NORTH-HOLLAND

Tuning Fuzzy Logic Controllers by Genetic Algorithms*

F. Herrera, M. Lozano, and J. L. Verdegay
Department of Computer Science and Artificial Intelligence
University of Granada, Spain

ABSTRACT

The performance of a fuzzy logic controller depends on its control rules and membership functions. Hence, it is very important to adjust these parameters to the process to be controlled. A method is presented for tuning fuzzy control rules by genetic algorithms to make the fuzzy logic control systems behave as closely as possible to the operator or expert behavior in a control process. The tuning method fits the membership functions of the fuzzy rules given by the experts with the inference system and the defuzzification strategy selected, obtaining high-performance membership functions by minimizing an error function defined using a set of evaluation input-output data. Experimental results show the method's good performance.

KEYWORDS: ***fuzzy logic control systems, tuning, genetic algorithms***

1. INTRODUCTION

Recently fuzzy control techniques have been applied to many industrial processes. Fuzzy logic controllers (FLCs) are rule-based systems which are useful in the context of complex ill-defined processes, especially those which can be controlled by a skilled human operator without knowledge of their underlying dynamics.

The essential part of the FLC system is a set of fuzzy control rules (FCRs) related by means of a fuzzy implication and the compositional rule of inference.

Address correspondence to Francisco Herrera, Dept. of Computer Science and A.I., ETS de Ingeniería Informática, University of Granada, 18071 Granada, Spain.

*This research has been supported under project PB92-0933

Received July 1993; accepted November 1994.

International Journal of Approximate Reasoning 1995; 12:299–315

655 Avenue of the Americas, New York, NY 10010
0888-613X/95/$9.50
SSDI 0888-613X(94)00033-Y

FCRs are usually formulated in linguistic terms, in the form of IF-THEN rules, and there are different modes for deriving them [1]. In all cases, the correct choice of the membership functions of the linguistic label set plays an essential role in the performance of an FLC, it being difficult to represent the experts' knowledge perfectly by linguistic control rules.

The fuzzy-control-rule base has many parameters, and its control depends on the tuning of the control system. Therefore, an FLC contains a number of sets of parameters that can be altered to modify the controller performance. They are [2]:

- the scaling factors for each variable,
- the fuzzy set representing the meaning of linguistic values,
- the IF-THE$_N$ rules.

Each of these sets of parameters have been used as controller parameters to be adapted in different adaptive FLCs.

In this paper we present an adaptive FLC that modifies the fuzzy set definitions (it alters the shapes of the fuzzy sets defining the meaning of linguistic values) to determine the membership functions that produce maximum FLC performance according to the inference system (fuzzy implication and compositional operator) and the defuzzification strategies used—that is, to tune the FCR so as to make the FLC behave as closely as possible to the operator or expert behavior. This method relies on having a set of training data against which the controller is tuned.

Recent work has centered on the use of mathematical optimization techniques to alter the set definition, so that the FLC matches a suitable set of reference data as closely as possible. This procedure is carried out off line and so tunes the controller before it is used. Among the proposed methods are the following: Nomura et al. reported a self-tuning method for fuzzy inference rules employing a descent method for Takagi-Sugeno fuzzy rules with constant outputs, and isosceles-triangular fuzzy numbers [3]. Glorennec presented an adaptive controller using fuzzy logic and connectionnist methods [4]. Guély and Siarry used the gradient descent method for optimizing Takagi-Sugeno rules with symmetric and asymmetric triangular membership functions and output functions (standard Sugeno rules), proposing the "centered Takagi-Sugeno rules" for avoiding a specific class of local minima [5]. Zheng proposed a computer-aided tuning technique for fuzzy control by gradient analysis, input variables with width on each side equal to the interval between the two adjacent peaks, and output variables using symmetrical, equal-width, triangular membership functions [6]. On the other hand, some approaches using genetic algorithms (GAs) for designing an adaptive FLC have been presented in the literature [7–10].

We propose a tuning method for obtaining high-performance fuzzy control rules by means of special GAs. The tuning method using GAs fits

the membership functions of the fuzzy rules dealing with the parameters of the membership functions, minimizing a squared-error function defined by means of an input-output data set for evaluation.

In order to do this, we set up the paper as follows. In the next section we describe the family of FLC systems that we study. Section 3 introduces the GA as a tool used for tuning knowledge-based rules. Section 4 presents a study of tuning fuzzy control rules, and Section 5 presents some examples. Finally we make some remarks thereon.

2. FUZZY LOGIC CONTROLLERS

Our FLC uses approximate reasoning to evaluate rules. We assume a family $S = (T', T, I, G, D)$ of FLCs, with T' and T fuzzy conjunction connectives (like t-norms), I an implication function, G a disjunctive connective (like a t-conorm), and D a defuzzification method. Then we can represent the fuzzy logic control process as

$$y^0 = S(x^0),$$

where x^0 represents the value of the state variables and y^0 the value of the control variables associated by means of the fuzzy controller.

More concretely:

1. The rule set is composed of a finite number of rules with the form

   IF x_1 is A_{i1} AND x_2 is A_{i2} AND ... AND x_n is A_{in} THEN y is B_i,

   where $x_1, \ldots, x_n$, and y are linguistic variables representing the process state variables and the control variables respectively; and $A_{i1}, \ldots, A_{in}$, B_i, $i = 1, \ldots, m$, are the linuguistic values of the linguistic variables $x_1, \ldots, x_n$, y in the universes of discourse $U_1, \ldots, U_n$, V.
2. Linguistic labels A_{ij} and B_i have the form of trapezoidal-shaped functions. Fuzzy sets (linguistic terms) are used for obtaining the fuzzy rules from the expert. The parametric representation of the trapezoidal membership functions is achieved by the 4-tuple (c_i, a_i, b_i, d_i), which characterizes the membership functions.
3. The fuzzy inference is made with T as a fuzzy conjunction and the generalized modus ponens constructed from T' and I. The inference process is the following:

   IF x_1 is A_{i1} AND x_2 is A_{i2} AND ... AND x_n is A_{in} THEN y is B_i
   x is A'
   ---
   y is B'

   $$B_i'(y) = \sup\{T'(A'(x), I(A_i(x), B_i(y)))\,|\,x \in R^n\},$$
   $$A_i(x) = T(A_{i1}(x_1), \ldots, A_{in}(x_n)).$$

Since the input is a point $x = x^0$,

$$A'(x) = \begin{cases} 1 & \text{if} \quad x^0 = x, \\ 0 & \text{if} \quad x^0 \neq 0; \end{cases}$$

then the result is translated into

$$B_i'(y) = T'\big(1, I\big(A_i(x^0), B_i(y)\big)\big) = I\big(A_i(x^0), B_i(y)\big).$$

4. The integration of all fuzzy rules is made by means of the disjunctive connective G:

$$B'(y) = G(\{B_i(y)\}).$$

5. The defuzzification method produces the control action that best represents the possibility distribution of an inferred fuzzy control action,

$$y^0 = D(B'(y)).$$

Several methods, for example, the center of gravity or center of area, mean of maximum, weight average of the center of gravity by the heights, and maximal-height methods (max criteria, max gravity center) (see [11, 12]), can be considered.

Most of the existing FLCs are based on the fuzzy reasoning method called the "min-max-gravity method" by Mamdani [13]. However, different fuzzy implications as well as compositional operators and defuzzification strategies often employed in an FLC have been described in the literature [14, 15, 11, 12]. We will use the max-min inference system (I = MIN, G = MAX), T = MIN, and all the aforementioned defuzzification strategies.

3. GENETIC ALGORITHMS

GAs are search algorithms that use operations found in natural genetics to guide the trek through a search space. GA are theoretically and empirically proven to provide robust search in complex spaces, offering a valid approach to problems requiring efficient and effective search (see [16]).

Any GA starts with a population of randomly generated solutions (chromosomes) and advances toward better solutions by applying genetic operators, modeled on genetic processes occurring in nature. In these algorithms we maintain a population of solutions for a given problem; this population undergoes evolution in a form of natural selection. In each generation, relatively good solutions reproduce to give offspring that replace the relatively bad solutions, which die. An evaluation or fitness

function plays the role of the environment in distinguishing between good and bad solutions.

Although there are many possible variants of the basic GA, the fundamental underlying mechanism operates on a population of individuals, and consists of three operations:

1. evaluation of individual fitness,
2. formation of a gene pool, and
3. recombination and mutation.

The initial population $P(0)$ is chosen randomly, and the individuals resulting from these three operations form the next generation's population. The process is iterated until the system ceases to improve. Generally, each individual in the population is represented by a fixed-length binary string which encodes values for variables.

During iteration t, the GA maintains a population $P(t)$ of solutions $x_1^t, \ldots, x_R^t$ (the population size R remains fixed). Each solution, x_i^t, is evaluated by a function $E(\cdot)$, with $E(x_i^t)$ being a measure of the fitness of the solution. The fitness value determines the relative ability of an individual to survive and produce offspring in the next generation. In the $(t + 1)$th iteration a new population is formed on the basis of the operators (2) and (3).

Figure 1 shows the structure of a simple GA.

The recombination is produced using the crossover operator, which combines the features of two parent structures to form two similar offspring; this is applied under a random position cross with a probability of performance (the crossover probability) P_c. The mutation operator arbitrarily alters one or more components of a selected structure so as to increase the structural variability of the population. Each position of each solution vector in the population undergoes a random change according to a probability defined by the mutation rate (the mutation probability) P_m.

```
Procedure genetic algorithm
begin (1)
  t = 0;
  inicialize P(t);
  evaluate P(t);
  While (Not terminationc-condition) do
  begin (2)
    t = t + 1
    select P(t) from P(t − 1);
    recombine P(t);
    evaluate P(t);
  end (2)
end (1)
```

Figure 1. GA structure.

Thus, it is generally accepted that any GA for solving a problem must take into account the following five points:

1. *a genetic representation of solutions to the problem,*
2. *a way to create an initial population of solutions,*
3. *an evaluation function that gives the fitness of each individual,*
4. *genetic operators that alter the genetic composition of children during reproduction, and*
5. *values for the parameters that the* GA *uses (population size, probabilities of applying genetic operators, etc.).*

The robustness and simple mechanism of GA make them a potentially useful tool to search for a good parameter configuration for the set of fuzzy control rules in the input/output spaces, which justify its use for tuning FLCs.

4. TUNING FUZZY CONTROL RULES

Training data (TRDs) are necessary for tuning fuzzy controllers. A group of TRDs is a pair of input-output data, in which the output data are desired output values, and the input data are relevant fuzzy input values. These tuning data represent the skilled-operator control behavior.

The difficulty in tuning a fuzzy controller can be attributed to the interference between fuzzy tunable parameters. A skilled operator's behavior can be described by many groups of tuning data, and one fuzzy control action comes from the synthesis of all rules' degrees of fulfillment [6], that is, the result of matching every input value in the antecedent with the corresponding membership functions.

Therefore, tuning any membership function usually affects more than one rule, and every rule may affect each fuzzy control action. Hence, a key problem is how to take tuning actions to provide the closest match for controller actions covering the entire range of tuning data.

We propose a tuning method for obtaining high-performance FCRs by means of special GAs, whose components are described as follows.

4.1. Representation

In the population of our special GAs a candidate solution C_r, $r = 1, \ldots, R$, represents an FCR, and a rule

$$\text{IF } x_1 \text{ is } A_{i1} \text{ AND } x_2 \text{ is } A_{i2} \text{ AND} \ldots \text{AND } x_n \text{ is } A_{in} \text{ THEN } y \text{ is } B_i$$

is represented by a piece of chromosome C_{ri}, $i = 1, \ldots, m$. Therefore, a base of m FCRs is represented by the chromosome C_r:

$$C_r = C_{r1}\ C_{r2}\ \ldots\ C_{rm}.$$

Since a label has the form of a trapezoidal-shaped function with a parametric representation by a 4-tuple, the A_{ij} will be represented by the 4-tuple $(c_{ij}, a_{ij}, b_{ij}, d_{ij})$, and B_i by (c'_i, a'_i, b'_i, d'_i), $i = 1, \ldots, m$, $j = 1, \ldots, n$. Then C_{ri} codes the vector values

$$C_{ri} = (c_{i1}, a_{i1}, b_{i1}, d_{i1}, \ldots, c_{in}, a_{in}, b_{in}, d_{in}, c'_i, a'_i, b'_i, d'_i).$$

In [17, p. 75] Michalewicz wrote:

The binary representation traditionally used in genetic algorithms has some drawbacks when applied to multidimensional, high-precision numerical problems. For example, for 100 variables with domains in the range $[-500, 500]$ where a precision of six digits after the decimal point is required, the length of the binary solution vector is 3000. This, in turn, generates a search space of about 10^{1000}. For such problems genetic algorithms perform poorly.

That is our case. For example with $n = 2$, $m = 7$, we have 84 real values; supposing a domain in the range $[-1, 1]$ where a precision of six digits after the decimal point is required, the length of the binary solution vector is 1.764. This generates a search space of about 10^{500}.

We propose to approach this problem with real coded genes together with special genetic operators for them. Then an FCR set would be a chromosome vector coded as a vector of floating-point numbers, the nearest to a natural representation of the problem. The precision of such an approach depends on the underlying machine, but is generally much better than that of the binary representation. We can always increase the precision of the binary representation by introducing more bits, but it increases the size of the search space exponentially, and this slows down the algorithm considerably (see [18, 17]).

Finally, we represent a population of R rules by C, and it is set up as follows:

$$C = \begin{pmatrix} C_1 & \cdots & C_R \end{pmatrix}.$$

4.2. Formation of an Initial Population or Gene Pool

The initial gene pool is created from the initial FCR set given by the expert. This initial FCR set is a chromosome, which is denoted as C_1. As we want to tune the FCRs, we define for every gene c_h of C_1, $h = 1, \ldots, H$, $H = (n + 1) \times 4$, an interval of performance for it, $[c_h^l, c_h^r]$, which will be the interval of adjustment of this variable, $c_h \in [c_h^l, c_h^r]$.

If $t \bmod 4 = 1$, then c_t is the left value of the support of a fuzzy number. The fuzzy number is defined by the four parameters $(c_t, c_{t+1}, c_{t+2}, c_{t+3})$, and the intervals of performance that we define are the following:

$$c_t \in [c_t^l, c_t^r] = \left[c_t - \frac{c_{t+1} - c_t}{2}, c_t + \frac{c_{t+1} - c_t}{2}\right],$$

$$c_{t+1} \in [c_{t+1}^l, c_{t+1}^r] = \left[c_{t+1} - \frac{c_{t+1} - c_t}{2}, c_{t+1} + \frac{c_{t+2} - c_{t+1}}{2}\right],$$

$$c_{t+2} \in [c_{t+2}^l, c_{t+2}^r] = \left[c_{t+2} - \frac{c_{t+2} - c_{t+1}}{2}, c_{t+2} + \frac{c_{t+3} - c_{t+2}}{2}\right],$$

$$c_{t+3} \in [c_{t+3}^l, c_{t+3}^r] = \left[c_{t+3} - \frac{c_{t+3} - c_{t+2}}{2}, c_{t+3} + \frac{c_{t+3} - c_{t+2}}{2}\right].$$

Figure 2 shows these.

The initialization process is very simple: we create a population of chromosomes, with C_1, and with the remaining chromosomes initialized randomly, each gene being in the respective interval of performance.

4.3. Evaluation of Individual Fitness

We consider a training input-output data set of size K,

$$\{(x_i, y_i) = (x_{i1}, \ldots, x_{in}, y_i), i = 1, \ldots, K\}.$$

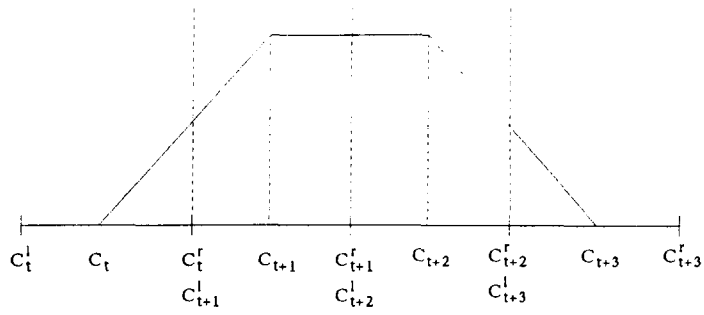

Figure 2. Intervals of performance.

The inference error of an FLC, S, with an FCR set BR and the input-output data set TRD, is calculated as the sum of the quadratic errors. It is specified by the function

$$E(S, \mathrm{BR}, \mathrm{TRD}) = \frac{1}{2} \sum_{i=1}^{K} [y_i - S(x_i)]^2.$$

The fitness function of the GA for a chromosome C_j coding an FCR set BR_j and the training data TRD is defined as follows:

$$F(C_j) = E(S, \mathrm{BR}_j, \mathrm{TRD}) = \frac{1}{2} \sum_{i=1}^{K} [y_i - S(x_i)]^2$$

in order to minimize the fitness function.

4.4. Genetic Operators

During the reproduction phase of the GA we use two classical genetic operators: mutation and crossover.

We propose to use the nonuniform mutation proposed by Michalewicz [17]. If $C_v^t = (c_1, \ldots, c_k, \ldots, c_H)$ is a chromosome and the element c_k was selected for this mutation (the domain of c_k is $[c_k^l, c_k^r]$), the result is a vector $C_v^{t+1} = (c_1, \ldots, c_k', \ldots, c_H)$, with $k \in 1, \ldots, H$, and

$$c_k' = \begin{cases} c_k + \Delta(t, c_{kr} - c_k) & \text{if a random digit is 0,} \\ c_k - \Delta(t, c_k - c_{kl}) & \text{if a random digit is 1,} \end{cases}$$

where the function $\Delta(t, y)$ returns a value in the range $[0, y]$ so that the probability of $\Delta(t, y)$ being close to 0 increases as t increases. This property causes this operator to make a uniform search in the initial space when t is small, and very locally at later stages.

In relation to the crossover operator, we use two different operators:

1. *Simple crossover.* It is defined in the usual way, with a crossover point. If $C_v^t = (c_1, \ldots, c_k, \ldots, c_H)$ and $C_w^t = (c_1', \ldots, c_k', \ldots, c_H')$ are to be crossed, and assuming that the crossover point is (randomly) selected before the kth gene, the two resulting offspring are

$$C_v^{t+1} = (c_1, \ldots, c_{k-1}, c_k', \ldots, c_H'),$$

$$C_w^{t+1} = (c_1', \ldots, c_{k-1}', c_k, \ldots, c_H).$$

2. *Max-min-arithmetical crossover.* If C_v^t and C_w^t are to be crossed, we generate

$$C_1^{t+1} = aC_w^t + (1-a)C_v^t, \qquad C_2^{t+1} = aC_v^t + (1-a)C_w^t,$$

$$C_3^{t+1} \text{ with } c_{3k}^{t+1} = \min\{c_k, c_k'\}, \qquad C_4^{t+1} \text{ with } c_{4k}^{t+1} = \max\{c_k, c_k'\}.$$

This operator can use a parameter a which is either a constant, or a variable whose value depends on the age of the population. The resulting offspring are the two best of the four aforementioned offspring.

4.5. Parameters

We carry out our experiments with the following parameters:

- Population size: 61.
- Probability of crossover: $P_c = 0.6$ (max-min-arithmetical crossover, $a = 0.35$).
- Probability of mutation:
  Probability of chromosome update: $P_m = 0.6$.
  Probability of gene mutation: $P_{m(\text{gen})} = 0.007140$.
- Selection procedure: Stochastic universal sampling; the number of offspring of any structure is bounded by the floor and ceiling of the expected number of offspring [19].

The following section presents some experimental results for such a genetic system for tuning fuzzy logic controllers.

5. EXPERIMENTAL RESULTS

An inverted pendulum is a very good example for control engineers to verify a modern control process. Figure 3 shows the system.

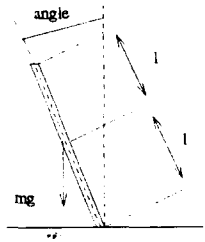

Figure 3. Inverted pendulum.

On the assumption that $|\theta| \ll 1$ (radian), the nonlinear differential equation which controls the behaviour of pendulum can be simplified to

$$m\frac{L}{3}\frac{d^2\theta}{dt^2} = \frac{L}{2}\left(-F + mg\sin\theta - k\frac{d\theta}{dt}\right),$$

where $k\, d\theta/dt$ is an approximation of the friction force.

The state variables are θ, the angle; ω, the angular speed; and the control variable f, the force. For every (θ_0, ω_0) we try to find the force that we must apply to the center of gravity of the pendulum for a constant time in order to take the pendulum to the vertical position.

We consider the linguistic rules proposed by Yamakawa [20], but instead of the speed of the cart pole considered by him, we consider the force as an output variable.

The linguistic term set is

{Negative Large (NL), Negative Medium (NM), Negative Small (NS),

Zero (ZR),

Positive Small (PS), Positive Medium (PM), Positive Large (PL)},

and the fuzzy linguistic rules are:

Rule 1. IF θ is PM AND ω is ZR THEN f is PM.
Rule 2. IF θ is PS AND ω is PS THEN f is PS.
Rule 3. IF θ is PS AND ω is NS THEN f is ZR.
Rule 4. IF θ is NM AND ω is ZR THEN f is NM.
Rule 5. IF θ is NS AND ω is NS THEN f is NS.
Rule 6. IF θ is NS AND ω is PS THEN f is ZR.
Rule 7. IF θ is ZR AND ω is ZR THEN f is ZR.

A pendulum weighing 5 kg and 5 m long has been considered in a real simulation, applying the force to the gravity center, for a constant time of 10 ms. Under these parameters the universes of discourse of the variables are the following:

$$\theta \in [-0.5240, 0.5240] \text{ rad}, \qquad \omega \in [-0.8580, 0.8580] \text{ rad/s},$$

$$f \in [-2980.0, 2980.0] \text{ N}.$$

Initially the membership functions corresponding to each element in the linguistic term set must be defined. We have considered the discretization of the universes presented in [21].

Since any optimum values always depend on specific models, we use the max-min inference system (I = MIN, G = MAX), T = MIN, and the follow-

Table 1. Error Reduction: Quadratic Error

	QE		
Strategy	Initial FCR	GA1-FCR	GA2-FCR
COA	669.406,2500	58.968,8984	64.044,7500
MOM	2.028.396,7500	193.374,7188	250.266,0312
WECOA	630.547,8750	54.619,2188	58.459,5000
MC	1.248.722,1250	796.287,8125	863.310,3750
MCOA	1.248.722,2500	501.209,1250	522.959,0625

ing defuzzification strategies: center of area (COA), mean of maximum (MOM), weight average of the center or area by the heights (WECOA), and the maximal-height methods: max-maximum criterion (MC) and max-center of area (MCOA) [11, 12].

Experimentally we have obtained a TRD with 68 input-output data (TRD-1) in the intervals

$$[-0.275, 0.275], \quad [-0.454, 0.454], \quad \text{and} \quad [1576.681, 1576.681]$$

for θ, ω, and f respectively.

We apply the two proposed GAs for obtaining the high-performance FCR: GA1 with simple crossover, GA2 with max-min-arithmetical crossover, for $T = 5000$ iterations.

Tables 1 and 2 show the behavior of the initial FCR, the FCR obtained by GA1 (GA1-FCR), and the FCR obtained by GA2 (GA2-FCR), presenting the quadratic error (QE) and the linear error (LE). In all cases the behavior of the final set of fuzzy linguistic rules has been improved.

REMARKS GA1 and GA2 have been run for every defuzzification method, obtaining high-performance FCR sets associated to every de-

Table 2. Error Reduction: Linear Error

	LE		
Strategy	Initial FCR	GA1-FCR	GA2-FCR
COA	4.149,4062	1.005,0128	1.174,9415
MOM	10.407,3691	3.270,4431	3.392,4426
WECOA	3.100,4968	0.730,9796	0.848,7941
MC	7.267,7412	6.131,2129	6.424,4795
MCOA	7.267,7417	5.439,5615	5.571,3335

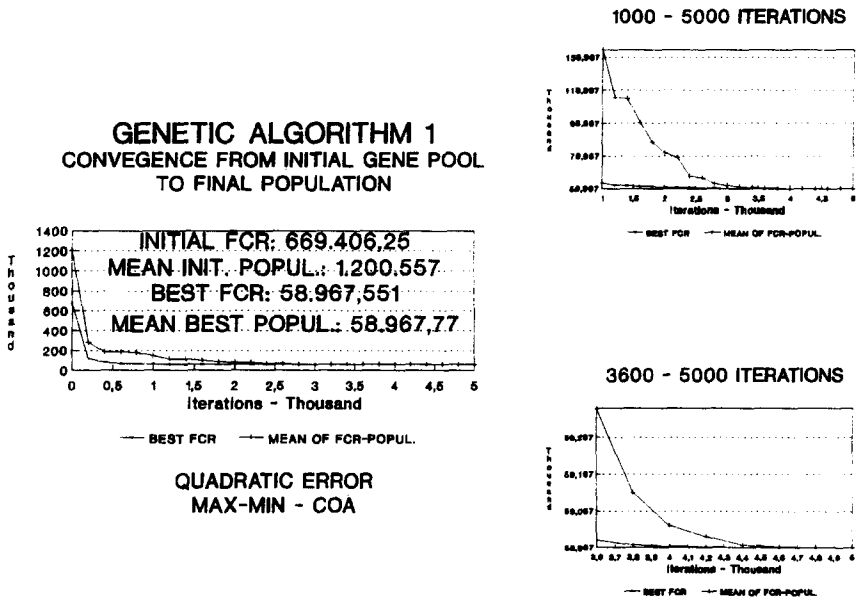

Figure 4. Convergence of genetic algorithm 1.

fuzzification method. In the second and third columns of the tables we evaluate the ten FCR sets obtained using GA1 and GA2.

Figures 4 and 5 show the behavior of both GAs in the case of COA defuzzification methods.

Figures 6–9 show the behavior of the inverted pendulum with the max-min inference system and the defuzzification methods COA and WECOA, and with the three fuzzy control rule sets. The force is applied every 600 ms. The graphics represent the position of the pendulum (θ in milliradians) and its stabilizations.

We observe that the best stabilization of the FLC in time occurs with the tuning FCR sets obtained by the GA that we have proposed.

Next, we use the MOM defuzzification method. As we can observe in the tables, the error of this method is greater than the error associated with COA and WECOA. Here we can notice that it is necessary to apply the force in smaller intervals of time, due to the bad behavior of the MOM method. In the case of applying force every 600 ms, the initial FCR and the GA1-FCR lose control of the inverted pendulum, whilst the GA2-FCR set keeps control. The initial FCR set has the worst behavior. Figures 8 and 9 show this.

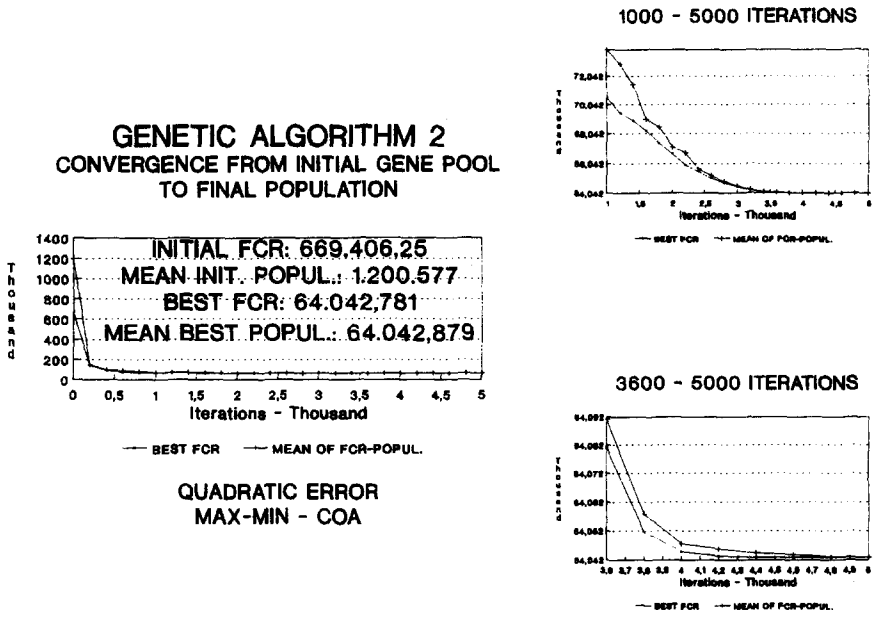

Figure 5. Convergence of genetic algorithm 2.

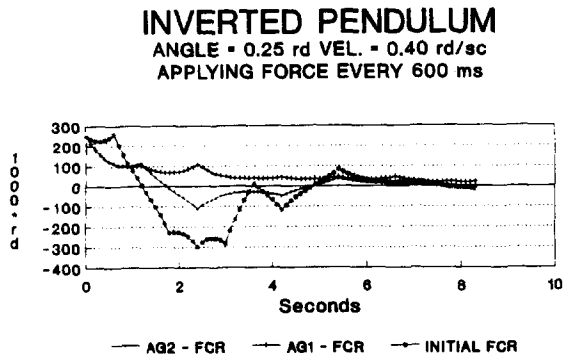

Figure 6. Inverted pendulum: max-min, COA.

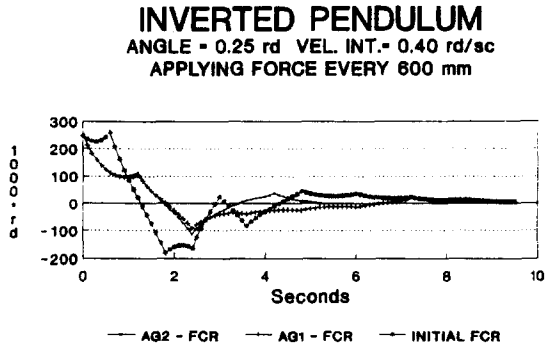

Figure 7. Inverted pendulum: max-min, WECOA.

6. CONCLUSIONS

In this paper we have presented a tuning method for fuzzy control rules by means of Gas. The usefulness of this method is verified by some numerical examples. As shown by the results, the FCRs obtained greatly improve the behavior of the FLC systems: clearly the quadratic and the linear error have been decreased. We conclude by pointing out the effectiveness of the proposed tuning method, and that high-performance FCRs are obtained.

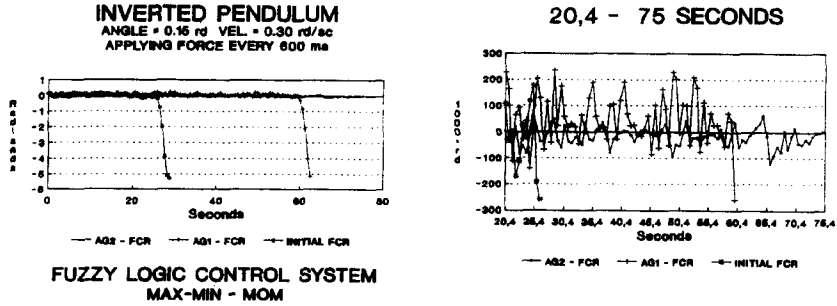

Figure 8. Inverted pendulum: max-min, MOM, 600 ms.

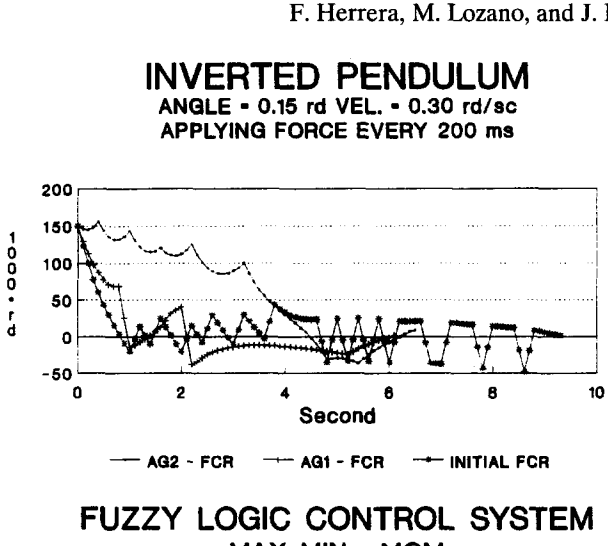

Figure 9. Inverted pendulum: max-min, MOM, 200 ms.

ACKNOWLEDGMENTS

The authors thank the referees for valuable comments which have improved the presentation of the paper.

References

1. Lee, C. C., Fuzzy logic in control systems: Fuzzy logic controller, Parts 1, 2, *IEEE Trans. Systems Man Cybernet.* 20, 404–418, 419–433, 1990.
2. Driankov, D., Hellendoorn, H., and Reinfrank, M., *An Introduction to Fuzzy Control*, Springer-Verlag, 1993.
3. Nomura, H., Hayashi, I., and Wakami, N., A self-tuning method of fuzzy control by descent method, *Proceedings of Fourth IFSA Congress*, Brussels, Vol. Engineering, 155–158, 1991.
4. Glorennec, Y. P., Adaptive fuzzy control, *Proceedings of Fourth IFSA Congress*, Brussels, Vol. Engineering, 33–36, 1991.
5. Guély, F., and Siarry, P., Gradient descent method for optimizing various fuzzy rule bases. *Proceedings of the Second IEEE International Conference on Fuzzy Systems*, San Francisco, 1241–1246, 1993.
6. Zheng, L., A practical computer-aided tuning technique for fuzzy control, *Proceedings of the Second IEEE International Conference on Fuzzy Systems*, San Francisco, 702–707, 1993.

7. Karr, C., Genetic algorithms for fuzzy controllers, *AI Expert*, Feb. 1991, 26–33.

8. Kropp, K., and Baitinger, U. G., Optimization of fuzzy logic controller inference rules using a genetic algorithm, *Proceedings of the First European Congress on Fuzzy and Intelligent Technologies*, Aachen, 1090–1096, 1993.

9. Hessburg, T., Lee, M., Takagi, H., and Tomizuka, M., Automatic design of fuzzy systems using genetic algorithms and its application to lateral vehicle guidance, *SPIE's International Symposium on Optical Tools for Manufacturing and Advanced Automation*, Vol. 2061, Boston, 1993.

10. Varsek, Urbancic, T., and Filipic, B., Genetic algorithms in controller design and tuning, *IEEE Trans. Systems Man Cybernet.* 23, 1330–1339, 1993.

11. Mizumoto, M., Improvement methods of fuzzy controls, *Proceedings of Third IFSA Congress*, Seattle, 60–62, 1989.

12. Zhao, R., and Govind, R., Defuzzification of fuzzy intervals, *Fuzzy Sets and Systems* 43, 45–55, 1991.

13. Mamdani, E. H., Applications of fuzzy algorithm for control of a simple dynamic plant, *Proc. IEEE* 121, 1974, 1585–1588.

14. Mizumoto, M., and Zimmermann, H. J., Comparison of fuzzy reasoning methods, *Fuzzy Sets and Systems* 8, 253–283, 1982.

15. Mizumoto, M., Fuzzy controls under various approximate reasoning methods, *Preprints of Second IFSA Congress*, Tokyo, 143–146, 1987.

16. Goldberg, D. E., *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, 1989.

17. Michalewicz, Z., *Genetic Algorithms + Data Structures = Evolution Programs*, Springer-Verlag, 1992.

18. Janikow, C. Z., and Michalewicz, Z., An experimental comparison of binary and floating point representations in genetic algorithms, *Fourth International Conference on Genetic Algorithms*, San Diego, 31–36, 1991.

19. Baker, J. E., Reducing bias and inefficiency in the selection algorithm, in *Proceedings of the Second International Conference on Genetic Algorithms* (J. J. Grefenstette, Ed.), Lawrence Erlbaum, Hillsdale, N.J., 14–21, 1987.

20. Yamakawa, T., Stabilization of an inverted pendulum by a high-speed fuzzy logic controller hardware system, *Fuzzy Sets and Systems* 32, 161–180, 1989.

21. Liaw, C-M., and Wang, J-B., Design and implementation of a fuzzy controller for a high performance induction motor drive, *IEEE Trans. Systems Man Cybernet.* 21, 921–929, 1991.

22. Takagi, T., and Sugeno, M., Fuzzy identification of systems and its applications to modeling and control, *IEEE Trans. Systems Man Cybernet.* SMC-15, 116–132, 1985.