ELSEVIER

Fuzzy Sets and Systems 100 (1998) 143–158

FUZZY sets and systems

A learning process for fuzzy control rules using genetic algorithms[1]

F. Herrera*, M. Lozano, J.L. Verdegay

Department of Computer Science and Artificial Intelligence, E.T.S. de Ingeniería Informática, University of Granada, 18071 Granada, Spain

Received March 1995; revised January 1997

Abstract

The purpose of this paper is to present a genetic learning process for learning fuzzy control rules from examples. It is developed in three stages: the first one is a fuzzy rule genetic generating process based on a rule learning iterative approach, the second one combines two kinds of rules, experts rules if there are and the previously generated fuzzy control rules, removing the redundant fuzzy rules, and the third one is a tuning process for adjusting the membership functions of the fuzzy rules. The three components of the learning process are developed formulating suitable genetic algorithms. © 1998 Elsevier Science B.V. All rights reserved

Keywords: Fuzzy logic control systems; Learning; Genetic algorithms

1. Introduction

Fuzzy-rule-based systems have been shown to be an important tool for modelling complex systems, in which due to the complexity or the imprecision, classical tools are unsuccessful. Fuzzy logic controllers (FLCs) are now considered as one of the most important applications of the fuzzy-rule-based systems. The experience of skilled operators and the knowledge of control engineers are expressed qualitatively by a set of fuzzy control rules.

The construction of fuzzy rules has been mainly based on the operator's control experience or actions. However, converting the experts' know-how into if–then rules is difficult and often results are incomplete, unnecessary and include conflicting knowledge, since operators and control engineers are not capable of specific details or cannot express all their knowledge including intuition and inspiration. Then, an alternative appears which consists of applying automatic techniques to obtain the fuzzy control rules. These techniques are focused on the use of sampled input–output data to help in the development of these rules.

Different approaches have been proposed to facilitate and automate the design of fuzzy control rules. In the last few years many different approaches have been presented taking the genetic algorithms (GAs) as a base of the learning process.

GAs have demonstrated to be a powerful tool for automating the definition of the fuzzy control rule knowledge base (KB), since adaptive control, learning, and self-organization may be considered in a lot of cases as optimization or search processes. Their advantages have extended the use of GAs in the

* Corresponding author. E-mail: herrera@decsai.ugr.es.

[1] This research has been supported by DGICYT PB92-0933.

0165-0114/98/$19.00 © 1998 Elsevier Science B.V. All rights reserved
PII: S0165-0114(97)00043-2

development of a wide range of approaches for designing FLCs over the last few years.

In particular, the application to the design, learning and tuning of KBs has produced quite promising results. These approaches can receive the general name of *genetic fuzzy systems* (GFSs) [6], a short description of them is included in Section 2.

The purpose of this paper is to present a fuzzy rule learning process based on the use of GAs under the following hypotheses:

- It may be linguistic information from the human controller's experience. But, linguistic rules alone are usually not enough for designing a successful control system, or might not be available.
- There is numerical information, from sampled input–output (state-control) pairs that are recorded experimentally.
- The combination of these two kinds of information may be sufficient for the successful design of a fuzzy control rule base.
- We include the possibility of not having any linguistic information, and having a complete numerical information.

According to the aforementioned hypotheses we wish to develop a learning process for fuzzy systems based on GAs, a GFS, with the following aims:

- to develop a process for generating fuzzy-control rules using numerical data pairs; and
- to develop a general approach that combines both kinds of information, expert rules and fuzzy-control rules obtained by the generating process, in a common framework, using both simultaneously and cooperatively to solve the control design problem.

In order to achieve these aims, we propose a methodology based on the design of the following three stages:

(a) A *genetic generating process* for obtaining desirable fuzzy rules capable of including the complete knowledge from the set of examples, based on an iterative rule learning approach.
(b) A *genetic process for combining rules and simplifying them*, thereby avoiding the possible overlearning and removing the redundant fuzzy rules.
(c) A *genetic tuning process* for adjusting the membership functions of the fuzzy rules.

To do so, this paper is organized as follows. Section 2 introduces, in short, the GAs and the GFSs; Section 3 presents an overall description of the GFS together with the fuzzy rule structure and some requirements on the KB; Section 4 shows the genetic generating process; Section 5 presents the combining-simplifying process; and Section 6 deals with the genetic tuning process. Then, and for sake of illustrating the learning process, Section 7 is devoted to develop some examples. Finally, some concluding remarks are made.

2. Preliminaries: genetic algorithms and genetic fuzzy systems

In the following, we present a short description of the GAs and GFSs.

2.1. Genetic algorithms

GAs are search algorithms that use operations found in natural genetics to guide the trek through a search space. GAs use a direct analogy of natural behaviour. They work with a population of chromosomes, each one representing a possible solution to a given problem. Each chromosome has assigned a fitness score according to how good a solution to the problem it is. GAs are theoretically and empirically proven to provide robust search in complex spaces, giving a valid approach to problems requiring efficient and effective searching [11].

Any GA starts with a population of randomly generated solutions, chromosomes, and advances toward better solutions by applying genetic operators, modeled on the 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 to distinguish between good and bad solutions. The process of going from the current population to the next population constitutes one generation in the execution of a GA.

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

(1) evaluation of individual fitness,

(2) formation of a gene pool (intermediate population) and
(3) recombination and mutation.

The next procedure shows the structure of a simple GA.

Procedure Genetic Algorithm
begin (1)
  $t = 0$;
  initialize $P(t)$;
  evaluate $P(t)$;
  While (**Not** *termination-condition*) **do**
  begin (2)
    $t = t + 1$;
    select $P(t)$ *from* $P(t-1)$;
    recombine $P(t)$;
    evaluate $P(t)$;
  end (2)
end (1)

A fitness function must be devised for each problem to be solved. Given a particular chromosome, a solution, the fitness function returns a single numerical fitness, which is supposed to be proportional to the utility or adaptation of the individual which that chromosome represents.

There are a number of ways of making this selection. We might view the population as mapping onto a roulette wheel, where each chromosome is represented by a space that proportionally corresponds to its fitness. By repeatedly spinning the roulette wheel, chromosomes are chosen using "stochastic sampling with replacement" to fill the intermediate population. The selection procedure proposed in [1], and called *stochastic universal sampling* is one of the most efficient, where the number of offspring of any structure is bound by the floor and ceiling of the expected number of offspring.

After selection has been carried out the construction of the intermediate population is complete, then the genetic operators, crossover and mutation, can occur.

A crossover operator combines the features of two parent structures to form two similar offspring. It is applied with a probability of performance, the crossover probability, P_c. A 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 a mutation rate, the mutation probability, P_m.

It is generally accepted that a GA to solve a problem must take into account the following five components:

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

The basic principles of GAs were first laid down rigorously by Holland [17], and are well described in many books such as [11, 23].

2.2. Genetic fuzzy systems

The GAs' properties make them suitable to use in order to design and optimize fuzzy systems. The automatic definition of the KB may be considered in many cases as optimization or search processes. The application to the learning and/or tuning of KB has provided fairly promising results.

As mentioned in the introduction, GAs are applied to modify/learning the data base (DB) and/or the rule base (RB), and it is possible to distinguish three different groups of GFSs depending on the KB components (DB and RB) included in the genetic learning process.

Genetic definition of the DB. The tuning of the fuzzy rule membership functions is an important task in the design of fuzzy systems. The tuning method using GAs fits the membership functions of the fuzzy rules dealing with their parameters according to a fitness function. Different approaches are presented in [3, 14, 19, 27].

Genetic derivation of the RB. All the methods belonging to this family suppose the existence of a collection of fuzzy set membership functions giving meaning to the labels, a DB, and learning a rule base. Some approaches are presented in [12, 13, 20, 28, 24].

Genetic learning of the KB. There are many approaches for the genetic learning processes of a complete KB, fuzzy rules and membership functions. We find approaches presenting variable

chromosome length, others coding a fixed number of rules and their membership functions, etc. Some approaches are presented in [4, 5, 21, 25, 29].

For a more detailed description see [6], for an extensive bibliography see [7] (Section 3.13), and some approaches may be found in [16].

The genetic learning processes belonging to the latter two classes can do the learning simultaneously or in various stages.

Classically, two alternative approaches in which GAs have been applied to learning processes have been mainly used, the Michigan [18] and the Pittsburgh [26] approaches. In the first one, the chromosomes correspond to classifier rules which are evolved as a whole, whereas in the Pittsburgh approach, each chromosome encodes a complete set of classifiers.

A third way is presented as an alternative to these models, an iterative rule learning approach, where each chromosome represents only one rule. In the latter model, as in the Michigan one, each chromosome in the population represents a single rule, but contrary to the Michigan one, only the best individual is considered as the solution, discarding the remaining chromosomes in the population. This model, used in [30] the first time, attempts to reduce the search space for the possible solutions.

In the following we present the GFS proposal in various stages based on the iterative rule learning approach.

3. On the genetic fuzzy system proposal

We propose a GFS methodology based on three stages. The first stage is a genetic generation process for obtaining desirable fuzzy rules capable of including the complete knowledge from the set of examples. The second one is a genetic simplifying process for combining rules and eliminating redundant rules, selecting the most cooperative set of rules. The third one is a genetic tuning process for adjusting the membership functions of the fuzzy rules. In the following we describe them in short.

(a) *Genetic generating process.* It is based on an iterative rule learning approach and consists of a fuzzy rule *generating method* together with an iterative *covering method* of the set of examples.
   - The fuzzy rule *generating method* is developed by means of a real coding GA (RCGA) that codes a single fuzzy rule in each chromosome. The GA finds the best rule in every run over the set of examples according to the features included in its fitness function.
   - The *covering method* is developed as an iterative process. It allows a set of fuzzy rules to be obtained covering the set of examples. In each iteration, it runs the generating method choosing the best chromosome (rule), considers the relative covering value this rule provokes over the example set and removes the examples with a covering value greater than a value ε provided by the controller designer.

   In this iterative model, the GA provides a partial solution to the problem of learning. This learning way is to allow "niches" and "species" formation. Species formation seems particularly appealing for concept learning, considering the process as the learning of multimodal concepts. This approach attempts to reduce the search space for the possible solutions.

(b) *Genetic simplification process.* Since two similar rules or one rule similar to another given by an expert process may be obtained in the generation, it is necessary to combine and simplify the complete RB obtained from the previous process for deriving the final RB. It is based on a binary-coded GA and a measure of the FLC performance in the control of the system being identified. It will save the overlearning that the previous component may cause selecting the most cooperative set of fuzzy rules.

(c) *Genetic tuning process.* The tuning method using GAs fits the membership functions of the fuzzy rules dealing with the parameters of the membership functions, minimizing a square error function defined by means of an input–output data set for evaluation. It is based on an RCGA and it will give the final KB as output by tuning the membership functions for each fuzzy control rule.

Afterwards we introduce the fuzzy rule structure used in our GFS, and some previous requirements on the KB to be taken into consideration for developing the process. They are to verify the completeness property and to have a high covering examples value.

3.1. Fuzzy rules structure

Let us suppose we know an incomplete set of rules given by the experts, R^e, and a set of training examples of the system consisting of the values that the variables take during an experiment in which the system is controlled by a human.

Expert rules. For extracting the knowledge from the experts, we suppose that the domains of the variables can be determined either by consulting the experts or by exploring the range of variables contained in the database.

We assume that each universe, U, contains a number of referential sets having their linguistic meaning, which form a finite set of fuzzy sets on U. For instance, if X is a variable on U for temperature, then one may define A_1 as "*low temperature*", A_i $(1<i<r)$ as "*medium temperature*" and A_r as "*high temperature*", etc. These referential fuzzy sets are characterized by their membership function $A_i(u)$: $U \to [0,1]$, $i=1,\ldots,r$. To ensure the performance of the fuzzy model and provide a uniform basis for further study it is essential that all the referential sets should be normal convex, and should satisfy the following completeness condition:

$$\forall u \in U \ \exists j,\ 1 \leqslant j \leqslant r, \text{ such that } A_j(u) \geqslant \delta$$

and δ is a fixed threshold, this being the *completeness degree* of the universes.

Learning rules. Let us suppose we know a set of p training examples $E_p = \{e_1,\ldots,e_p\}$ of the system consisting of the values that the variables take during an experiment in which the system is controlled by an expert: "in the time $t=k$, the value of the variable vectors X and Y are ex^k and ey^k, respectively"

We shall focus on Mamdani's model for MISO systems, where the knowledge base of a fuzzy controller consists of a collection of fuzzy rules (with the logical connective ALSO between the rules) describing the control actions in the form

$$R_i: \quad \text{IF } x_1 \text{ is } A_{i1} \text{ and } \ldots \text{ and } x_n \text{ is } A_{in} \text{ THEN } y \text{ is } B,$$

where $x_1,\ldots,x_n$ and y are the process state variables and the control variable, respectively; and $A_{i1},\ldots,A_{in}$, B are fuzzy sets in the universes of discourse $U_1,\ldots,U_n$, V.

These fuzzy sets are characterized by their membership functions

$$A_{ij}(B)\text{: } U_j(V) \to [0,1], \quad j=1,\ldots,n.$$

In our study we consider every fuzzy set associated with a normalized triangular membership or a normalized trapezoidal membership function. A computational way to characterize them is either by using a parametric representation achieved by means of the 3-tuple (a_{ij},b_{ij},c_{ij}), (a_i,b_i,c_i), $j=1,\ldots,n$ or by means of the 4-tuple $(a_{ij},b_{ij},c_{ij},d_{ij})$, (a_i,b_i,c_i,d_i), $j=1,\ldots,n$.

The classical Mamdani model is a linguistic model based on collections of *IF–THEN* rules with fuzzy quantities associated with linguistic labels, and the fuzzy model is essentially a qualitative expression of the system. A KB in which the fuzzy sets giving meaning (semantic) to the linguistic labels are uniformly defined for all rules included in the RB constitutes a *descriptive* approach since the linguistic labels represent a real-world semantic.

It can be considered a KB for which fuzzy rules either present different meaning for the same linguistic terms or the fuzzy quantities have not any associated linguistic label. In this case, the KB and the FLC using it, present a different philosophy, the approach is *approximative*. In the second approach we say that the rules present *free semantic*.

We will centre on the second approach and consider rules with a free semantic for the generating process, without any linguistic syntax associated to the rules. Each rule defines a typical feature of the system behaviour according to some requirements and is independent from a fuzzy partition, that is, no restrictions are imposed on the membership function locations and shapes.

3.2. Completeness property and covering value

If we wish to generate a set of rules describing the behaviour of a system, then it is necessary to establish a condition over the set of rules, R. This is the requirement of covering all possible situation–action pairs, $e_k \in E_p$, the *completeness property*. This may be formalized for a constant $\tau \in [0,1]$, it requires the non-zero union of fuzzy sets $A_i(\cdot)$, $B_i(\cdot)$, $i=1,\ldots,T$, $T=|R|$, and is formulated

by the following expressions:

$$C_R(e_k) = \bigcup_{i=1\ldots T} R_i(e_k) \geqslant \tau, \quad k = 1, \ldots, p,$$

$$R_i(e_k) = *(A_i(ex^k), B_i(ey^k)),$$

$$A_i(ex^k) = *(A_{i1}(ex_1^k), \ldots, A_{in}(ex_n^k)),$$

where $*$ is a t-norm, and $R_i(e_k)$ is a compatibility degree between the rule R_i and the example e_k. This property is required in the simplifying process. It eliminates redundant and unnecessaries rules under the condition of maintaining a minimal completeness degree on the training set.

Given a set of rules R, we define the *covering value* of an example e_k with a base of rules R as

$$CV_R(e_k) = \sum_{i=1}^{T} R_i(e_k),$$

and we require the following condition in the genetic generating process:

$$CV_R(e_k) \geqslant \varepsilon, \quad k = 1, \ldots, p.$$

The aforementioned covering method is based on this idea. We remove an example when it has a covering value higher than or equal to ε.

4. Genetic generating process

We focus this section on the description of the fuzzy rule *genetic generating process* consisting of a *generating method* for desirable fuzzy rules together with an iterative *covering method* of the set of examples.

4.1. Generating method

The generating method of fuzzy rules is developed by means of an RCGA, where a chromosome represents a fuzzy rule. The RCGA finds the best rule in every run over the set of examples, according to different features including in the fitness function.

In order to describe the RCGA we present the fundamental underlying mechanisms of a GA: representation, formation of an initial gene pool, fitness function, genetic operators and running parameters.

4.1.1. Representation

In the RCGA population, a candidate solution C_r, $r = 1, \ldots, M$, represents a fuzzy rule

IF x_1 is A_{r1} ... and x_n is A_{rn} THEN y_1 is B

where the real values $a_{rj}, b_{rj}, c_{rj}, d_{rj}, a_r, b_r, c_r, d_r$ characterize the membership functions of A_{rj}, $j = 1, \ldots, n$ and B_r, respectively.

Thus, C_r codes the vector values

$$(a_{r1}, b_{r1}, c_{r1}, d_{r1}, \ldots, a_{rn}, b_{rn}, c_{rn}, d_{rn}, a_r, b_r, c_r, d_r).$$

This happens when we use trapezoidal membership functions. Three parameters per variable would be used with triangular membership functions.

We propose approaching this problem with real-coded genes [15] together with special genetic operators developed for them. Then a rule would be a chromosome vector coded as a vector of floating point numbers.

Finally, we represent a population of M chromosomes (rules) by C, and it is set up as follows:

$$C = (C_1, \ldots, C_M).$$

4.1.2. Formation of an initial gene pool

We denote the domain of every input variable X_j as a close real interval $U_j = [a_j, b_j]$. Similarly, the domain of the output variable Y is the real interval $V = [c, d]$.

The initial gene pool is created partially from $E_t \subseteq E_p$ (t chromosomes) and the remaining ($M - t$ chromosomes) initiated randomly, as follows:

- Let $t = \min\{|E_p|, M/2\}$; then we select t examples from E_p at random, and for each one we determine the chromosome (rule) belonging to the initial gene pool as follows. Suppose the example $e^k \in E_t$ and the component $ex_j^k \in [a_j, b_j]$, $\Delta ex_j^k = \min\{ex_j^k - a_j, b_j - ex_j^k\}$, let $\delta(ex_j^k)$ be a random value in the range $[0, \Delta ex_j^k]$, then we form the membership function using the 4-tuple

  $$(ex_j^k - \delta(ex_j^k), ex_j^k, ex_j^k, ex_j^k + \delta(ex_j^k)).$$

  The procedure is the same for the remaining components of e_k.
- The remaining $M - t$ chromosomes of the initial population are chosen at random, each gene in its

respective interval,

$C_r = (c_{r1}, \ldots, c_{rl})$,

$l = 4(n+1)$, with requirements $c_{4s+1} \leqslant c_{4s+2} \leqslant c_{4s+3} \leqslant c_{4s+4}$, $s = 0, \ldots, n$ for trapezoidal membership functions, and in a similar way for triangular membership functions.

4.1.3. Evaluation of individual fitness

We define the fitness function according to five features with the objective of selecting fuzzy rules covering a lot of examples (two first criteria), with a few negative examples, small or fixed membership function width and finally with high symmetrical membership functions. Their formulations are described in the following:

(a) *High-frequency value*: The frequency of a fuzzy rule, R_i, through the set of examples, E_p, is defined as [9]:

$$\Psi_{E_p}(R_i) = \frac{\sum_{k=1}^{p} R_i(e_k)}{p}$$

with $R_i(e_k)$ being the compatibility degree.

(b) *High average covering degree over positive examples*: The set of positive examples to R_i with compatibility degree greater than or equal to ω is defined as

$$E^+(R_i) = \{e_k \in E_p / R_i(e_k) \geqslant \omega\}$$

with $n_{R_i}^+ = |E^+(R_i)|$ being the number of positive examples, and

$$G_{R_i} = \sum_{e_k \in E^+(R_i)} R_i(e_k)/n_{R_i}^+$$

the *average covering degree* on $E^+(R_i)$.

(c) *Small negative examples set*: The set of the negative examples to R_i is defined as

$$E^-(R_i) = \{e_k \in E_p / R_i(e_k) = 0 \text{ and } A_i(ex^k) > 0\}$$

with $n_{R_i}^- = |E^-(R_i)|$ being the number of negative examples.

An example is considered negative for a rule when it matches better with some other rule that has the same antecedent but a different consequent.

The *penalty function on the negative examples set* will be

$$g_n(R_i^-) = \begin{cases} 1 & \text{if } n_{R_i}^- \leqslant 5, \\ \dfrac{1}{n_{R_i}^- - 5 + \exp(1)} & \text{otherwise,} \end{cases}$$

where we permit up to 5 negative examples per rule without any penalty.

In every case, we consider the negative examples over the complete set of training data.

(d) *Small membership function width*: The rule variable width (RW) and rule modal variable width (RMW) of a fuzzy rule R_i are defined as

$$RW_i = \frac{\sum_{j=1}^{n+1} WVR_{ij}/DW_j}{n+1}$$

and

$$RMW_i = \frac{\sum_{j=1}^{n+1} WVMR_{ij}/WVR_{ij}}{n+1}$$

where

$$WVR_{ij} = d_{ij} - a_{ij}, \qquad WRMR_{ij} = c_{ij} - b_{ij}$$

being $(a_{ij}, b_{ij}, c_{ij}, d_{ij})$, $j = 1, \ldots, n+1$ (n input variables plus the output variable), the 4-tuple associated to every membership function, and DW_j is the domain interval width per variable.

We define the *membership width rate*, *MWR*, of the rule R_i as a function of the two parameters above, defined by the following expression:

$$MWR(R_i) = g_1(RW_i)\, g_2(RMW_i)$$

where the functions g_i represent the required relationships with respect to the width.

- If we wish a small width then we may consider the function

  $g_i(x) = e^{1-ax}$.

- If we prefer a constant relationship between the domain width and the variable width or between the two variable intervals width then we can consider the function

  $g_i(x) = e^{-|1-ax|}$.

  with a being the constant relationship.

(e) *High symmetrical membership functions*: The rate of symmetry is defined in order to achieve symmetry for the fuzzy numbers, and to prevent the bad covering of extreme points. It is defined as follows:

$$RS(R_i) = \frac{1}{d^i}$$

where

$$d^i = \operatorname*{Max}_{j=1\ldots n+1} \{d_i^j\}, \text{ with } d_i^j = \operatorname{Max}\left\{\frac{d_{i1}^j}{d_{i2}^j}, \frac{d_{i2}^j}{d_{i1}^j}\right\}$$

and

$$d_{i1}^j = b_{ij} - a_{ij},\ d_{i2}^j = d_{ij} - c_{ij}.$$

Clearly, $RS \leqslant 1$, and if the rule has a symmetry semantic, then $RS = 1$.

An *evaluation function* to the rule R_i, and therefore, a fitness function to the associated chromosome C_i can be defined as follows:

$$F(R_i) = \Psi_{E_p}(R_i) G_{R_i} g_n(R_i^-) MWR(R_i) RS(R_i).$$

with the objective of maximizing the fitness function. We have selected the product operator for combining the criteria because it is strictly increasing in the variables and can provide us a good equilibrium among the criteria, if a criterion value is low, then the fitness function is low.

4.1.4. Genetic operators

During the reproduction phase of the genetic algorithm we use two classical genetic operators, mutation and crossover. We use the non-uniform mutation proposed by Michalewicz [23], and the max–min-arithmetical crossover used in [14], and the selection procedure is the stochastic universal sampling [1]. A short description of them is given below.

Non-uniform mutation: 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 $[c_{kl}, c_{kr}]$), the result is a vector $C_v^{t+1} = (c_1, \ldots, c_k', \ldots, c_H)$, with $k \in 1, \ldots, H$, and

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

where a is a random number that can have a value of zero or one, and, the function $\Delta(t, y)$ returns a value in the range $[0, y]$ such that the probability of $\Delta(t, y)$ being close to 0 increases as t increases:

$$\Delta(t, y) = y(1 - r^{(1 - t/T)^b}),$$

where r is a random number in the interval $[0, 1]$, T is the maximum number of generations and b is a parameter chosen by the user, which determines the degree of dependency with the number of iterations. This property causes this operator to make a uniform search in the initial space when t is small, and very locally at later stages.

Max–min-arithmetical crossover: 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, we generate the following four offspring:

$$C_1^{t+1} = aC_w^t + (1 - a)C_v^t,$$

$$C_2^{t+1} = aC_v^t + (1 - a)C_w^t,$$

$$C_3^{t+1} \quad \text{with } c_{3k}^{t+1} = \min\{c_k, c_k'\},$$

$$C_4^{t+1} \quad \text{with } c_{4k}^{t+1} = \max\{c_k, c_k'\},$$

and we select the two best offspring chromosomes.

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 descendants are the two best of the four aforesaid offspring.

Selection procedure: The selection procedure is the stochastic universal sampling, the number of offspring of any structure is limited by the floor and ceiling of the expected number of offspring [1], together with the elitist selection.

4.1.5. Parameters

We carry out our experiments using 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: $P_{m(gene)} = P_m/$ chromosome length.

4.2. Covering method

The covering method is developed as an iterative process. It allows to be obtained as a set of fuzzy

rules covering the set of examples. In each iteration, it runs the generating method, it chooses the best chromosome (rule), assigns the relative covering value to every example, and removes the examples with a covering value greater than ε.

We denote by R^e the set of fuzzy-control rules obtained from an expert. Using the aforesaid generating method, the covering method is developed as follows:

1. Initialization:
   - To introduce ω and ε.
   - To assign $CV[k] \leftarrow CV_{R^e}(e_k)$, $k = 1, \ldots, p$.
   - If $CV[k] \geqslant \varepsilon$ then to remove e_k from E_p, $k = 1, \ldots, p$.
2. Over the set of examples E_p, to apply the generating method.
3. To select the best chromosome C_r with R_r the associated fuzzy rule.
4. To introduce R_r in the set of rules R^g, which is initially empty.
5. For every $e_k \in E_p$ do

   $CV[k] \leftarrow CV[k] + R_r(e_k)$:

   If $CV[k] \geqslant \varepsilon$ then remove it from E_p.
6. If $E_p = \emptyset$ then Stop else return to Step 2.

5. Combining–simplifying rules process

Due to the iterative nature of the genetic generation process, an overlearning phenomenon may appear. This occurs when some examples are covered at a higher degree than the desired one and it makes the obtained RB perform worse. Since two similar rules or one rule similar to another given by an expert may be obtained in the generating process, it is necessary to combine and simplify the complete RB obtained from the previous process for deriving the final RB, thereby allowing the system to be controlled.

The set of rules obtained from experts' knowledge, denoted as R^e, together with the rules obtained with the generating process, denoted as R^g, are joined together in order to form the set of candidate rules R:

$$R = R^e \cup R^g, \quad |R| = m.$$

This process is based on a binary-coded GA, in which the selection of the individuals is developed using the aforementioned stochastic universal sampling procedure together with an elitist selection scheme, and the recombination is put into effect by using the classical binary multipoint crossover (performed at two points) and uniform mutation operators.

The coding scheme generates fixed-length chromosomes. Considering the m rules contained in R counted from 1 to m, an *m-bit string* $C = (c_1, \ldots, c_m)$ represents a subset of candidate rules to form the RB finally obtained as this stage output, B^s, such that

$$\text{If } c_i = 1 \text{ then } R_i \in B^s \text{ else } R_i \notin B^s$$

The initial population is generated by introducing a chromosome representing the complete previously obtained rule set R^g, that is, with all $c_i = 1$. The remaining chromosomes are selected at random.

As regards to the fitness function, $E(\cdot)$ it is based on an application-specific measure, the medium square error (MSE) over a training data set, E_{TDS}, which is represented by the following expression:

$$E(C_j) = \frac{1}{2|E_{TDS}|} \sum_{e_l \in E_{TDS}} (ey^l - S(ex^l))^2,$$

where $S(ex^l)$ is the output value obtained from the FLC using the RB coded in C_j, $R(C_j)$, when the state variables values are ex^l, and ey^l is the known desired value.

Anyway, there is a need to keep the *control rule completeness* property considered in the previous stage. An FLC must always be able to infer a proper control action for every process state. We will ensure this condition by forcing every example contained in the training set to be covered by the encoded RB at a degree greater than or equal to τ,

$$C_{R(C_j)}(e_l) = \bigcup_{j=1 \ldots T} R_j(e_l) \geqslant \tau,$$
$$\forall e_l \in E_{TDS} \text{ and } R_j \in R(C_j),$$

where τ is the *minimal training set completeness degree* accepted in the simplification process.

Therefore, we define a *training set completeness degree* of $R(C_j)$ over the set of examples E_{TDS} as

$$TSCD(R(C_j), E_{TDS}) = \bigcap_{e_l \in E_{TDS}} C_{R(C_j)}(e_l).$$

The final fitness function penalizing the lack of the completeness property is

$$F(C_j) = \begin{cases} E(C_j) & \text{if } TSCD(R(C_j), E_{\mathrm{TDS}}) \geqslant \tau, \\ \frac{1}{2}\sum_{e_l \in E_{\mathrm{TDS}}} (ey^l)^2 & \text{otherwise.} \end{cases}$$

6. Tuning process

In [14] it was presented in-depth the genetic tuning process used in this stage. The method relies on having a set of training data, E_{TDS}, against which the controller is tuned, and acts by means of an RCGA. The tuning method using GAs fits the membership functions of the fuzzy rules dealing with the parameters of the membership functions, minimizing a square error function defined by means of an input–output data set for evaluation.

A fuzzy rule set is represented as a chromosome, and we try to obtain the chromosome with the best adaptation. A rule R_i is represented by a piece of chromosome C_{ri},

$$C_{ri} = (a_{i1}, b_{i1}, c_{i1}, d_{i1}, \ldots, a_{in}, b_{in}, c_{in}, d_{in}, a_i, b_i, c_i, d_i),$$

therefore a base of t rules, R, is represented by the chromosome C_r,

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

The initial gene pool is created from the initial fuzzy rule set to be tuned. The initial rule set is a chromosome, which is denoted as C_1. We define for every gene c_h of C_1, $h = 1 \ldots (n+m) \times 4$ an interval of performance, $[c_h^l, c_h^r]$, which will be the interval of adjustment for 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 four parameters $(c_t, c_{t+1}, c_{t+2}, c_{t+3})$ and the intervals of performance are the following (see Fig. 1):

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

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

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

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

Therefore, we create a population of chromosomes with C_1 as the initial base of fuzzy control rules, and the remaining chromosomes initiated at random, with each gene being in its respective interval of performance.

By using a training input–output data set, E_{TDS}, we define the fitness function of a chromosome as the square medium error of its associated rule set. It is represented by the following expression:

$$E(C) = \frac{1}{2|E_{\mathrm{TDS}}|} \sum_{e_k \in E_{\mathrm{TDS}}} (ey^k - S(ex^k))^2.$$

The genetic operators and parameters are the same as in the generating process.

7. Experiments

To test the GFS proposal we have carried out experiments simulating the inverted pendulum problem and two n-dimensional functions used for deriving three-dimensional surfaces.

We use the first one for checking the use of expert rules in the GFS, and the second one for comparing the GFS with other learning approaches.

7.1. Inverted pendulum problem

An inverted pendulum is a simple example for control engineers, and the behaviour of the GFS proposal can be shown:

7.1.1. The model

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

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

where $k\,\mathrm{d}\theta/\mathrm{d}t$ is an approximation of the friction force.

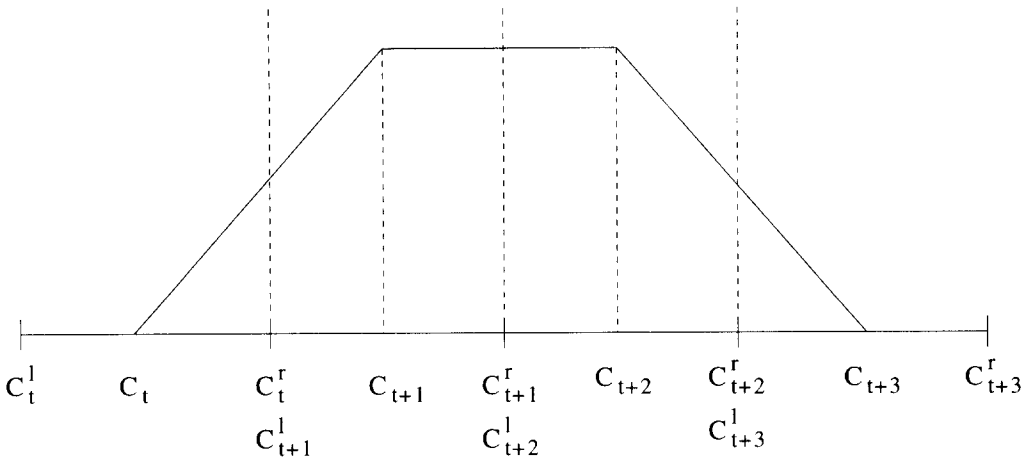

Fig. 1. Intervals of performance.

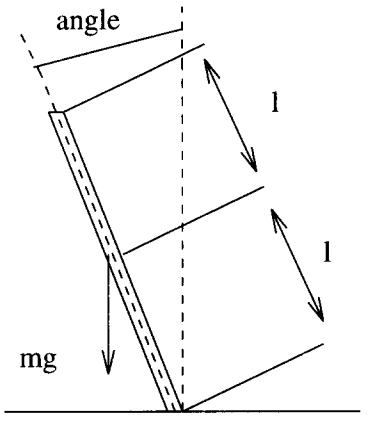

Fig. 2. Inverted pendulum.

Fig. 2 shows the system.

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 gravity centre of the pendulum for a constant amount time to move the pendulum to the vertical position.

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

$\theta \in [-0.5240, 0.5240]$ rad,

$\omega \in [-0.8580, 0.8580]$ rad/s,

$f \in [-2980.0, 2980.0]$ N.

Experimentally, we obtained two training data sets in the intervals $[-0.277, 0.277]$, $[-0.458, 0.458]$ and $[1592.273, 1592.273]$ for θ, ω and f, respectively. The first set with 213 input–output data is a training data set denoted as E_{TDS} and it is used in the learning processes components. The second with 125 input–output data set is a test data set denoted as E_{CD} and it is used for evaluation, using the measure of medium square method on it.

As regards to the FLC reasoning method employed, we have selected the *minimum t-norm* playing the role of the implication and conjunctive operators, and the *centre of gravity weighted by the matching* strategy acting as the defuzzification operator [8].

7.1.2. Expert rules

We consider the well-known linguistic rules that can be found in [32], where instead of the speed of the cart pole considered, we consider 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 three expert rules considered are:

Rule #3: IF θ is PS AND ω is NS THEN f is ZR,

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.

For managing these rules we consider the discourse universes partition presented in [22] in which the fuzzy numbers have normalized trapezoidal membership functions. We also consider triangular membership functions with the same support and the centre point of the modal interval as a modal value.

7.1.3. Experiments

The experiments were carried out with 1000 iterations for the generating and simplifying methods, and 2000 iterations for the tuning method.

We apply the process using the following framework:

1. For considering triangular membership functions (TMF1) or trapezoidal membership functions (TMF2).
2. Expert knowledge
   1.a Without expert knowledge (WEK)
   1.b Existence of expert knowledge
   We assume that an expert can give us information about the process. We shall consider two cases:
   - Expert knowledge 1 (EK1)
   The expert gives us the rule that represents the equilibrium point.

   Rule #7: IF θ is ZR AND ω is ZR
   THEN f is ZR.

   - Expert knowledge 2 (EK2)
   The expert gives us the rules where it is not necessary to apply force.

   Rule #3: IF θ is PS AND ω is NS
   THEN f is ZR,

   Rule #6: IF θ is NS AND ω is PS
   THEN f is ZR,

Table 1
Associated parameters

Trapezoidal memb. func.			Triangular memb. func.		
ε	ω	τ	ε	ω	τ
1.0	0.1	0.1	1.0	0.1	0.1
1.5	0.3	0.3	1.5	0.3	0.3
1.5	0.5	0.5	1.5	0.5	0.5
2.5	0.7	0.5	2.5	0.7	0.5

Rule #7: IF θ is ZR AND ω is ZR
THEN f is ZR,

3. Fitness functions in the generating process

For the generating process we use the fitness functions for obtaining the smallest width:

$$g_1(x) = e^{1-x} \quad \text{and} \quad g_2(x) = e^{1-x}.$$

We have carried out the experiments with the associated parameters (covering value, compatibility degree and completeness degree) as given in Table 1.

Tables 2–4 show the results for one configuration of TMF2 and two configurations for TMF1 on the test data set.

The medium square error of the first column is based on the set of generated rules together with the expert knowledge rules of the associated framework if there are, either WKE or EK1 or EK2.

Analyzing the results, we can observe the following:

- As regards to the use of triangular or trapezoidal membership functions, it seems better to use triangular membership functions instead of trapezoidal ones.
- As regards to the use of expert information, the behaviour of the learnt KB without expert knowledge is better than the KB that combines expert knowledge with learnt rules in the second case, in which it shows the best results of the experiments, and they present an intermediate behaviour in the other two cases. These results show a good performance of the proposed approach for learning good rules according to the available examples.

7.2. Three-dimensional surfaces

For analyzing the accuracy of the method proposed, we have selected two functions for using them to derive two three-dimensional surfaces.

Table 2
Triangular membership functions: $\varepsilon = 1.5$, $\omega = 0.5$, $\tau = 0.5$

	Generating process			Simplification process			Tuning process	
	MSE	$\|R\|$	$TSCD(\cdot,\cdot)$	MSE	$\|R\|$	$TSCD(\cdot,\cdot)$	MSE	$TSCD(\cdot,\cdot)$
TMF1-WEK	64863.83	18	0.5348	41735.33	11	0.5348	40.35	0.1854
TMF1-EK1	51818.03	16	0.5724	19989.27	9	0.5146	46.12	0.2337
TMF1-EK2	52576.84	18	0.6036	25861.34	11	0.5955	35.50	0.3155

Table 3
Trapezoidal membership functions: $\varepsilon = 2.5$, $\omega = 0.7$, $\tau = 0.5$

	Generating process			Simplification process			Tuning process	
	MSE	$\|R\|$	$TSCD(\cdot,\cdot)$	MSE	$\|R\|$	$TSCD(\cdot,\cdot)$	MSE	$TSCD(\cdot,\cdot)$
TMF2-WEK	97016.59	19	0.7516	67147.80	11	0.5598	153.29	0.4440
TMF2-EK1	92768.33	20	0.7940	51985.66	12	0.5242	241.81	0.2643
TMF2-EK2	86286.27	20	0.7940	35826.89	9	0.5242	32.17	0.2951

Table 4
Triangular membership functions: $\varepsilon = 2.5$, $\omega = 0.7$, $\tau = 0.5$

	Generating process			Simplification process			Tuning process	
	MSE	$\|R\|$	$TSCD(\cdot,\cdot)$	MSE	$\|R\|$	$TSCD(\cdot,\cdot)$	MSE	$TSCD(\cdot,\cdot)$
TMF1-WEK	64859.17	25	0.6335	31635.72	12	0.5352	23.71	0.1541
TMF1-EK1	60566.21	25	0.5734	13090.77	9	0.4992	67.06	0.2801
TMF1-EK2	62227.62	26	0.6284	13619.46	10	0.4942	40.04	0.3264

7.2.1. The models

The functions and the variable universes of discourse considered are shown below. The *spherical model*, F_1, is an unimodal function while the *generalized Rastrigin function*, F_2, is a strongly multimodal one, as may be observed in their graphical representations (Fig. 3):

$$F_1(x_1,x_2) = x_1^2 + x_2^2,$$
$$x_1,x_2 \in [-5,5],\ F_1(x_1,x_2) \in [0,50],$$
$$F_2(x_1,x_2) = x_1^2 + x_2^2 - \cos(18x_1) - \cos(18x_2),$$
$$x_1,x_2 \in [-1,1],\ F_2(x_1,x_2) \in [2,3.5231].$$

For each one of the functions, a training data set uniformly distributed in the three-dimensional definition space has been obtained experimentally. In this way, two sets with 1681 values have been generated by taking 41 values for each one of the two state variables considered to be uniformly distributed in their respective intervals.

Two other data sets have been generated for their use as test sets for evaluating the performance of the learning method, avoiding any possible bias related to the data in the training set. The size of these data sets is a percentage of the corresponding training set one, a 10% to be precise. The data are obtained by generating, at random, the state variable values into the concrete universes of discourse for each one of them, and computing the associated output variable value. Hence, two test sets formed by 168 data are used to measure the accuracy of the FLCs designed by computing the medium square error for them.

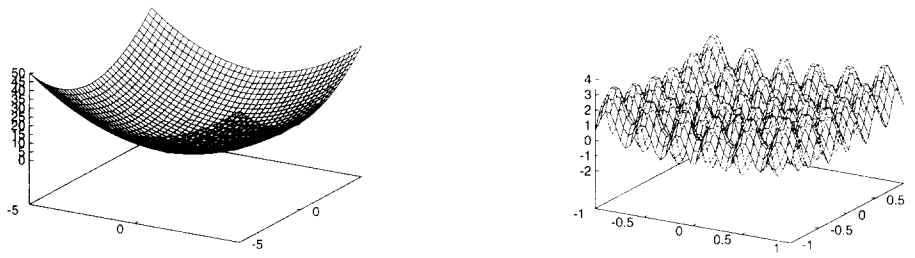

Fig. 3. Graphical representation of F_1 and F_2.

Table 5
Triangular membership functions: $\varepsilon = 1.5$, $\omega = 0.05$, $\tau = 0.25$

	Generating process		Simplification process		Tuning process
	MSE	$\|R\|$	MSE	$\|R\|$	MSE
GLP-F1	6.146344	88	3.986195	64	0.768021
WM+Tuning-F1	4.6518	49			0.9507
GLP-F2	0.424792	167	0.300555	119	0.270455
WM+Tuning-F2	2.0940	49			1.2180

7.2.2. *Experiments*

In order to compare the proposed method we have compared it with a two-stage method: obtaining a complete KB by deriving the RB by means of the Wang and Mendel's (WM) method [31], and defining the DB by means of the genetic tuning process.

The initial DB used in the WM process is constituted by three primary fuzzy partitions (two corresponding to the state variables and one associated to the control one) formed by *seven linguistic terms* with triangular-shaped fuzzy sets giving meaning to them (as shown in Fig. 4), and the adequate scaling factors to translate the generic universe of discourse into the one associated with each problem variable.

Regarding our GFS we also use triangular membership functions. The following parameter values, corresponding to the first two stages, are used: $\varepsilon = 1.5$, $\omega = 0.05$ and $\tau = 0.25$.

Finally, as regards to the FLC reasoning method employed, we again have selected the *minimum t-norm* playing the role of the implication and conjunctive operators, and the *centre of gravity weighted by the matching* strategy acting as the defuzzification operator [8].

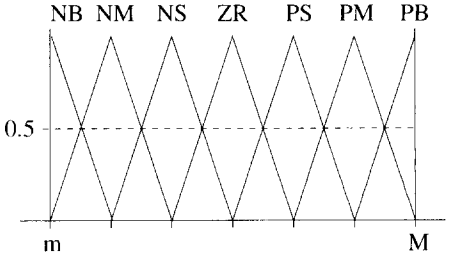

Fig. 4. Graphical representation of the fuzzy partition.

The results obtained are shown in Table 5, each one of them associated to both of the functions considered and to the test sets.

Analyzing these results, we can observe the good behaviour presented by the proposed method; in both functions it presents the best results. More concretely for the second one, the strongly multimodal one, the difference is higher between the two learning algorithms.

A drawback may be associated to the proposed process, the KB loses its readability. We overcome the lack of flexibility due to the rigid partitioning of the

input and output spaces [2] and the loss of KB readability may be justified by the benefit obtained by the improved FLC accuracy, which is a main concern in many real control problems.

8. Conclusions

We have focused this paper on the development of a GFS for learning fuzzy-control rules from examples, where it is possible to combine expert knowledge represented as linguistic control rules and fuzzy control rules generated from numerical examples.

An advantage of this GFS proposal is that in the first stage, the fuzzy rule generating process based on an iterative rule learning approach, it considerably reduces the space of search because it looks for only one fuzzy rule in each sequence of iterations, and stages second and third provide tools that can improve the RB and DB, respectively. The first stage establishes a competition among the fuzzy rules whilst the aim of the second and third stage is to find the best possible cooperation among the rules for obtaining a good KB.

Future works will be directed to the development of this GFS methodology.

References

[1] J.E. Baker, Reducing bias and inefficiency in the selection algorithm, in: J.J. Grefenstette (Ed.), Proc. 2nd Internat. Conf. on Genetic Algorithms, Lawrence Erlbaum, Hillsdale, NJ, 1987, pp. 14–21.

[2] A. Bastian, How to handle the flexibility of linguistic variables with applications, Internat. J. Uncertainty Fuzziness Knowledge-Based Systems 2 (1994) 463–484.

[3] F. Bolata, A. Nowè, From fuzzy linguistic specifications to fuzzy controllers using evolution strategies, Proc. 4th IEEE Internat. Conf. on Fuzzy Systems, Yokohama, 1995, pp. 1089–1094.

[4] B. Carse, T.C. Fogarty, A. Munro, Evolving fuzzy rule based controllers using genetic algorithms, Fuzzy Sets and Systems 80 (1996) 273–293.

[5] M. Cooper, J.J. Vidal, Genetic design of fuzzy controllers: the cart and jointed pole problem. Proc. 3rd IEEE Internat. Conf. on Fuzzy Systems, Orlando, 1994, pp. 1332–1337.

[6] O. Cordón, F. Herrera, A general study on genetic fuzzy systems, in: J. Periaux, G. Winter, M. Galan, P. Cuesta (Eds.), Genetic Algorithms in Engineering and Computer Science, Wiley, New York, 1995, pp. 33–57.

[7] O. Cordón, F. Herrera, M. Lozano, A classified review on the combination fuzzy logic-genetic algorithms bibliography, Tech. Report #DECSAI–95129, Dept. of Computer Science and A.I., University of Granada, 1995. Available at the URL address: http://decsai.ugr.es/~herrera/fl-ga.html.

[8] O. Cordón, F. Herrera, A. Peregrin, Applicability of the fuzzy operators in the design of fuzzy logic controllers, Fuzzy Sets and Systems, to appear.

[9] M. Delgado, A. González, A frequency model in a fuzzy environment, Internat. J. Approx. Reasoning 11 (1994) 159–174.

[10] D. Driankov, H. Hellendoorn, M. Reinfrank, An Introduction to Fuzzy Control, Springer, Berlin, 1993.

[11] D.E. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley, Reading, MA, 1989.

[12] A. González, R. Pérez, Structural learning of fuzzy rules from noisy examples, Proc. FUZZ-IEEE/IFES '95, Yokohama, Vol. III, 1995, pp. 1323–1330.

[13] A. González, R. Pérez, A learning system of fuzzy control rules, in: F. Herrera, J.L. Verdegay (Eds.), Genetic Algorithms and Soft Computing, Physica-Verlag, Wurzburg, 1996, pp. 202–225.

[14] F. Herrera, M. Lozano, J.L. Verdegay, Tuning fuzzy logic controllers by genetic algorithms, Internat. J. Approx. Reasoning 12 (1995) 299–315.

[15] F. Herrera, M. Lozano, J.L. Verdegay, Tackling real-coded genetic algorithms: operators and tools for the behaviour analysis, Artificial Intelligence Rev., to appear.

[16] F. Herrera, J.L. Verdegay, (Eds.), Genetic Algorithms and Soft Computing, Physica-Verlag, Wurzburg, 1996.

[17] J.H. Holland, Adaptation in Natural and Artificial Systems, Ann Arbor, 1975 (MIT Press, New York, 1992).

[18] J.H. Holland, J.S. Reitman, Cognitive systems based on adaptive algorithms, in: D.A. Waterman, F. Hayes-Roth (Eds.), Pattern-Directed Inference Systems, Academic Press, New York, 1978.

[19] C. Karr, Genetic algorithms for fuzzy controllers, Artificial Intelligence Expert 6 (1991) 26–33.

[20] C. Karr, Applying Genetic Algorithms to Fuzzy Logic, Artificial Intelligence Expert 6 (1991) 38–43.

[21] M.A. Lee, H. Takagi, Embedding a priori knowledge into an integrated fuzzy system design method based on genetic algorithms, Proc. 5th Internat. Fuzzy Systems Association World Congress, Seoul, 1993, pp. 1293–1296.

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

[23] Z. Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs, Springer, Berlin, 1992.

[24] D.T. Pham, D. Karoboga, Optimum design of fuzzy logic controllers using genetic algorithms, J. Systems Eng. 1 (1991) 114–118.

[25] A. Satyadas, K. Krishanakumar, GA-optimized fuzzy controller for spacecraft attitude control, Proc. 3rd IEEE Internat. Conf. on Fuzzy Systems, Orlando, 1994, pp. 1979–1984.

[26] S.F. Smith, A learning system based on genetic adaptive algorithms, Ph.D. Thesis, University of Pittsburgh, 1980.

[27] H. Surmann, A. Kanstein, K. Goser, Self-organizing and genetic algorithms for an automatic design of fuzzy control and decision systems, Proc. 1st European Congress on Fuzzy and Intelligent Technologies, Aachen, 1993, pp. 1097–1104.

[28] P. Thrift, Fuzzy logic synthesis with genetic algorithms, Proc. 4th Internat. Conf. on Genetic Algorithms, San Diego, 1991, pp. 509–513.

[29] J.R. Velasco, L. Magdalena, Genetic learning applied to fuzzy rules and fuzzy knowledge bases, Proc. 6th Internat. Fuzzy Systems Association World Congress, Sao Paulo, 1995, pp. 257–260.

[30] G. Venturini, SIA: A supervised inductive algorithm with genetic search for learning attribute based concepts, Proc. European Conf. on Machine Learning, Vienna, 1993, pp. 280–296.

[31] L.X. Wang, J.M. Mendel, Generating fuzzy rules by learning from examples, IEEE Trans. Systems Man Cybernet. 22 (1992) 1414–1427.

[32] T. Yamakawa, Stabilization of an inverted pendulum by a high-speed fuzzy logic controller hardware system, Fuzzy Sets and Systems 32 (1991) 161–180.