typically follow the trace of messages they understand. This characteristic causes some ants to reach the global minimum while others are kept at the edges. It is expected that after some iterations the number of ants that find the food source will increase as the language between both groups converges.

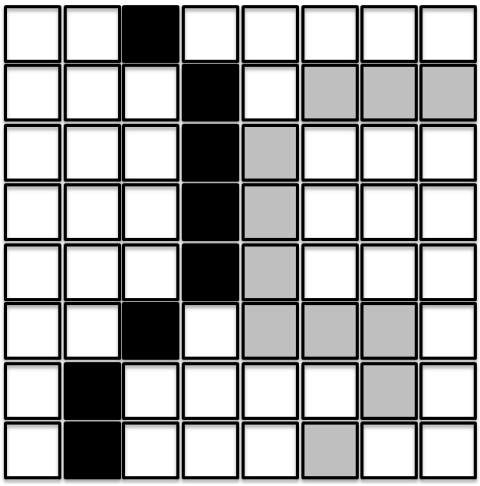


Fig. 3 Example of the pheromone trace of two different groups of ants

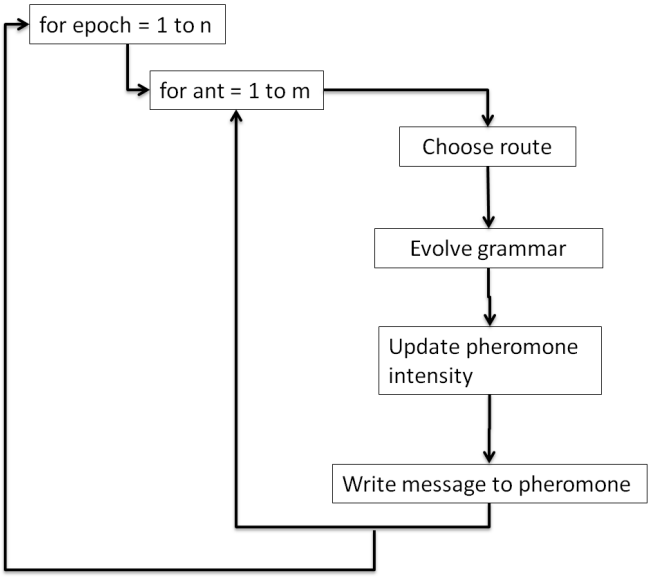


Fig. 4 Diagram of the simulation's steps

The simulation is divided into epochs, during each epoch ants must choose a route based on the dominant pheromone left behind during the previous epoch, they must also evolve their grammar and write messages to the deposits. Figure 4 is a chart of the steps made during each epoch, these steps are further explained in later paragraphs.

The pseudo code for the main body of the simulation is as follows.

```
1:        ant[n] is an array of n ants
2:        Each ant is assign one of two grammars given in Chomsky Normal Form
3:        Each ant is placed randomly in the two dimensional plane
4:        for epoch = 0 to maximum number of epochs do
5:        for i = 0 to n do
6:        ant[i] chooses a route
7:        end for
8:        update pheromone intensity
9:        update messages in the pheromone
10:       end for
```

An ant forms a route incrementally by stepping into a new deposit and then selecting from the eight possible adjacent spaces. The ant takes into account each deposit's pheromone intensity and its embedded message, in order to select the deposit the ant must be able to parse the message. Figure 5 gives an example of this selection. The circle in the middle of the grid is an ant, and it must choose which of the eight adjacent spaces to include in its route. Each space has a number representing the intensity of the pheromone. Spaces are marked with different colors to represent that the message in them are originated from different Fuzzy Grammars. A route is completed once it reaches a maximum number of allowed movements. Thus the probability of choosing deposit i is:

$$P(i) = \left(\frac{\tau(i)}{\sum_{j=1}^{8} \tau(j)} \right) (1 - \Delta(i)) \tag{5}$$

where $\tau(i)$ is the pheromone intensity at deposit i and Δ is the uncertainty of the message in i as given by 3.

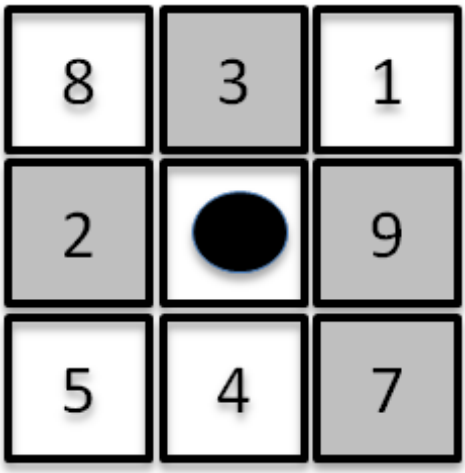


Fig. 5 Eight deposits with intensity

The following pseudo-code illustrates how a route is generated.

```
1:        route[n] is an array of deposits of size n
2:        best is the best solution found so far for
De Jong's function
3:        Set route[0] = best
4:        for j = 1 to n do
5:        position[8] is an array of eight possible
positions an ant can choose as its next step
6:        Set position with the message and intensity
of the dominant pheromone
7:        for i = 0 to 8 do
8:        Set route[j] = position[i] with probability
given by equation 5
9:        end for
10:       if route[j] is a better solution then best
when evaluated under De Jong then
11:       Set best = route[j]
12:       end if
13:       end for
```

In order to parse the embedded message an algorithm given by Cocke [6], Younger [26] and Kasami [14], also known as the CYK algorithm, is used. One common implementation of the CYK algorithm uses a three dimensional boolean matrix to store true or false values as the parsing tree is built. In order to calculate *G(x)* as given in 2 a modification that allows storage of grades of membership of each production rule is made to the CYK algorithm.

```
1:        S = a1a2...an is the message to parse of
length n
2:        G(S) is the grade of membership of S in the
fuzzy grammar
3:        The fuzzy grammar contains r non terminal
variables
4:        P[n, n, r] is a three dimensional matrix
with real values, each position stores both the lower
and upper grades of membership of a production rule
5:        for i = 0 to n do
6:          for all unit productions Rj →ai do
7:            P[0, i, j] = μ(Rj → ai)
8:          end for
9:        end for
10:       for i = 1 to n do
11:         for j = 0 to n - i do
12:           for k = 0 to i do
13:             for all productions RA → RBRC do
14:               if P[k, j, B] > 0 and P[i-k-1, j+k+1;
C] > 0 then
15:                 P[i, j, A] = min(μ(RA → RBRC), P[k,
j, B], P[i-k-1, j+k+1, C])
16:               end if
17:             end for
18:           end for
19:         end for
20:       end for
21:       G(S) = P [n    1; 0; 0]
```

Grades of membership for the fuzzy production rules have to be adjusted after the route has been selected. The grammar rules that were used are reinforced by raising their grade of membership, while those that weren't are lowered. As epochs pass the rules most often used will have a higher grade of membership while the least used will eventually cease to be part of the grammar.

```
1:        P [n] is an array of production rules of
size n
2:        μ(P[i]) is the grade of membership of pro-
duction P[i]
3:        α is the degree by which μ(P[i]) is lowered
or increased
4:        set α = 0.01
5:        for i = 0 to n do
6:          if Production P[i] was used during parsing
then
7:            Set μ̲(P[i]) = μ̲(P[i]) + (μ̲(P[i]) * α)
8:            Set μ̄(P[i]) = μ̄(P[i]) + (μ̄(P[i]) * α)
9:          else
10:           Set μ̲(P[i]) = μ̲(P[i]) - (μ̲(P[i]) * α)
11:           Set μ̄(P[i]) = μ̄(P[i]) + (μ̄(P[i]) * α)
12:         end if
13:       end for
```

Each ant is equipped with a method for deducing the grammar rules corresponding to a message it could not understand. This allows the ant to follow a pheromone trace of a different group, the advantage being that if a different group is more successful in finding food then the ant will eventually integrate itself into that group by adopting their language. The method is based on the CYK algorithm and is the same as described in [1] but the grade of membership for the new rules are given in an interval. The pseudo code is:

```
1:        S = a1a2...an is the message to parse of
length n
2:        The fuzzy grammar contains r non terminal
variables
```

```
3:        P[n, n, r] is a three dimensional matrix
with real values, each position stores both the lower
and upper grades of membership of a production rule
4:        for i = 0 to n do
5:          if the unit productions Rj → ai doesn't
exist then
6:            add μ(Rj → ai) = 0.01 to the fuzzy gram-
mar
7:            Set P[0, I, j] = 0.01
8:          end if
9:        end for
10:       for i = 1 to n do
11:         for j = 0 to n i do
12:           for k = 0 to i do
13:             if There doesn't exists a production RA
→ RBRC such that P[k, j, B] > 0   and P[i-k-1; j+k+1;
C] > 0 then
14:               add μ(RA → RBRC) = 0.01 to the fuzzy
grammar
15:               add μ̄ (RA →RBRC) = 0.0075 to the fuzzy
grammar
16:               Set P[i, j, A] = [0.0075; 0.01]
17:             end if
18:           end for
19:         end for
20:       end for
```

The last step of each epoch updates the pheromone intensity in all deposits, the following equation is used:

$$\tau(i) = (1-\rho)\tau(i) + \frac{Q}{1+f(j)} \tag{6}$$

Where $\tau(i)$ is the current pheromone intensity in deposit i, ρ is the forgetting factor, the constant Q is a value in the same order as $f(j)$ and $f(j)$ is the result of evaluating De Jong's function with the best solution found so far by ant j.

The pseudo code for updating the pheromone intensity is:

```
1:        ant[n] is an array of n ants
2:        deposits[m] is an array of pheromone depo-
sits of size m
3:        for i = 0 to n do
4:          for j = 0 to m do
5:            if ant[j] visited deposit[j] then
6:              Use equation 6 to set depo-
sit[j].intensity
7:              Set deposit[j].message =
ant[i].getMessage
8:            end if
9:          end for
10:       end for
```

4 Experiments and Results

4.1 Experiment 1

The first experiment was the control case, it consisted of two groups of ten ants each, each group's grammar had $\Delta(G) = 0$, in other words, the uncertainty in the language was eliminated. This simulation ran for fifty epochs with no evolution, the results are given in table 1. The first group minimized the function before the twentieth epoch, while the second group couldn't explore the problem space because the dominant pheromone wasn't understood. This experiment illustrates how a group of ants can quickly reach a solution if there's no uncertainty in the language and it also shows how a group that doesn't understand the dominant language is kept away from the solution.

Table 1 Two groups of ants over 50 epochs with no language evolution and $\Delta(G) = 0$

Epoch	Group 1	Group 2
5	7.152	17.704
10	1.1	17.704
15	0.288	17.704
20	0	17.704
25	0	17.704
30	0	17.704
35	0	17.704
40	0	17.704
45	0	17.704
50	0	17.704

4.2 Experiment 2

During the second experiment each group of ants had a grammar G with $\Delta(G) = 0.3$. In this case, language evolution was introduced to study its effects. As is shown in table 2, due to the level of uncertainty the first group managed to minimize the function on the fortieth epoch (instead of the twentieth as before), while the second group reached a lower solution than before but still didn't reach the global minimum. This simulation experienced a slowdown in the search due to the introduction of uncertainty.

Table 2 Two groups of ants over 50 epochs with language evolution and $\Delta(G) = 0.3$

Epoch	Group 1	Group 2
5	6.716	14.416
10	3.204	8.364
15	0.9	5.428
20	0.376	4.3
25	0.188	1.436
30	0.072	1.084
35	0.004	0.852
40	0	0.296
45	0	0.084
50	0	0.008

4.3 Experiment 3

In the third experiment the level of uncertainty is reduced to $\Delta(G) = 0.2$ in order to view the impact this would have on the results. Table 3 shows how it only took group one thirty epochs to reach a solution, but the level of uncertainty is still high enough that the second group couldn't reach it also. It's possible that with more epochs both groups would be able to find the solution.

Table 3 Two groups of ants over 50 epochs with language evolution and $\Delta(G) = 0.2$

Epoch	Group 1	Group 2
5	8.712	14.356
10	4.228	8.836
15	2.944	5.372
20	2.456	1.98
25	1.416	1.036
30	0	0.444
35	0	0.208
40	0	0.208
45	0	0.036
50	0	0.036

4.4 Experiment 4

For the fourth experiment $\Delta(G)$ was reduced even further to $\Delta(G) = 0.1$. The results in table 4 show both groups of ants reaching the same solution in fewer epochs than in previous experiments, the level of uncertainty was low enough to allow this and both groups converged on compatible languages.

Table 4 Two groups of ants over 50 epochs with language evolution and $\Delta(G) = 0.1$

Epoch	Group 1	Group 2
5	8.304	9.556
10	3.908	4.104
15	0.872	2.776
20	0.096	2.436
25	0.088	0.432
30	0	0.164
35	0	0
40	0	0
45	0	0
50	0	0

4.5 Experiment 5

In order to test the effect of a third language, the fifth experiment was carried out, it consisted of adding a third group to the simulation and $\Delta(G)$ for all three languages was maintained at 0.1. The results in table 5 show how all three groups where constantly reducing De Jong's function but weren't able to reach the global minimum. It is possible that adding a third language slowed down the convergence for all groups. In the following experiment the simulation was executed over more epochs to visualize when all languages converge on one.

Table 5 Three groups of ants over 50 epochs with $\Delta(G) = 0.1$

Epoch	Group 1	Group 2	Group 3
5	14.064	12.24	10.612
10	11.488	11.428	8.876
15	7.74	8.656	7.088
20	5.432	5.892	4.548
25	3.748	2.552	2.996
30	2.608	1.96	2.42
35	1.336	1.624	2.252
40	0.664	1.504	1.496
45	0.428	1.216	0.52
50	0.236	1.08	0.352

4.6 Experiment 6

The sixth experiment is similar to the fifth but the number of epochs was increased to two hundred. As expected, by allowing more time for the languages to converge the groups managed to reach the global minimum in 95 epochs.

Table 6 Three groups of ants over 200 epochs with $\Delta(G) = 0.1$

Epoch	Group 1	Group 2	Group 3
5	10.08	17.12	14.82
10	6.116	14.124	10.784
15	4.236	11.348	7.408
20	2.556	7.42	5.468
25	1.724	5.38	3.58
30	1.288	3.02	1.82
35	0.836	2.668	1.296
40	0.136	1.612	0.804
45	0.06	0.728	0.352
50	0.032	0.42	0.304
55	0.024	0.24	0.212
60	0.016	0.06	0.2
65	0.016	0.028	0.136
70	0	0.012	0.052
75	0	0.008	0.052
80	0	0.008	0.004
85	0	0.004	0.004
90	0	0.004	0
95	0	0	0

4.7 Experiment 7

In the seventh experiment an additional group was added and $\Delta(G)$ was maintained at 0.1, this was done to see if by adding a fourth language the behavior

Table 7 Four groups of ants over 50 epochs with $\Delta(G) = 0.1$

Epoch	Group 1	Group 2	Group 3	Group 4
5	14.6	11.256	13.884	13.884
10	12.284	9.48	11.94	12.472
15	9.64	8.172	9.592	10.596
20	6.58	7.848	7.28	9.516
25	5.576	6.06	6.028	8.176
30	4.584	4.268	5.208	6.72
35	4.236	3.66	3.804	6.204
40	3.288	3.352	3.544	4.816
45	2.156	2.808	2.404	4.052
50	2.056	2.292	1.948	3.104

experience so far would still manifest. This experiment behaved similarly to experiment 5, note that the solutions on the final epoch (Table 7) are worse in this experiment than in experiment 5, it is possible that by adding the fourth group the convergence once again slowed down.

4.8 Experiment 8

The final experiment had the same conditions as the previous experiment but ran for two hundred epochs. In this case, two groups converged on the global minimum before the one hundredth epoch while the other two found the minimum before the one hundred and fiftieth epoch. As was expected, more epochs were required to allow all languages to converge.

Table 8 Four groups of ants over 200 epochs with $\Delta(G) = 0.1$

Epoch	Group 1	Group 2	Group 3	Group 4
5	12.928	20.108	14.4	14.22
10	11.352	18.828	11.92	12
15	10.276	14.944	10.972	10.576
20	7.348	11.792	8.584	8.824
25	5.672	6.916	6.308	7.02
30	4.204	5.148	5.44	6.192
35	3.08	4.42	4.472	4.936
40	2.86	2.54	4.368	3.476
45	2.588	2.264	3.996	2.596
50	1.952	1.092	3.672	2.028
55	1.036	0.288	2.928	1.716
60	0.728	0.252	2.352	1.508
65	0.532	0.156	1.788	0.936
70	0.224	0	1.688	0.764
75	0.116	0	1.608	0.624
80	0.116	0	1.608	0.58
85	0.056	0	1.1	0.376
90	0.02	0	1.052	0.256
95	0	0	1.052	0.1
100	0	0	1.02	0.1
105	0	0	0.7	0.1
110	0	0	0.604	0.1
115	0	0	0.596	0.1
120	0	0	0.452	0.1
125	0	0	0.288	0.064
130	0	0	0.288	0.064
135	0	0	0.2	0.04
140	0	0	0.2	0.008
145	0	0	0.2	0.004
150	0	0	0.08	0
155	0	0	0	0

5 Conclusions

The available literature was researched and it was found that Fuzzy Grammars haven't been explored as a means to study Language Evolution. Since there's a divide between Nativist and Non-Nativist scientist regarding the origin of human language, there are plenty of opportunities to provide them with the tools they need, such as Fuzzy Grammars, to expand their research.

By formally defining Type-2 Fuzzy Grammars the work done in [1] and [19] was expanded in order to simulate the uncertainty experienced by an individual learning a new language.

In order to test the feasibility of Type-2 Fuzzy Grammars, ACO was modified in several ways, first each ant was equipped with a Type-2 Fuzzy Grammar that allows it to parse and create messages. Second, the pheromone was modified to carry a message, thus an ant must be able to parse the message with a low uncertainty in order to follow the trace. Different grammars where used as a way to distinguish between multiple groups of ants exploring the same problem space.

The experiments show that if multiple groups of ants attempt to solve the same problem and they are unable to understand the each other's language, only one will reach the solution; but if all groups are able to assimilate each other's language then both will converge on the solution. Also, uncertainty plays an important role in finding a solution; experiments show that a large enough uncertainty will slow down the search.

The experiments presented here can be divided into two groups. In the first group of experiments gradually uncertainty was introduced to illustrate its impact in finding a solution. The first experiment had no uncertainty and no language evolution and the results show a fast solution by one group. Language evolution and uncertainty were then introduced to the ant population (experiments two, three and four) and it was shown that the higher the uncertainty the longer it takes for ants to reach the global minimum in De Jong's function.

The second group of experiments demonstrate how a search for a solution is furthered delayed by adding even more languages to the ant population, but how given enough time all ants will reach the solution and converge on a compatible language.

This leads to conclude that Type-2 Fuzzy Grammars are a viable tool in language evolution research.

6 Future Work

The question of language origin is a divisive one, and even though it may never be known for certain how humans developed language, there is a modern example of language emergence in the case of the Nicaragua Sign Language ([17], [18] and [25]). This example can be the foundation of future experiments in which Fuzzy Grammars can be employed to simulate emergence. Also, it can be used to study the use of words (evolution of syntax) and the use of symbols (evolution of communication).

The experiments presented here are only an abstraction of problem solving in a social environment, but real world social and economical phenomenon can be modeled as well, such as migration and social integration where language proficiency can be linked to higher wages and better work positions ([6], [11] and [28]).

In [11] it's suggested that proficiency in English is associated with eighteen to twenty percent higher earnings in the UK and that language proficiency is an important factor in determining probability of employment.

Chiswik and Miller in [6] found that an immigrant's socioeconomic status and integration into the culture of a host country is influenced by the languages an immigrant can speak and the level of proficiency of the destination language. In other words, the value of a worker with language and profession skills is greater than a worker of similar professional skills but with a language deficiency. The extent to which an immigrant practices the host language is determined by factors such as age of migration, educational attainment and duration of residence (persons that migrated on a younger age are more likely to speak the host language). They also found that immigrants in areas with a large concentration of people that speak their language of origin are less likely to speak the host language.

Modeling these social interactions with Fuzzy Grammars could give new insight as well as provide knowledge that could guide future policies regarding immigrant groups.

Acknowledgements. We would like to express our gratitude to the Masters and PhD in Science and Engineering program (MyDCI) of the Division of Graduate Studies and Research of the Autonomous University of Baja California, Mexico and the Computational Intelligence group in the Faculty of Chemical Sciences and Engineering at the UABC campus in Tijuana.

We would also like to thank our colleagues at the Tijuana Institute of Technology.

Finally, we are very grateful for the financial support provided by our sponsors in the National Council for Science and Technology of Mexico (CONACYT).

References

1. Alvarado-Magaña, J.P., Rodríguez-Díaz, A., Castro, J.R., Castillo, O.: Simulation of Language Evolution using Fuzzy Grammars. In: 31st Annual NAFIPS Meeting (2012)
2. De Boer, B.: Emergent Vowel Systems in a Population of Agents. In: Proceedings of ECAL 1997 (1997)
3. Briscoe, T.: Language Acquisition: The Bioprogram Hypothesis and the Baldwin Effect. MS, Computer Laboratory, University of Cambridge (1997)
4. Castro, J.R., 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)
5. Castillo, O., Melin, P., Pedrycz, W.: Design of interval type-2 fuzzy models through optimal granularity allocation. Appl. Soft Comput. 11(8), 5590–5601 (2011)
6. Chiswick, B.R., Miller, P.W.: A Model of Destination-Language Acquisition: Application to Male Immigrants in Canada. Demography 38(3) (2001)
7. Chomsky, N.: Government and Binding, Foris, Dordrech (1981)

8. Cocke, J., Schwartz, J.T.: Programming Languages and Their Compilers. Courant Institute of Mathematical Sciences, New York (1970)
9. Deacon, T.: The Symbolic Species. Penguin, London (1997)
10. Dorigo, M., Di Caro, G.: The Ant Colony Optimization Meta-Heuristic. In: New Ideas in Optimization. McGraw-Hill (1999)
11. Dustmann, C., Fabbri, F.: Language Proficiency and Labour Market Performance of Immigrants in the UK. The Economic Journal 113(489) (2003)
12. Fernald, A., Simon, T.: Expanded intonation contours in mother's speech to newborns. Developmental Psychology 20, 104–113 (1984)
13. Hashimoto, T., Ikegami, T.: Emergence of net-grammar in communicating agents. BioSystems 38, 1–14 (1998)
14. Hutchins, E., Hazelhurst, B.: How to invent a lexicon. The development of shared symbols in interaction. In: Artificial Societies: The Computer Simulation of Social Life. UCL Press, London
15. Jim, K.-C., Lee Giles, C.: Talking Helps: Evolving Communicating Agents for the Predator-Prey Pursuit Problem. Artificial Life 6(3), 237–254 (2000)
16. Kasami, T.: An efficient recognition and syntax analysis algorithm for context-free languages. Technical Report AFCRL-65-758, Air Force Cambridge Research Laboratory, Bedford, Massachusetts (1965)
17. Kegl, J.: Conference Report: Linguistic Society of America Meeting. Signpost 7(1) (1994)
18. Kegl, J.: The Nicaraguan Sign Language Project: An Overview. Signpost 7(1) (1994)
19. Lee, E.T., Zadeh, L.A.: Note on fuzzy languages. Information Sciences 1, 421–434 (1969)
20. Mendel, J.M.: Uncertain Rule-Based Fuzzy Logic Systems: Introduction and New Directions. Prentice-Hall, Upper Saddle River (2001)
21. Perfors, A.: Simulated Evolution of Language. Journal of Artificial Societies and Social Simulation 5(2) (2000)
22. Pinker, S., Bloom, P.: Natural language and natural selection. Behavioural and Brain Sciences 13, 707–784 (1990)
23. Pinker, S.: The Language Instinct. William Morrow and Company (1994)
24. Pinker, S.: Why the child holded the baby rabbits: A case study in language acquisition. In: Language: An invitation to Cognitive Science, 2nd edn., vol. 1, pp. 107–133. MIT Press, Cambridge (1995)
25. Senghas, A.: The Development of Nicaraguan Sign Language via the Language Acquisition Process. In: BUCLD 19: Proceedings of the 19th Annual Boston University Conference on Language Development. Boston Cascadilla Press (1995)
26. Shipley, E., Kuhn, I.: A constraint on comparisons: equally detailed alternatives. Journal of Experimental Child Psychology 35, 195–222 (1983)
27. Tomasello, M.: First words: A case study of early grammatical development. Cambridge University Press, Cambridge (1992)
28. Veglery, A.: Differential Social Integration among First Generation Greeks in New York. International Migration Review 22(4) (1988)
29. Younger, D.H.: Recognition and parsing of context-free languages in time n3. Information and Control 10(2), 372–375 (1967)
30. Zadeh, L.A.: The Concept of a Linguistic Variable and Its Application to Approximate Reasoning. Information Sciences 8, 199–249 (1975)

Methodology of Design: A Novel Generic Approach Applied to the Course Timetabling Problem

Soria-Alcaraz Jorge A., Carpio Martin, Puga Héctor, Terashima-Marin Hugo, Cruz Reyes Laura, and Sotelo-Figueroa Marco A.

División de Estudios de Posgrado e Investigación, Instituto Tecnológico de León, León Guanajuato, México
{Soajorgea,masotelof}@gmail.com, jmcarpio61@hotmail.com, pugahector@yahoo.com

Abstract. The Course Timetabling problem is one of the most difficult and common problems inside a university. The main objective of this problem is to obtain a timetabling with the minimum student conflicts between assigned activities. A Methodology of design is a framework of solution applied to a heuristic algorithm for timetabling problem. This strategy has recently emerged and aims to improve the obtained results as well as provide a context-independent layer to different versions of the timetabling problem. This methodology offers the researcher the advantage of solving different set instances with a single algorithm; which it is a new paradigm in the timetabling problem state of art. In this chapter the proposed methodology is described and tested with several metaheuristic algorithms over some well-known set instances, Patat 2002 and 2007. The main objectives in this chapter are: to show the construction of a two-phase algorithm based in a novel generic approach called design methodology and to find which metaheuristic algorithm shows a better performance in terms of quality. The design methodology generates set of generic structures: MMA, LPH, LPA and LPS. These structures build an independent context layer, so the two-phase algorithm only needs to solve the problem coded into them. No further specification or explicit codification of any problem-dependent constraint is needed inside the algorithm. This guarantee that in order to solve other instance of the Course timetabling problem, only it is needed the translation of the incoming instance into the proposed structures. With these structures the proposed methodology searches, in the first phase, for at least one feasible solution (a solution that has no conflict in the hard constraints). In a second phase the methodology utilizes this feasible solution in order to intensify the search around it, looking for the *perfect solution* (a solution with no conflict in any constraint hard or soft). Precisely for this two phases it is necessary the use of metaheuristic algorithms. This kind of algorithms does not guarantee to obtain the

P. Melin, O. Castillo (Eds.): *Soft Computing Appli. in Optimization,* STUDFUZZ 294, pp. 287–319.
DOI: 10.1007/978-3-642-35323-9_12

global optima but offers an opportunity to obtain a good solution in a reasonable time. The algorithms chosen to be tested along with the design methodology are from the area of evolutionary computation, Cellular algorithms and Swarm Intelligence. It is important to say that there exist several previous implementations of these metaheuristic algorithms over CTTP problems but this is the first time that these algorithms will be evaluated under a generic approach like the Methodology of design. Finally our experiments use some non-parametric statistical tests like Sing test, Kruskal-Wallis test and Wilcoxon signed rank test in order to identify the metaheuristic algorithm with the best performance over the course timetabling problem using the Methodology of Design.

1 Introduction

The timetabling problem is one of the most difficult, common and diverse problems inside the industry. This problem tries to assign several activities into a *Timelsots* making a *Timetabling*. The main objective of this problem is to obtain a timetabling with the minimum conflicts between assigned activities [21].

The timetabling problem is a wide problem that can be seen on different places for example: airports, train stations, delivery companies…etc. in this chapter the timetabling problem is seen from the point of view of an superior educational institution or university.

There are diverse timetabling problems inside an university as the ones described by Adriaen et.al [1]:

A) **Faculty Timetabling:** This timetabling problem assigns teachers to a set of specific subjects or topics.
B) **Class-teacher Timetabling:** This timetabling problem assigns subjects to a fixed and specific group of students.
C) **Classroom-assigment:** This timetabling problem ensures that every pair teacher-subject has an assigned classroom.
D) **Examination Timetabling:** This timetabling problem assigns one-time events like final exams or especial lectures to individual students.
E) **Course Timetabling:** This timetabling problem assigns subjects to individual students.

This paper focus on the *Course timetabling problem* (CTTP). In this problem is assigned a set of subjects to individual students looking for minimum conflicts, usually time-conflicts, between the events.

Like most timetabling problems, the *Course timetabling*has been reported as a NP-Complete problem [13] [31]. This is commonly attributed to the huge combinatorial explosion of possible events assigned into timeslots as well as the constraints that each university uses in the course timetabling creation. Due to this complexity and the fact that, even now many of these course timetabling constructions are making by hand; It is necessary to automate the timetabling construction process this also will improve the performance of the solutions reached by the human expert [21].

If we consider that every university usually needs to implement a new course timetabling algorithm in order to achieve a good solution (basically due its internal policies) then it exists an important obstacle; for example an algorithm that solves the problem in a university may not be able to provide at least a feasible solution for another university.

We called to this situation: *a high dependency between the problem instance and the solution algorithm.* This is not a new problem; it has been documented by Shaerf [23]. This situation means that there exist a big dependency between a problem instance and an algorithm highly specialized and adapted to solve it. The main problem is that, if it is necessary to change something in the original problem instance (due university policies) or if it is necessary to solve another university, then is highly probably that the specialized algorithm cannot obtain a good solution or in the worst case that algorithm cannot been at least executed in that new environment.

In the worst case the researcher usually needs: to code a new algorithm, to make more experiments in order to find the useful strategies and finally to solve his new instance. Basically that means for the researcher the return to the design table to build and test a new solver.

In this context a new methodology of solution has emerged, the methodology of design [25] [26] [27] [28]. Thismethodology of design builds a layer where the university policies from the original course timetabling input are traslatedto a set of generic structures for its treatment by means of metaheuristic algorithms. This design provides a context-independent layer allowing metaheuristic algorithms to work and solve several course timetabling problems without using any explicit constraints.

The main advantage of this approach is that if exist a change in the original course timetabling instance it is only necessary to translate this new instance to the set of generic structures in order to solve it. This means that the researcher do not need to utilize time and efford to construct a new solution algorithm. he only needs to apply the generic approach to obtain a solution with reasonable quality.

Other advantage of this approach is that it can be used as an benchmarking framework. In the state of art of the course timetabling problem commonly appears the problem of how to compare two algorithms that solve CTTP. This is an important problembecause the compared algorithms usually are higly dependent to its own different instances, so is too hard to find a way to identify the best algorithm for an specific objetive. This problem can be relaxed by the application of the design methodology; since this approach offers a generic framework of solution, two algorithms that utilizes the same independent layer could be easily compare in order to determine which is better in terms of student conflicts.

The design methodology uses metaheuristics in order to find at least a feasible solution, this kind of solution means that it does not have violations in any hard constraints. Once this feasible solution is achieved it is necessary to intensify the search around it, in order to obtain a perfect solution. The perfect solution means a solution that does not have any violation in all the constraints hard or soft. This

methodology of solution is called Two-phase algorithm. The present work is only focused in the feasibility phase, leaving for future work the application of a second phase.

The two phase algorithms has been utilized in previous works [12][15][33] these algorithms have several advantages and disadvantages shown by Lewis [21]. The main advantages are:

A) In early stages of the solution process the algorithm can detect if exist some constraints in conflict. So it is posible to identify if the problem do not have a solution due a fail in the constraint design.

B) Once the feasible solution is achieved this solution can be applied to the real work at any moment, it is not necesary to finish the algorithm execution to ensure to have at least an applicable solution.

On another hand the main disadvantage of this kind of algorithms is that this kind of algorithm needs a way to ensure the modification of a feasible solution without produce any violation in the previously solved constraints. The methodology of design offers a way to deal with this disadvantage: due the usage of generic structures the algorithm will never produce an unfeasible solution. The main effort of the design methodology is to find the perfect solution inside the bounds of feasibility provided by the generic structures. These generic structures will be explained on the next section but can be summarized as:

A) **MMA:** Generic structure that shows the possible number of conflicts between two events if these two events are assigned into the same timeslot.

B) **LPH:** Generic structure that represents the time domain of each subject, this structure shows the possible time-related assignations that every subject needs to satisfy.

C) **LPA:** Generic structure that represents the space domain of each subject, this structure shows the possible space-related assignations that every subject needs to satisfy.

D) **LPS:** Generic structure that represent the demands for every student, in the practice this structure shows the usage of time proposed by the student itself.

The first 3 structures represent the hard constraints and the LPS structure represents the soft constraints. A complete CTTP assignation is obtained when the proposed methodology achieve a solution that satisfy all the structures. In order to search inside these structures a metaheuristic algorithm is used, The metaheuristics algorithms has been characterized for offer good results in a reasonable time. There exist a huge variety of metaheuristics applicable to a wide range of problems, but the no-free lunch theorem [32] indicate us that there no exist such a metaheuristic capable to give a good solution for every possible problem. The selection of the best metaheuristic algorithm has then a great importance over the final performance for the generic proposed approach. This paper shows besides the generic design methodology, a comparative study between several different metaheuristics and its performance over a generic approach for the CTTP.

The algorithms chosen to be tested along with the design methodology are: Classic Genetic Algorithm (sGA), a novel Frequency Genetic Algorithm (fGA), Eclectic Genetic Algorithm (eGA), Cellular Genetic Algorithm (cGA), Differential Evolution (DE/rand/1) , Particle Swarm Optimization (PSO) and Great Deluge Algorithm (GDA).It is important to say that there exist several previous implementations of these metaheuristic algorithms over the CTTP problem[24][27][15][26][7] but this is the first time that these algorithms will be evaluated under a generic approach like the Methodology of design.

All the experiments will be realized over a set of well-known and referenced instances: PATAT 2002 and PATAT 2007. The chapter is organized as follows. Section 2 presents the design methodology for the course timetabling, The metaheuristics chosen for comparison and its justification. Section 3 contains the experimental setup, results, analysis and discussion. Finally Section 4 include some conclusions and future work.

2 Methodology of Design

In the literature it can see that there is a problem with the diversity of course timetabling instances due different policies. This situation directly impacts in the reproducibility and comparison of timetabling algorithms [23]. The state of art indicates some strategies to solve this problem. For example, more formal problem-formulations as well as the construction of benchmark instances [21] are methods constantly used. 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 [23]. In this context, it is proposed a new context-independent layer to the course timetabling resolution process. This new layer integrates timetabling constraints into four basic structures MMA matrix, LPH list and LPA list and LPS list (explained in subsequent sections). This approach has been applied together to Genetic Algorithms with direct representation [26] and Hyperheuristics [27] with encouraging results over real university instances at Leon Institute of Technology, but so far it has not been applied to a 2-phase algorithm or an international testing benchmark like ITC 2002 and ITC 2007 provided by PATAT. The 2-phase algorithm is a kind of timetabling solver[21]. This solver works with the timetabling problem in two phases. The first one tries to solve only the Hard constraints i.e. Constraints that cannot be violated or the solution simply could not be applied to reality. Once the feasible solution is achieved the algorithm enhances it usually by means of a heuristic local search in order to obtain a Perfect Solution i.e. A Solution that solves both Hard and Soft constraints. In this research it is used a two-phase algorithm with several Metaheuristics as well as a generic approach in order to apply the first phase to both ITC 2002 and ITC 2007.

These set of instances ITC 2002 and ITC 2007 belongs to the first and second timetabling competition sponsored by WATT and PATAT (Practice and Theory of

Automated Timetabling). ITC 2002 as well as ITC 2007 has been recognized as some of the most important course timetabling instances in the world.

2.1 Problem Definition

A clear and concise definition of the CTTP is given by Conant-Pablos [12]: A set of events(courses or subjects) $E = e_1, e_2, \ldots, e_n$ is the basic element of a CTTP. Also there are a set of periods of time or timeslots $T = t_1, t_2, \ldots, t_s$, a set of places (classrooms) $P = p_1, p_2, \ldots, p_m$, and a set of agents (students registered in the courses) $A = a_1, a_2, \ldots, a_o$. 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 an assignment is a quadruple (e, t, p, S). A timetabling solution is a complete set of n assignments, one for each event. It is important to notice that each assignation must satisfy a set of $C = \{c_1, c_2, c_3, \ldots c_k\}$ constraits usually defined by each university. In the practice the constraint set is divided into hard constraints and soft constraints. The hard constraits must be satisfy and the soft constraits are prefered to be satisfy. This problem has been documented to be at least NP-complete problem [13] [31].

As the reader can see, the construction of the constraint set is arbitrary and depends exclusively for each university itself. The high number of possible constraints makes very difficult to design an algorithm capable to solve all the posibilities. Fortunately Corne et.al[11] groups the mayority of these constraints into 5 classes, so the proposed methodology of design utilizes this classificacion in order to reach a high rate of generalization. The five classes are:

A) **Unary Constraints:** These constraints involves only one variable. *event* e_2 *must be assigned into timeslot* t_7.
B) **Binary Constraints:** These constraints involves two variables. *Event* e_2 *must be asigned into an timeslot before/after that the timeslot assigned by the event* e_6
C) **Capacity Constraints:** These constraints involves the space-domain of each variable. *The classroom* p_8 *only can use by 20 students at the same timeslot.*
D) **Event Spread Constraints:** These are constraints that concern requierements such as the "spreading-out" or "clumping-together" of events within the timetable in order to ease student/teacher worload, and/or agree with university's timetabling policity (usually soft constraints).
E) **Agent constraint:** These are constraints that are imposed in order to promote the preferences of people who will use the timetables.(it can be soft or hard).

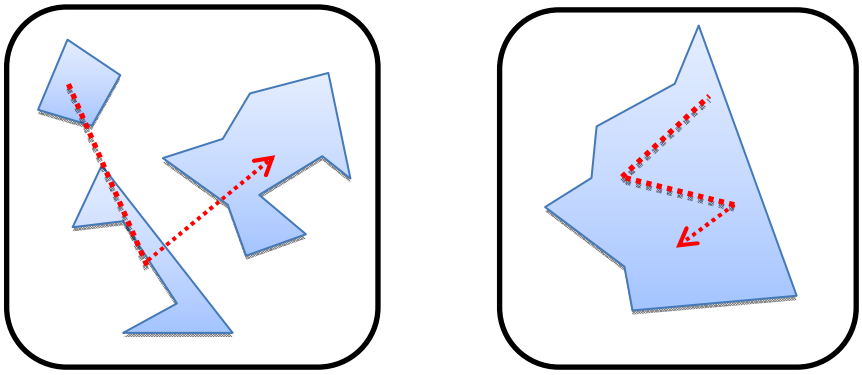
It is considered that these main five classes represents a wide range of CTTP instances, so the generic approach is based in this five classes in order to offer a good rate of generalization.

2.2 *Methodology of Design for the Course Timetabling Problem*

As seen on previous section 2.1 there exist several different types of constraints inside a CTTP problem. This situation makes difficult to apply a previously adapted algorithm to a new CTTP instance. This work propose the usage of a context-independent layer that transforms the original inputs/constraints into a set of generic structures, so theoretically; it does not matter the configuration of the original CTTP instance, because once translated a generic algorithm can solve the instance using generic structures.

This layer of context-independency is named "Methodology of design"[27][28] and its principal objetive is to solve *bydesign* the mayor number of constraints in order to build a search space only with feasible solutions, where a heuristic strategy can search for a solution working only with a minimun number of variables.

The expression "by design" means that by the use of generic structures it is posible to build a search space where all the constraints appears in an implicit way and all the posible solutions in that space were feasible to most of the original constraints. The main effort of this approach is to search inside this space of feasibility in order to find the optimal solution, where all the constraints are satisfied.



(a) Original CTTP (b) Generic Structures CTTP

Fig. 1 In the original CTTP problem the feasible reagions are spread over the search space, solver needs to manage unfeasible solutions in order to travel between feasible regions. In the Generic structures space the feasible region is only one, so at any moment the algorithm have a feasible solution.

In order to reach a feasible space like the shown on Fig 1, several generic structures are needed, The structures are MMA matrix, LPH list, LPA list and LPS list. The first 3 represents the hard constraints and the last one represent the soft constraints. The definition of these structures are:

MMA Matrix: This 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. This matrix shows the number of students that demands simultanously the row subject and the column subject. An example of this matrix can be seen on the Figure 2 and the algorithm utilized for its construction on Algorithm 1.[25][26]

	ACM0403	SCM0414	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

Fig. 2 MMA matrix

Algorithm 1. MMA Construction

Require: int **N**= Students, int[][] **LD**= Students Demands
1: **for***i=0***to** N **do**
2: *Starr = LD[i]*
3: **for***j=0***to***size(Starr)***do**
4: **for***k=j+1***to***size(Starr)***do**
5: *MMA[Starr[j]][Starr[k]]+=1*
6: *MMA[Starr[k]][Starr[j]]+=1*
7: **end for**
8: **end for**
9: **end for**
10: **Return***MMA*

The MMA matrix is used in order to determine the quality of solutions reached by the two-phase algorithm, in the practice this matrix is useful to evaluate the number of student conflicts in a complete timetabling, the task of the two-phase algorithm is to find a timetabling with a zero student conflict.

The MMA matrix is the main structure because is directly used by the fitness function, so every heuristic strategy applied to the Design methodology needs it. Its construction detailed in algorithm 1 requires the student demands, that demands are the enrolled subjects that each student must take on the timetabling period.

LPH List: This structure shows the feasible time-domain of each variable. This domain is obtained by the application of *node and arc consitancy* algorithms in the original CTTP inputs. This structure informs the correct search space of each

variable. The LPH list shows in its rows all events and in its colums all the days each cell shows the possible assignation for an specific event into an specific day.An example of this structure can be seen on table 1.[27]

Table 1 LPH List

	Day 1	Day 2	…	Day p
e_1	$< t_3 >$	$< t_2 >$	⋯	$< t_2$ or $t_1 >$
e_2	$< t_2 >$	$< t_2$ or $t_1 >$	⋯	$< t_2$ or $t_1 >$
⋮	⋮	⋮	⋮	⋮
e_n	$< t_2 >$	$< t_2$ or $t_1 >$	⋯	$< t_2$ or $t_1 >$

The LPH list contains the time-domain of each event for the CTTP that means the heuristic algorithm only needs to search inside these valid options for each variable to ensure feasibility. This is an advantage against non-generic approaches, because these approaches need to work with non feasible solutions waiting to reach a feasible zone. The LPH list optimizes the cpu usage only searching in feasible spaces.

The LPH list also provides a generic layer because the algorithm do not need to have an explicit codification of the problem constraints. It is enough to search inside the LPH list to ensure a non-violation of the time space domain. The construction of this LPH list needs the application of consistancy algorithms as well as the application of agents constraints because this list offers the possible timeslots for each event or subject in the CTTP problem. node and Arc consistancy algorithms are algorithms designed to stablish a feasible region of the search space in order to search around it, the *node consistancy[21]* in the CTTP instance is use to, for example; if an specific subjects needs to be taken by 30 student then no classroom with capacity under 30 is allowed to be assigned to thisspecific subject. This simple exercise reduce the search space. The Arc consistancy algorithm searches the valid domains for *chains* of variables, for example in the CTTP problem, if event *i* must be assigned in the 4th timeslot and there exist an order constraint that stablish the assignation of event *j* after event *i*, then we know that the possible time-domain for the event *j* must be from the 4th timeslot.

This structure is important because it establish the feasible search space in terms of time constraints. Time constraints implicity allows the heuristic strategies to focus in the search of the best possible solution.

LPA List: This list contains the feasible space domain for each event. This domain is obtained by the application of *node consitancy* algorithms between all the posible rooms and the features-demands of each event. The LPA list shows in its rows each event and in its columns the valids classrooms for each event. An example of this LPA list as well as the construction algorithm is detailed on table 2 and algoritm 2.

Table 2 LPA list

event	*Classrooms*
e_1	$< p_4, p_{l1}, p_{c2} >$
e_2	$< p_{lab}, p_{c2} >$
e_3	$< p_6, p_{b2}, p_{b3}, p_{b4} >$
e_4	$< p_{lab}, p_{l2} >$
⋮	⋮
e_n	$< p_{d7} >$

Algorithm 2. LPA Construction

Require: int **Nm**= Subjects, int[][] **CA**= Room Features, int[][] **DA**= Subjects Demands, int[] **Rms** = Room List.

```
1: for i=0 to Nm do
2:   for j=0 to size(CA) do
3:     for k=0 to size(CA[j]) do
4:       if DA[i] <= CA[j][k] then
5:         LPA[i].add(Rms[k])
6:       end if
7:     end for
8:   end for
9: end for
10: Return LPA
```

The LPA list constructs the feasible domain in terms of space related constraints. This structure allows the algorithm to only search in the possible valid space values for each event. Like the LPH list this structure represents implicity all the space constraints, this means that the usage of this list ensures the satisfaction of these constraints, so the heuristic strategy only needs to search inside all the possible assignations represented by the LPA list.

This list is constructed by the application of node and arc consistancy algorithms as well as algorithm 2. This algorithm needs the features from each room and the demands for each subjects, with this information the node consistancy algorithm stablish a feasible domain for all the events.

LPS List: This structure represents the point of view of the student in the CTTP instance commonly this list is used as a set of soft constraints. The LPS list is constructed by taken directly from the students (agent constraint) their proposed usage of time, the task for the heuristic search is to satify most students as possible, but of course any violations in MMA, LPH and LPA are not allowed. An example of this structure is shown in table 3.

Table 3 LPS List

Student	Max	Min	Csc	Timetable
S_1	5	2	2	$<t_2,t_4,t_6,>$
S_2	5	1	3	$<t_4,>$
S_3	3	2	2	$<t_2,t_4>$
⋮	⋮	⋮	⋮	⋮
S_p	4	0	2	$<t_2,t_4,t_6,t_9,t_5>$

The LPS structure constraints several columns for each student, the first one the **Max** column specify the maximun number of desired subjects by the student for each day. The **Min** column shows the minimun number of desired subject by the student for each day, the **Csc** columns shows the number of desired subjects in a consecutive way, for example; the student S_3 prefers to have a timetabling with 2 or less subjects in consecutive timeslots, so it is a violation (only for this student) to assign him 3 or more subjects in a row. Finally the **Timetable** column shows the preference of usage of time for each student, for example student S_p prefers his subjects into the t_2,t_4,t_6,t_9 and t_5 timeslots.

As it can be seen these structures represent most of the CTTP constraints from the five main clases by Corne et. al[11]. The first 3 structures provides a feasible search space and the LPS list provides the soft constraint search space. Each structure provides a generalization layer so the search heuristic do no need to have any code for the constraints in an explicit way. This allows to solve diferrent CTTP instances once translated to the generic structures.

One important point in the construction of these structues is their simplicity. This simplicity means that is relatively easy to obtain the proposed structures from a real CTTP instance and therefor is easy to upgrade them if a change happens in the real instance.[27]

Other advantage of the usage of these structures can be seen if for example the researcher found that the LPH list cannot be constructed or an specific event simply do no have any feasible timeslot or classroom, then this means that the problem have constraints in conflict so it is not possible to find at least a feasible solution. The researcher does not need to run a complex algorithm, simply by looking in the structures he knows that the instance have no solution.

Finally it is time to talk about the most important element in the design methodology: the concept of vector. This vector is a binary representation of an event.[17][16] It can be constructed as seen on table 4a where each v_i is a vector that represents an event e_i.

The vectors can be easily added and subtracted in order to construct $sets$. the symbols used for these vectors sets are $V_A, V_B, \ldots V_N$. One characteristic is that the number of vectors sets is always related with the number of timeslots offered by the current timetabling. The main idea about vectors is to have a space where it

can be worked with events without assigned them yet to a fixed timeslot. This independent layer of context generalizes even more the solution process of the CTTP problem. One example of the Vector Set construction can be seen on Table 4b. It is important to see that the number of the vector sets is the same that the timeslots offered in the current timetabling, that is:$N = |T|$. Finally these vector sets have the next properties $\bigcap_{I \in N} V_I = \emptyset$ and$\bigcup_{I \in N} V_I = E$.

The main problem is now how to construct a fixed number of vectors sets (usually the cardinality of timeslots set) in order to obtain zero conflict on MMA, LPH and LPA. It is precisely for the vector sets construction that it is needed a heuristic algorithm, but if any other CTTP problem can be expressed by means of the Methodology of design's generic structures then the same algorithm can be applied without any modification in order to solve it.[25][26][27][28]

Table 4a Vector Construction

Events	e_1	e_2	$\cdots$	e_{n-1}	e_n
v_1	1	0	…	0	0
v_2	0	1	…	0	0
$\vdots$	$\vdots$	$\vdots$	…	$\vdots$	$\vdots$
v_{n-1}	0	0	…	1	0
v_n	0	0	…	0	1

The Methodology of design have only one explicit variable that is necesary to be solved in order to make a complete solution, this variable is represented in the MMA structure as the *student conflict*. The main task now is how to deal with *students conflicts* only (MMA Matrix). The heuristic algorithm works with these conflicts by means of the next minimization function[25][27]:

$$min(FA) = \sum_{i=1}^{k} FA_{V_i} \tag{1}$$

$$FA_{V_j} = \sum_{s=1}^{(M_{V_j})-1} \sum_{l=1}^{M_{V_i}-s} \left(A_{j,s} \wedge A_{j,s+l} \right) \tag{2}$$

Where: FA = Student conflicts of current timetabling. V_i = Student conflicts from "Vector Set" 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 set" j.

A is a student that demands subject s in a timetabling j.

Table 4b Vector set Construction

Events	e_1	e_2	e_3	e_4	e_5	e_6	⋯	e_{n-1}	e_n
Vectors	v_1	v_2	v_3	v_4	v_5	v_6	⋯	v_{n-1}	v_n
Vector Set	V_A			V_B		V_I	⋯	V_N	

2.3 Metaheuristics Adapted to the Methodology of Design

As we can see from section 2.2 it is necesary to construct a particular vector set where the number of student conflicts between the assigned subjects be the minimum. The construction of this set can be seen as a combinatorial problem, despite of the wide variety of metaheuristics that can be applied to this kind of problem, we only chose metaheuristics that have been tested over similar problems with a good reported performance over its respective instances. It is important to say that this is the first time that these algorithms will be tested with an generic approach like the Methodology of Design, and by the No-free lunch theorem [32] the fact that these algorithms had shown a good performance in its particular approaches does not mean that it can be expected a similar good behaviour for all metaheuristics in the proposed generic methodology.

The main effort of the implemented metaheuristic will be to find a solution with 0 conflicts in accordance the MMA matrix, at the same time that it searches inside LPH and LPA list. Once a strategy achieves 0 or a minimun conflict in the MMA matrix a second phase will be executed, in this phase the best obtained solution will be intensified in order to satisfy most soft constraint displayed by the LPS list.

In this sectioneach metaheuristic implemented will be detailed. The set of proposed metaheuristic had been used in previous work with encouraging results over diferrent CTTP instances. This work will test these metaheuristics with the objective to find which metaheuristic shown a better performance over the methodology of design.

The selected metaheuristic are: Classic Genetic Algorithm (sGA), a novel Frecuency Genetic Algorithm (fGA), Eclectic Genetic Algoritm (eGA), Cellular Genetic Algorithm (cGA), Diferential Evolution (DE/rand/1) , Particle Swarm Optimization (PSO) and Great Deluge Algorithm (GDA).

Genetic Algorithm (sGA): As seen on the work of Xin-She [34] Genetic algorithms are probably the most popular evolutionary algorithms in terms of diversity and applications. This heuristic solver paradigm was developed by John Holland, whose book *adaptation in natural and artificial systems* (1975) was instrumental in creating a new breach of heuristic optimization: *evolutionary computation*, As the name can express this heuristic solver is highly based in Darwin's

Evolutionary theory in the sense that individuals with a better adaptation to the environment have bigger chances to past its genes to a new generation. This is indeed the basic idea of GAs.In the Course timetabling state of art this algorithm has reported a good results for early works like [35] [15] and [10]. The legacy of this algorithm and its ease of implementation allows us to selected it as the first metaheuristic tool for our generic approach.

The essence of genetic algorithms involves the encoding of an optimization solution as arrays of bits or character strings to represent chromosomes, the operators applied then to the chromosomes tries to mix the genetic material (characteristic of each timetable for the CTTP) in order to produce decedents, the comparison of these new individuals by a selection operator according the fitness function provides a way to identify the best solutions. The basic pseudo code for a generic GA can be seen on algorithm 3 taken from Xin-She [34].

Algorithm 3. Simple Genetic Algorithm

Require: Objetive Function **f(x)**,$x=(x_{1,....,}x_n)^T$
1: *Encode the solution into Chromosomes (binary strings)*
2: *Define fitness* **F** (usually *f(x)*)
3: *Generate initial population*
4: *Initial probabilities of crossover* (P_c)*and mutation* (P_m)
5: **while**(*t < Max number of generations*)
6: *Generate new solution by crossover and mutation*
7: **if** P_c>*rand, Crossover;* **end if**
8: **if**P_m>*rand, Mutate;* **end if**
9: *Accept the new solution if its fitness increases*
10: *Select the current best for the next generation (elitism)*
11: **end while**
12: **Return***Best Solution from Population*

As seen on Algorithm 3 the GA needs some parameters, these parameters usually are : *Population number, Generation number, Crossover probability, Mutation probability and Elistism Percentage.*Population number means the number of desired chromosomes in each iteration of the GA, this number ussually depends to the problem itself. Generation number means the number of iterations executed by the GA, the value in this parameter is ussually in emphirical way. Crossover Probability or (P_c) means the propability that a chromosome will be reproduce with another chromosome. This parameter is commonly set between 0.7 or 0.99. Mutation Probability (P_m) means the propability to the chromosome to be changed arbitrary at the end of a generation, this parameter is ussually set in low values (0.01 to 0.15) . Elitism Percentage means the percentage of the best chromosomes that will pass to the next generation without any change. This parameter is ussually considered like the *memory* of the GA and its values are set emphirically.

Basically the GA is an simple metahueristic that needs only a fitness function and a adequate representation in its solution. In terms of the current CTTP the GA uses a direct representation previously reported by Soria et al [27] where each gene represents an event to be assigned into a timeslot or vector. A more detailed explaination of each adaptation is shown next:

A) Fitness Funtion: Taken form equation 1 and 2.

B) Solution Representation: The representation is direct where each gene represent and integer that indicates the pair of the timeslot (reported by LPH) and classroom (reported in the LPA structure). An example of this representation is shown on table 5.

Table 5 sGA representation

Events	Value
e_1	3 (timeslot 2, classroom 4)
e_2	12 (timeslot 1, classroom 12)
e_3	7 (timeslot 4, classroom 1)
.	.
e_n	9 (timeslot 2, classroom 4)

C) Selection Operator: The CTTP problem seen from the point of view of the Methodology of Design is a minimization Problem, in order to assign a bigger probability of selection to the chromosomes/individuals with less value in its fitness the next equation is proposed.

$$P_{e_i} = \frac{\left(1 - \left(\frac{f_{e_i}}{\sum_{i=0}^{n} f_{e_i}}\right)\right)}{n-1} \tag{3}$$

WhereP_{e_i}means the probability to select event e_i. f_{e_i}means fitness value of the event e_i. n means the number of events in the current timetabling. This ecuation is proposed as a minimization Roulette and its objetive is to give more probability to less fitness values, bigger values means that by the equation 3 the chromosome will be less propable to be selected .

D) Crossover Operator: As seen on the proposed representation there is no problem in the repetition of values inside the chromosome, so for the crossover the Single point crossover will be use.

This operator simple selects uniformly a random point inside the chromosome and then from this point the genetic material is interchanged between two chromosomes, the random point is changed at each iteration.

F) Mutation Operator: The mutation operator simply selects randomly a variable, then changes the value of this gen in the chromosome in a uniformly random way. In each mutation a new gen is selected and a new value is assigned. It is important to say that at maximun one mutation is performed in every iteration.

Frecuency Genetic Algorithm (fGA): This kind of GA was developed during this invetigation, this GA uses the concept of *execution by frecuency*. The *frecuency* in the CTTP means the number of events that have the same cardinality in its LPH list. An example of this frecuency value can be seen of table 6

Table 6 Example of selection by frecuency

LPH	Day 1	Day 2	Day3	Day 4
e_1	$< t_1 >$	$< t_2 >$	<>	$< t_2$ or $t_1 >$
e_2	$< t_3 >$	$< t_2 , t_4 >$	$< t_2 >$	$< t_2$ or $t_1 >$
e_3	$< t_2 >$	$< t_2 >$	<>	$< t_2 >$
e_4	$< t_2 >$	$< t_2$ or $t_1 >$	$< t_2 >$	$< t_2$ or $t_1 >$

Frecuency	Events
1	e_3
2	e_1 , e_4
3	
4	e_2

From table 6 it can be noticed that the frecuency of events 1 and 4 is 2 so it is easy for the GA to work with these 2 events in the same iteration (the representation for both events have the same number of posibilities: 2 integer values). The main diference between sGA and fGA is that the fGA algorithm have a dynamic chromosome length , so in the firsts iterations the chromosome length will be the*l* events that have the same *frecuency* or number of posibles timeslots in the LPH list, A proposed way to select the quantity of this *l* events is equation 4.

$$Fr = \begin{cases} n * \frac{1}{2} & if\ i \leq (Gt * 0.33) \\ n * \frac{1}{3} & if (Gt * 0.33) < i \leq (Gt * 0.66) \\ n * \frac{1}{4} & if (Gt * 0.66) < i \leq Gt \end{cases} \quad (4)$$

Where *Fr* means the number of *frecuencies* to solve in the iteration *i.n* means the number of total frecuencies in the CTTP (taken from the LPH list). *Gt* represents the parameter *generations* in the fGA. As it can be seen from the equation 4 the sucession of the selected *frecuencies* is nothing more than the *armonic succession* divided between *k* execution groups, in this work the number of execution groups was set in 3.

The main idea of this fGA algorithm is to optimize the usage of cpu working only with new events each *k* iteration groups. From figure 3 it can be seen the performance of sGA in the CTTP with Methodology of Design and on figure 4 the performance of fGA in similar conditions.

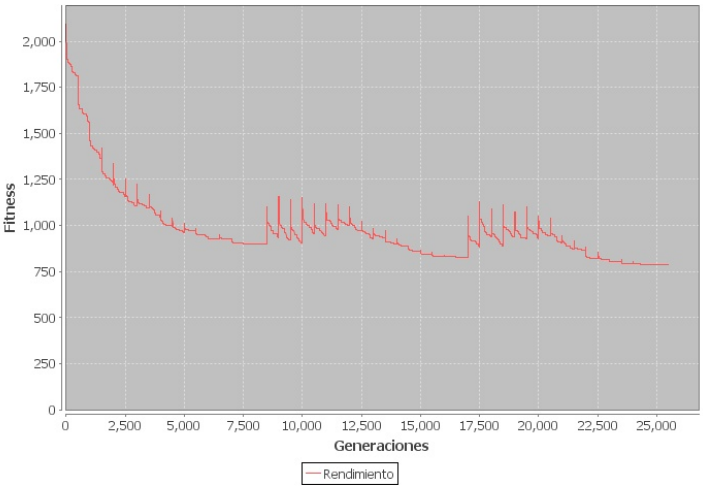


Fig. 3 sGA Performance over CTTP with Design Methodology

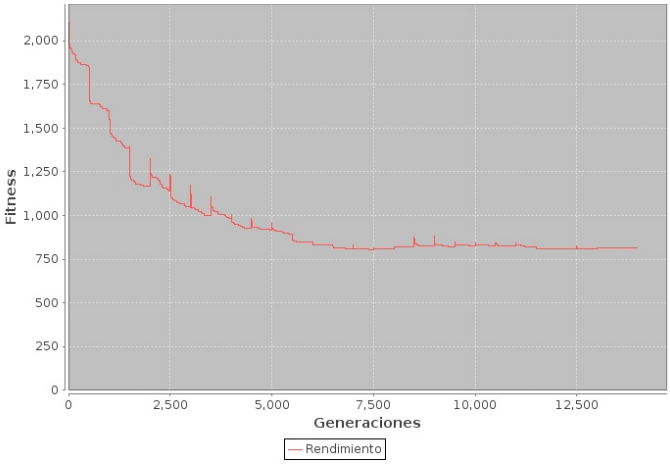


Fig. 4 fGA performance over CTTP with Design Methodology

The fGA algorithm was designed to achieve at least the same value of fitness than sGA with less iterations and execution time. The operators used in this fGA algorithm besides the frequency selection are exactly the same described in the previous sGA section.

Eclectic Genetic Algorithm (eGA): Genetic Algorithm with Vasconcelos selection and auto-adaptation in its parameters. This Genetic algorithm was developed by Angel Kuri [20], and has shown a good performance over high-constrained problems. The auto-tunning allows this algorithm to escape from local optima by itself. This main chracteristic incorporated in this GA are Kuri[19][20]:

- **A)** Full elitism over a set of *n* size of the last population. Given that it has been tested *nk* individuals by generation *k*, the population consist of the best *n* up to that point.
- **B)** A deterministic selection scheme (opposed to the traditional stochastic selection methods). The main idea is to emphasize genetic variety by imposing a strategy which enforces crossover of predefined individuals. In this scheme, the *j-th* individual is crossed with the *(n-i+1)-th*individual (Vazconselos strategy).
- **C)** Annular crossover
- **D)** Population self-adaptation of the following parameters: The number of the Offspring, Crossover probability and mutation Probability.

These considerations were adapted to the CTTP problem with Design Methodology. The chromosome codification was the same used in the previous sGA and fGA algorithms so the chromosome is a simple integer chain with the length of the number of events. For the selection operator, the Vazconselos strategy was implemented, that strategy sorts al the chromosomes from the best fitness to the worse, then the crossover operator is applied to every pair *(i,n-i)* where *n* is the number of current population. On the figure 5 it can be seen an example of this strategy.

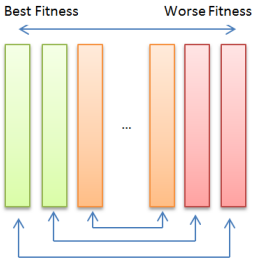


Fig. 5 Vazconselos selection strategy.

The crossover operation is the *annular* crossover; this operator considers two chromosomes as annular chains, this operator selects two arbitrary points inside the chromosomes and then interchanges its genetic material, this operator works as seen on figure 6.

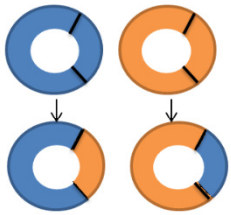


Fig. 6 Annular Crossover

The Crossover and mutation probability parameters are included in each chromosome, so essentially each individual carries on with its own crossover and mutation probabilities, in order to fix these parameters values at the beginning of the generation, the next equations are applied.

$$(P_m)_k = \frac{1}{n}\sum_{i=1}^{n}(P_m)_i \tag{5}$$

Where $(P_m)_k$meanshe probability of mutation in the *k*-th iteration, $(P_m)_i$means the probability of mutation coded in the *i-th* chromosome and *n* means the total number of chromosomes in teh current iteration.

$$(P_c)_k = \frac{1}{n}\sum_{i=1}^{n}(P_c)_i \tag{6}$$

In a similar way the equation 6 details the probability of crossover for the *k-th iteration;*Where $(P_c)_k$meanshe probability of crossover in the *k*-th iteration, $(P_c)_i$means the probability of crossover coded in the *i-th* chromosome and *n* means the total number of chromosomes in the current iteration.

Finally, both; the fitness function and the offspring selection are the same as sGA and FGA algorithm.

Cellular Genetic Algorithm (cGA): Genetic Algorithm with high parallelism developed by Alba et.al [2] [3]. This GA limits each individual to a specific neighbourhood (NEWS neighbourhood in this work), also each individual is placed in a toroidal grid. This kind of algorithm admits sub-populations that work at the same time in different regions of the search space, but gathers information with a migration operator. Several adaptations of common GA operators are made in order to use them in this cGA, For example the selection operator only selects neighbours for each individual/cell. The Elitism operator is changed as well, making only possible to choose the best individual for each subpopulation, i.e in each subpopulation the best cell cannot be modify ,but this cell can modify (cross-over) others.

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 special structure defined as a connected graph, where each vertex is a common GA chromosome or Cell that is only allowed to communicate with its nearest

neighbours. Particularly, individuals are conceptually fixed in a toroidal mesh and are only allowed to recombine with close individuals.[5] An example of this type of interaction can be seen on the figure 7.

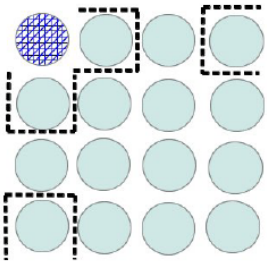


Fig. 7 Simple NEWS toroidal grid interaction.

A pseudo-code of the canonical version of cGA proposed by Alba et al[3] can be seen on Algorithm.

Algorithm 4. Canonical Cellular Genetic Algorithm

Require: Fitness Function **f(x)**,$x=(x_{1,\ldots,}x_n)^T$
1: *Encode the solution into Chromosomes (binary strings)*
2: *Define fitness* **F** (usually *f(x)*)
3: *Generate initial population*
4: *Initial probabilities of crossover* (P_c)*and mutation* (P_m)
5: **while**(*t < Max number of generations*)
6: **for**i **to** *population size (total cells)* ***do***
7: *Define Neigborhood for Cell i.*
8: *Selects a Neighbor for Cell i.*
9: *TempCell = recombitation(Cell i, Selected Neighbor)*
10: *Update Cell I with TempCell*
11: **end for**
12: *Mutate (Grid)*
12: **Return***Best Solution from Grid*

Further cGA adaptations applied in order to work over the CTTP with Design Methodology are:

- **A)** Each Cell in the toroidal grid represents a complete timetable as an integer chain of all the events. Each integer is a value that coded a pair (time-slot-classroom).
- **B)** The neighborhood model used is the NEWS model (as seen on figure 7), in order to select a neighbor to execute the crossover the minimization roulette is applied around each cell (equation 3)
- **C)** The crossover operator applied is the single point crossover (the same that sGA).

D) The mutation operator is the same applied in sGA. A simple arbitrary change in an event over a single cell in the grid.
E) The elitism operator is applied every iteration and this operator simply locks the best cell to any change. It is necessary to say that this kind of lock only denied any change in the cell itself, but this cell can modify its neighbors.
F) The grid used is 4x4
G) The fitness function is the same seen on equation 1 and 2.

The use of an elitism operator do not appears in the original cGA algorithm, in this work this elitism is a proposed strategy to construct a cGA with *memory*, this memory has the objective to stop any non-desired change or a situation when the best cell updates with a worse genetic material.

Finally, this algorithms admits the usage of Sub-populations, this subpopulation usage means the execution of two or more similar grids at the same time. The grids interchange information every k iterations, where k is a parameter usually defined by the user. In this work, 2 grids both of 4 x 4 with similar configuration, interchanges information every 250 function points.

Differential Evolution (DE): The Differential Evolution (DE) is a evolutionary strategy designed for problems of continuous nature. This algorithm has been reported [22] [29] as a good algorithm capable to work with high constrained problems in a small time. Developed by Storn and Price, It is a vector-based algorithm and can be considered as a further development to GA. This stochastic search algorithm with self-organizing tendency do not uses information of derivatives (as a GA). Unlike GA, DE carries out operations over each component of the vector (in our case each variable is coded inside the chromosome). This kind of operators can be expected to be more efficient when the optimal solution is near to the current point coded in an chromosome/individual [34]. Despite of the discrete nature of the CTTP instance, the DE can work over a discrete representation like the proposed GA chromosome where each *component* is an CTTP event whit a well-defined pool of timeslot choices(LPH). So the representation used is, as well as previous GA's, a chain with integer values.Despite of the fraccionary nature of DE, each value produced by a DE operation will be rounded to the nearest integer that coded a LPH-LPA value.

Diferential evolution consist of three main steps: mutation, crossover and selection. Mutation is executed by an mutation scheme. For each variable $x_i$at any time or generation/iteration t , first it is randomly selected three distincs chromosomes x_p, x_q and x_r, an then it is generated a *donor* vector by the equation 7 Where $F = \epsilon[0,2]$ is a parameter refered as diferential weight.

$$v_i = x_p^t + F(x_p^t - x_r^t) \tag{7}$$

The crossover operator is controlled by a crossover probability $C_r = \epsilon[0,1]$ and the actual crossover is executed in this work as a binomial scheme. The binomial scheme performs crossover on each of the d components (events or variables). By generating a uniformly distributed random number $r_i = \epsilon[0,1]$, the *j-th* component of v_i is manipulated as seen on equation 8.

$$u_{j,i}^{t+1} = \begin{cases} u_{j,i} & if\ r_i \le C_r \\ x_{j,i} & otherwise \end{cases}, \quad j = 1,2,\dots,d \tag{8}$$

This way, each component can be decided randomly whether to exchange with donor vector.

Selection is essentially the same as that used in the previous GA's seen on equation 3. Therefore the update of each component is executed by the equation 9.

$$x_i^{t+1} = \begin{cases} u_i^{t+1} & if\ f(u_i^{t+1}) \le f(x_i^t) \\ x_i^t & otherwise \end{cases} \tag{9}$$

These operator can be seen in the next DE pseudocode taken from Xin-She[34].

Algorithm 5. Canonical Differential Evolution

Require: Fitness Function **f(x)**,$x=(x_{1,\dots,}x_n)^T$
1: *Initialize the population* $\boldsymbol{x}$ *with randomly generated solutions*
2: **while**(*stopping criterion*)
3: **for***i= 1 to n* **do**
4: *for each* x_i*, randomly choose 3 distinct vectors* x_p*,* x_q *and* x_r
5: *Generate a new vector* $\boldsymbol{v}$ *by equation 7*
6: *Generate a random index* $J_r = \epsilon[1,2,\dots,d]$ *by permutation*
7: *Generate a randomly distributed number* $r_i = \epsilon[0,1]$
8: **for** j=1 to d **do**
9: *for each component* $\boldsymbol{v_{j,i}}$ *update*
10:

$$u_{j,i}^{t+1} = \begin{cases} u_{j,i} & if\ r_i \le C_r\ or\ j = J_r \\ x_{j,i} & if\ r_i > C_r\ and\ j \ne J_r \end{cases}, \quad j = 1,2,\dots,d$$

11: **end for**
12: *Select and update the solution by equation 9*
13: **end for**
14: *Update the counters such as t=t+1*
15: **end while**
16: **Return***Best Solution*

Particle Swarm Optimization (PSO): Particle Swam Optimization is based on the swarm behaviour was developed by Kennedy and Eberhart(1995) [17]. Since then the PSO has been applied to almost every area in optimization, computational intelligence and design/scheduling applications [16][24]. This algorithm searches the space of an objective function by adjusting the trajectories coded inside each particle (in our case the time values of each variable/event) in a quasi-stochastic manner. Each particle is attracted toward the position of the current global best g^* and its best location x^*_i in history, while at the same time it has a tendency to move randomly. This algorithm can be adapted to CTTP instances with the same considerations seen on the DE algorithm. This means that the PSO will manage an vector represented by a integer chain where every component is a CTTP event and every operation will be rounded to the nearest integer that coded a LPH-LPA value.

The essential steps of the PSO algorithm can be summarized as the pseudo-code taken from Xin-She[34] and showned on algorithm 6. Let $\mathbf{x_i}$ and $\mathbf{v_i}$be the position vector (CTTP complete assignation) and the velocity for particle **i** (Solution). The new velocity vector is detrmined by the following formula

$$v_i^{t+1} = v_i^t + \alpha\varepsilon_1 \odot [g^* - x_i^t] + \beta\varepsilon_2 \odot [x_i^* - x_i^t] \tag{10}$$

Where ε_2 and ε_1 are two random vectors, and each entry taking the values between 0 and 1. The hadaman product od two matrices $u \odot v$ is defined as the entrywise product, that is $[u \odot v]_{ij} = u_{ij} v_{ij}$. The parameters α and β are the learning parameters or acceleration constants, which can typically taken as ≈ 2.

The initial locations of all particles should distributed uniformly so that can sample over most regions. The initial velocity of a particle can be taken as 0, that is $v_i^{t=0} = 0$, The new position can be update by equation 11.

$$x_i^{t+1} = x_i^t + v_i^{t+1} \tag{11}$$

Algorithm 6. Canonical Particle Swarm Optimization

Require: Fitness Function **f(x)**,$x=(x_1,\ldots,x_n)^T$
1: *Initialize locations* $\boldsymbol{x_i}$ *and velocity* $\boldsymbol{v_i}$ *of n particles*
2: *Find* g^* *from min*$\{f(x_1),\ldots, f(x_n)\}$
3: **while (***stopping criterion***)**
4: *t=t+1*
5: **for** j=1 **to** *n (Particles)*
6: *Generate a new velocity* v_i^{t+1} *using equation 10*
7: *Calculate new locations usign equation 11*
8: *Evaluate fitness funtion at new locations* x_i^{t+1}
9: *Find the current best for each particle* x_i^*
10: **end for**
11: *Find the current global best* g^*
12: **end while**
13: **Return**g^* *Solution*

Great Deluge Algorithm (GDA): The Great Deluge Algorithm was developed by Dueck, 1993 [14] based on simulated annealing. This algorithm uses only one parameter *time execution*. It has been observed [7] that a enough big execution time impacts positively in the final solution granted by this algorithm. The GDA strategy works with an quasi-stochastic search, looking for the best possible solution in the fitness landscape but its constrained to: Do not chose a worse solution and Do not take a solution with a fitness value smaller that the remain time execution units. This strategy ensures for example that if GDA has 100 remain execution units(time) no solution with a fitness bigger that 100 will be accepted. This algorithm have been tested over CTTP instances with good results.[33]

The GDA algorithm uses the concept of *Neighbourhood*. That concept is applied to the current solution making smaller changes into in in order to achieve a new solution with the best characteristic of the previous one.

In order to make a change to a prevous solution the algorithm selects randomly a single variable to change (CTTP event) inside the integer chain representation once selected,several heuristics are proposed to update its value:

A) Sequential Selection: It selects the next pair (LPH-LPA) from the order constructed by its Cartesian Product that defined the variable's domains.

B) Min-conflict in Soft Constraints: Chooses the pair (LPH-LPA) that participates in the least number of conflicts with constraints.

C) Random Selection: Chooses (LPH-LPA) values in a random way.

These heuristics are usefull to make minor changes in the current solution willing for a possible positive change but without having a big lost in terms of the current quality of the solution.Also, these heuristic were taken from previous works with the CTTP [12] where it have shown encouraging results. The GDA algorithm adapted to Methodology of Design can be seen on the algorithm 7.

Algorithm 7. Great Deluge for CTTP with Design Methodology

Require: Fitness Function **f(x)**,$x=(x_{1,\ldots,}x_n)^T$
1:*Set parameter* ***ExecutionTime****(miliseconds)*
2: Construct initial solution ***s*** *Randomly*
3: *Calculate initial fitness funtion from solution* ***s*** *f(s)*
4: *Set initial Boundary level* $\boldsymbol{B=B_0=}$ *f(s)*
5: *Set initial decay rate *B =0*
6: Set t_i *= current time (miliseconds)*
7: **while (***B>0)*
8: *Create Neighbor *****s*** *from the random application of heuristics to* ***s***
9: *Calculate f(*s)*
10*:* **if** *f(*s) <= f(s)* ***or*** *(f(*s)<=B)***then**
11: *Update s=*s*
12: **end if**
13**;***Set* t_f *= current time (miliseconds)*
14: *Update *B*
15: *lower Boundary B =* B_0*-*B*
16: **end while**
17: **Return***Solution* ***s***

The most important line in the algorithm 7 is line 14, where the decay rate is updated. This parameter is usually set indirectly by the user during the assignation of execution time units. The proposed linear equation used to update the parameter ***B** can be seen on equation 12.

$$*B= \left[\left(-\frac{B_0}{Executiontime}\right)\left(t_f-(t_i+Executiontime)\right)\right] \quad (12)$$

The equation 12 supposes that the desired limit at the end of *Executiontime*is 0, this means that the theorically boundary in the last executions will be closer to an

optimal solution. The GDA algorithm behaves at early stages of execution as a simulated annealing algorithm accepting a wide range of solutions. In late executions the GDA intesifyits current solution. Finally this algorithm can be easily adapted to any constrained problem where it will possible the creation of a neighbor solution.

2.4 Test Instances

The methodology of design allows to solve several different set instances as long as these instances can be expressed in terms of the generic structures (MMA, LPH, LPA and LPS), That is the principal advantage of this generic approach. Two well know and referenced set instances are taken to make the comparison experiment over our generic approach, these set instances PATAT 2002 and PATAT 2007 were made for the first and second International Timetabling Competition respectively.

There are 20 test instances for Patat 2002 and 24 for Patat 2007, the main characteristics are share between these sets like the main data as well as some constraints. The last two hard constraints marked by (*) are only utilized in ITC 2007.

Patat 2002 and 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 that 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.

3 Experiment Design

The comparison between the selected metaheuristics was made with PATAT 2002 an 2007. Basically once each metaheuristic is adapted to the proposed generic approach, that adaptation is used to solve both test instances. It should be noted

that exists previous works [9] [18] where some metaheuristics were tested for patat 2002 or patat 2007, however this is the first time where a single generic algorithm is capable to solve both instance sets with no special adaptation for each case. For the present comparison each metaheuristic execute 100 independent experiments in order to assume statistical normality as well as 1000 functions points per independent run. A xGA generation containts x funtion points where x is equal to the population parameter. The parameters used for each metaheuristic can be seen on table 7. These parameters were taken from the literature and empirical evidence obtained in this paper.

Table 7 Parameters for each metaheuristic

Algorithm	Parameter	Value
sGA	Elitism	0.3
	Cross-over	0.85
	Mutation	0.15
	Population	256
fGA	Elitism	0.3
	Cross-over	0.95
	Mutation	0.1
	Selection	Harmonic
	Population	256
eGA	Elitism	0.3
	Cross-over	Auto adaptable
	Mutation	Auto adaptable
	Population	256
cGA	Elitism	1 per sub-population
	Cross-over	0.937
	Mutation	0.1
	sub-populations	16
	individuals	256
	Neighborhood	NEWS
DE	f	0.9
	Cr	0.5
	Population	256
PSO	gBest	0.8
	lBest	0.4
	Inertia	0.95
	Population	256
GDA	Time	Until 1000 F.points

3.1 Results

The results achieved for each metaheuristic can be seen on table 8 and 9.

Table 8 Results for PATAT 2002 instances

Instance	Results	sGA	fGA	eGA	cGA	DE	PSO	GDA
2002-1	Mean Fitness	324	305	308	188	303	288	279
	Std deviation	18.8	19.8	20.4	14.9	12.2	19.6	15.5
2002-2	Mean Fitness	305.5	215	260	179	184	252	204
	Std deviation	18.2	15.5	19.9	14	13.5	17.6	18.4
2002-3	Mean Fitness	330.4	278.58	248.04	202.7	295.9	239.6	252.5
	Std deviation	15.3	16.8	15.6	13	12.5	16.8	18.4
2002-4	Mean Fitness	485.5	451.2	397.2	304	438.2	349.8	390.5
	Std deviation	23.9	24	22.8	22.9	19.5	26.5	28.4
2002-5	Mean Fitness	480.18	360	423.6	293.3	348.6	440	310.2
	Std deviation	26.57	25.5	28.5	21.4	23.3	22.4	21.9
2002-6	Mean Fitness	481.8	374.8	416.5	294.8	280.5	439.1	402.7
	Std deviation	26.56	25.4	26.85	21.06	25.4	30	27.87
2002-7	Mean Fitness	503.4	371.2	282.3	287.9	268.5	400.1	339.9
	Std deviation	30.55	29.3	27.6	24.9	12.45	18.66	24.4
2002-8	Mean Fitness	371.94	228.6	316.5	210.2	180.6	350.2	237.6
	Std deviation	25.14	28.4	22.65	18.65	17.8	28.1	24.6
2002-9	Mean Fitness	346.8	254.1	288.6	207.9	274.5	266.5	281.5
	Std deviation	20.54	19.5	17.65	16.56	12.45	16.4	20.2
2002-10	Mean Fitness	335.15	277.9	202.5	201.9	213.7	271.1	218.5
	Std deviation	21.24	19.8	16.5	14.2	10.4	17.4	15.4
2002-11	Mean Fitness	350.4	277.6	223.5	208.2	233.1	255.4	330.5
	Std deviation	19.4	18.8	17.8	14.56	12.45	13.66	14.56
2002-12	Mean Fitness	312.1	200.7	256.6	188.5	215.8	298	176.8
	Std deviation	19.15	18.5	16.5	14.25	13.6	12.5	15.4
2002-13	Mean Fitness	396.44	261.2	249.5	231.6	290.6	337.4	265.5
	Std deviation	24.26	19.6	15.5	19.54	16.6	19.5	20.8
2002-14	Mean Fitness	520.14	349.6	401.2	313.3	313.9	308.6	319.4
	Std deviation	28.17	28.6	25.6	23.2	20.15	26.5	29.9
2002-15	Mean Fitness	449.5	378.8	281.5	261.4	329.4	366.6	275.2
	Std deviation	28.65	24.5	26.6	22.44	18.6	23.5	29
2002-16	Mean Fitness	369.67	278.5	266.9	223.6	254.8	277.8	305.4
	Std deviation	20.44	17.6	18.6	18.22	13.36	17.4	21.1
2002-17	Mean Fitness	468.9	266.9	399.9	289.4	289.8	281.5	336.5
	Std deviation	29.22	28.9	26.6	22.19	19.6	23.6	27.5
2002-18	Mean Fitness	307.14	256.9	213.6	181.5	210	236.3	200.5
	Std deviation	20.3	17.5	19.9	14.98	12.32	19.5	14.5
2002-19	Mean Fitness	497.1	439.6	426.3	297.7	356	271.3	402.1
	Std deviation	28.95	16.7	29.5	23	19.5	22.5	27.6
2002-20	Mean Fitness	449.14	375	299.6	266	311.3	336.5	361.5
	Std deviation	26.35	24.6	28.9	21.54	18.56	20.3	26.3

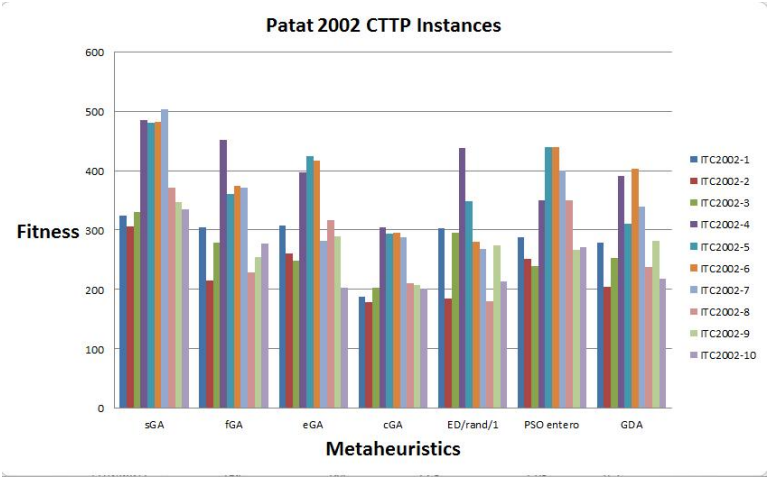


Fig. 8 Performance over Patat 2002 Instances

Table 9 Results for Patat 2007 instaces

Instance	Result	sGA	fGA	eGA	cGA	DE	PSO	GDA
2007-1	M.Fit	1362.50	1315.60	1160.20	975.30	1064.15	1281.45	1279.50
	Std dev.	55.00	52.73	47.10	45.96	54.76	54.59	47.93
2007-2	M. Fit	1388.50	1123.23	1189.30	999.00	1068.32	1315.20	986.23
	Std dev.	56.87	48.38	47.84	42.78	55.28	44.00	50.08
2007-3	M. Fit	556.80	445.30	378.60	286.75	336.80	366.50	359.20
	Std dev.	57.80	57.58	38.89	27.00	26.45	39.43	27.84
2007-4	M. Fit	619.20	488.30	334.50	342.80	426.30	329.50	567.20
	Std dev.	44.00	36.18	42.99	30.55	41.72	39.23	39.82
2007-5	M. Fit	805.30	697.50	689.23	564.4	520.51	576.30	650.04
	Std dev.	35.70	34.39	35.52	32.28	34.61	33.78	33.54
2007-6	M. Fit	794.81	623.50	666.56	552.18	550.36	526.30	713.20
	Std dev.	36.72	33.91	35.54	33.15	34.77	36.63	35.03
2007-7	M. Fit	334.90	284.50	200.69	187.8	272.27	196.52	230.13
	Std dev.	28.96	17.56	17.15	15.5	27.48	20.13	21.14
2007-8	M. Fit	375.00	284.50	280.42	196.30	320.40	316.50	220.50
	Std dev.	36.14	18.86	21.23	18.72	29.50	27.28	21.30
2007-9	M. Fit	1403.10	1206.50	1163.20	979.23	1488.50	1154.30	994.50
	Std dev.	69.77	67.04	58.40	52.67	57.95	67.15	62.59
2007-10	M. Fit	1400.30	1305.60	1068.20	1011.60	1159.50	1235.40	1056.20
	Std dev.	53.58	51.12	50.59	47.38	56.50	47.95	50.61
2007-11	M. Fit	612.84	425.20	343.50	319.21	325.12	328.50	309.20
	Std dev.	58.64	36.51	45.06	32.2	30.40	49.96	49.88
2007-12	M. Fit	588.11	375.20	426.50	317.74	402.92	394.50	370.50
	Std dev.	65.15	59.91	60.47	28.55	33.43	41.32	50.84
2007-13	M. Fit	843.75	680.50	601.50	590.17	706.50	716.50	748.20
	Std dev.	34.24	34.03	34.07	31.19	31.68	34.18	33.78
2007-14	M. Fit	813.11	729.50	648.20	572.86	565.76	723.50	646.80
	Std dev.	36.41	35.86	30.03	26.77	29.50	29.18	30.74
2007-15	M. Fit	396	173.73	171.12	154	204.56	249.50	246.50
	Std dev.	30.23	12.52	29.06	12.20	15.34	17.96	21.03

Table 9 (*continued*)

2007-16	M. Fit	326.85	165.05	168.23	167.21	236.04	198.50	261.20
	Std dev.	34.47	24.68	29.90	16.82	26.13	27.74	20.98
2007-17	M.Fit	432.15	268.50	309.5	160.15	243.85	98.60	204.50
	Std dev.	69.81	51.63	66.95	35.05	56.32	45.74	45.05
2007-18	M. Fit	1000.59	736.50	894.50	635.15	560.50	636.15	623.50
	Std dev.	56.14	50.23	55.58	38.87	40.85	50.65	55.46
2007-19	M. Fit	814.96	682.45	571.23	482.50	713.62	703.54	576.20
	Std dev.	58.10	53.79	54.50	32.52	41.70	48.69	32.55
2007-20	M. Fit	879.07	501.12	774.23	497.52	613.50	643.15	570.12
	Std dev.	59.34	58.66	44.92	38.96	55.69	50.69	53.32
2007-21	M.Fit	847.17	688.95	668.20	614.27	561.23	679.23	576.20
	Std dev.	28.99	28.80	28.50	26.82	27.02	27.11	28.55
2007-22	M. Fit	1556.15	1305.20	1245.10	1185.74	1284.20	1445.20	1251.20
	Std dev.	59.93	68.32	70.65	75.48	66.07	62.28	65.05
2007-23	M. Fit	2882.60	2365.20	2078.20	2137.85	2019.30	2643.20	2347.20
	Std dev.	114.88	98.73	102.86	88.12	105.59	112.15	110.42
2007-24	M. Fit	886.48	562.80	580.20	527.90	620.50	670.20	636.26
	Std dev.	52.31	35.82	45.80	35.2	41.24	37.06	50.86

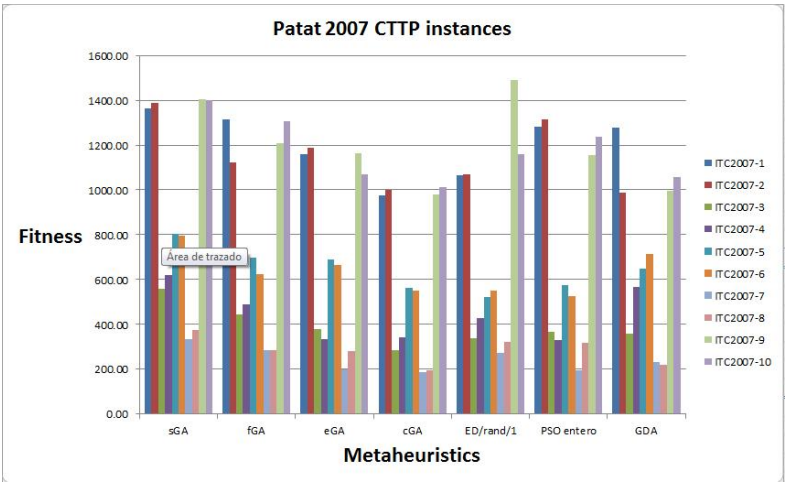


Fig. 9 Performance over Patat 2007 instances

3.2 Discussion

Once obtained the results shown in table 5 and 6 the non-parametric Kruskal-Wallis test is applied. This test is a method for testing whether samples originate from the same distribution. It is used for comparing more than two samples that are independent, or not related. The factual null hypothesis is that the populations from which the samples originate have the same median (in our case this means that each metahuristic have the same median performance). When the Kruskal-Wallis test leads to significant results, then at least one of the samples is different

from the other samples. The test does not identify where the differences occur or how many differences actually occur.

The Kruskal-Wallis test rejects the null hypothesis and accept the alternative hypothesis: at least one of the distributions have different mean. This means that at least one of the algorithms have a different performance with the generic approach over the test instances. it can be seen some evidence that reinforces this result on figures 3 and 4, the figures shows different performance between the proposed algorithms.

Once it is known that the algorithms present different performance, it is needed to identify which metaheuristic present the best performance (Minimum conflicts), in order to do that Wilcoxon signed rank test is applied to every possible pair or metaheuristics. The Wilcoxon signed-rank test is a non-parametric statistical hypothesis test used when comparing two related samples, matched samples, or repeated measurements on a single sample to assess whether their population mean ranks differ. In this the wilcoxon signed rank test is applied in order to evaluate which metahuristic has the best performance in terms of minimun conflicts.

By means of Wilcoxon signed rank test the cGA algorithm has shown a better performance in term of quality solution (conflicts) over our of test instances. The second best algorithm was the ED/Rand/1 also, this algorithm was also the fastest in execution. The second fastest algorithm was PSO however, this algorithm has not shown a good performance in terms of quality over the test instances. On the figures 3 and 4 it can be seen that the cGA algorithm has the best performance, this evidence support the results with Wilconxon Signed Rank test.

A simple sign test is realized on the cGA algorithm on table10 for patat 2002 and 11 for patat 2007. In this test the overall wins over every pair cGA-X algorithm areanalized in order to find enough statistical data about the cGA results.

This sign test considers the number of wins as a binomial distribution; for a greater number of cases, the number of wins is under the null hypothesis distributed according to n ($n/2$, $\sqrt{n}/2$), which allows for the use of the z-test; if the number of wins is at least $n/2+1.96 \cdot \sqrt{n}/2$, then the algorithm is significantly better with $p < 0.05$.

Table 10 Sign test for pairwise comparisons over PATAT 2002. In each test cGA shows significant imrpovement over the other algorithms with a level of significance α=0.5.

cGA	sGA	fGA	eGA	DE	PSO	GDA
Wins(+)	20	20	19	19	20	20
Loses(-)	0	0	1	1	0	0
	α=0.5	α=0.5	α=0.5	α=0.5	α=0.5	α=0.5

Table 11 Sign test for pairwise comparisons over PATAT 2007. In each test cGA shows significant imrpovement over the other algorithms with a level of significance α=0.5.

cGA	sGA	fGA	eGA	DE	PSO	GDA
Wins(+)	25	25	23	21	25	25
Loses(-)	0	0	2	4	0	0
	α=0.5	α=0.5	α=0.5	α=0.5	α=0.5	α=0.5

4 Conclusions and Future Work

This chapter has shown a comparison between several different meta heuristics over an generic approach for the course timetabling problem. This generic approach has been use to solve both well known an referenced international test instances PATAT 2002 and PATAT 2007. The Design methodology was capable to solve both set of instances with a single algorithm.

This chapter has gathered evidence about the good performance of the cGA algorithm as an metaheuristic tool to solve the CTT problem by means of a generic approach. The cGA algorithm uses an overlapped neighbourhood as well as a fixed toroidal structure, these concepts allows the cGA algorithm to diversify the genetic material in its individuals and preserve the best traits and characteristics of the best solutions founded. The cGA algorithm also utilizes a parallel scheme that accelerates the time needed to achive a solution. this algorithm uses a sub-population approach in order to search in different areas of the fitness landscape at the same time. This parallelism and sub-population techniques have shown a positive impact in the solution of CTT problem over an generic approach like the Design methodology.

For future work is proposed to analyse the performance of the cGA algorithm over a different set of instances like UNITIME.org with the same generic approach, also to make more test over different neighbourhood schemes for the cGA. The integration of the migration concept could be benefit for the cGA since this operator can be implemented in a parallel scheme, more test over this idea are suggested.

Acknowledgment. The authors thanks Consejo Nacional de Ciencia y Tecnologia (CONACYT) for the obtained support for this research.

References

1. Adriaen, M., Causmaecker, P., Demeester, P.: Tackling the university course timetabling problem with an aggregation approach. In: Burke, K., Rudova, H. (eds.) Proceedings PATAT 2006, pp. 330–335 (2006)
2. Alba, E., Dorronsoro, B.: Introduction to Cellular Genetic Algorithms. Cellular Genetic Algorithms, 3–20 (2008)
3. Alba, E., Dorronsoro, B.: The State of the Art in Cellular EvolitionaryAlgorithms. Cellular Genetic Algorithms 1, 21–34 (2008)
4. Alba, E., Troya, J.M.: A survey of parallel distributed genetic algorithms, complexity, vol. 4(4), pp. 31–52 (1999)
5. Alba, E., Troya, J.M.: Cellular Evolutionary Algorithms: Evaluating the Influence of Ratio. In: Deb, K., Rudolph, G., Lutton, E., Merelo, J.J., Schoenauer, M., Schwefel, H.-P., Yao, X., et al. (eds.) PPSN 2000. LNCS, vol. 1917, pp. 29–38. Springer, Heidelberg (2000)
6. Alba, E., Troya, J.M.: Improving flexibility and efficiancy by adding parallelism to genetic algorithms. Statistics and Computing 12(2), 91–114 (2002)

7. Burke, E., Bykov, Y.,Newall, J., Petrovic, S.: A time-predefined local search approach to exam timetabling problems. Computer Science TEchnical Report No. NOTTCS-TR-2001-6, 1 (2001)
8. Martín, C., Jorge A., S.-A., Héctor J., P., Rosario, B., Manuel, O., Ernesto, M.L.: Variable Length Number Chains Generation without Repetitions. In: Melin, P., Kacprzyk, J., Pedrycz, W. (eds.) Soft Computing for Recognition Based on Biometrics. SCI, vol. 312, pp. 349–364. Springer, Heidelberg (2010)
9. Cambazard, H., Hebrard, E., O'Sullivan, B., Papadopoulos, A.: Submission to ICT: Track 2. International Timetabling Compertition 2007 (2008)
10. Colorni, A., Dorigo, M., Maniezzo, V.: Genetic Algorithms and Highly Constrained Problems: The Time-Table Case. In: Schwefel, H.-P., Männer, R. (eds.) PPSN 1990. LNCS, vol. 496, Springer, Heidelberg (1991)
11. Corne, D., Ross, P., Fang, H.: Fast Practical Evolutionary Timetabling. In: Fogarty, T.C. (ed.) AISB-WS 1994. LNCS, vol. 865, pp. 251–263. Springer, Heidelberg (1994)
12. Conant-Pablos, S.E., Magaña-Lozano, D.J., Terashima-Marín, H.: Pipelining Memetic Algorithms, Constraint Satisfaction, and Local Search for Course Timetabling. In: Aguirre, A.H., Borja, R.M., Garciá, C.A.R. (eds.) MICAI 2009. LNCS, vol. 5845, pp. 408–419. Springer, Heidelberg (2009)
13. Cooper Tim, B., Kingston, J.H.: The Compexity of Timetable Construction Problems. PhD thesis, The University of Sydney, 1995.
14. Dueck, G.: New Optimization Heuristics: The Great Deluge Algorithm and the Record-to-Record Travel. Journal of Computational Physics 104, 86–92 (1993)
15. Erben, W.: A Grouping Genetic Algorithm for Graph Colouring and Exam Timetabling. In: Burke, E., Erben, W. (eds.) PATAT 2000. LNCS, vol. 2079, p. 132. Springer, Heidelberg (2001)
16. Sheau, H., Deri, F.: University course timetable planning using hybrid particle swarm optimization. In: Proceedings of the first ACM/SIGEVO Summit on Genetic and Evolutionary Computation, pp. 239–246 (2009)
17. Kennedy, J., Eberhart, R.: Particle Swarm Optimization. In: Proccedings of IEEE International Conference on Neural Networks, vol. 1, pp. 1942–1948 (1995)
18. Kostuch. P.: Timetabling Competition-SA-based Heuristic. Metaheuristics Network (2003)
19. Angel, K.M.: A solution to the prisioner's dilemma using an eclectic genetic algorithm. IPN 1 (2000)
20. Angel, K.M., Quezada, C.V.: A universal Eclectic Genetic Algorithm for constrained optimization. ITAM 1 (1998)
21. Lewis, R.: Metaheuristics for University Course Timetabling. PhD thesis, University of Notthingham (August 2006)
22. Price, K., Storn, R., Lampinen: Differential Evolution: A pratical approach to global optimization. Springer (2005)
23. Andrea, S., Di Gaspero, L.: Measurability and Reproducibility in University Timetabling Research: Discussion and Proposals. In: Burke, E.K., Rudová, H. (eds.) PATAT 2007. LNCS, vol. 3867, pp. 40–49. Springer, Heidelberg (2007)
24. Sheau, F., Safaai, D., Hashim, S.: A study on PSO-Based University Course Timetabling Problem. In: International Conference on Advanced Computer Control, ICACC 2009, pp. 648–651 (2009)
25. Jorge. A., S.-A.: Diseño de horarios con respecto al alumno mediante técnicas de cómputo evolutivo. Master's thesis, Instituto Tecnologico de León, 2010.

26. Jorge. A., S.-A., Carpio, M., Puga, H.: Diseño de Horarios mediante algoritmos genéticos. Décima Primera Reunión de Otoño de Potencia. In: Electrónica y Computación del IEEE, XI ROPEC, Morelia, vol. 1, pp. 24–35 (2009)
27. Jorge. A., S.-A., Terashima-Marin, H., Carpio, M.: Academic Timetabling Design using Hyper-heuristics. In: Advances in Soft Computing, ITT, vol. 1, pp. 158–164 (2010)
28. Jorge. A., S.-A., Martin, C., Terashima-Marin, H.: Several Strategies to Improve the Performance of Hyperheuristics for Academic Timetabling Design Problem. In: EEE Electronics, Robotics and Automative Mechanics Conference 2010. IEEE Computer Society, México (2010) ISBN: 978-0-7695-4204-1
29. Storn, R.: On the usage of differential evolution for function optimization. In: Biennial Conference of the North America Fuzzy Information Processing Society (NAFIPS), pp. 519–523 (1996)
30. Storn, R., Price, K.: Differential evolution - a simple and efficient heuristic for gobal optimization over continuous spaces. Journal of Global Optimization 11, 341–359 (1997)
31. Willemen Robertus, J.: School Timetable Constructrion: Algorithms and complexity. PhD thesis, Institute for Programming research and Algorithms (2002)
32. Wolpert, H., Macready, G.: No free lunch Theorems for Search. Technical report The Santa Fe Institute, 1 (1996)
33. Yang, Y., Petrovic, S.: A Novel Similarity for Heuristic Selection in Examination Timetabling. In: Burke, E.K., Rudová, H. (eds.) PATAT 2007. LNCS, vol. 3867, p. 1. Springer, Heidelberg (2007)
34. Xin-She, Y.: Nature-Inspired Metaheuristics Algorithms, 2nd edn. Luniver Press (2010)
35. Yu, E., Sung, K.S.: A Genetic Algorithm for a University Wekly Courses Timetabling Problem. International Transactions in Operational Research 9, 703–717 (2002)

High-Performance Architecture for the Modified NSGA-II

Josué Domínguez, Oscar Montiel-Ross, and Roberto Sepúlveda

Instituto Politécnico Nacional - CITEDI. Av. del Parque 1310,
Tijuana, B., C., México
jdominguez@citedi.mx, {o.montiel,r.sepulveda}@ieee.org

Abstract. NSGA-II is one of the most popular algorithms for solving Multi-objective Optimization Problems. It has been used to solve different real-world optimization problems; however, NSGA-II has been criticized for its high computational cost and bad performance on applications with more than two objective functions. In this paper, we propose a high-performance architecture for the NSGA-II using parallel computing, for evaluation functions and genetic operators. In the proposed architecture, the Mishra Fast Algorithm for finding the Non Dominated Set was used. In this paper, we propose a modification in the sorting process for the NSGA-II that improves the distribution of the solutions in the Pareto front. Results for five different test functions using distinct crossover and mutation operators to test performance are presented.

Keywords: Genetic Algorithm, Multi-Objective Optimization, Pareto Optimal, NSGA – II.

1 Introduction

Optimization refers to obtain the values of decision variables, which correspond to the maximum or minimum of one or more objective functions [1]. Many applications consider only one objective function, probably due to the available computational resources; however, most real problems involve one or more objectives, which are very difficult to solve because of its high computational cost. The way of finding optimal solutions of such a problem is known as multiobjective optimization (MOO).

In single-objective optimization, the search space is often well defined; when we try to optimize several objectives at the same time, the search space also becomes partially ordered. A multiobjective optimization problem could be written in the form minimize $[f_1(x), f_2(x),..., f_k(x)]$ for k objective functions $f_i : \Re^n \rightarrow \Re$ subject to some equality and inequality constraints. For $x = [x_1^*, x_2^*, ..., x_n^*]^T$, the vector of decision variables, our task is to determine the set $\boldsymbol{F}$

P. Melin, O. Castillo (Eds.): *Soft Computing Appli. in Optimization,* STUDFUZZ 294, pp. 321–341.
DOI: 10.1007/978-3-642-35323-9_13

of all vectors which satisfy all the constraints, the particular set of values $\boldsymbol{x} = [x_1^*, x_2^*, \ldots, x_n^*]^T$, and also yields the optimum values for all the objective functions [2].

If all objective functions are for minimization, a feasible solution $\boldsymbol{x}$ is said to dominate another feasible solution $\boldsymbol{y}$ $(\boldsymbol{x} \succ \boldsymbol{y})$, if and only if, $f_i(\boldsymbol{x}) \leq f_i(\boldsymbol{y})$ for $i = 1, \ldots, k$, where k is the number of objective functions, and $f_j(\boldsymbol{x}) < f_j(\boldsymbol{y})$ for at least one objective function j [3]. One general approach in MOO is to determine an entire Pareto Optimal solution set or a representative subset. A Pareto optimal set is a set of solutions that are non-dominated with respect to each other, as shown in Fig. 1.

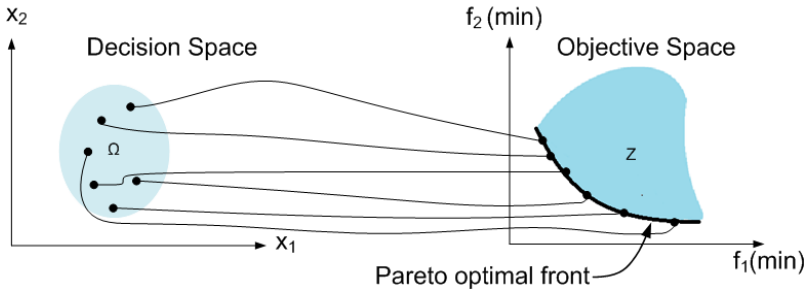


Fig. 1 Decision variable and objective space relationship

The main purpose of a multiobjective optimization algorithm is to identify solutions in the Pareto optimal set. There are three principal methods of dealing with multiple objectives:

1. Combine all the objectives into a single scalar value, typically as a weighted sum, and optimize the scalar value.
2. Solve for the objectives hierarchically, optimizing for a first objective, if there is more than one solution, optimize these solutions for a second objective, and repeat for a third, etc., if it is appropriate.
3. Obtain a set of alternative, non - dominated solutions, each of which must be considered equivalent in the absence of further information regarding the relative importance of each of the objectives.

Generating the Pareto set can be computationally expensive and is often infeasible because the complexity of the application prevents exact methods from being applied. For this reason, a number of stochastic search strategies such as evolutionary algorithms, tabu search, Ant Colony Optimization, and others have been developed, and they usually find a good approximation, i.e., a set of solutions whose objective vectors are not too far away from the optimal objective vectors.

2 Pareto-Optimality

Definition 1. Domination: A decision vector, $\boldsymbol{x}$ dominates a decision vector $\boldsymbol{y}$ (denoted by $\boldsymbol{x} \succ \boldsymbol{y}$), if and only if

- $\boldsymbol{x}$ is not worse than $\boldsymbol{y}$ in all objectives, i.e. $f_i(\boldsymbol{x}) \leq f_i(\boldsymbol{y})$, $\forall i = 1,2, \ldots, k$, and.
- $\boldsymbol{x}$ is strictly better than $\boldsymbol{y}$ in at least one objective, i.e.‹ $f_i(\boldsymbol{x}) \leq f_i(\boldsymbol{y})$, $\forall i = 1,2, \ldots, k$. The concept of dominance is illustrated in Fig. 2.

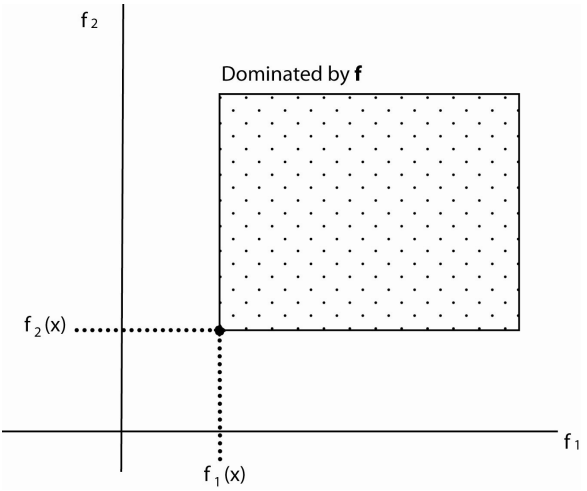


Fig. 2 Dominance concept. Point $(f_1(x),f_2(x))$ dominates all other points.

Definition 2. Pareto-optimal: A decision vector $\boldsymbol{x}^* \in \Omega$ is Pareto - optimal if there, does not exist a decision vector $\boldsymbol{x} \neq \boldsymbol{x}^* \in \Omega$ that dominates it. An objective vector,$\boldsymbol{f}^*(\boldsymbol{x})$, is Pareto-optimal if $\boldsymbol{x}$ is Pareto-optimal.

3 Genetic Algorithms

Holland and his colleagues proposed the concept of Genetic Algorithms (GA) in the 1960s and 1970s [4]. GA is inspired by the evolutionist theory explaining the origin of species. In nature, weak and unfit species within their environment are faced with extinction by natural selection. The strong ones have greater opportunity to pass their genes to future generations via reproduction. In the long run, species carrying the correct combination in their genes become dominant in their population. Sometimes, during the slow process of evolution, random changes may occur in genes. If these changes provide additional advantages in the challenge for survival, new species evolve from the old ones. Unsuccessful changes are eliminated by natural selection.

A solution vector $\boldsymbol{x} \in \Omega$ is called an individual or a chromosome. Chromosomes are made of discrete units called genes and each gene controls one or more features of the chromosome [5]; genes are assumed to be binary bits. The population is a collection of *N* chromosomes, and it is normally randomly initialized.

3.1 Operators to Genereate New Individuals

GA uses two operators to generate new solutions from existing ones: crossover and mutation. In crossover, generally two chromosomes, called parents, are combined to form different chromosomes, called offspring. The parents are selected among existing chromosomes in the population with preference towards fitness so that offspring is expected to inherit good genes, which make the parents fitter. By iteratively applying the crossover operator, genes of selected chromosomes are expected to appear more frequently in the population, eventually leading to convergence to an overall good solution, which is illustrated in Figure 3.

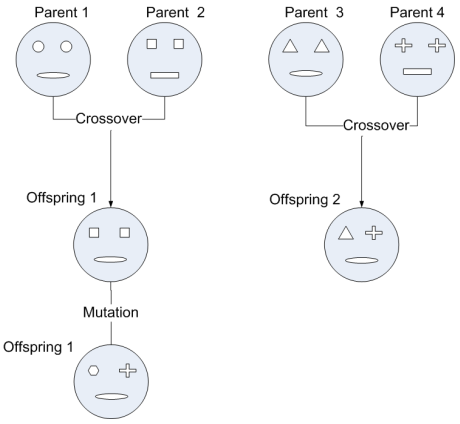


Fig. 3 Illustration of crossover and mutation operator process

The mutation operator introduces random changes into characteristics of chromosomes. A mutation is generally applied at the gene level, in typical GA implementations the mutation rate (probability of changing the properties of a gene) is very small and depends on the length of the chromosome. Therefore, the new chromosome produced by mutation will not be very different from the original one. Reproduction involves selection of chromosomes for the next generation; the fitness of an individual determines the probability of its survival for the future generation. There are different selection procedures in GA, such as proportional selection, ranking and tournament selection that are the most popular procedures. In the following subsections, we describe some crossover and mutation operators that are used in this work.

The general procedure for GA given in Algorithm 1 summarizes a general GA.

Algorithm 1 GA general procedure

Let $t = 0$ be the generation counter;
Create and initialize an $n_x -$ dimensional population: $P(0)$;
while Stopping condition != true **do**
  Evaluate the fitness, $f(x_i)$, of each individual, x_i, in the population, $P(t)$;
  Perform cross-over to produce offspring;
  Perform mutation on offspring;
  Select population $P(t+1)$ of new generation;
  Advance to the new generation, i.e. $t = t + 1$;
end while

3.1.1 Crossover Operators

Linear Crossover (LX). One of the first crossover operators implemented to AGCR was linear crossover operator proposed by Wright [6]. Where two parents are chosen randomly, i.e., $x^{(1)} = (x_1, x_2, ..., x_n)$ and $x^{(2)} = (x_1, x_2, ..., x_n)$ from which you build a solution that is in the middle of the two parents, such it is shown in (1)

$$y^{(1)} = 0.5(x^{(1)} + x^{(2)}) \tag{1}$$

and the two others that are displaced to the extremes using (2) and (3).

$$y^{(2)} = 1.5x^{(1)} - 0.5x^{(2)} \tag{2}$$

$$y^{(3)} = -0.5x^{(1)} + 1.5x^{(2)} \tag{3}$$

Depending on the separation of parents, offspring will also end separated. Wright proposes to eliminate the worst offspring of the two extremes and thus generate only two children [6], but also it is possible to keep all three and this is shown in Figure 4.

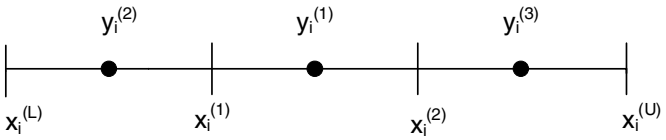


Fig. 4 Distribution of the offspring in the linear crossover operator

Heuristic Crossover (HX) [7]**.** The heuristic crossover operator generates only one offspring $y^{(1)}$ from two parents $x^{(1)}$ and $x^{(2)}$ using the rule of the equation (4).

$$y^{(1)} = r(x^{(2)} - x^{(1)}) + x^{(2)} \tag{4}$$

Where r is a random number uniformly distributed between 0 and 1, the parent $x^{(2)}$ should not be worse than $x^{(1)}$ in terms of this objective function; this is done to keep the search in the right direction.

Blend Crossover. The BLX-α crossover was suggested by Eshelman and Schaffer [8]. From two parents $x^{(1)}$ and $x^{(2)}$ (assuming that $x^{(1)}$ is better than $x^{(2)}$) the BLX operator randomly generates a solution in the interval $[x_i^{(1)} - \alpha(x_i^{(2)} - x_i^{(1)}), x^{(2)} + \alpha(x_i^{(2)} - x_i^{(1)})]$, this new solution is calculated using (5).

$$y^{(1)} = (1 - \gamma_i)x_i^{(1)} + \gamma_i x_i^{(2)} \tag{5}$$

where $\gamma_i = (1 - 2\alpha)u_i - \alpha$, and u_i is a random number between 0 and 1. According to Deb [10], the best performance of this operator is when $\alpha = 5$; if the difference between parents is small, the difference between the child and parents would also be small. This property is an adaptive search, which is illustrated in Figure 5.

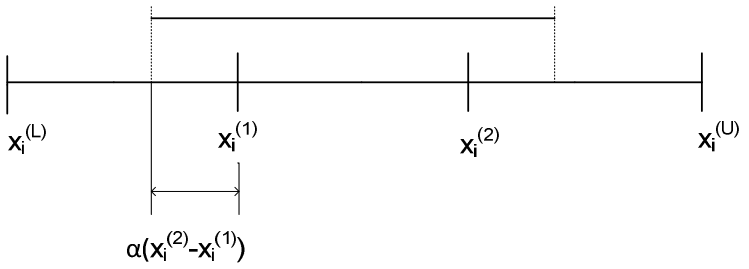


Fig. 5 Probability distribution of the offspring in the BLX-α operator

Simulated Binary Crossover. Deb and his students developed the algorithm SBX [9], which creates two offspring from two parent solutions. As it name suggests simulates the working principle with the operator of single-point crossover in binary strings. With this operator, from two parents, two solutions are calculated as it is shown in equations (6), (7) and (8).

$$y^{(1)} = 0.5[(1 + \beta_{qi})x_i^{(1)} + (1 - \beta_{qi})x_i^{(2)}] \tag{6}$$

$$y^{(2)} = 0.5[(1 - \beta_{qi})x_i^{(1)} + (1 + \beta_{qi})x_i^{(2)}] \tag{7}$$

where,

$$\beta_{qi} = \begin{cases} (2u_i)^{1/(n_c+1)}, & u_i \le 0.5 \\ \left(\frac{1}{2(1-u_i)}\right)^{1/(n_c+1)}, & others \end{cases} \tag{8}$$

u_i is a random number between 0 and 1, n_c is a parameter chosen by the user who is dependent on the probability that a child is created by the father; the recommended value for n_c is any between 0 and 10, and this is shown in Figure 6.

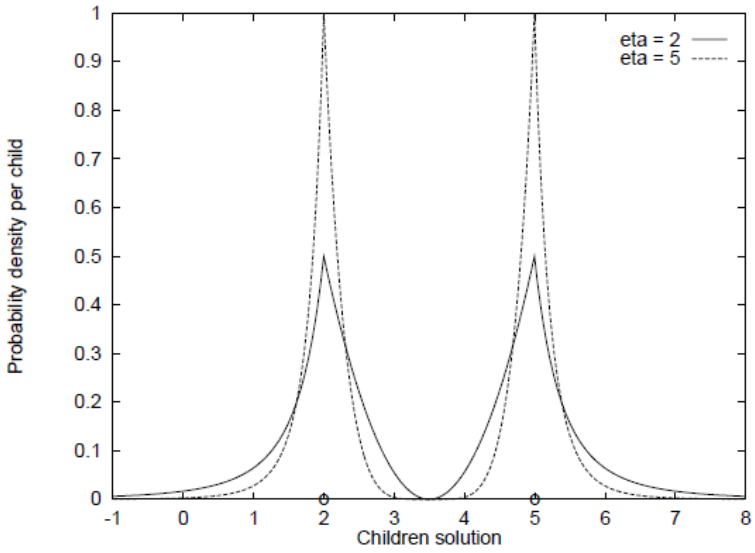


Fig. 6 Probability distribution for creating children solutions of continuous variables [10]

Laplace Crossover. This operator was proposed by Deep and Thakurn [11], it is also known as LX operator, which produces two children from two parents using equations (9), (10) and (11).

$$y^{(1)} = x_i^{(1)} + \beta \mid x_i^{(1)} - x_i^{(2)} \mid \tag{9}$$

$$y^{(1)} = x_i^{(2)} + \beta \mid x_i^{(1)} - x_i^{(2)} \mid \tag{10}$$

$$\beta = \begin{cases} a - b\ln(u), & u \le 0.5 \\ a + b\ln(u), & u > 0.5 \end{cases} \tag{11}$$

where β is a function between 0 and 1, and u_i is a random number uniformly distributed. β is obtained by inverting the Laplace distribution function. The parameters $a \in R$ and $b \neq 0$ are called location parameters, usually taken a = 0

and b> 0 are known as scaling parameters. Deep [11], experimented using values b = 0, 5 and b = 1, which is indicated in Figure 7.

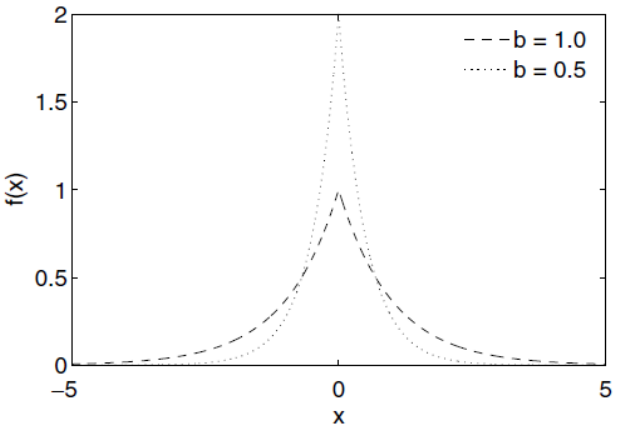


Fig. 7 Density function of Laplace distribution (a = 0, b = 0.5 and b = 1)

3.1.2 Mutation Operators

Uniform mutation (UM) [12]**.** It is the simplest mutation operator for real coding given by (12), it creates a random solution throughout the search space,

$$y^{(1)} = r_i(x_i^{(U)} - x_i^{(L)}) \tag{12}$$

where r_i is a random number in [0, 1], $x_i^{(L)}$ is the lower limit of the variable x_i and $x_i^{(U)}$ is the upper limit. This operator is independent of parents and is equivalent to a random initialization of the population. If the objective is to modify a parent using this operator, it can be performed with the equation (13),

$$y^{(1)} = x_i^{(1)} + (r_i - 0.5)\Delta_i \tag{13}$$

where Δ_i is the maximum perturbation defined by the user. Special care must be taken that this disturbance does not produce solutions beyond the limits, which is illustrated in Figure 8.

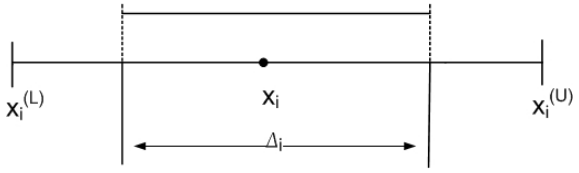


Fig. 8 Spread of the new solution with UM operator

Non Uniform Mutation (NUM). This operator was proposed by Michalewicz [12]. Here, the probability of creating a solution near the father is greater than the probability of creating an offspring away. The probability of create a solution close to the parent increases in each generation (*t*),

$$y^{(1)} = x_i^{(1)} + \tau(x_i^{(U)} - x_i^{(L)})(1 - r_i^{(1-t/t_{\max})^b}) \tag{14}$$

In (14) τ can take the -1 or 1 value, each one with a probability of 0.5. The parameter $t_{\max}$ is the allowed maximum number of generations, b is a user defined parameter, in [12] the value $b = 2$ was used.

Mäkinen, Periaux and Toivanen Mutation. This operator is also known as MPTM [13] is generated from points $x^{(1)} + (x_1, x_2, ...x_n)$ and point $y^{(1)} + (y_1, y_2, ...y_n)$, which is created using (15) to (17), where r_i is a uniformly distributed random number where $r_i \in [0,1]$. Then the mutation is given using (15).

$$y^{(1)} = (1 - \hat{t})x_i^{(L)} + \hat{t}x_i^{(U)}) \tag{15}$$

where

$$\hat{t} = \begin{cases} t - t(\dfrac{t-r}{t})^b, & r < t \\ t, & r = t \\ t + (1-t)(\dfrac{r-t}{1-t})^b, & r > t \end{cases} \tag{16}$$

and

$$t = \frac{x_i^{(1)} - x_i^{(L)}}{x_i^{(U)} - x_i^{(1)}} \tag{17}$$

The parameter b defined by the distribution of the mutation is also called the exponent of mutation. If $b = 1$, is a uniform mutation.

4 NSGA-II

The NSGA-II, Non Dominated Sorting Genetic Algorithm, is an improved version of the algorithm NSGA proposed by Srinivas and Deb [14]. This is a scheme for solving multiobjective optimization problems using the concept of non-dominance introduced by Goldberg [15], and a genetic algorithm to produce new solutions.

The schematic of this algorithm is shown in Fig 9, which shows that the first randomly generates a population P_t with N individuals and from them creates an R_t set of offspring solutions using the genetic operators of crossover and mutation. The parents and offspring generated are added to a new set called R_t, in this set a non-dominated sorting is made and then different fronts of non-dominated are obtained, so in each solution a crowding distance is assigned. A new parent population is formed from the fronts with the better range of non-dominance and the last frontier choosing solutions with the best crowding distance. The pseudocode for the NSGA-II is given in the algorithm 2:

Algorithm 2 NSGA-II pseudocode

$R_t = P_t \bigcup Q_t$
$F = fast - nondominated - sort(R_t)$
while $|P_{t+1}| < N$ **do**
  $crowding - distance - assignment(F_i)$
  $P_{t+1} = P_{t+1} \bigcup F_i$
end while
$Q_{t+1} = make - new - pop(p_{t+1})$
$t = t + 1$

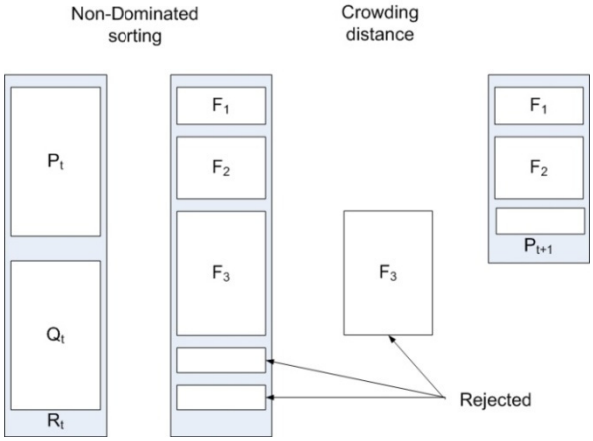


Fig. 9 Illustration of the NSGA-II process

The fast non-dominated sorting procedure separates the population into different fronts according to their level of dominance, see algorithm 3.

Algorithm 3 fast non-dominated sort

```
for all p ∈ P do
  for all p ∈ P do
    if q ≺ p then
      S_p = S_p ⋃ q
    else
      if p ≺ q then
        n_p = n_p + 1
      end if
    end if
    if n_p = 0 then
      F_i = F_1 ⋃ p
    end if
  end for
end for
i = 1
while F_i ≠ ∅ do
  H = ∅
  for all p ∈ F_i do
    for all p ∈ S_p do
      n_q = n_q − 1
      if n_q = 0 then
        H = H ⋃ q
      end if
    end for
  end for
end while
i = i + 1
F_i = H
```

This algorithm is used to maintain diversity of the population in each non-dominated frontier; for each individual, a density estimator named *Crowding distance* is *applied.* It is an average separation between two contiguous individuals; the extreme points of each front are assigned with an infinite distance. The pseudocode for this procedure is shown in algorithm 4.

Algorithm 4 Crowding distance procedure

```
l = |F|
for all i ∈ F do
  d_i = 0
  for all objetivo m do
    I = Sort (l, m)
    d_1^m = ∞ d_l^m = ∞
    for j ← 2, (l − 1) do
      d_j^m = d_j^m + F_{j+1}^m − F_{j−1}^m
    end for
  end for
end for
```

5 MNSGA-II

The MNSGA-II is an algorithm based in the NSGA-II structure with different sorting procedure, elite mechanism and genetic operators. The first step is to initialize a population of size *2N* where *N* is the population size. Each individual is an array of real numbers of size *k+m+2* as it is shown in Fig. 10; where *k* is the number of variables, *m* is the objective function and other two places for Pareto Rank (*R*), and Crowding Distance (d). The calculation of their objective values is realized in parallel form.

The next step is to assign Pareto rank to each individual using the parallel sorting procedure executing the following steps.

- **Step 1.** Sort all the solutions *R* in decreasing order of their first objective function (S_i).
- **Step 2.** Divide the population *R* in equal number of the parallel workers R_1, $R_2 \ldots$, R_W.
- **Step 3.** For each parallel worker, assign Pareto Rank to R_W and return the subpopulation P_W.
- **Step 4.** Synchronize the workers and join the population *P*.

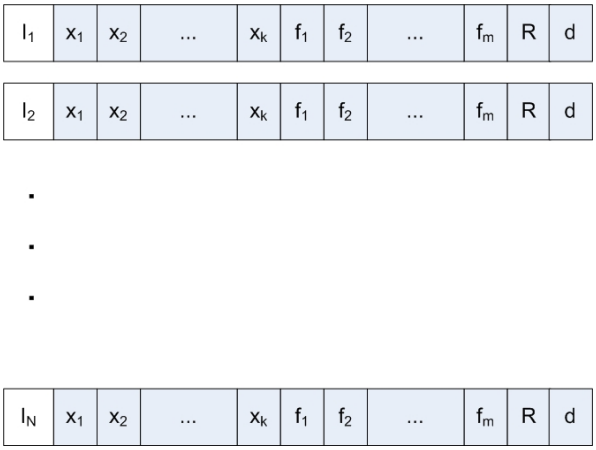


Fig. 10 Encoding of individuals in a population

The process of assigning the Pareto Rank is different from the NSGA-II, rather than separate the population in non-dominated fronts, each individual is assigned a Pareto Rank equal to their non-dominated fronts. In the MNSGA-II, the Pareto Rank is assigned separately to each subpopulation R_W. The first step is to find the non-dominated set of R_W using the Mishra fast algorithm, assigns Pareto Rank 1 to its set, add them to the new set P_W and delete them from R_W. For the new population R_W, find the non-dominated set, assign Pareto Rank 2 to each individual of the non-dominated set, add this set to P_W and delete it from R_W; this process continues until R_W is an empty set.

In this algorithm, one of two strategies to archive truncation is applied. The first one choose the solutions with the best Pareto Rank; for these strategies, the first step is to sort *P* in ascendant order of their Rank, if any solutions of the first *N* has Pareto Rank greater than 1, delete the last *N* individual and assign the crowding distance. If this conditional is not satisfied, then preserve only the solutions with Rank 1 and compute the crowding distance.

If the size of the population *P* is bigger than *N,* the second archive truncation strategy is applied. For this strategy, a crowding distance is assigned to each individual; then, sort *P* in decreasing order of their crowding distance and preserve only the best *N* solutions. This elitism strategy causes that if already found the *N* non-dominated set, and the algorithm is still running even more generations do not converge to a single point only generate more solutions along the Pareto frontier.

For computing the crowding distance to the population *P,* each solution of Rank *i,* is moved to a new subset F_i. Sort each F_i in ascendant order of their first objective function, and assign infinite distance to the first and the last solutions. For the other solutions compute the average between adjacent solutions, $d_j^m = d_j^m + F_{j+1}^m - F_{j-1}^m$; this process is iterative for each objective function.

Applying the elitism strategy, we have *N* Parents *P* with the best Rank and/or crowding distance. With the crowding comparator, we choose the best solutions and move to the mating pool to generate new offspring solutions.

The crowding comparator is a binary tournament that compares two solutions randomly selected from *P*, and the winner has the better Rank. If the Rank is equal, the winner has the biggest crowding distance and move it to the mating pool *B,* this process continues until *B* have *N* solutions.

To create new solutions, the crossover and mutation operators are executed in a parallel form with *W* workers at the same time. For this process, the mating pool *B* is a global memory, and each worker generates *N*=*W* offspring, and after this, they are synchronized to the offspring population *Q*.

Each worker chooses two random solutions from *B*, after generate two random numbers *c* (crossover probability) and *m* (mutation probability), if *p* is bigger than the selected crossover probability the worker chooses other parents, else apply the crossover operator. If *m* is bigger than the mutation probability the new solution is added to the offspring population *Q;* else, the mutation operator is applied, and the new solution is added to the offspring population *Q*, this process continues until have *N/W* offspring. If some offspring is out of the variable boundaries, it is replaced for some parents.

6 Results

The MNSGA-II algorithm was tested with five functions; the first two were proposed by Schaffer SCH1, SCH 2 [16], equations (18) and (19). The other three functions were proposed by Zitzler [17]; they are known as ZDT1, ZDT2 and ZDT3 functions and defined by equations (20) to (22).

All these functions have two objectives to be minimized, and its borders are known as Pareto optimal.

$$SCH1\begin{cases} f_1(x) = x^2 \\ f_1(x) = (x-2)^2 \end{cases} \tag{18}$$

$$SCH2\begin{cases} f_1(x) = \begin{cases} -x, & x \le 1 \\ x-2, & 1 \le x \le 3 \\ 4-x, & 3 \le x \le 4 \\ x-4, & x > 4 \end{cases} \\ f_2(x) = (x-5)^2 \end{cases} \tag{19}$$

$$ZDT1\begin{cases} f_1 = x_1 \\ g = 1 + \dfrac{9}{n+1} \quad (\sum_{i=2}^{n} x_i) \\ f_1 = 1 - \sqrt{x_1 / g} \end{cases} \tag{20}$$

$$ZDT2\begin{cases} f_1 = x_1 \\ g = 1 + \dfrac{9}{n+1} \quad (\sum_{i=2}^{n} x_i) \\ f_1 = 1 - (x_1 / g)^2 \end{cases} \tag{21}$$

$$ZDT3\begin{cases} f_1 = x_1 \\ g = 1 + \dfrac{9}{n+1} (\sum_{i=2}^{n} x_i \\ f_1 = 1 - \sqrt{x_1 / g} - (x_1 / g)\sin(10\pi x_i) \end{cases} \tag{22}$$

Table 1 shows the results obtained by the algorithm MNSGA-II for the SCH1 function using a population of 100 individuals and 10 generations. For all tests, we used the non-uniform mutation operator combined with different crossover operators (BLX-α, SBX, LX and LinX). The table shows the average of all individuals in the decision variable (x_1), the minimum, maximum and average crowding distance.

Table 1 Results of the algorithm MNSGA-II for function SCH1

		x_1			
		BLX	SBX	LX	LINX
NUM	Mean	1.02367404	1.01508669	0.99508727	1.03645464
	Min	-0.00069813	0.0093411	0.00771846	0
	Max	2.00827094	1.99994377	1.99943546	1.996125
	Crowding	0.12089923	0.12632004	0.12329614	0.12664984

For function SCH1, the Pareto optimality is in the interval $x \in [0,2]$; Table 1, shows that any operator recombination, used in all the obtained solutions, are within this range, the crowding distance shows that all tests have similar distribution in the Pareto front.

Table 2 shows the results obtained for different operators for the SCH2 function, and the NUM, RM and MPTM mutation operator, considering that crossover operator BLX-α was used, a population of 100 individuals and 10 generations. Pareto optimality is in $x \in [1,2] \cup [4,5]$; for all the experiments, the results are within this range, and the separations of the solutions are similar. Fig. 11 shows the non-dominated set obtained after 10 generations using BLX-α and non-uniform mutation operators. MNSGA-II algorithm for functions SCH1 and SCH2 have rapid convergence, and the resulting solutions are close to Pareto optimality.

Table 2 Results of the algorithm MNSGA-II for function SCH2

		x_1		
		NUM	RM	MPTM
BLX-α	Mean	2.044492032	2.154450159	2.043589034
	Min	0.998364469	0.998663251	1.001569329
	Max	4.982431601	4.999366373	4.995915783
	Crowding	0.329002044	0.322671774	0.327572686

Table 3 shows the results obtained with the algorithm MNSGA-II for ZDT1 problem using a population of 100 individuals, as this problem is more complex than the two discussed above algorithm, it was run for 200 generations. The Pareto optimal for this function occurs when $x_1^* \in [1,2]$ and $x_i^* = 0$ for *i=2,3,..,30*, to analyze the results, variables x_2 *to* x_{30} are shown in the same column; in each row, the average of all the minimum values, maximum and average crowding distance are shown.

For this function using the BLX-0.5, combined with any of the three mutation operators, solutions close to the true Pareto optimal were obtained. With the SBX operator, only in combination with the non-uniform mutation operator (NUM) good results were obtained. For the Laplace crossover operator (LX), using it with

random mutation operators (RM) and Mäkinen, Periaux and Toivanen (MPTM), were not obtained good results. With linear crossover operator (LinX) and all mutation operators, good results were obtained. Fig. 12 shows that using the Laplace crossover operator and non-uniform mutation operator the best result was obtained. The worst result was achieved using the Laplace operator with the RM.

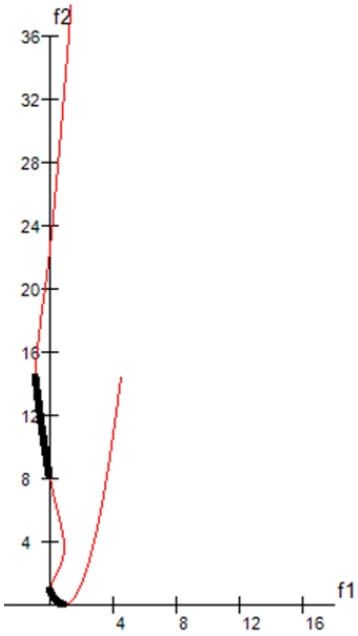


Fig. 11 Solution of the problem SCH2 using BLX-0.5 and NUM operators

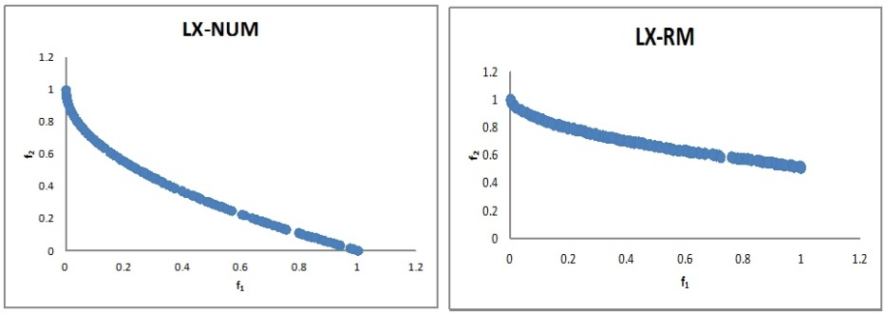


Fig. 12 Best and worst solutions of the problem ZDT1 using MNSGA-II algorithm

The analysis of the results obtained for the ZDT2 problem is shown in Table 4, the format presented is the same as for ZDT2 function. The Pareto optimal for this not convex function is $x_1^* \in [1,2]$ and $x_i^* = 0$ for $i=2,3,..,30$. The algorithm was run for each experiment 200 generations with a population of 100 individuals.

Table 3 Results of the algorithm MNSGA-II for function ZDT1

		NUM		RM		MPTM	
		X_1	$X_{2\text{-}30}$	X_1	$X_{2\text{-}30}$	X_1	$X_{2\text{-}30}$
BLX	Mean	0.4339473	0.00690591	0.42883798	0.01570027	0.42200358	0.00735316
	Min	7.8297E-06	0.00113467	0	0.00099613	1.7811E-05	0.00101034
	Max	0.99897266	0.02097337	0.99316471	0.46624688	0.99964575	0.02511185
	Crowding	0.03042085	0	0.03173915	0	0.03088147	0
SBX	Mean	0.50126557	0.00874785	0.45659	0.23042746	0.4802173	0.33604821
	Min	0	0	0	0.00539689	0	0.09194388
	Max	0.99999376	0.04601851	0.99969187	0.84533201	0.99959848	0.69220241
	Crowding	0.03686594	0	0.02695703	0	0.02325005	0
LX	Mean	0.41278814	0.00101626	0.46800612	0.4064682	0.49739841	0.38528937
	Min	0	0	0	0.02727913	4.0914E-06	0.14511811
	Max	0.99959856	0.00493435	0.99506886	0.9248732	0.99594635	0.78231683
	Crowding	0.0335355	0	0.02726643	0	0.02549761	0
LX	Mean	0.39255204	0.00106852	0.4165725	0.00134428	0.42498091	0.00120388
	Min	0	0	0	0	0	0
	Max	0.9936501	0.01089806	0.99999953	0.01006621	0.99982034	0.013028
	Crowding	0.03114298	0	0.03151869	0	0.03167497	0

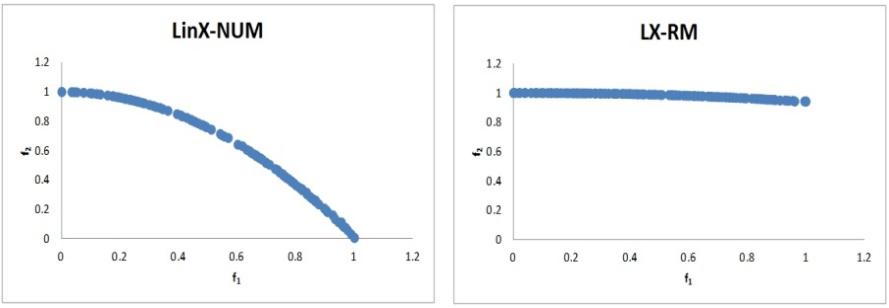


Fig. 13 Best and worst solutions of the problem ZDT2 using MNSGA-II algorithm

The best result for the problem ZDT2 is achieved using linear crossover operator and non uniform mutation. Laplace operator combined with random mutation is the worst, see Fig.13. For this problem, the best results were obtained

Table 4 Results of the algorithm MNSGA-II for function ZDT2

		NUM		RM		MPTM	
		X_1	$X_{2\text{-}30}$	X_1	$X_{2\text{-}30}$	X_1	$X_{2\text{-}30}$
BLX	Mean	0.5981419	0.00392449	0.57187174	0.01209529	0.55150488	0.00990163
	Min	0.00016666	0.00024646	0	0.00045374	0	0.00056134
	Max	0.99935171	0.01411728	0.99848268	0.51312742	0.9942452	0.35321205
	Crowding	0.02975808	0	0.03054849	0	0.02996389	0
SBX	Mean	0.54964984	0.16408595	0.54773128	0.25473612	0.5236946	0.26904922
	Min	0.00022273	0.08708248	0.00012835	0.0818626	7.5804E-06	0.16226497
	Max	0.99959916	0.25942415	0.99908381	0.53855168	0.99952305	0.43568235
	Crowding	0.01869885	0	0.01850633	0	0.01730054	0
LX	Mean	0.51527241	0.24490449	0.486541	0.40772558	0.50735896	0.28952409
	Min	0	0.15979978	0	0.05219498	0.00011129	0.15755669
	Max	0.99812493	0.66417564	0.99879269	0.88404645	0.99466544	0.69930188
	Crowding	0.01884936	0	0.0203338	0	0.01926876	0
LINX	Mean	0.55578411	0.00083943	0.59212062	0.0045958	0.57124406	0.00458354
	Min	0	0	0	0	0	0
	Max	0.9991459	0.00478766	0.99827288	0.39116457	0.99996375	0.22249817
	Crowding	0.03111558	0	0.03105473	0	0.03238907	0

with the combination of the BLX operator with any of the three mutation operators. With the simulated crossover binary and Laplace mutation operators, results were far from the true Pareto optimal.

For the multi-objective optimization problem ZDT3, the results are shown in Table 5; this function has the Pareto optimal $x_i^* = 0$ for *i=2,3,..,30* and some values in the range $0 \le x_1^* \le 0$ which causes a discontinuous Pareto optimal region. This discontinuity causes that the convergence to the optimal is difficult; for this reason, each experiment was run for 300 iterations with 100 individuals in the population. For this function, the best results were obtained with the linear crossing operator and closer to the Pareto optimal frontier was combined with non-uniform mutation, see Fig 14. With the BLX, we obtained good results using any of the three mutation carriers. With the Laplace and simulated binary crossover operators, results were far from the Pareto optimal frontier.

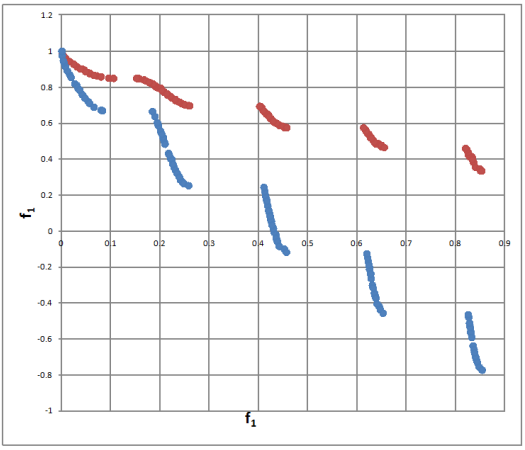


Fig. 14 Best and worst solutions of the problem ZDT3 using MNSGA-II algorithm

Table 5 Results of the algorithm MNSGA-II for function ZDT2

		NUM		RM		MPTM	
		X_1	$X_{2\text{-}30}$	X_1	$X_{2\text{-}30}$	X_1	$X_{2\text{-}30}$
BLX	Mean	0.42570259	0.01228299	0.40358189	0.02659435	0.39185833	0.01808241
	Min	3.1506E-05	0.00206034	0	0.00691783	0	0.00119918
	Max	0.85452111	0.04154692	0.85230299	0.6657291	0.8531614	0.43645557
	Crowding	0.04619395	0	0.04565728	0	0.04625058	0
SBX	Mean	0.35315376	0.25562093	0.34343446	0.33213008	0.34237021	0.29840395
	Min	8.4748E-06	0.03394595	0	0.06226514	5.7935E-05	0.11630487
	Max	0.851223	0.65050606	0.8497918	0.81487461	0.85213701	0.52063588
	Crowding	0.02803276	0	0.02681256	0	0.02732888	0
LX	Mean	0.34810305	0.25239654	0.36111901	0.35738852	0.37908293	0.3742102
	Min	0	0.19787363	0	0.27278216	2.3712E-06	0.06594863
	Max	0.84875631	0.64973904	0.85324118	0.69013451	0.85176722	0.81087185
	Crowding	0.03066285	0	0.02837957	0	0.02859414	0
LINX	Mean	0.3956875	0.00170911	0.41892444	0.00185217	0.41556922	0.00236408
	Min	0	0	0	0	0	0
	Max	0.8527739	0.01171929	0.85189492	0.01567379	0.85211903	0.0235468
	Crowding	0.0482462	0	0.04695875	0	0.04802958	0

7 Conclusions

The MNSGA-II is an improved version of the NSGA-II because it handles elitism in two different ways. The first way is selecting the best Pareto ranks, from which a mating pool is generated to obtain the parents using tournament selection to generate the offsprings. The second way applies when we have *N* non-dominated individuals, for which we select the individuals with the best crowding distance.

The used elitism strategy is an improvement to the algorithm NSGA-II, because the area of MOOEA has been given great importance to the spread of solutions in the Pareto optimal front. Being the only stop criterion the number of generations; if the algorithm is iterated for more generations needed to reach the Pareto frontier, the only thing that will occur is an improvement of the spread.

The proposed scheme for applying genetic operators in parallel can be used for any genetic algorithm, the suggestion to use shared memory allows maintaining the diversity of the population, but this strategy only works if parallelization is used for large granularity.

In this work we have presented a comparative analysis of the MNSGA-II using different crossover and mutation operator. The successful of the different combinations depends on the problem; therefore, all the operators can work for the MNSGA-II being difficult to choose one for a big diversity of applications.

Acknowledgment. The authors would like to thank the Instituto Politécnico Nacional (IPN), Comisión de Operación y Fomento de Actividades Académicas (COFAA), and the Mexican Consejo Nacional de Ciencia y Tecnología (CONACYT) for supporting our research activities.

References

1. Rangaiah, G.P.: Multi-Objective Optimization: Techniques and Applications in Chemical Engineering. World Scientific Publishing CO. Pthe. Ltd. (2009)
2. Abraham, A., Jain, L.C., Goldberg, R.: Evolutionary Multiobjective Optimization: Theoretical Advances And Applications. Springer (2005)
3. Konak, A., Coit, D.W., Smith, A.E.: Multi - objective optimization using genetic algorithms: A tutorial. Reliability Engineering and System Safety 91 (2006)
4. Holland, J.H.: Adaptation in natural and artificial systems. University of Michigan Press, Ann Arbor (1975)
5. Mitchell, M.: An introduction to Genetic Algorithms. MIT Press, Cambridge (1998)
6. Wright, A.H.: Genetic algorithms for real parameter optimization. In: Foundations of Genetic Algorithms, pp. 205–218. Morgan Kaufmann (1991)
7. Michalewicz, Z., Logan, T.: Evolutionary operators for continuous convex parameter space. In: Sebald, L.A.V. (ed.) Proceeding of 3rd Annual Conference on Evolutionary Programming, p. 8497. World Scientific (1994)
8. Eshelman, L.J., Schaffer, J.D.: Real-coded genetic algorithms and interval-schemata. In: Whitley, D.L. (ed.) Foundation of Genetic Algorithms 2, pp. 187–202. Morgan Kaufmann, San Mateo (1993)
9. Agrawal, R.B., Deb, K., Agrawal, R.B.: Simulated binary crossover for continuous search space. Tech. Rep. (1994)

10. Deb, K., Georg Beyer, H.: Self-adaptive genetic algorithms with simulated binary crossover. Complex Systems 9, 431–454 (1999)
11. Deep, K., Thakur, M.: A new crossover operator for real coded genetic algorithms. Applied Mathematics and Computation 188(1), 895–911 (2007)
12. Michalewicz, Z.: Genetic algorithms + data structures = evolution programs, 3rd edn. Springer, London (1996)
13. Makinen, R.A., Toivanen, J., Toivanen, M.J., Periaux, J.: Multidisciplinary shape optimization in aerodynamics and electromagnetics using genetic algorithms
14. Srinivas, N., Deb, K.: Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms. Evolutionary Computation (1994)
15. Goldberg, D.E.: Genetic Algorithms in Search. In: Optimization and Machine Learning, Addison-Wesley Longman Publishing Co., Inc. (1989)
16. Schaffer, J.D.: Some experiments in machine learning using vector evaluated genetic algorithms (artificial intelligence, optimization, adaptation, pattern recognition), Vanderbilt University (1984)
17. Zitzler, E.: Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications (1999)
18. Deb, K., Pratap, A., Agarwal, S.R., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation (2002)
19. Deb, K., Pratap, A., Agarwal, S.R., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation (2002)
20. Corne, D.W., Jerram, N.R., Knowles, J.D., Oates, M.J., Martin, J.: PESA-II: Region-based Selection in Evolutionary Multiobjective Optimization. In: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2001 (2001)
21. Li, M., Liu, L., Lin, D.: A fast steady-state epsilon-dominance multi-objective evolutionary algorithm. Comput. Optim. Appl. (2011)
22. Deb, K.: Multi-Objective Optimization using Evolutionary Algorithms. Wiley (2001)
23. Coello, C.A.: Evolutionary Algorithms for Solving Multi-Objective Problems. Springer (2007)
24. Abido, M.A.: Multiobjective evolutionary algorithms for electric power dispatch problem. IEEE Trans. Evolutionary Computation (2006)
25. Formiga, K.T.M., Chaudhry, F.H., Cheung, P.B., Reis, L.F.R.: Optimal Design of Water Distribution System by Multiobjective Evolutionary Methods. In: Fonseca, C.M., Fleming, P.J., Zitzler, E., Deb, K., Thiele, L. (eds.) EMO 2003. LNCS, vol. 2632, pp. 677–691. Springer, Heidelberg (2003)
26. Ahmed, F., Deb, K.: Multi-objective path planning using spline representation. In: ROBIO (2011)
27. Eshelman, L.J., Schaffer, J.D.: Real-coded genetic algorithms and interval-schemata. In: Foundation of Genetic Algorithms, vol. 2, pp. 187–182 (1993)

Author Index