ELSEVIER

Information Sciences 136 (2001) 85–107

INFORMATION SCIENCES
AN INTERNATIONAL JOURNAL

www.elsevier.com/locate/ins

A genetic learning process for the scaling factors, granularity and contexts of the fuzzy rule-based system data base ☆

O. Cordón [a,*], F. Herrera [a], L. Magdalena [b], P. Villar [c]

[a] *Department of Computer Science and Artificial Intelligence, E.T.S. de Ingeniería Informática, Universidad de Granada, 18071 Granada, Spain*
[b] *Dept. de Matemática Aplicada, E.T.S.I. Telecomunicación, Universidad Politécnica de Madrid, 28040 Madrid, Spain*
[c] *Dept. de Informática, Escola Superior de Enxeñería Informática, Universidade de Vigo, Campus As Lagoas, 32004 Ourense, Spain*

Abstract

In this contribution, we propose a new method to automatically learn the Knowledge Base of a Fuzzy Rule-Based System by finding an appropriate Data Base using a Genetic Algorithm and considering a simple generation method to derive the Rule Base. Our genetic process learns all the components of the Data Base (number of labels, working ranges and membership function shapes for each linguistic variable) using a non-linear scaling function to adapt the fuzzy partition contexts. © 2001 Elsevier Science Inc. All rights reserved.

Keywords: Fuzzy rule-based systems; Data base; Learning; Genetic algorithms; Granularity; Context adaptation; Scaling functions

1. Introduction

The generation of the Knowledge Base (KB) of a Fuzzy Rule-Based System (FRBS) presents several difficulties since the KB depends on the concrete

☆ This research has been supported by CICYT under project PB98-1319.
* Corresponding author.
E-mail addresses: ocordon@decsai.ugr.es (O. Cordón), herrera@decsai.ugr.es (F. Herrera), llayos@mat.upm.es (L. Magdalena), pvillar@uvigo.es (P. Villar).

application. This makes the accuracy of the FRBS directly depend on its composition.

Many approaches have been proposed to automatically learn the KB from numerical information. Most of them have focused on the Rule Base (RB) learning, using a predefined Data Base (DB) [2,7,13,20,25–27,37,39]. This operation mode makes the DB have a significant influence on the FRBS performance. In fact, some studies have showed that the system performance is much more sensitive to the choice of the semantics in the DB than to the composition of the RB [5,12,40].

The usual solution to improve the FRBS performance by dealing with the DB components involves a tuning process of the preliminary DB definition once the RB has been derived [4,5,8,22,30]. This process only adjusts the membership function definitions and does not modify the number of linguistic terms in each fuzzy partition since the RB remains unchanged. In opposite to this a posteriori DB learning, there are some approaches that learn the different DB components a priori [9,12,14,17,18,29,36,38].

In this paper, we propose a new process to automatically generate the KB of a Mamdani FRBS based on a new learning approach [14,18] composed of two methods with different goals:

- A genetic learning process for the DB that allows us to define:
  - The number of labels for each linguistic variable.
  - The variable domain (working range).
  - The form of each fuzzy membership function in non-uniform fuzzy partitions, using a non-linear scaling function that defines different areas in the variable working range where the FRBS has a higher or a lower relative sensibility, i.e., the fuzzy partition contexts.
- A quick ad hoc data-driven method [7] that derives the RB considering the DB previously obtained. This method is run from each DB definition generated by the Genetic Algorithm (GA), thus allowing the proposed hybrid learning process to finally obtain the whole definition of the KB (DB and RB) by means of the cooperative action of both methods.

Besides, we present an extension of the proposed method that changes the fitness function with the aim of obtaining FRBSs with a better generalization capability and whose RB is comprised by a lesser number of rules.

In order to do that, this paper is organized as follows. Sections 2 and 3 show some preliminaries about the KB learning in FRBSs and fuzzy context adaptation, respectively. In Section 4, our method is presented, describing the context adaptation approach considered and the components of the genetic process: coding of the solutions, initial population, evaluation function and genetic operators. Section 5 shows some experimental results. Section 6 presents the extension of the proposed method. Finally, in Section 7, some conclusions are pointed out.

2. Automatic learning of the KB

Two problems arise when generating the KB of a FRBS:

- The DB learning, that comprises the specification of the universes of discourse and the number of labels for each linguistic variable, as well as the fuzzy membership functions associated to each label.
- The RB derivation, involving the determination of the number of rules and of the composition of each one of them (i.e., of the specific labels associated to each linguistic variable).

As said, most of the approaches proposed to automatically learn the KB from numerical information have focused on the RB learning, using a predefined DB. The usual way to define this DB involves choosing a number of linguistic terms for each linguistic variable (an odd number between 3 and 9, which is normally the same for all the variables) and setting the values of the system parameters by an uniform distribution of the linguistic terms into the variable universe of discourse. The RB learning methods are based on different techniques such as ad hoc data-driven algorithms [2,13,27,39], least square methods [2], Simulated Annealing [7] and GAs [20,25,26,37]. Fig. 1(a) graphically shows this type of KB learning.

This operation mode makes the DB have a significant influence on the FRBS performance. In fact, studies such as the ones developed in [5,40] show, for the case of Fuzzy PI Controllers, that the system performance is much more sensitive to the choice of the semantics in the DB than to the composition of the RB. Considering a previously defined RB, the performance of the Fuzzy Controller is sensitive to four aspects in the following order: scaling factors, peak values, width values and rules.

In [12], the influence of fuzzy partition granularity (number of linguistic terms per variable) in the FRBS performance is studied, showing that using an appropriate number of terms for each linguistic variable, the FRBS accuracy can be improved with no need to use complex RB learning methods.

With the aim of making the FRBS perform better, some approaches try to improve the preliminary DB definition once the RB has been derived. To do so, a tuning process considering the whole KB obtained (the preliminary DB and the derived RB) is used a posteriori to adjust the membership function parameters. Nevertheless, the tuning process only adjusts the shapes of the membership functions and not the number of linguistic terms in each fuzzy partition, which remains fixed from the beginning of the design process. A graphical representation of this kind of learning is showed in Fig. 1(b). For some examples of tuning methods based on Simulated Annealing and GAs, refer to [4,5,8,22,30].

Other approaches try to learn the two components of the KB simultaneously. This kind of learning is depicted in Fig. 1(c). Working in this way,

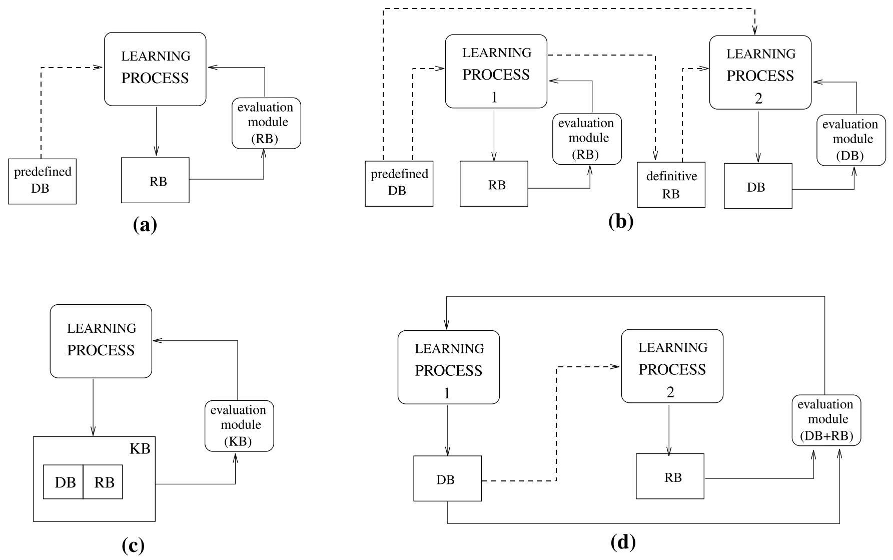

Fig. 1. Graphical representation of the different KB learning approaches.

they have the possibility of generating better definitions but they deal with a larger search space that makes the learning process more difficult and slow. For some examples, refer to [6,11,16,31–33].

Finally, there is another way to generate the whole KB that considers two different processes to derive both components, DB and RB. A DB generation process wraps a RB learning one working as follows: each time a DB has been obtained by the DB definition process, the RB generation method is used to derive the rules, and some type of error measure is used to validate the whole KB obtained. We should note that this operation mode involves a partitioning of the KB learning problem. Whilst the learning processes belonging to the previous family (Fig. 1(c)) look for solutions in a complex global search space (DB + RB), the processes belonging to the current group are composed of two different (and independent) learning processes looking for solutions in two simpler search spaces (DB and RB ones) to obtain complete solutions. This type of KB learning is represented in Fig. 1(d). The following processes are examples of this kind of KB learning:

- The method proposed in [12] uses Simulated Annealing to learn an appropriate fuzzy partition granularity for each variable and various RB learning methods to derive the rules.
- In the approach presented in [18], a genetic process that obtains a KB for simplified TSK rules is proposed.
- The method presented in [29] deals with a GA to design a FRBS for pattern classification problems.

- The approach proposed in [17] uses a GA to learn the definition points of the membership functions and the Wang and Mendel's rule generation method [39] to derive the RB.
- A method that simultaneously adjusts the number of labels per variable and the membership functions parameters is presented in [14].

Our method will belong to this group. We use a genetic process to learn all the components of the DB (number of labels, working ranges and membership functions shapes) and a simple ad hoc data-driven algorithm to derive the RB.

In opposite to the *local* tuning of the membership functions developed in many approaches, the method presented in this paper implements a *global* tuning of the membership functions (a change in the parameter values origins a change in the whole fuzzy partition). Besides, the FRBSs obtained are more readable because *strong* fuzzy partitions are considered. A fuzzy partition with l labels $A = (A_1, A_2, \ldots, A_l)$ is called *strong* if:

$$\sum_{i=1}^{l} \mu_{A_i}(x) = 1, \quad x \in U.$$

In order to evaluate its performance, the FRBSs obtained from it will be compared with others designed by the usual way, that is, learning of the RB using a predefined DB, with and without a tuning process of the DB maintaining fixed the RB previously obtained (Figs. 1(a) and (b) respectively), and with the ones obtained from another method belonging to the same family [17] (Fig. 1(d)).

3. Context adaptation in fuzzy processing

The use of contextual transformation functions to adjust membership functions in Fuzzy Systems can be a good form to find the fuzzy sets that best represent the linguistic terms they are associated with.

Considering concepts representable by fuzzy sets, the main effect of a context can be related by some sort of filtering. That is, the same base concept can be perceived in different situations, provided it is filtered to suit the context particulars [21].

The scaling functions map the input and output variables onto the universe of discourse of the fuzzy sets definitions. From a linguistic point of view, the scaling function can be interpreted as a sort of context information. While the membership functions describe the relative semantics (context-independent) of the linguistic variables contained in the rules, the union of the scaling functions and the membership functions generates the absolute semantics of the linguistic variables (context-dependent through the scaling functions).

Different results may be obtained when modifying the scaling function working on a certain variable, obtaining two kinds of contexts, linear and non-linear ones. In a non-linear context, the membership function shape can be adjusted. This is not possible when using linear context adaptation.

As an example of context adaptation, the concept of *flexible linguistic variable* is introduced in [3]. The linguistic terms of those variables are defined by a flexible semantic KB. Considering the flexible linguistic variable named AGE, a possible linguistic value for it is *old*. Usually, nobody tends to judge his own age as old. So the meaning of the fuzzy set *old* is context dependent because it may change according to the speaker age. Hence, the semantic KB to adapt the fuzzy set *old* could be a single rule:

$$\mu_{old} = (own\_age + 5, own\_age + 10, own\_age + 15).$$

In [21], more details about fuzzy context adaptation can be found, such us a formalization of the context adaptation procedure, the formal requirements for the scaling function and some context adaptation approaches.

The next proposals are FRBS learning methods based on context adaptation:

- The approach proposed in [35] adjusts the membership functions of the input variables and the gain for the output variable in the framework of Fuzzy Control.
- The method presented in [21] is a genetic learning method for the parameters of a determinated non-linear scaling function, in order to find an adequate fuzzy partition per variable. A similar study based on Neural Networks is proposed in [36].
- The approach proposed in [32] uses a GA to adapt the whole KB definition of a Fuzzy Controller. For the learning of the DB, the GA evolves the working ranges of the linguistic variables and the membership functions shapes, by means a non-linear scaling function that produces high sensibility either for the medium values or for the extreme values of the linguistic variables. This approach is refined in [33] by adding a new parameter to the non-linear scaling function that may produce high sensibility in only one of the working range limits of the linguistic variables.

The learning method proposed in this paper uses a GA to define the parameters of the non-linear scaling function proposed in [33], including the possibility of an enlargement of the initial working ranges of the linguistic variables (translating to the Fuzzy Modelling field the idea proposed in [32] for Fuzzy Controllers).

The main contribution of our method is that it also evolves the number of labels per linguistic variable, an information that has not been considered to be relevant for the context adaptation methods mentioned before. The fuzzy partition granularity of a linguistic variable can also be viewed as a sort of context information. Considered a specific label set for a variable, the scaling functions can adapt the membership functions to a determinated real situation. However, in other contexts, some labels can result irrelevant, that is, they can contribute nothing and even can cause confusion. In other cases, it would be necessary to add new labels to appropriately differentiate the values of the variable.

4. Learning the DB of a FRBS using GAs

In this section, we propose a new process to automatically generate the KB of a Mamdani FRBS based on a new learning approach composed of two methods with different goals:

- A genetic learning process for the DB that allows us to define:
  - The number of labels for each linguistic variable.
  - The variable domain (working range).
  - The form of each fuzzy membership function in non-uniform fuzzy partitions, using a non-linear scaling function that defines different areas in the variable working range where the FRBS has a higher or a lower relative sensibility, i.e., the fuzzy partition contexts.
- A quick ad hoc data-driven method [7] that derives the RB considering the DB previously obtained. This method is run from each DB definition generated by the GA, thus allowing the proposed hybrid learning process to finally obtain the whole definition of the KB (DB and RB) by means of the cooperative action of both methods.

The granularity level per variable has a great influence in the final FRBS performance, as stated in [12]. Our method also evolves the variable working range, usually considered as a fixed part of the problem.

4.1. Context adaptation proposal

All the components of the DB will be adapted throughout a genetic process. Since it is interesting to reduce the dimensionality of the search space for that process, the use of non-linear scaling functions is conditioned by the necessity of using parameterized functions with a reduced number of parameters. We consider the scaling function proposed in [32], that has a single sensibility parameter called a ($a \in \mathbb{R}$). The function used is ($f : [-1, 1] \to [-1, 1]$)

$$f(x) = \operatorname{sign}(x) \cdot |x|^a \quad \text{with } a > 0.$$

The final result is a value in $[-1, 1]$ where the parameter a produces uniform sensibility ($a = 1$), higher sensibility for center values ($a > 1$), or higher sensibility for extreme values ($a < 1$). In this paper, triangular membership functions are considered due to their simplicity. So, the non-linear scaling function will only be applied on the three definition points of the membership function (which is equal to transform the scaling function in a continuous piecewise linear function), in order to make easier the structure of the generated DB and to simplify the defuzzification process. Fig. 2 shows a graphical representation of these three possibilities.

We should note that the previous scaling function is recommended to be used with symmetrical variables since it causes symmetrical effects around the center point of the interval. For example, it cannot produce higher sensibility in only one of the working range extents.

In [15], a genetic learning method for the DB using the previous scaling function is proposed. In the method presented in this paper, we add a new parameter (called S) to the non-linear scaling function as described in [33]. S is a parameter in $\{0, 1\}$ to distinguish between non-linearities with symmetric shape (lower sensibility for middle or for extreme values, Fig. 2) and asymmetric shape (lower sensibility for the lowest or for the highest values).

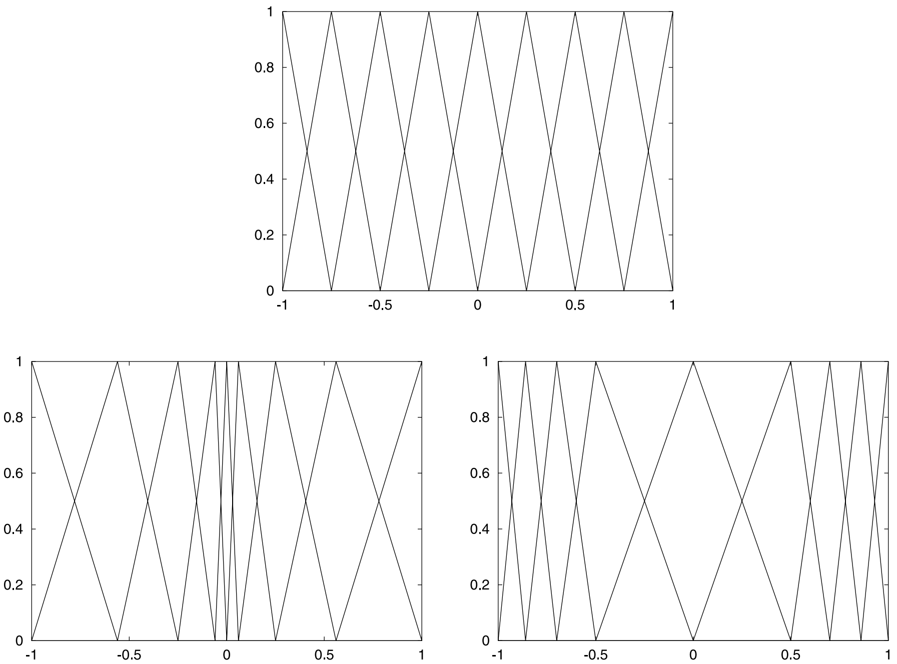

Fig. 2. Fuzzy partitions with $a = 1$ (top), $a > 1$ (down left), and $a < 1$ (down right).

Therefore, the context adaptation process involves three or four steps to build the fuzzy partition associated to each variable depending on the value of the parameter S:

- The first step builds a uniform fuzzy partition considering the number of labels of the variable and its working range ($[v_{\min}, v_{\max}]$).
- If $S = 0$ or $a = 1$:
  - The second step uses the working range ($[v_{\min}, v_{\max}]$) to produce a linear mapping of the fuzzy partition from $[v_{\min}, v_{\max}]$ to $[-1, 1]$.
  - The third step introduces the non-linearity through the scaling function $f(x) = \mathrm{sign}(x) \cdot |x|^a$, that maintains the extremes of the interval unchanged ($[-1, 1]$).
- If $S = 1$ and $a \neq 1$:
  - The second step uses the working range ($[v_{\min}, v_{\max}]$) to produce a linear mapping of the fuzzy partition from $[v_{\min}, v_{\max}]$ to $[0, 1]$.
  - The third step introduces the non-linearity through the scaling function:

$$f' : [0,1] \to [0,1], \quad f'(x) = \begin{cases} |x|^a & \text{when } a < 1, \\ 1 - (1 - |x|)^{1/a} & \text{when } a > 1, \end{cases}$$

  which extends the previously defined one by permitting a symmetric effect in the two interval extremes.
  - In the four step, a second linear mapping transforming the resulting fuzzy partition from $[0, 1]$ to $[-1, 1]$ is applied.

The overall result is a non-linear mapping from $[v_{\min}, v_{\max}]$ to $[-1, 1]$. A graphical representation of the fuzzy partitions when $S = 1$ is showed in Fig. 3

4.2. Genetic learning process

GAs [19,34] are search and optimization techniques based on a formalization of natural genetics. The genetic process starts with a population of solutions called chromosomes, that constitutes the first generation ($G(0)$), and

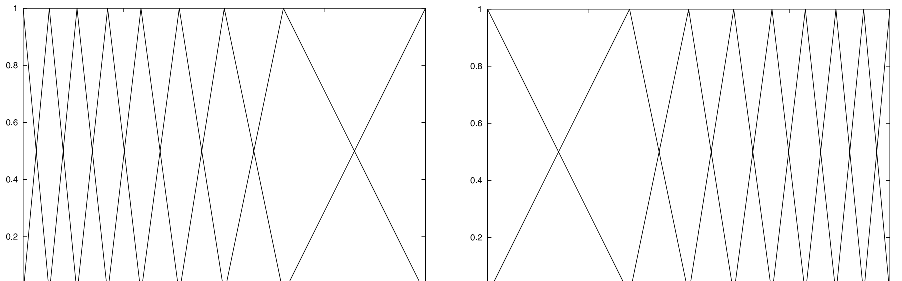

Fig. 3. Fuzzy partitions with $S = 1$ (left with $a > 1$ and right with $a < 1$).

undergoes evolution over it. While a certain termination condition is not met, each chromosome is evaluated by means of an evaluation function (a fitness value is assigned to the chromosome) and a new population is created $(G(t+1))$, by applying a set of genetic operators to the individuals of generation $G(t)$. Different proposals that use GAs in order to design FRBSs are contained in [9,23].

The important questions when using GAs are: how to code each solution, how to evaluate these solutions and how to create new solutions from existing ones. Moreover, it is relatively important the choice of the initial population, because we can obtain good solutions more quickly if an appropriate initial gene pool is chosen.

In our process, each chromosome represents a complete DB definition by encoding the said parameters. To evaluate a chromosome, we use an ad hoc data-driven method to learn the RB considering the DB contained in it, generating a complete KB, and then the accuracy of the FRBS obtained on a training data set is measured.

The next four sections describe the main components of the genetic learning process.

4.2.1. Encoding the DB

As said, the three main components of the DB are the number of linguistic terms for variable, the membership functions that define their semantics and the scaling factors. The latter component allows us to change the variable working range.

As regards the membership functions, we initially consider triangular-shaped functions, symmetrical and uniformly distributed across the variable working range. Then, we apply the non-linear function described in the previous section on the three definition points of each label. Thus, the working ranges, the number of labels and the sensibility parameters for each variable are the only information needed to define the whole fuzzy partition.

Therefore, each chromosome will be composed of three parts:

- Number of labels (C_1): For a system with N variables (including the input and output ones), the number of labels per variable is encoded into an array of length N. In this contribution, the possible values considered are the set $\{3,\ldots,9\}$.
- Sensibility parameters (C_2): An array of length $N \times 2$, where the sensibility parameters (a,S) are stored for each variable. In our case, the range considered for the parameter a is the interval $(0,10)$.
- Working ranges (C_3): An array of $N \times 2$ real values stores the variable working range $([r^{\inf}, r^{\sup}])$. If the initial domain of the variable i is $[v_i^{\min}, v_i^{\max}]$, and d is the interval dimension $(d = v_i^{\max} - v_i^{\min})$, the ranges considered for the variable domain limits are:

$$\begin{aligned}
&\text{lower limit}: && [v_i^{\min} - (1/4 \ * \ d), v_i^{\min}],\\
&\text{upper limit}: && [v_i^{\max}, v_i^{\max} + (1/4 \ * \ d)].
\end{aligned}$$

Hence, the structure of the chromosome is summarized next (considering that $R_i = \{r_i^{\inf}, r_i^{\sup}\}$):

$$\begin{aligned}
&C_1 = (l_1, \ldots, l_N),\\
&C_2 = (a_1, \ldots, a_N, S_1, \ldots, S_N),\\
&C_3 = (R_1, \ldots, R_N),\\
&C = C_1C_2C_3.
\end{aligned}$$

4.2.2. Initial gene pool

The initial population is composed of three parts, the first two having $\#val \times 5$ chromosomes, with $\#val$ being the cardinality of the term set (in our case $\#val = 7$, corresponding to the seven possibilities for the number of labels, $3, \ldots, 9$). Therefore, the number of chromosomes (M) has to be at least greater than $\#val \times 10$. The generation of the initial gene pool is described next:

- The first $\#val \times 5$ chromosomes will have the same number of labels and the initial working range in all its variables. For each possible number of labels, five individuals with the main possibilities for the sensibility parameters will be created: one with $a = 1$, two with $a < 1$ (one with $S = 0$ and another with $S = 1$) and the other two with $a > 1$ (one with $S = 0$ and another with $S = 1$). The latter four values of parameter a are generated at random.
- The second $\#val \times 5$ chromosomes are equal to the first group, but randomly changing the variable working range. Each chromosome will have the same number of labels in all its variables. For each possible number of labels, five individuals are created as in the first part of the population (one with $a = 1$, two with $a < 1$ and the other two with $a > 1$). For the third part of the chromosomes, two random values in the variable working range interval (lower and upper) are selected.
- In the rest of the initial population, the remaining $M - (\#val \times 10)$ chromosomes, all the components are selected at random. In our case, this part is comprised by 30 chromosomes, so, the total population length is 100.

4.2.3. Evaluating the chromosome

There are three steps that must be done to compute the fitness of each chromosome:

- Generate the fuzzy partitions for all the linguistic variables using the information contained in the chromosome as introduced in Section 4.1.
- Generate the RB by running a fuzzy rule learning method considering the DB obtained.

- Calculate the mean square error (MSE) over the training set using the KB obtained (genetically derived DB + RB) by means of the expression:

$$\mathrm{MSE} = \frac{1}{2|E|} \sum_{e_i \in E} (ey^i - S(ex^i))^2$$

with $E = \{e_1, \ldots, e_i, \ldots, e_p\}$ being the example set (training or test), $S(ex^i)$ being the output value obtained from the FRBS when the input variable values are $ex^i = (ex^i_1, \ldots, ex^i_n)$, and ey^i being the known desired value.

4.2.4. Genetic operators

A set of genetic operators is applied to the genetic code of the DBs contained in $G(t)$, to obtain $G(t+1)$. Due to the special nature of the chromosomes involved in this DB definition process, the design of genetic operators able to deal with it becomes a main task. Since there is a strong relationship among the three chromosome parts, operators working cooperatively in C_1, C_2 and C_3 are required in order to make best use of the representation tackled.

Taking into account these aspects, the following operators are considered:

4.2.4.1. Selection. The reproduction operator is Baker's stochastic universal sampling [1], in which the number of any structure offspring is limited by the floor and ceiling of the expected number of offspring, together with the elitist selection.

4.2.4.2. Crossover. As regards the recombination process, two different crossover operators are considered depending on the two parents' scope:

- *Crossover when both parents have the same granularity level per variable:* If the two parents have the same values in C_1 (each variable has the same number of labels in the two parents), the genetic search has located a promising space zone that has to be adequately exploitated. This task is developed by applying the max-min-arithmetical (MMA) crossover operator in the chromosome parts based on real-coding scheme (parameters a_i, second part of C_2, and working ranges, C_3) and obviously by maintaining the parent C_1 values in the offspring. If the parameter S_i is equal in the two parents, this value is also maintained in the offspring. If it is different, both combinations are tested and the best value is selected. This crossover operator is proposed in [24] and works in the way shown below.

If $C^t_v = (c_1, \ldots, c_k, \ldots, c_H)$ and $C^t_w = (c'_1, \ldots, c'_k, \ldots, c'_H)$ are to be crossed, the following four offsprings are generated (with $d \in [0, 1]$)

$$\begin{aligned}
&C_1^{t+1} = dC_w^t + (1-d)C_v^t,\\
&C_2^{t+1} = dC_v^t + (1-d)C_w^t,\\
&C_3^{t+1} \text{ with } c_{3k}^{t+1} = \min\{c_k, c'_k\},\\
&C_4^{t+1} \text{ with } c_{4k}^{t+1} = \max\{c_k, c'_k\}.
\end{aligned}$$

If the values for the parameter S_i are different in the two parents, eight offsprings are generated, the previous four with $S = 0$ and the same four with $S = 1$.

This operator uses a parameter d which is either a constant, or a variable whose value depends on the age of the population. The resulting descendents are the two best of the four (or eight) aforesaid offsprings.

• *Crossover when the parents encode different granularity levels:* This second case highly recommends the use of the information encoded by the parents to explore the search space in order to discover new promising zones. Hence, when C_1 is crossed at a certain point, the values in C_2 and C_3 corresponding to the crossed variables are also crossed in the two parents. In this way, an standard crossover operator is applied over the two parts of the chromosomes. This operator performs as follows: a crossover point p is randomly generated in C_1 and the two parents are crossed at the pth variable in C_1. The crossover is developed this way in the three chromosome parts, thereby producing two meaningful descendents.

Let us look at an example in order to clarify the standard crossover application. Consider:

$$\begin{aligned}
C_t = (&l_1, \ldots, l_p, l_{p+1}, \ldots, l_N, a_1, \ldots, a_p, a_{p+1}, \ldots, a_N,\\
&S_1, \ldots, S_p, S_{p+1}, \ldots, S_N, R_1, \ldots, R_p, R_{p+1}, \ldots, R_N)
\end{aligned}$$

$$\begin{aligned}
C'_t = (&l'_1, \ldots, l'_p, l'_{p+1}, \ldots, l'_N, a'_1, \ldots, a'_p, a'_{p+1}, \ldots, a'_N,\\
&S'_1, \ldots, S'_p, S'_{p+1}, \ldots, S'_N, R'_1, \ldots, R'_p, R'_{p+1}, \ldots, R'_N)
\end{aligned}$$

as the individuals to be crossed at point p, the two resulting offsprings are:

$$\begin{aligned}
C_t = (&l_1, \ldots, l_p, l'_{p+1}, \ldots, l'_N, a_1, \ldots, a_p, a'_{p+1}, \ldots, a'_N,\\
&S_1, \ldots, S_p, S'_{p+1}, \ldots, S'_N, R_1, \ldots, R_p, R'_{p+1}, \ldots, R'_N)
\end{aligned}$$

$$\begin{aligned}
C'_t = (&l'_1, \ldots, l'_p, l_{p+1}, \ldots, l_N, a'_1, \ldots, a'_p, a_{p+1}, \ldots, a_N,\\
&S'_1, \ldots, S'_p, S_{p+1}, \ldots, S_N, R'_1, \ldots, R'_p, R_{p+1}, \ldots, R_N)
\end{aligned}$$

Hence, the complete recombination process will allow the GA to follow an adequate exploration–exploitation balance in the genetic search. The expected behavior consists of an initial phase where a high number of standard crossovers and a very small of MMA ones (equal to zero in the great majority of the cases) are developed. The genetic search will perform a wide exploration in this first stage, locating the promising zones and sampling the population individuals at them in several runs. At this moment, a new phase begin, characterized by the increase of the exploitation of these zones and the decrease of the space exploration. Therefore, the number of MMA crossovers rises a lot and the application of the standard crossover decreases. This way to perform an appropriate exploration–exploitation balance in the search was successfully applied in [11,14].

4.2.4.3. Mutation. Three different operators are used, each one of them acting on different chromosome parts. A brief description of them is given below:

• *Mutation on C_1*: The mutation operator selected for C_1 is similar to the one proposed by Thrift in [37]. When a mutation on a gene belonging to the first part of the chromosome is going to be performed, a local modification is developed by changing the number of labels to the immediately upper or lower value (the decision is made at random). When the value to be changed is the lowest (3) or highest one (9), the only possible change is developed.

• *Mutation on the first part of C_2 (parameters a_i) and C_3*: Since both parts are based on a real-coding scheme, Michalewicz's non-uniform mutation operator is employed [34].

If $C_v^t = (c_1, \ldots, c_k, \ldots, c_H)$ is a chromosome and the gene c_k was selected for 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

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

where e is a random number that may 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:

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

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 an uniform search in the initial space when t is small, and a very local one in later stages.

• *Mutation on the second part of C_2 (parameters S_i)*: As the possible values of the parameter S are $\{0, 1\}$, a simple binary mutation is developed.

5. Experimental results

We have considered three different problems for the experiments developed:

• $P1$: An electrical network distribution problem in northern Spain [10]. The system tries to estimate the length of the low voltage line installed in a determined village. The problem has two input variables: the *population of the village* and its *radius*, and one output variable: *the length of the installed line*. We were provided with real data of 495 villages. The training set contains 396 elements and the test set contains 99 elements, randomly selected from the whole sample.

• *P2*: A problem with estimations of minimum maintenance costs which are based on a model of the optimal electrical network for spanish towns [10]. The problem has four input variables: *sum of the lengths of all streets in the town*, *total area of the town*, *area that is occupied by buildings* and *energy supply to the town* and one output variable: *maintenance costs of medium voltage line*. These values are somewhat lower than the real ones, but companies are interested in an estimation of the minimum costs. Of course, real maintenance costs are exactly accounted but a model that relates these costs to any characteristic of simulated towns with the optimal installation is important for the electrical companies. We were provided with data concerning the four characteristics of the towns and their minimum maintenance costs in a sample of 1059 simulated towns. The training set contains 847 elements and the test set contains 212 elements, randomly selected from the whole sample.

• *P3*: The modeling of a tridimensional function [8] defined by:

$$F(x_1,x_2) = 10\frac{x_1 - x_1x_2}{x_1 - 2x_1x_2 + x_2}, \quad x_1,x_2 \in [0,1], \;\; F(x_1,x_2) \in [0,10].$$

Fig. 4 shows its graphical representation. The training set contains 674 uniformly generated elements, and the test set contains 67 randomly generated elements.

The parameters values used in the experiments are presented in Table 1. A quick and simple RB generation algorithm, the Wang and Mendel's rule generation method [1], will be considered. We have chosen this method due to its simplicity and good performance. Our genetic process is independent of the RB generation method, so any of them can be used to derive the RB.

For every benchmark, the best results obtained by our genetic learning process and the other methods considered for comparison purposes are showed in Tables 2–4, which contains the following columns:

- Method (M): Learning process used to obtain the KB:
  - M1: Wang and Mendel's rule generation method [39]. This row shows the results of the FRBS with best MSE over the training set (MSE_{tra}), considering the interval $\{3,\ldots,9\}$ as possible values for the number of labels, with all the variables having the same granularity and uniform fuzzy partitions.

[1] See Appendix A for a brief description of this method.

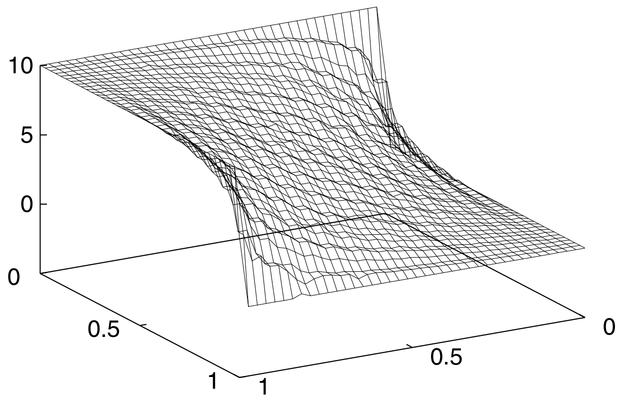

Fig. 4. Graphical representation of the mathematical function.

Table 1
Parameter values considered

Parameter	Value
Population size	100
Crossover probability	0.6
Mutation probability	0.1
Parameter *b* (non-uniform mutation)	5
Parameter *d* (MMA crossover)	0.35
Number of generations	1000

Table 2
Best result for the low voltage line length problem (P1)

M	DB components				#*R*	MSE_{tra}	MSE_{tst}	% tra	% tst
M1	gr.	9	9	9	29	197,613	283,645	–	–
M2	gr.	9	9	9	29	141,022	251,898	28.6	11.2
	gr.	9	9	9	29	144,831	223,734	26.7	21.1
M3	gr.	9	9	9	34	**133,763**	423,639	32.3	−49.3
	gr.	7	7	7	25	152,969	161,245	22.5	43.1
M4	gr.	9	9	9					
	a	2.5	0.9	1.1	37	146,957	180,384	25.6	36.4
	S	1	1	1					
	gr.	7	8	9					
	a	1.1	1.1	0.8	**19**	163,249	**147,062**	17.3	48.1
	S	0	0	0					

- M2: Wang and Mendel's rule generation method + tuning. A genetic tuning process [8] is applied to refine the previous FRBS DB.
- M3: Filipic and Juricic's method [17] (mentioned in Section 2). As this method imposes that the granularity of each variable must be an odd

Table 3
Best result for the optimal electrical network problem (P2)

M	DB components						$\#R$	MSE_{tra}	MSE_{tst}	% tra	% tst
M1	gr.	9	9	9	9	9	130	32,337	33,504	–	–
M2	gr.	9	9	9	9	9	130	13,442	17,585	58.4	47.5
M3	gr.	9	9	9	9	9	133	17,441	21,184	46.1	36.7
	gr.	9	9	9	9	9	139	18,654	19,112	42.3	42.9
M4	gr.	5	6	9	9	9					
	a	0.9	0.3	0.8	0.5	1.1	**87**	**9841**	**10,466**	69.5	68.7
	S	0	0	0	0	0					

Table 4
Best result for the three-dimensional function modeling problem (P3)

M	DB components				#*R*	MSE$_{tra}$	MSE$_{tst}$	% tra	% tst
M1	gr.	9	9	9	81	0.11875	0.03402	–	–
M2	gr.	9	9	9	81	0.03455	0.03156	70.9	7.2
M3	gr.	9	9	9	**80**	0.04427	0.06641	62.7	−95.2
	gr.	9	9	9	**80**	0.04656	0.04979	60.7	−46.3
M4	gr.	9	9	9					
	a	0.3	0.7	0.6	81	**0.01539**	**0.01422**	87.2	58.1
	S	0	0	0					

number, the possible values are $\{3, 5, 7, 9\}$. All the variables will have the same granularity.
  - M4: The genetic learning process proposed in this paper.

Due to the non-deterministic nature of the latter three methods (M2, M3 and M4), four runs have been developed for each of them with different seeds for the pseudo-random number generator. This is the reason why two rows appear for M2, M3 and M4 in the different tables presented. The first row corresponds to the FRBS which obtains the best MSE$_{tra}$ and the other to the FRBS which obtains the best average between the MSE$_{tra}$ and the MSE over the test set (MSE$_{tst}$). Of course, if these two FRBSs are the same, only one row is shown in the table.

- DB components: This column shows the main components of the DB. It has three items:
  - gr: Granularity of the linguistic variables,
  - *a*: Parameter *a*,
  - *S*: Parameter *S*.

Of course, the two latter components only appear in M4.

- #*R*: The number of rules of the FRBS RB.
- MSE$_{tra}$: MSE over the training set.
- MSE$_{tst}$: MSE over the test set.
- % tra: Improvement percentage of the MSE$_{tra}$ obtained respect to the MSE$_{tra}$ obtained using the Wang and Mendel's method (M1).
- % tst: Improvement percentage of the MSE$_{tst}$ obtained respect to the MSE$_{tst}$ obtained using the Wang and Mendel's method (M1).

As can be observed, in the majority of cases our method shows a high accuracy improvement of the final FRBS compared with the usual way to design FRBSs, using the same RB learning method. As regards the method that considers the same KB learning approach (M3), it is also clearly overcome by our learning process. Although some *MSE*$_{tra}$ results of methods M2 and M3 are better than the ones obtained with M4 in Table 2, the MSE$_{tst}$ values associated are very high, that is, the former FRBSs are over-fitted to the training data. Our method has a better generalization capability.

Moreover, the FRBSs obtained by our method normally generates RBs with a lesser number of rules than anyone obtained by the other methods. This result is of a significant importance since more accurate fuzzy models can be obtained with a lesser number of rules on the RB (hence, more interpretable FRBSs) by considering a learning approach different than the usual one.

6. An extension to the proposed method

In this section, we propose a new way to calculate the evaluation function of each chromosome in order to avoid the possible overfitting, thus improving the generalization capability of the final FRBS. To do that, we will lightly penalize FRBSs with a high number of rules to obtain more compact linguistic models. Therefore, once the RB has been generated and its MSE over the training set has been calculated, the fitness function is based on the one proposed in [28]:

$$F_C = \omega_1 \mathrm{MSE} + \omega_2 \#R$$

We consider $\omega_1 = 1$ and ω_2 is calculated taking two values as a base: the MSE of the FRBS obtained with the RB generation method considering the DB with the maximum number of labels per variable and uniform fuzzy partitions ($\mathrm{MSE}_{\max\_lb}$), and the number of rules of that RB ($\#R_{\max\_lb}$):

$$\omega_2 = \alpha \frac{\mathrm{MSE}_{\max\_lb}}{\#R_{\max\_lb}}$$

with α being a weighting percentage.

Table 5 shows the best results obtained by the proposed method with this new evaluation function when solving of the three problems considered in the

Table 5
Best results obtained considering the new evaluation function

B	DB components						#R	$\mathrm{MSE}_{\mathrm{tra}}$	$\mathrm{MSE}_{\mathrm{tst}}$	% tra	% tst
P1	gr.		8	9	8						
	a		1.2	2.3	0.6		29	154,722	218,152	21.7	23.1
	S		0	0	0						
	gr.		7	4	7						
	a		0.2	1.2	1.1		**7**	171,137	155,589	13.4	45.1
	S		0	0	0						
P2	gr.	3	6	9	9	9					
	a	0.1	0.1	2.0	0.5	0.9	**74**	**9238**	**8644**	71.4	74.2
	S	0	0	1	0	0					
P3	gr.		9	9	9						
	a		0.7	0.3	0.5		81	0.01543	**0.01421**	87.1	58.2
	S		0	0	0						

previous section. The parameter values for the GA are the same that in the previous experiments developed (Table 1) and α has been set to three possible values ($\{0.05, 0.1, 0.2\}$). For every benchmark, there are two rows (the FRBS with best MSE_{tra} and the FRBS with best average between the MSE_{tra} and the MSE_{tst}). As in the previous tables of results (Tables 2–4), if these two FRBSs are the same, only one row is showed. All the columns in the table stand for the same aspects that in the tables collected in the previous section, but column "method" (M), that is substituted by column "benchmark" (B). The percentages showed in the last two columns are the improvement percentages of the MSE_{tra} and MSE_{tst} with respect to the corresponding values obtained using the Wang and Mendel's method (method M1 in Tables 2–4).

Notice that the values highlighted in boldface in Table 5 are those corresponding to the cases when better results are obtained using the new fitness function than in the case of the previous experimentation showed in Tables 2. In view of these results, the consideration of the new fitness function leads to obtain, in the most of the cases, a lesser number of rules in the RB and a lesser MSE_{tst} than those obtained with the original evaluation function. On the other hand, as supposed, the values obtained for the MSE_{tra} considering this new evaluation function are, generally, lightly higher than the previous ones. The only exception to the latter is the FRBS designed for problem P2 which overcomes both in MSE_{tra} and MSE_{tst} the best fuzzy models obtained with the remaining techniques.

7. Concluding remarks

This paper has presented the usual ways to automatically obtain the KB of a FRBS, and has analysed some proposals that try to adapt the fuzzy partition contexts. A new genetic process has been proposed to automatically learn the whole KB: the DB being evolved by a GA (adapting the granularity, the contexts using a non-linear scaling function and a possible enlargement of the variable domain), and the RB being generated by a simple rule generation method. It has obtained good results in three different applications.

Our genetic process may be applied to any RB learning method, having in mind its run time since the RB generation method must be run many times within the DB learning process.

Appendix A. The Wang and Mendel learning method

The ad hoc data covering RB generation process proposed by Wang and Mendel in [39] has been widely known because of its simplicity and good performance. The generation of the RB is put into effect by means of the following steps:

1. *Consider a fuzzy partition of the input variable spaces*: It may be obtained from the expert information (if it is available) or by a normalization process. If the latter is the case, perform a fuzzy partition of the input variable spaces dividing each universe of discourse into a number of equal or unequal partitions, select a kind of membership function and assign a fuzzy set to each subspace.
2. *Generate a preliminary linguistic rule set*: This set will be formed by the rule best covering each example (input–output data pair) contained in the input–output data set. The structure of these rules is obtained by taking a specific example, i.e., an $n+1$-dimensional real array (n input and 1 output values), and setting each one of the variables to the linguistic label best covering every array component.
3. *Give an importance degree to each rule*: Let $R_l = IF\ x_1$ is A_1 *and* ... *and* x_n is A_n *THEN* y is B be the linguistic rule generated from the example $e_l = (x_1^l, \ldots, x_n^l, y^l)$. The importance degree associated to it will be obtained as follows:
$$G(R_l) = \mu_{A_1}(x_1^l) \cdots \mu_{A_n}(x_n^l) \cdot \mu_B(y^l)$$
4. *Obtain a final RB from the preliminary fuzzy rule set*: The rule with the highest importance degree is chosen for each combination of antecedents.

References

[1] J.E. Baker, Reducing bias and inefficiency in the selection algorithm, in: Proceedings of the Second International Conference on Genetic Algorithms (ICGA'87), Hillsdale, 1987, pp. 14–21.

[2] A. Bardossy, L. Duckstein, Fuzzy Rule-based Modelling with Application to Geophysical, Biological and Engineering System, CRC Press, Boca Raton, 1995.

[3] A. Bastian, How to handle the flexibility of linguistic variables with applications, International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 2 (4) (1994) 463–484.

[4] J.M. Benítez, J.L. Castro, I. Requena, FRUTSA: Fuzzy rule tuning by simulated annealing, International Journal of Approximate Reasoning (to appear).

[5] P.P. Bonissone, P.S. Khedkar, Y.T. Chen, Genetic algorithms for automated tuning of fuzzy controllers, a transportation application, in: Proceedings of the Fifth IEEE International Conference on Fuzzy Systems (FUZZ-IEEE'96), New Orleans, 1996, pp. 674–680.

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

[7] J. Casillas, O. Cordón, F. Herrera, A methodology to improve ad hoc data-driven linguistic rule learning methods by inducing cooperation among rules, Technical Report #DECSAI-000101, University of Granada, Spain, February, 2000.

[8] O. Cordón, F. Herrera, A three-stage evolutionary process for learning descriptive and approximative fuzzy logic controller knowledge bases from examples, International Journal of Approximate Reasoning 17 (4) (1997) 369–407.

[9] O. Cordón, F. Herrera, F. Hoffmann, L. Magdalena, Genetic fuzzy systems evolutionary tuning and learning of fuzzy knowledge bases, World Scientific (2001) (in press).
[10] O. Cordón, F. Herrera, L. Sánchez, Solving electrical distribution problems using hybrid evolutionary data analysis techniques, Applied Intelligence 10 (1999) 5–24.
[11] O. Cordón, F. Herrera, Hybridizing genetic algorithms with sharing scheme and evolution strategies for designing approximate fuzzy rule-based systems, Fuzzy Sets and Systems 118 (2) (2000) 235–255.
[12] O. Cordón, F. Herrera, P. Villar, Analysis and guidelines to obtain a good uniform fuzzy partition granularity for fuzzy rule-based systems using simulated annealing, International Journal of Approximate Reasoning 25 (3) (2000) 187–216.
[13] O. Cordón, F. Herrera, A proposal for improving the accuracy of linguistic modeling, IEEE Transactions on Fuzzy Systems 8 (3) (2000) 335–344.
[14] O. Cordón, F. Herrera, P. Villar, Generating the knowledge base of a fuzzy rule-based system by the genetic learning of the data base, IEEE Transactions on Fuzzy Systems (2001) (to appear).
[15] O. Cordón, F. Herrera, L. Magdalena, P. Villar, A genetic learning process for the scaling factors, granularity and context of the fuzzy rule-based system data base, in: Proceedings of the Eighth International Conference on Information Processing and Management of Uncertainty in Knowledge-based Systems (IPMU'00), Madrid, 2000, pp. 430–437.
[16] M. Delgado, A.F. Gómez-Skarmeta, F. Martín, A fuzzy clustering based rapid prototyping for fuzzy rule-based modelling, IEEE Transactions on Fuzzy Systems 5 (2) (1997) 223–233.
[17] B. Filipic, D. Juricic, A genetic algorithm to support learning fuzzy control rules from examples, in: F. Herrera, J.L. Verdegay (Eds.), Genetic Algorithms and Soft Computing, Physica, Wurzburg, 1996, pp. 403–418.
[18] P. Glorennec, Constrained optimization of FIS using an evolutionary method, in: F. Herrera, J.L. Verdegay (Eds.), Genetic Algorithms and Soft Computing, Physica, Wurzburg, 1996, pp. 349–368.
[19] D.E. Goldberg, Genetic Algorithms in Search, Optimization & Machine Learning, Addison Wesley, Reading, MA, 1989.
[20] A. González, R. Pérez, SLAVE: A genetic learning system based on the iterative approach, IEEE Transactions on Fuzzy Systems 7 (2) (1999) 176–191.
[21] R. Gudwin, F. Gomide, W. Pedrycz, Context adaptation in fuzzy processing and genetic algorithms, International Journal of Intelligent Systems 13 (1998) 929–948.
[22] F. Herrera, M. Lozano, J.L. Verdegay, Tuning fuzzy controllers by genetic algorithms, International Journal of Approximate Reasoning 12 (1995) 299–315.
[23] F. Herrera, J.L. Verdegay (Eds.), Genetic Algorithms and Soft Computing, Physica, Wurzburg, 1996.
[24] F. Herrera, M. Lozano, J.L. Verdegay, Fuzzy connectives based crossover operators to model genetic algorithms population diversity, Fuzzy Sets and Systems 92 (1) (1997) 21–30.
[25] F. Hoffmann, G. Pfister, Evolutionary design of a fuzzy knowledge base for a mobile robot, International Journal of Approximate Reasoning 17 (4) (1997) 447–469.
[26] W.R. Hwang, W.E. Thompson, Design of fuzzy logic controllers using genetic algorithms, in: Proceedings of the Third IEEE International Conference on Fuzzy Systems (FUZZ-IEEE'94), Orlando, 1994, pp. 1383–1388.
[27] H. Ishibuchi, K. Nozaki, H. Tanaka, Y. Hosaka, M. Matsuda, Empirical study on learning in fuzzy systems by rice test analysis, Fuzzy Sets and Systems 64 (1994) 129–144.
[28] H. Ishibuchi, K. Nozaki, N. Yamamoto, H. Tanaka, Selecting fuzzy IF–THEN rules for classification problems using genetic algorithms, IEEE Transactions on Fuzzy Systems 3 (3) (1995) 260–270.

[29] H. Ishibuchi, T. Murata, A genetic-algorithm-based fuzzy partition method for pattern classification problems, in: F. Herrera, J.L. Verdegay (Eds.), Genetic Algorithms and Soft Computing, Physica, Wurzburg, 1996, pp. 555–578.

[30] C. Karr, Applying genetics to fuzzy logic, AI Expert 6 (3) (1991) 38–43.

[31] M.A. Lee, H. Takagi, Embedding apriori knowledge into an integrated fuzzy system design method based on genetic algorithms, in: Proceedings of the Fifth International Fuzzy Systems Association World Congress (IFSA'93), Seoul, 1993, pp. 1293–1296.

[32] L. Magdalena, Adapting the gain of an FLC with genetic algorithms, International Journal of Approximate Reasoning 17 (4) (1997) 327–349.

[33] L. Magdalena, J.R. Velasco, Evolutionary based learning of fuzzy controllers, in: W. Pedrycz (Ed.), Fuzzy Evolutionary Computation, Kluwer Academic Publishers, Dordrecht, 1997, pp. 249–268.

[34] Z. Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs, Springer, Berlin, 1996.

[35] Y. Oh, D.J. Park, Self-tuning fuzzy controller with variable universe of discourse, in: Proceedings of the IEEE International Conference on System, Man and Cybernetics, Vancouver, 1995, pp. 2628–2632.

[36] W. Pedrycz, R. Gudwin, F. Gomide, Nonlinear context adaptation in the calibration of fuzzy sets, Fuzzy Sets and Systems 88 (1997) 91–97.

[37] P. Thrift, Fuzzy logic synthesis with genetic algorithms, in: Proceedings of the Fourth International Conference on Genetic Algorithms (ICGA'91), 1991, pp. 509–513.

[38] J.R. Velasco, S. López, L. Magdalena, Genetic fuzzy clustering for the definition of fuzzy sets, in: Proceedings of the Sixth IEEE International Conference on Fuzzy Systems (FUZZ-IEEE'97), Barcelona, 1997, pp. 1665–1670.

[39] L.X. Wang, J.M. Mendel, Generating fuzzy rules by learning from examples, IEEE Transactions on Systems, Man, and Cybernetics 22 (1992) 1414–1427.

[40] L. Zheng, A practical guide to tune proportional and integral (PI) like fuzzy controllers, in: Proceedings of the First IEEE International Conference on Fuzzy Systems (FUZZ-IEEE'92), San Diego, 1992, pp. 633–640.