# Evolutionary Computing for Topology Optimization of Type-2 Fuzzy Controllers

Oscar Castillo, Gabriel Huesca, and Fevrier Valdez

Department of Computer Science, Tijuana Institute of Technology, Tijuana, Mexico

**Abstract.** We describe in this paper the use of hierarchical genetic algorithms for fuzzy system optimization in intelligent control. In particular, we consider the problem of optimizing the number of rules and membership functions using an evolutionary approach. The hierarchical genetic algorithm enables the optimization of the fuzzy system design for a particular application. We illustrate the approach with the case of intelligent control in a medical application. Simulation results for this application show that we are able to find an optimal set of rules and membership functions for the fuzzy system.

## 1 Introduction

We describe in this paper the application of a Hierarchical Genetic Algorithm (HGA) for fuzzy system optimization (Man et al. 1999). In particular, we consider the problem of finding the optimal set of rules and membership functions for a specific application (Yen and Langari 1999). The HGA is used to search for this optimal set of rules and membership functions, according to the data about the problem. We consider, as an illustration, the case of a fuzzy system for intelligent control.

Fuzzy systems are capable of handling complex, non-linear and sometimes mathematically intangible dynamic systems using simple solutions (Jang et al. 1997). Very often, fuzzy systems may provide a better performance than conventional non-fuzzy approaches with less development cost (Procyk and Mamdani 1979). However, to obtain an optimal set of fuzzy membership functions and rules is not an easy task. It requires time, experience and skills of the designer for the tedious fuzzy tuning exercise. In principle, there is no general rule or method for the fuzzy logic set-up, although a heuristic and iterative procedure for modifying the membership functions to improve performance has been proposed. Recently, many researchers have considered a number of intelligent schemes for the task of tuning the fuzzy system. The noticeable Neural Network (NN) approach (Jang and Sun 1995) and the Genetic Algorithm (GA) approach (Homaifar and McCormick 1995) to optimize either the membership functions or rules, have become a trend for fuzzy logic system development.

The HGA approach differs from the other techniques in that it has the ability to reach an optimal set of membership functions and rules without a known fuzzy system topology (Tang et al. 1998). During the optimization phase, the membership functions need not be fixed. Throughout the genetic operations (Holland 1975), a reduced fuzzy system including the number of membership functions and fuzzy rules will be generated (Yoshikawa et al. 1996). The HGA approach has a number of advantages:

1. An optimal and the least number of membership functions and rules are obtained
2. No pre-fixed fuzzy structure is necessary, and
3. Simpler implementing procedures and less cost are involved.

We consider in this paper the case of automatic anesthesia control in human patients for testing the optimized fuzzy controller. We did have, as a reference, the best fuzzy controller that was developed for the automatic anesthesia control (Karr and Gentry 1993, Lozano 2003), and we consider the optimization of this controller using the HGA approach. After applying the genetic algorithm the number of fuzzy rules was reduced from 12 to 9 with a similar performance of the fuzzy controller. Of course, the parameters of the membership functions were also tuned by the genetic algorithm. We did compare the simulation results of the optimized fuzzy controllers obtained with the HGA against the best fuzzy controller that was obtained previously with expert knowledge, and control is achieved in a similar fashion. Since simulation results are similar, and the number of fuzzy rules was reduced, we can conclude that the HGA approach is a good alternative for designing fuzzy systems. We have to mention that Type-2 fuzzy systems are considered in this research work, which are more difficult to design and optimize.

## 2 Genetic Algorithms for Optimization

In this paper, we used a floating-point genetic algorithm (Castillo and Melin 2001) to adjust the parameter vector $\theta$, specifically we used the Breeder Genetic Algorithm (BGA). The genetic algorithm is used to optimize the fuzzy system for control that will be described later (Castillo and Melin 2003). A BGA can be described by the following equation:

$$\text{BGA}=(P_g^0, N, T, \Gamma, \Delta, HC, F, \text{term}) \qquad (1)$$

where: $P_g^0$=initial population, $N$=the size of the population, $T$=the truncation threshold, $\Gamma$=the recombination operator, $\Delta$=the mutation operator, $HC$=the hill climbing method, $F$=the fitness function, $term$=the termination criterion.

The BGA uses a selection scheme called truncation selection. The %T best individuals are selected and mated randomly until the number of offspring is equal the size of the population. The offspring generation is equal to the size of the population. The offspring generation replaces the parent population. The best individual found so far will remain in the population. Self-mating is prohibited (Melin and Castillo 2002). As a recombination operator we used "extended intermediate recombination", defined as: If $x =(x_1,...x_n)$ *and* y $y=(y_1,...,y_n)$ are the parents, then the successor $z=(z_1,...,z_n)$ is calculated by:

$$z_i=x_i+\alpha_i(y_i-x_i) \qquad i=1,\ldots n \qquad (2)$$

The mutation operator is defined as follows: A variable $x_i$ is selected with probability $p_m$ for mutation. The BGA normally uses $p_m = 1/n$. At least one variable will be mutated. A value out of the interval $[-range_i, range_i]$ is added to the variable. $range_i$ defines the mutation range. It is normally set to $(0.1 \text{ x } searchinterval_i)$. searchinterval$_i$ is the domain of definition for variable $x_i$. The new value $z_i$ is computed according to

$$z_i=x_i\pm range_i\cdot\delta \qquad (3)$$

The + or – sign is chosen with probability 0.5. δ is computed from a distribution which prefers small values. This is realized as follows

$$\delta=\sum_{i=0}^{15}\alpha_i 2^i \qquad \alpha_i \in 0,1 \qquad (4)$$

Before mutation we set $\alpha_i$=0. Then each $\alpha_i$ is mutated to 1 with probability $p_\delta$=1/16. Only $\alpha_i$=1 contributes to the sum. On the average there will be just one $\alpha_i$ with value 1, say $\alpha_j$. Then δ is given by

$$\delta=2^{-j} \qquad (5)$$

The standard BGA mutation operator is able to generate any point in the hypercube with center *x* defined by $x_i \pm range_i$. But it generates values much more often in the neighborhood of x. In the above standard setting, the mutation operator is able to locate the optimal $x_i$ up to a precision of $ramge_i \cdot 2^{-150}$.

To monitor the convergence rate of the LMS algorithm, we computed a short term average of the squared error $e^2(n)$ using

$$ASE(m) = \frac{1}{K} \sum_{k=n+1}^{n+K} e^2(k) \tag{6}$$

where *m=n/K=1,2,....* The averaging interval K may be selected to be (approximately) K=10N. The effect of the choice of the step size parameter Δ on the convergence rate of LMS algorithm may be observed by monitoring the ASE(m).

## 2.1 Genetic Algorithm for Optimization

The proposed genetic algorithm is as follows:

1. We use real numbers as a genetic representation of the problem.
2. We initialize variable i with zero (i=0).
3. We create an initial random population $P_i$, in this case ($P_0$). Each individual of the population has n dimensions and, each coefficient of the fuzzy system corresponds to one dimension.
4. We calculate the normalized fitness of each individual of the population using linear scaling with displacement (Melin and Castillo 2002), in the following form:

$$f_i' = f_i + \frac{1}{N} \sum |f_i| + \left| \min_i (f_i) \right| \qquad \forall i$$

5. We normalize the fitness of each individual using:

$$F_i = \frac{f_i'}{\sum_{i=1}^{N} f_i'} \qquad \forall\, i$$

6. We sort the individuals from greater to lower fitness.
7. We use the truncated selection method, selecting the %T best individuals, for example if there are 500 individuals and, then we select 0.30*500=150 individuals.

8. We apply random crossover, to the individuals in the population (the 150 best ones) with the goal of creating a new population (of 500 individuals). Crossover with it self is not allowed, and all the individuals have to participate. To perform this operation we apply the genetic operator of extended intermediate recombination as follows:

   If $x=(x_1,...,x_n)$ and $y=(y_1,...,y_n)$ are the parents, then the successors $z=(z_1,...,z_n)$ are calculated by, $z_i=x_i+\alpha_i(y_i-x_i)$ *for* $i=1,...,n$ where α is a scaling factor selected randomly in the interval $[-d,1+d]$. In intermediate recombination $d=0$, and for extended $d>0$, a good choice is $d=0.25$, which is the one that we used.
9. We apply the mutation genetic operator of BGA. In this case, we select an individual with probability $p_m=1/n$ (where n represents the working dimension, in this case n=25, which is the number of coefficients in the membership functions). The mutation operator calculates the new individuals $z_i$ of the population in the following form: $z_i=x_i \pm range_i\, \delta$ we can note from this equation that we are actually adding to the original individual a value in the interval: $[-range_i,range_i]$ the range is defined as the search interval, which in this case is the domain of variable $x_i$, the sign ± is selected randomly with probability of 0.5, and is calculated using the following formula,

$$\delta = \sum_{i=0}^{m-1} \alpha_i 2^{-i} \qquad \alpha_i \in 0,1$$

   Common used values in this equation are $m=16$ y $m=20$. Before mutation we initiate with $_i=0$, then for each $_i$ we mutate to 1 with probability $p=1/m$.
10. Let i=i+1, and continue with step 4.

## 3 Evolution of Fuzzy Systems

Ever since the very first introduction of the fundamental concept of fuzzy logic by Zadeh in 1973, its use in engineering disciplines has been widely studied. Its main attraction undoubtedly lies in the unique characteristics that fuzzy logic systems possess. They are capable of handling complex, non-linear dynamic systems using simple solutions. Very often, fuzzy systems provide a better performance than conventional non-fuzzy approaches with less development cost.

However, to obtain an optimal set of fuzzy membership functions and rules is not an easy task. It requires time, experience, and skills of the operator for the tedious fuzzy tuning exercise. In principle, there is no general rule or method for the fuzzy logic set-up. Recently, many researchers have considered a number of intelligent techniques for the task of tuning the fuzzy set. Here, another innovative scheme is described (Tang et al. 1998). This approach has the ability to reach an optimal set of membership functions and rules without a known overall fuzzy set topology. The conceptual idea of this approach is to have an automatic and intelligent scheme to tune the membership functions and rules, in which the conventional closed loop fuzzy control strategy remains unchanged, as indicated in Figure 1.

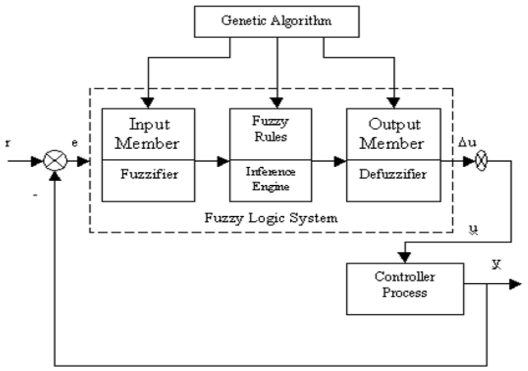

**Fig. 1** Genetic algorithm for a fuzzy control system.

In this case, the chromosome of a particular system is shown in Figure 2. The chromosome consists of two types of genes, the control genes and parameter genes. The control genes, in the form of bits, determine the membership function activation, whereas the parameter genes are in the form of real numbers to represent the membership functions.

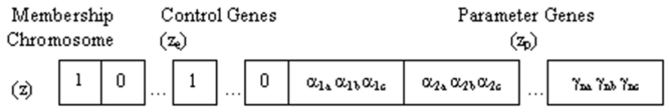

**Fig. 2** Chromosome structure for the fuzzy system.

To obtain a complete design for the fuzzy control system, an appropriate set of fuzzy rules is required to ensure system performance. At this point it should be stressed that the introduction of the control genes is done to govern the number of fuzzy subsets in the system. Once the formulation of the chromosome has been set for the fuzzy membership functions and rules, the genetic operation cycle can be performed. This cycle of operation for the fuzzy control system optimization using a genetic algorithm is illustrated in Figure 3. There are two population pools, one for storing the membership chromosomes and the other for storing the fuzzy rule chromosomes. We can see this in Figure 3 as the membership population and fuzzy rule population, respectively. Considering that there are various types of gene structure, a number of different genetic operations can be used. For the crossover operation, a one-point crossover is applied separately for both the control and parameter genes of the membership chromosomes within certain operation rates. There is no crossover operation for fuzzy rule chromosomes since only one suitable rule set can be assisted.

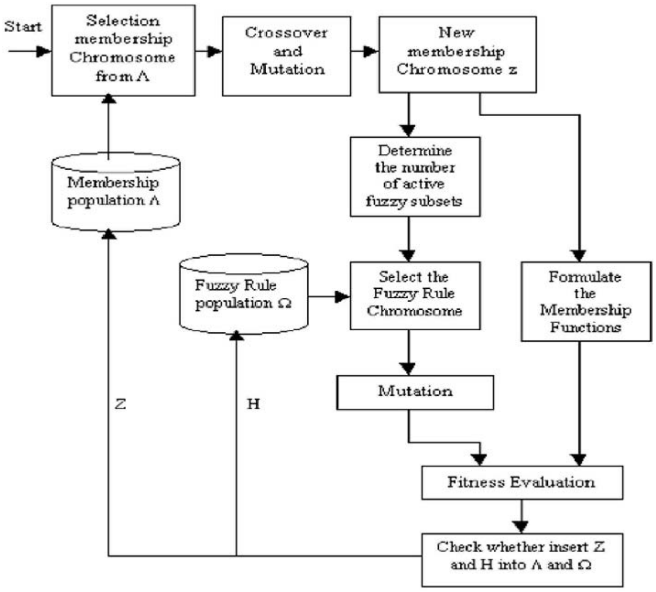
Start
Selection membership Chromosome from Λ
Crossover and Mutation
New membership Chromosome z
Determine the number of active fuzzy subsets
Membership population Λ
Fuzzy Rule population Ω
Select the Fuzzy Rule Chromosome
Formulate the Membership Functions
Mutation
Z
H
Fitness Evaluation
Check whether insert Z and H into Λ and Ω

**Fig. 3.** Genetic cycle for fuzzy system optimization.

Bit mutation is applied for the control genes of the membership chromosome. Each bit of the control gene is flipped if a probability test is satisfied (a randomly generated number is smaller than a predefined rate). As for the parameter genes, which are real number represented, random mutation is applied.

The fitness function can be defined in this case as follows:

$$f_i = \Sigma / y(k) - r(k) / \qquad (7)$$

where Σ indicates the sum for all the data points in the training set, and y(k) represents the real output of the fuzzy system and r(k) is the reference output. This fitness value measures how well the fuzzy system is approximating the real data of the problem.

## 4 Type-2 Fuzzy Logic

The concept of a type-2 fuzzy set, was introduced by Zadeh (Melin and Castillo 2002) 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 (Castillo and Melin 2001, Melin and Castillo 2002) 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 (Castillo and Melin 2001). 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.

Example: Consider the case of a fuzzy set characterized by a Gaussian membership function with mean m and a standard deviation that can take values in $[\sigma_1,\sigma_2]$, i.e.,

$$\mu(x)=\exp\{-½[(x-m)/\sigma]^2\}; \quad \sigma \in [\sigma_1,\sigma_2] \tag{8}$$

Corresponding to each value of $\sigma$, we will get a different membership curve (Figure 4). So, the membership grade of any particular x (except x=m) can take any of a number of possible values depending upon the value of $\sigma$, i.e., the membership grade is not a crisp number, it is a fuzzy set. Figure 4 shows the domain of the fuzzy set associated with x=0.7.

The basics of fuzzy logic do not change from type-1 to type-2 fuzzy sets, and in general, will not change for any type-n (Castillo and Melin 2003). A higher-type number just indicates a higher “degree of fuzziness”. Since a higher type changes the nature of the membership functions, the operations that depend on the membership functions change; however, the basic principles of fuzzy logic are independent of the nature of membership functions and hence, do not change. In Figure 5 we show the general structure of a type-2 fuzzy system. We assume that both antecedent and consequent sets are type-2; however, this need not necessarily be the case in practice.

The structure of the type-2 fuzzy rules is the same as for the type-1 case because the distinction between type-2 and type-1 is associated with the nature of the membership functions. Hence, the only difference is that now some or all the sets involved in the rules are of type-2. In a type-1 fuzzy system, where the output sets are type-1 fuzzy sets, we perform defuzzification in order to get a number, which is in some sense a crisp (type-0) representative of the combined output sets. In the type-2 case, the output sets are type-2; so we have to use extended versions of type-1 defuzzification methods. Since type-1 defuzzification gives a crisp number at the output of the fuzzy system, the extended defuzzification operation in the type-2 case gives a type-1 fuzzy set at the output. Since this operation takes us from the type-2 output sets of the fuzzy system to a type-1 set, we can call this operation "type reduction" and call the type-1 fuzzy set so obtained a "type-reduced set". The type-reduced fuzzy set may then be defuzzified to obtain a single crisp number; however, in many applications, the type-reduced set may be more important than a single crisp number. Type-2 sets can be used to convey the uncertainties in membership functions of type-1 fuzzy sets, due to the dependence of the membership functions on available linguistic and numerical information.

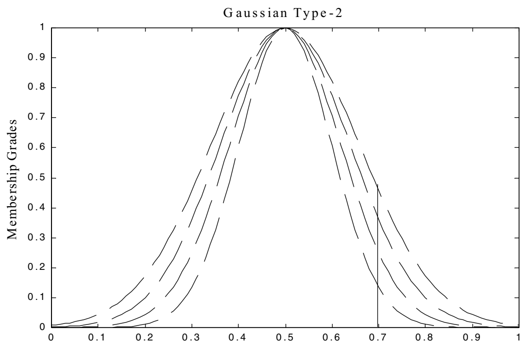

**Fig. 4.** A type-2 fuzzy set representing a type-1 set with uncertain deviation.

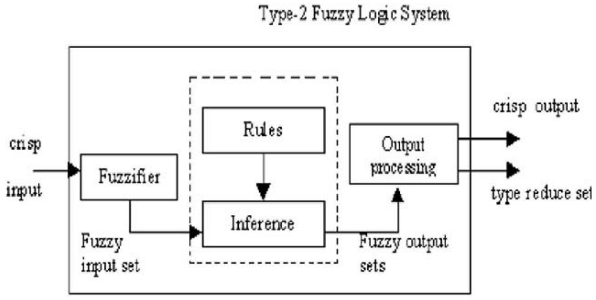

**Fig. 5.** Structure of a type-2 fuzzy system.

## 5 Application to Intelligent Control

We consider the case of controlling the anesthesia given to a patient as the problem for finding the optimal fuzzy system for control (Lozano 2003). The complete implementation was done in the MATLAB programming language. The fuzzy systems were build automatically by using the Fuzzy Logic Toolbox, and genetic algorithm was coded directly in the MATLAB language. The fuzzy systems for control are the individuals used in the genetic algorithm, and these are evaluated by comparing them to the ideal control given by the experts. In other words, we compare the performance of the fuzzy systems that are generated by the genetic algorithm, against the ideal control system given by the experts in this application.

### 5.1 Anesthesia Control Using Fuzzy Logic

The main task of the anesthesist, during and operation, is to control anesthesia concentration. In any case, anesthesia concentration can't be measured directly. For this reason, the anesthesist uses indirect information, like the heartbeat, pressure, and motor activity. The anesthesia concentration is controlled using a medicine, which can be given by a shot or by a mix of gases. We consider here the use of isoflurance, which is usually given in a concentration of 0 to 2% with oxygen. In Figure 6 we show a block diagram of the controller.

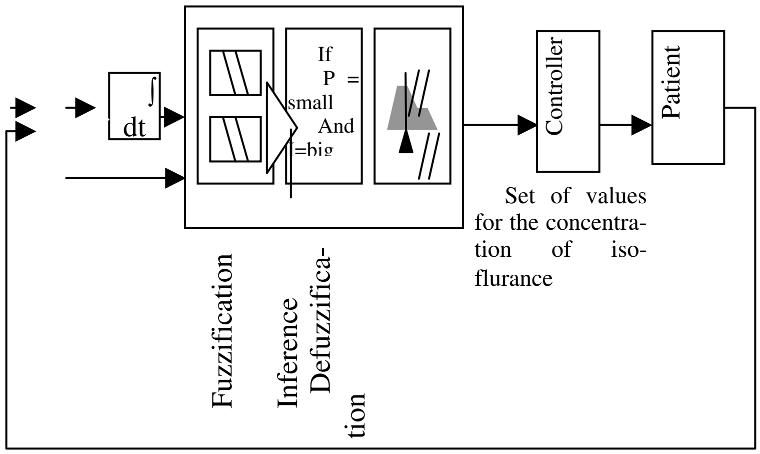

**Fig. 6.** Architecture of the fuzzy control system.

The air that is exhaled by the patient contains a specific concentration of isoflurance, and it is re-circulated to the patient. As consequence, we can measure isoflurance concentration on the inhaled and exhaled air by the patient, to estimate isoflurance concentration on the patient's blood. From the control engineering point of view, the task by the anesthesist is to maintain anesthesia concentration between the high level W (threshold to wake up) and the low level E (threshold to success). These levels are difficult to be determine in a changing environment and also are dependent on the patient's condition. For this reason, it is important to automate this anesthesia control, to perform this task more efficiently and accurately, and also to free the anesthesist from this time consuming job. The anesthesist can then concentrate in doing other task during operation of a patient.

The first automated system for anesthesia control was developed using a PID controller in the 60's. However, this system was not very succesful due to the non-linear nature of the problem of anesthesia control. After this first attempt, adaptive control was proposed to automate anesthesia control, but robustness was the problem in this case. For these reasons, fuzzy logic was proposed for solving this problem.

## 5.2 Characteristics of the Fuzzy Controller

In this section we describe the main characteristics of the fuzzy controller for anesthesia control. We will define input and output variable of the fuzzy system. Also, the fuzzy rules of fuzzy controller previously designed will be described.

The fuzzy system is defined as follows:

1. Input variables: Blood pressure and Error
2. Output variable: Isoflurance concentration
3. Nine fuzzy if-then rules of the optimized system, which is the base for comparison
4. 12 fuzzy if-then rules of an initial system to begin the optimization cycle of the genetic algorithm.

The linguistic values used in the fuzzy rules are the following:

PB = Positive Big
PS = Positive Small
ZERO = zero
NB =Negative Big
NS = Negative Small

We show below a sample set of fuzzy rules that are used in the fuzzy inference system that is represented in the genetic algorithm for optimization.

if Blood pressure is NB and error is NB then conc_isoflurance is PS
if Blood pressures is PS then conc_isoflurance is NS
if Blood pressure is NB then conc_isoflurance is PB
if Blood pressure is PB then conc_isoflurance is NB
if Blood pressure is ZERO and error is ZERO then conc_isoflurance is ZERO
if Blood pressure is ZERO and error is PS then conc_isoflurance is NS
if Blood pressure is ZERO and error is NS then conc_isoflurance is PS
if error is NB then conc_isoflurance is PB
if error is PB then conc_isoflurance is NB
if error is PS then conc_isoflurance is NS
if Blood pressure is NS and error is ZERO then conc_isoflurance is NB
if Blood pressure is PS and error is ZERO then conc_isoflurance is PS.

## 5.3 Genetic Algorithm Specification

The general characteristics of the genetic algorithm that was used are the following:
**NIND = 40;** % Number of individuals in each subpopulation.
**MAXGEN = 300;** % Maximum number of generations allowed.
**GGAP = .6;** %"Generational gap", which is the percentage from the complete population of new individuals generated in each generation.
**PRECI = 120;** % Precision of binary representations.
**SelCh = select('rws', Chrom, FitnV, GGAP);** % Roulette wheel method for selecting the indivuals participating in the genetic operations.
**SelCh = recombin('xovmp',SelCh,0.7);** % Multi-point crossover as recombination method for the selected individuals.
**ObjV = FuncionObjDifuso120_555(Chrom, sdifuso)**; Objective function is given by the error between the performance of the ideal control system given by the experts and the fuzzy control system given by the genetic algorithm.

## 5.4 Representation of the Chromosome

In Table 1 we show the chromosome representation, which has 120 binary positions. These positions are divided in two parts, the first one indicates the number of rules of the fuzzy inference system, and the second one is divided again into fuzzy rules to indicate which membership functions are active or inactive for the corresponding rule.

**Table 1.** Binary Chromosome Representation.

| Bit assigned | Representation |
|---|---|
| 1 a 12 | Which rule is active or inactive |
| 13 a 21 | Membership functions active or inactive of rule 1 |
| 22 a 30 | Membership functions active or inactive of rule 2 |
| ... | Membership functions active or inactive of rule... |
| 112 a 120 | Membership functions active or inactive of rule 12 |

## 6 Simulation Results

We describe in this section the simulation results that were achieved using the hierarchical genetic algorithm for the optimization of the fuzzy control system, for the case of anesthesia control. The genetic algorithm is able to evolve the topology of the fuzzy system for the particular application. We used 300 generations of 40 individuals each to achieve the minimum error. We show in Figure 7 the final results of the genetic algorithm, where the error has been minimized. This is the case in which only nine fuzzy rules are needed for the fuzzy controller. The value of the minimum error achieved with this particular fuzzy logic controller was of 0.0064064, which is considered a small number in this application.

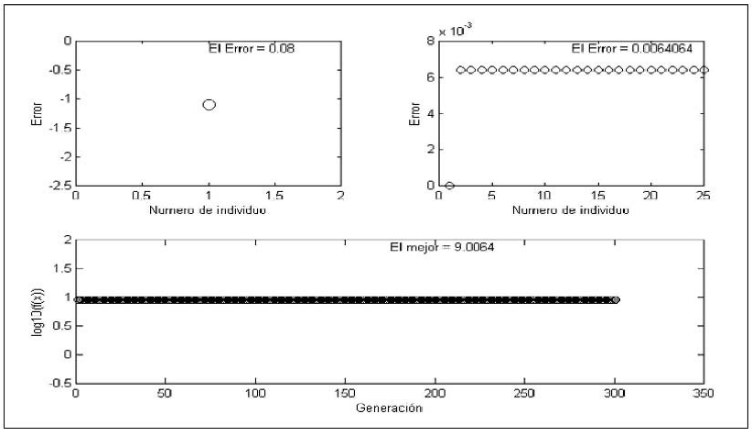

**Fig. 7.** Plot of the error after 300 generations of the HGA.

In Figure 8 we show the simulation results of the fuzzy logic controller produced by the genetic algorithm after evolution. We used a sinusoidal input signal with unit amplitude and a frequency of 2 radians/second, with a transfer function of [1/(0.5s +1)]. In this figure we can appreciate the comparison of the outputs of both the ideal controller (1) and the fuzzy controller optimized by the genetic algorithm (2). From this figure it is clear that both controllers are very similar and as a consequence we can conclude that the genetic algorithm was able to optimize the performance of the fuzzy logic controller. We can also appreciate this fact more clearly in Figure 9, where we have amplified the simulation results from Figure 8 for a better view.

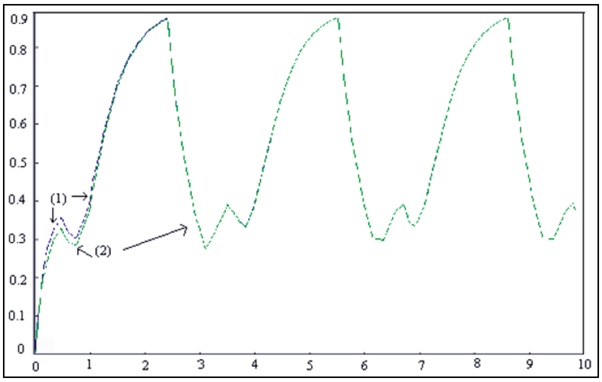

**Fig. 8.** Comparison between outputs of the ideal controller (1) and the fuzzy controller produced with the HGA (2).

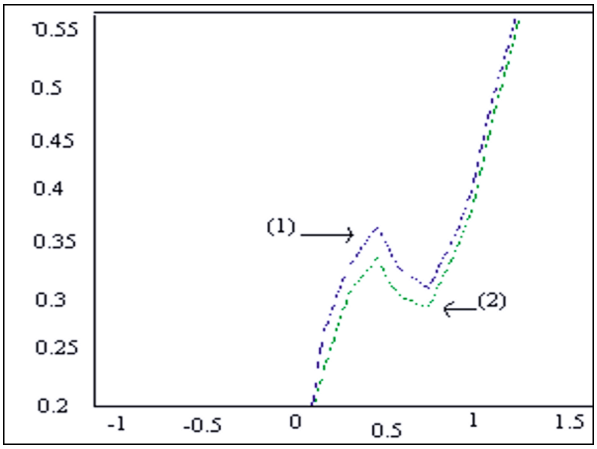

**Fig. 9.** Zoom in of figure 8 to view in more detail the difference between the controllers.

Finally, we show in Figure 10 the block diagram of the implementation of both controllers in Simulink of MATLAB. With this implementation we are able to simulate both controllers and compare their performances.

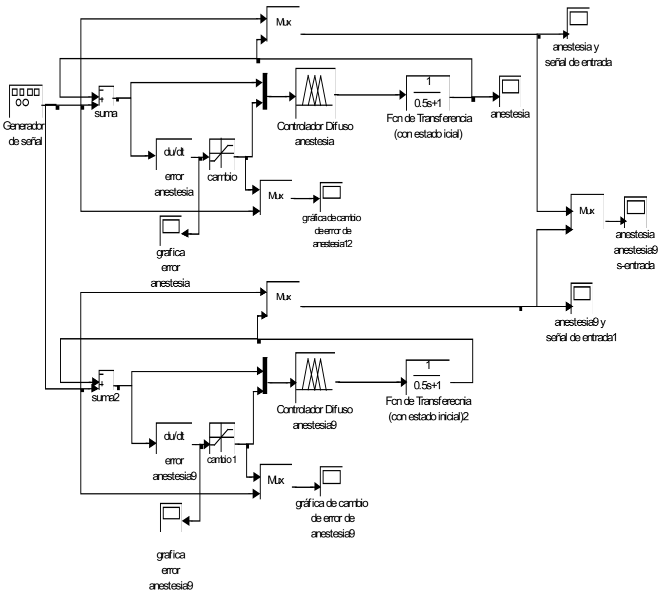

**Fig. 10.** Implementation in Simulink of MATLAB of both controllers for comparison of their performance.

## 7 Conclusions

We consider in this paper the case of automatic anesthesia control in human patients for testing the optimized fuzzy controller. We did have, as a reference, the best fuzzy controller that was developed for the automatic anesthesia control (Karr and Gentry 1993, Lozano 2003), and we consider the optimization of this controller using the HGA approach. After applying the genetic algorithm the number of fuzzy rules was reduced from 12 to 9 with a similar performance of the fuzzy controller. Of course, the parameters of the membership functions were also tuned by the genetic algorithm. We did compare the simulation results of the optimized fuzzy controllers obtained with the HGA against the best fuzzy controller that was obtained previously with expert knowledge, and control is achieved in a similar fashion.

## Acknowledgments

We would like to thank the Research Grant Committee of COSNET for the financial support given to this project (under grant 424.03-P). We would also like to thank CONACYT for the scholarships given to the students that work in this research project (Gabriel Huesca and Fevrier Valdez).

## References

O. Castillo and P. Melin (2001), “Soft Computing for Control of Non-Linear Dynamical Systems”, Springer-Verlag, Heidelberg, Germany.

O. Castillo and P. Melin (2003), “Soft Computing and Fractal Theory for Intelligent Manufacturing”, Springer-Verlag, Heidelberg, Germany.

J. Holland, (1975), "Adaptation in natural and artificial systems" (University of Michigan Press).

A. Homaifar and E. McCormick (1995), “Simultaneous design of membership functions and rule sets for fuzzy controllers using genetic algorithms”, *IEEE Trans. Fuzzy Systems*, vol. 3, pp. 129-139.

J.-S. R. Jang and C.-T. Sun (1995) “Neurofuzzy fuzzy modeling and control”, *Proc. IEEE*, vol. 83, pp. 378-406.

J.-S. R. Jang, C.-T. Sun, and E. Mizutani (1997), "Neuro-fuzzy and Soft Computing, A computational approach to learning and machine intelligence", , Prentice Hall, Upper Saddle River, NJ.

C.L. Karr and E.J. Gentry (1993), “Fuzzy control of pH using genetic algorithms”, *IEEE Trans. Fuzzy Systems*, vol. 1, pp. 46-53.

A. Lozano (2003), "Optimización de un Sistema de Control Difuso por medio de algoritmos genéticos jerarquicos", Thesis, Dept. of Computer Science, Tijuana Institute of Technology, Mexico.

K.F. Man, K.S. Tang, and S. Kwong (1999), "Genetic Algorithms: Concepts and Designs", Springer Verlag.

P. Melin and O. Castillo (2002), “Modelling, Simulation and Control of Non-Linear Dynamical Systems”, Taylor and Francis, London, Great Britain.

T.J. Procyk and E.M. Mamdani (1979), “A linguistic self-organizing process controller” *Automatica*, vol. 15, no. 1, pp 15-30.

K.-S. Tang, K.-F. Man, Z.-F. Liu and S. Kwong (1998), “Minimal fuzzy memberships and rules using hierarchical genetic algorithms”, *IEEE Trans. on Industrial Electronics*, vol. 45, no. 1.

J. Yen, and R. Langari (1999), "Fuzzy Logic: intelligence, control and information", Prentice Hall, Inc.

T. Yoshikawa, T. Furuhashi and Y. Uchikawa (1996), “Emergence of effective fuzzy rules for controlling mobile robots using NA coding method”, *Proc. ICEC’96*, Nagoya, Japan, pp. 581