Generating the Knowledge Base of a Fuzzy Rule-Based System by the Genetic Learning of the Data Base

Oscar Cordón, Francisco Herrera, and Pedro Villar

***Abstract*—A new method is proposed to automatically learn the knowledge base (KB) by finding an appropiate data base (DB) by means of a genetic algorithm while using a simple generation method to derive the rule base (RB). Our genetic process learns the number of linguistic terms per variable and the membership function parameters that define their semantics, while a rule base generation method learns the number of rules and their composition.**

***Index Terms*—Fuzzy rule-based systems, data base, learning, genetic algorithms.**

I. Introduction

THE generation of the knowledge base (KB) of a fuzzy rule-based system (FRBS) presents several difficulties because the KB depends on the concrete application, and 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) [3], [7], [13], [18], [22]–[24], [33], [39]. This operation mode makes the DB have a significant influence on the FRBS performance. In fact, some studies have shown 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 for improving 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], [19], [27]. 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 contrast to this, *a posteriori* DB learning, there are some approaches that learn the different DB components *a priori* [9], [12], [15], [16], [26], [32], [37].

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.

Manuscript received August 10, 2000; revised November 2, 2000. This work was supported by CICYT under PB98-1319.

O. Cordón is with the Department of Computer Science and Artificial Intelligence, University of Granada, 18071 Granada, Spain (e-mail: ocordon@decsai.ugr.es).

F. Herrera is with the Department of Computer Science and Artificial Intelligence, University of Granada, 18071 Granada, Spain (e-mail: herrera@decsai.ugr.es).

P. Villar is with the Department of Computing, University of Vigo, ESEI, 32004 Ourense, Spain (e-mail: pvillar@uvigo.es).

Publisher Item Identifier S 1063-6706(01)04530-1.

1) A genetic learning process for the DB that allows us to define:
   a) the number of labels for each linguistic variable;
   b) the definition points of each fuzzy membership function.
2) 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.

In order to do that, this paper is organized as follows. Section II shows some preliminaries about the KB learning in FRBSs. In Section III, our method is presented, describing the components of the genetic process coding of the solutions, initial population, evaluation function, and genetic operators. In Section IV, some experimental results are shown. Finally, in Section V, some conclusions are pointed out.

II. Automatic Learning of the KB

Two problems arise when generating the KB of a FRBS:

1) 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;
2) the RB derivation, involving the determination of the number of rules and of the specific composition of each one of them (i.e., of the specific labels associated to each linguistic variable).

The next two sections describe, respectively, the more usual ways to learn the KB and the approach followed by the method proposed in this paper.

A. Usual Solutions for the KB Learning

a) RB learning using a predefined DB: Most of the approaches proposed to automatically learn the KB from numerical information have focused on this kind of KB learning. The usual way to define this DB involves choosing a number of linguistic terms for each linguistic variable (an odd number between three and nine, which is normally the same for all the variables) and setting the values of the system parameters by a 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 [3], [13], [24], [39], least square methods [3], simulated annealing,

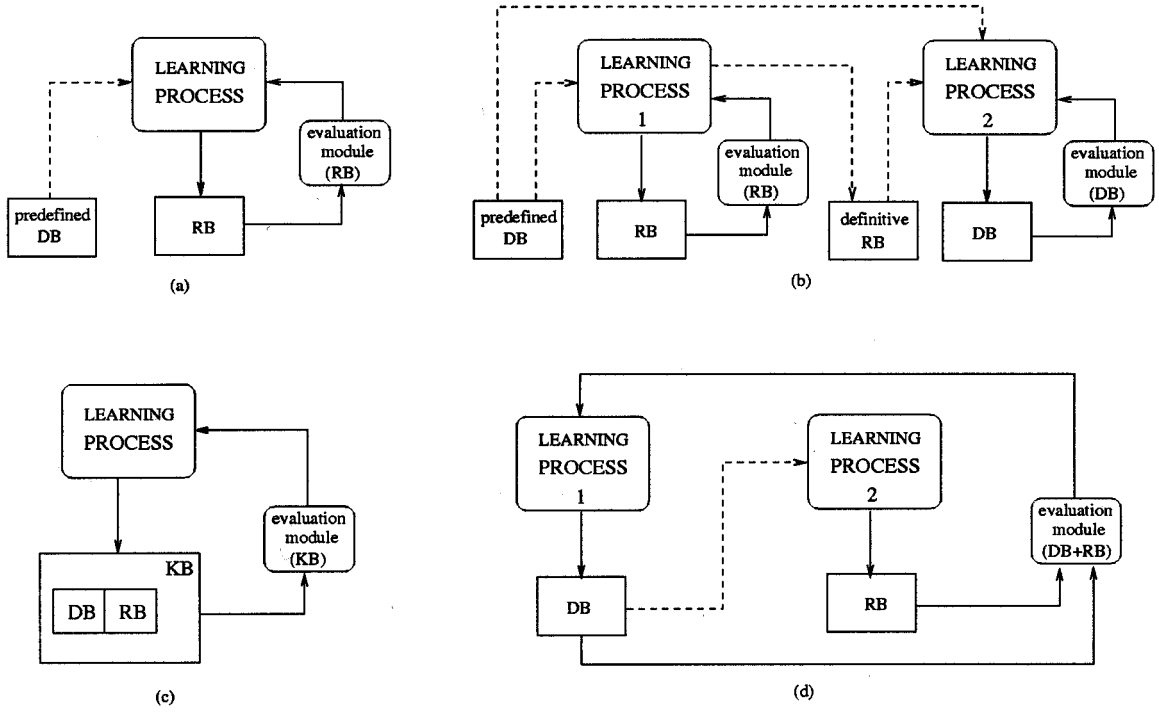

Fig. 1. Graphical representation of the different KB learning approaches.

[7] and GAs [18], [22], [23], [33]–[36]. 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 for variable) in the FRBS performance is studied, showing that using an appropiate number of terms for each linguistic variable, the FRBS accuracy can be improved with no need to use a complex RB learning method.

b) Tuning membership functions: 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 begining of the designing process. A graphical representation of this kind of learning is shown in Fig. 1(b). For some examples of tuning methods based on simulated annealing and GAs, refer to [4], [5], [8], [19], [27].

c) Learning the KB components simultaneously: Other approaches try to learn the two components of the KB at the same time [Fig. 1(c)]. Working in this way, 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], [28], [29], [31], [38].

There is another way to generate the whole KB that considers two different processes for deriving both components (DB and RB), based on the DB learning *a priori*. This approach will be described in the next section.

B. KB Derivation by Learning the DB A Priori

In this type of KB learning, 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 [Fig. 1(d)]. We should note that this operation mode involves a partitioning of the KB learning problem. While the learning processes belonging to the third family analyzed in the previous section [Fig. 1(c)] look for solutions in a complex global search space (DB + RB), the processes in 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.

The next processes are examples of the KB learning approach of Fig. 1(d).

1) The work proposed in [12] uses simulated annealing to learn an appropiate fuzzy partition granularity for each variable, maintaining the usual uniform distribution for the fuzzy sets in each universe of discourse. Each time a different granularity definition is generated, a specific RB generation method is applied from that DB definition to obtain the whole KB. Some results for different RB learning methods are shown.

2) In [16], a genetic process that obtains a KB for simplified TSK rules is proposed. The method is composed of an evolutionary process to learn the DB and by a gradient descent method to generate the singleton consequents for all the possible rules. This work imposes several constraints over the DB in order to preserve the readability of the final FRBS.
3) The method proposed in [26] deals with a GA to design a FRBS for pattern classification problems. The coding scheme generates binary chromosomes of fixed length, with a segment per variable. Each segment has a predefined length that determines its maximum granularity. In the chromosome, a one indicates the peak value of a triangular membership function and both extremes of the neighbor membership functions. This representation preserves the same constraints of the previous method. Finally, a data covering algorithm is run to obtain the class associated to each antecedent combination.
4) The approach proposed in [15] uses a GA to learn the DB and the Wang and Mendel's rule generation method [39] to derive the RB. The method always chooses an odd number for the fuzzy partition granularity of each variable and these values remain unchanged for the whole learning process. This method only codifies two points for each label (center and width) so all the triangles are isoceles.

Our method will belong to this group. We use a genetic process to learn the DB and a simple ad hoc data-driven algorithm to derive the RB as in [15]. In our approach we also include the learning of the fuzzy partition granularity per variable.

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]. Also, with another one belonging to the same family, the last method mentioned on the previous paragraph [15].

III. Learning the DB of a FRBS Using Genetic Algorithms

GAs [17], [30] 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 undergoes evolution. 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 FRBS are contained in [9], [20]. GAs have been basically applied to the learning of the different components of the KB (RB in isolation [18], [22], [23], [33]–[36], or both DB and RB [6], [11], [28], [29], [31], [38]) and to adjust a preliminary DB definition maintaining fixed an RB previously derived [5], [8], [19], [27].

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 the better solutions more quickly if an adequate initial gene pool is chosen.

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.

1) A genetic learning process for the DB that allows us to define:
   a) the number of labels for each linguistic variable;
   b) the definition points of each fuzzy membership function.

   Triangular membership functions are considered due to their simplicity.
2) A quick ad hoc data-driven method 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.

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, obtaining a complete KB and then the accuracy of the FRBS obtained on a training data set is measured.

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

A. Encoding the DB

The two components of the DB to be encoded are the number of linguistic terms for variable (granularity) and the membership functions that define their semantics. Therefore, each chromosome will be composed of two parts.

1) Number of labels (C_1). For a system with N variables (including input and output variables), the number of labels per variable is stored into an integer array of length N. In this contribution, the possible values considered are the set $\{3,\ldots,9\}$.
2) Membership functions (C_2). As we deal with triangular functions, a real number array of $N \times 9 \times 3$ positions is used to store the membership functions (N variables, with nine as maximum number of labels for each variable, and each label defined by three real values). Of course, if a chromosome does not have the maximum number of labels in one variable, the space reserved for the values of these labels is ignored in the evaluation process.

If l_i is the granularity of variable i, $P^1_{ij}, P^2_{ij}, P^3_{ij}$ are the definition points of the label j of the variable i, and C_{2i} is the information about the fuzzy partition of variable i in C_2 (all its labels), a graphical representation of the chromosome is shown next

$$
\begin{aligned}
C_1 &= (l_1, l_2, \ldots, l_N)\\
C_{2i} &= \left(P^1_{i1}, P^2_{i1}, P^3_{i1}, \ldots, P^1_{il_i}, P^2_{il_i}, P^3_{il_i}\right)\\
C_2 &= (C_{21}, C_{22}, \ldots, C_{2N})\\
C &= C_1 C_2.
\end{aligned}
$$

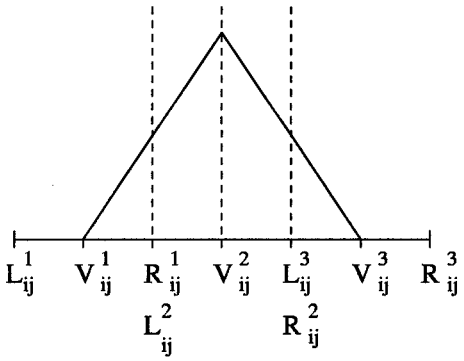

Fig. 2. Variation intervals for the membership function definition points.

With the aim of avoiding incoherent situations, such as the peak value of a label being greater than the peak value of the next label, a variation interval is defined for each one of the membership function definition points. These intervals are calculated taking uniform fuzzy partitions for each variable as a base. The variation intervals of each definition point of the label j of the variable i, $(P_{ij}^1, P_{ij}^2, P_{ij}^3)$, are defined next

$$P_{ij}^1 \in [L_{ij}^1, R_{ij}^1] = \left[V_{ij}^1 - \frac{V_{ij}^2 - V_{ij}^1}{2}, V_{ij}^1 + \frac{V_{ij}^2 - V_{ij}^1}{2}\right]$$

$$P_{ij}^2 \in [L_{ij}^2, R_{ij}^2] = \left[V_{ij}^2 - \frac{V_{ij}^2 - V_{ij}^1}{2}, V_{ij}^2 + \frac{V_{ij}^3 - V_{ij}^2}{2}\right]$$

$$P_{ij}^3 \in [L_{ij}^3, R_{ij}^3] = \left[V_{ij}^3 - \frac{V_{ij}^3 - V_{ij}^2}{2}, V_{ij}^3 + \frac{V_{ij}^3 - V_{ij}^2}{2}\right].$$

The graphical representation of these intervals is shown in Fig. 2.

Of course, when the number of labels of a variable changes by the action of a genetic operator, a new uniform fuzzy partition for that new granularity is built and the new variation intervals are calculated.

B. Initial Gene Pool

The initial population is composed of four groups with the same number of individuals. The generation of the initial gene pool is described next.

1) In the first part, each chromosome will have the same number of labels in all its variables and the membership functions are uniformly distributed across the variable working range.
2) In the second group, each chromosome can have a different granularity per variable (different values in C_1) and the membership functions are uniformly distributed as in the first part.
3) In the third part, each chromosome will have the same number of labels in all its variables. Then a uniform fuzzy partition is built for each variable as in the first part and the variation intervals of all the definition points are calculated. Finally, a value for all the definition points is randomly chosen from the correspondent variation interval.
4) In the last group, each chromosome can have different number of labels per variable as in second part and the membership functions are calculated in the same way as in the third part, a random value in the variation interval.

The aim of generating the initial population in these four different groups is to sample it to achieve an appropiate diversity. Although GAs have proven to be robust and get good solutions starting from randomly generated populations (group four), a quick convergence can be obtained using the knowledge available about the problem to sample the population in a biased way.

C. Evaluating the Chromosome

There are two steps that must be done to evaluate each chromosome.

1) Run the RB generation method using the DB definition contained in the chromosome, obtaining a complete KB.
2) Calculate the mean square error (MSE) over the training set using the KB obtained (genetically derived DB+RB). In order to improve the generalization capability of the final FRBS, we will lightly penalize the FRBSs with a high number of rules (NR). Therefore, the fitness function is based on the one proposed in [25]

$$F_C = \omega_1 \cdot \mathrm{MSE} + \omega_2 \cdot \mathrm{NR}.$$

In this contribution, 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, the DB with the maximum number of labels per variable and uniform fuzzy partitions $(\mathrm{MSE}_{\max\_\mathrm{lb}})$, and the number of rules of that RB $(\mathrm{NR}_{\max\_\mathrm{lb}})$

$$\omega_2 = \alpha \cdot \frac{\mathrm{MSE}_{\max\_\mathrm{lb}}}{\mathrm{NR}_{\max\_\mathrm{lb}}}.$$

With α being a weighting percentage.

D. Genetic Operators

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 two chromosome parts, operators working cooperatively in C_1 and C_2 are required in order to make best use of the representation used.

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

1) Selection: The selection probability calculation follows *linear ranking* [1]. Chromosomes are sorted in order of raw fitness and then the selection probability of each chromosome $p_s(\mathrm{C}_i)$ is computed according to its rank $\mathrm{rank}(\mathrm{C}_i)$ [with $\mathrm{rank}(\mathrm{C}_{\mathrm{best}}) = 1$] by using the following nonincreasing assignment function:

$$p_s(\mathrm{C}_i) = \frac{1}{\mathrm{NC}} \cdot \left(\eta_{\max} - (\eta_{\max} - \eta_{\min}) \cdot \frac{\mathrm{rank}(\mathrm{C}_i) - 1}{\mathrm{NC} - 1}\right)$$

where NC is the population size and $\eta_{\min} \in [0, 1]$ specifies the expected number of copies for the worst chromosome (the best one has $\eta_{\max} = 2 - \eta_{\min}$ expected copies). In the experiments $\eta_{\min} = 0.75$.

Linear ranking is performed along with *stochastic universal sampling* [2]. This procedure guarantees that the number of copies of any chromosome is bounded by the floor and by the ceiling of its expected number of copies.

Our reproduction operator includes the elitist selection.

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

1) *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 exploited. This task is developed by applying the max-min-arithmetical (MMA) crossover operator in C_2 and obviously, by maintaining the parent C_1 values in the offspring. This crossover operator is proposed in [21] and works in the way shown below.

   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, the following four offspring are generated:

$$\begin{aligned} C_1^{t+1} &= \left(c_{11}^{t+1}, \ldots, c_{1k}^{t+1}, \ldots, c_{1H}^{t+1}\right) \\ c_{1k}^{t+1} &= dc_k + (1-d)c'_k \\ C_2^{t+1} &= \left(c_{21}^{t+1}, \ldots, c_{2k}^{t+1}, \ldots, c_{2H}^{t+1}\right) \\ c_{2k}^{t+1} &= dc'_k + (1-d)c_k \\ C_3^{t+1} &= \left(c_{31}^{t+1}, \ldots, c_{3k}^{t+1}, \ldots, c_{3H}^{t+1}\right) \\ c_{3k}^{t+1} &= \min\{c_k, c'_k\} \\ C_4^{t+1} &= \left(c_{41}^{t+1}, \ldots, c_{4k}^{t+1}, \ldots, c_{4H}^{t+1}\right) \\ c_{4k}^{t+1} &= \max\{c_k, c'_k\}. \end{aligned}$$

   This operator uses a parameter d that is either a constant, or a variable whose value depends on the age of the population. We use $d = 0.35$, value taking from [21]. The resulting descendents are the two best of the four aforesaid offspring.

2) *Crossover when the parents encode different granularity levels:* This second case highly recommends the use of the information encoded by the parents for exploring 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 corresponding to the crossed variables are also crossed in the two parents. In this way, a 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 p-th variable in C_1. The crossover is developed this way in the two chromosome parts, C_1 and C_2, thereby, producing two meaningful descendents.

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

$$\begin{aligned} C_t &= (l_1, \ldots, l_p, l_{p+1}, \ldots, l_N, C_{21}, \ldots, C_{2p} \\ &\quad C_{2p+1}, \ldots, C_{2N}) \\ C'_t &= (l'_1, \ldots, l'_p, l'_{p+1}, \ldots, l'_N, C'_{21}, \ldots, C'_{2p}, \\ &\quad C'_{2p+1}, \ldots, C'_{2N}) \end{aligned}$$

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

$$\begin{aligned} C_t &= (l_1, \ldots, l_p, l'_{p+1}, \ldots, l'_N, C_{21}, \ldots, C_{2p}, \\ &\quad C'_{2p+1}, \ldots, C'_{2N}) \\ C'_t &= (l'_1, \ldots, l'_p, l_{p+1}, \ldots, l_N, C'_{21}, \ldots, C'_{2p}, \\ &\quad C_{2p+1}, \ldots, C_{2N}). \end{aligned}$$

Hence, the complete recombination process will allow the GA to follow an adequate exploration and exploitation rate in the genetic search. The expected behavior consists of an initial phase where a high number of standard crossovers and a very small number 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 in several runs. At this moment a new phase begins, characterized by the increase exploitation of these zones and the decrease of the space exploration. Therefore, the number of MMA crossovers rise a lot and the application of the standard crossover decreases. This way, to perform an appropiate exploration- exploitation balance in the search was succesfully applied in [11].

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

1) *Mutation on C_1*: The mutation operator selected for C_1 is similar to the one proposed by Thrift in [33]. 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 of the variable 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. Once a new value is selected, a uniform fuzzy partition for this variable is stored in its corresponding zone of C_2.

2) *Mutation on C_2*: Since both parts are based on a real-coding scheme, Michalewicz's nonuniform mutation operator is employed [30].

   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

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

   with t being the current generation, e being a random number that may have a value of zero or one, and the function $\triangle(t, y)$ returns a value in the range $[0, y]$ such that the probability of $\triangle(t, y)$ being close to 0 increases as t increases

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

   with r being a random number in the interval $[0, 1]$, T being the maximum number of generations and b being 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 a very local one in later stages. In the experiments, we consider $b = 5$, that is the value proposed for the author in [30].

E. Restart

Due to the large size of the search space tackled and to the special coding considered with two different information levels, the genetic search can get trapped in local optima. In some preliminary experiments the population did converge so fast to a determinated granularity level (C_1) and other promising zones were not explored. This process is known as *genetic drift* and it is usual when working with multimodal search spaces [17]. To avoid this problem, a restart operator is used when the difference between the fitness of the best individual and the average fitness of the population is less than a 5% of the first.

A usual way to proceed involves copying the best individual in the new population and creating the remaining chromosomes, starting from that individual, by randomly changing the 70% of the genes and maintaining fixed the remainder 30% [14]. In this case, as it has not the same effect to modify a value in C_1 or in C_2, we will first decide over which part a change will be developed. If it is a value in C_2, a new value is randomly chosen from the variation interval of this point, while if it is a value in C_1, a new number of labels is randomly selected and a uniform fuzzy partition for this variable is stored in its corresponding zone of C_2.

IV. Experimental Results

We have considered three different problems for the experiments developed.

1) **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*; 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.
2) **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.
3) **P3**: The modeling of a tridimensional mathematical function defined by.

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

Fig. 3 shows its graphical representation. We were provided with data of 741 points. The training set contains **674** elements and the test set contains **67** elements, randomly selected from the whole sample.

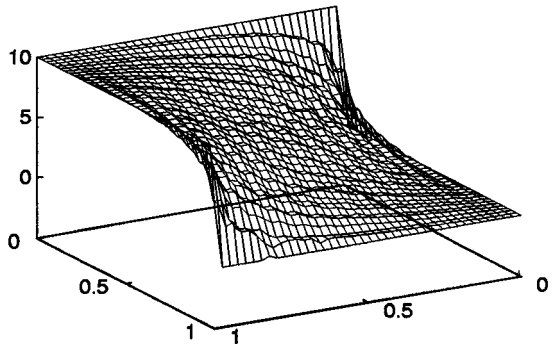

Fig. 3. Graphical representation of the mathematical function.

TABLE I
Genetic Algorithm Parameter Values

Parameter	Value
Population size	64
Crossover probability	0.6
Mutation probability	0.1
Parameter b (non-uniform mutation)	5
Parameter d (MMA crossover)	0.35
Number of generations	1000
Parameter α (weight ω_2 in fitness function)	$\{0.05, 0.1, 0.2\}$

The genetic parameters used in the expermients are presented in Table I. 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 its simplicity and to compare our learning process with the one presented in [15] that used it to derive the RB. Our genetic process is independent of the RB generation method, so anyone 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 shown in Tables II, III, and IV, which contain the following columns.

1) Method: learning process used to obtain the KB.
   a) WM: Wang and Mendel's rule generation method [39]. This line shows the results of the FRBS with best MSE over the training set $(\mathrm{MSE}_{\mathrm{tra}})$ considering the interval $\{3, \ldots, 9\}$ as possible values for the number

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

TABLE II
BEST RESULTS FOR THE LOW VOLTAGE LINE LENGTH PROBLEM (P1)

method	granularity	# rul.	MSE_{tra}	MSE_{tst}	% tra	% tst
WM	9 9 9	29	197613.4	283645.5	—	—
WM+T	9 9 9	29	141022.4	251898.4	28.6%	11.2%
	9 9 9	29	144831.2	223734.8	26.7%	21.1%
FJ	9 9 9	34	133763.6	423639.8	32.3%	-49.3%
	7 7 7	25	152969.3	161245.4	22.5%	43.1%
Gr+MF	7 9 9	28	146741.1	191360.2	25.7%	32.5%
	7 7 7	25	149832.4	152723.7	24.1%	46.1%

TABLE III
BEST RESULTS FOR THE OPTIMAL ELECTRICAL NETWORK PROBLEM (P2)

method	granularity	# rul.	MSE_{tra}	MSE_{tst}	% tra	% tst
WM	9 9 9 9 9	130	32337.4	33504.9	—	—
WM+T	9 9 9 9 9	130	13442.5	17585.7	58.4%	47.5%
FJ	9 9 9 9 9	133	17441.1	21184.6	46.1%	36.7%
	9 9 9 9 9	139	18654.5	19112.8	42.3%	42.9%
Gr+MF	4 3 9 9 9	96	9163.5	11121.3	71.6%	66.8%
	3 3 9 7 9	68	9987.7	10414.1	69.1%	68.9%

TABLE IV
BEST RESULTS FOR THE MATHEMATICAL FUNCTION PROBLEM (P3)

method	granularity	# rul.	MSE_{tra}	MSE_{tst}	% tra	% tst
WM	9 9 9	81	0.70125	0.67637	—	—
WM+T	9 9 9	81	0.13677	0.16381	80.5%	75.7%
FJ	9 9 9	80	0.04427	0.06641	93.6%	90.1%
	9 9 9	80	0.04656	0.04979	93.3%	92.6%
Gr+MF	7 9 9	80	0.02407	0.02586	96.5%	96.1%
	7 7 7	80	0.02537	0.01712	96.3%	97.4%

of labels, with all the variables having the same granularity and uniform fuzzy partitions.

b) WM + T: Wang and Mendel's rule generation method + tuning. A genetic tuning process [8] is applied to the previous FRBS DB.

c) FJ: Filipic and Juricic's method [15]. This method was briefly described in Section II. As the granularity of each variable must be an odd number, the posible values are $\{3, 5, 7, 9\}$. All the variables will have the same granularity.

d) Gr + MF: The genetic learning process proposed in this paper (granularity + membership functions).

Due to the non deterministic nature of the last three methods (WM + T, FJ and Gr + MF), four runs have been developed for them. This is the reason why two lines appear in the tables for WM+T, FJ and Gr+MF. The first one shows the FRBS with best $\mathrm{MSE_{tra}}$ and the other shows the FRBS with best average between the $\mathrm{MSE_{tra}}$ and the MSE over the test set ($\mathrm{MSE_{tst}}$). If these two FRBSs are equal, only one line appears in the table.

2) Granularity: the number of labels for each problem variable.
3) Number rule: the number of rules of the FRBS RB.
4) $\mathrm{MSE_{tra}}$: MSE over the training set.
5) $\mathrm{MSE_{tst}}$: MSE over the test set.
6) %tra: improvement percentage of the $\mathrm{MSE_{tra}}$ obtained respect to the $\mathrm{MSE_{tra}}$ obtained using the Wang and Mendel's method (WM).
7) %tst: improvement percentage of the $\mathrm{MSE_{tst}}$ obtained respect to the $\mathrm{MSE_{tst}}$ obtained using the Wang and Mendel's method (WM).

As it 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 (FJ), there is also a significant improvement. Although some results of methods WM + T and FJ in $\mathrm{MSE_{tra}}$ are better than the ones obtained with Gr + MF (in Table II), their $\mathrm{MSE_{tst}}$ are very high, that is, the former FRBSs are over-fitted to the training data. The method Gr + MF has a better generalization capability.

Moreover, the FRBSs obtained by our method always generates RBs with a lesser (or at most, equal) number of rules than any obtained by the other methods. This result is of significant importance since more accurate fuzzy models can be obtained with a lesser number of rules on the RB by considering a learning approach different than the usual approach.

V. CONCLUDING REMARKS

This paper has presented the usual ways to automatically design the KB of a FRBS and has analyzed results that consider the DB definition task in the FRBS design process. A new genetic process has been proposed to automatically design the whole KB; the DB being evolved by a GA and the RB being generated by a simple rule generation method.

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.

As a possible extension of the presented method, an input selection can be developed. This process could be previous to the learning of the DB or integrated in the GA.

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 expert information (if 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 one 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

one 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 $R_l = \text{IF} x_1$ 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:

$$G(R_l) = \mu_{A_1}\left(x_1^l\right) \cdot \ldots \cdot \mu_{A_n}\left(x_n^l\right) \cdot \mu_B\left(y^l\right).$$

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.

ACKNOWLEDGMENT

The authors would like to thank the anonymous referees for their constructive and useful comments.

REFERENCES

[1] J. E. Baker, "Adaptive selection methods for genetic algorithms," in *Proc. First Int. Conf. Genetic Algor.*, J. J. Grefenstette, Ed., Hillsdale, 1985, pp. 101–111.

[2] ——, "Reducing bias and inefficiency in the selection algoritmh," in *Proc. Sec. Int. Conf. Genetic Algor. (ICGA'87)*, Hillsdale, 1987, pp. 14–21.

[3] A. Bardossy and L. Duckstein, *Fuzzy Rule-Based Modeling With Application to Geophysical, Biological and Engineering Systems*. Boca Raton, FL: CRC , 1995.

[4] J. M. Benítez, J. L. Castro, and I. Requena, FRUTSA: Fuzzy rule tuning by simulated annealing, in Int. J. Approx. Reason.. To appear.

[5] P. P. Bonissone, P. S. Khedkar, and Y. T. Chen, "Genetic algorithms for automated tuning of fuzzy controllers, a transportation aplication," *Proc. Fifth IEEE Int. Conf. Fuzzy Syst. (FUZZ-IEEE'96)*, pp. 674–680, 1996.

[6] B. Carse, T. C. Fogarty, and A. Munro, "Evolving fuzzy rule based controllers using genetic algorithms," *Fuzzy Sets Syst.*, vol. 80, pp. 273–293, 1996.

[7] J. Casillas, O. Cordón, and F. Herrera, "A Methodology to Improve ad hoc Data-Driven Linguistic Rule Learning Methods by Inducing Cooperation Among Rules," University of Granada, Spain, Technical Report #DECSAI-000 101, Feb. 2000.

[8] O. Cordón and F. Herrera, "A three-stage evolutionary process for learning descriptive and approximative fuzzy logic controller knowledge bases from examples," *Int. J. Approx. Reason.*, vol. 17, no. 4, pp. 369–407, 1997.

[9] O. Cordón, F. Herrera, F. Hoffmann, and L. Magdalena, Genetic Fuzzy Systems. Evolutionary Tuning and Learning of Fuzzy Knowledge Bases, World Scientific, 2001. In press.

[10] O. Cordón, F. Herrera, and L. Sánchez, "Solving electrical distribution problems using hybrid evolutionary data analysis techniques," *Appl. Intell.*, vol. 10, pp. 5–24, 1999.

[11] O. Cordón and F. Herrera, "Hybridizing genetic algorithms with sharing scheme and evolution strategies for designing approximate fuzzy rule-based systems," *Fuzzy Sets Syst.*, vol. 118, no. 2, pp. 235–255, 2001.

[12] O. Cordón, F. Herrera, and P. Villar, "Analysis and guidelines to obtain a good uniform fuzzy partition granularity for fuzzy rule-based systems using simulated annealing," *Int. J. Approx. Reason.*, vol. 25, no. 3, pp. 187–215, 2000.

[13] O. Cordón and F. Herrera, "A proposal for improving the accuracy of linguistic modeling," *IEEE Trans. Fuzzy Syst.*, vol. 8, pp. 335–344, , 2000.

[14] L. J. Eshelman, "The CHC adaptative search algorithm: How to have safe search when engaging in non traditional genetic recombination," in *Foundations of Genetic Algorithms*, G. J. E. Rawlins, Ed. San Mateo, CA: Morgan Kaufmann, 1991, pp. 265–283.

[15] B. Filipic and D. Juricic, "A genetic algorithm to support learning fuzzy control rules from examples," in *Genetic Algorithms and Soft Computing*, F. Herrera and J. L. Verdegay, Eds: Physica-Verlag, 1996, pp. 403–418.

[16] P. Glorennec, "Constrained optimization of FIS using an evolutionary method," in *Genetic Algorithms and Soft Computing*, F. Herrera and J. L. Verdegay, Eds. Berlin, Germany: Physica-Verlag, 1996, pp. 349–368.

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

[18] A. González and R. Pérez, "SLAVE: A genetic learning system based on the iterative approach," *IEEE Tran. Fuzzy Syst.*, vol. 7, pp. 176–191, 1999.

[19] F. Herrera, M. Lozano, and J. L. Verdegay, "Tuning fuzzy controllers by genetic algorithms," *Int. J. Approx. Reason.*, vol. 12, pp. 299–315, 1995.

[20] F. Herrera and J. L. Verdegay, Eds., *Genetic Algorithms and Soft Computing*. Berlin, Germany: Physica-Verlag, 1996.

[21] F. Herrera, M. Lozano, and J. L. Verdegay, "Fuzzy connectives based crossover operators to model genetic algorihtms population diversity," *Fuzzy Sets Syst.*, vol. 92, no. 1, pp. 21–30, 1997.

[22] F. Hoffmann and G. Pfister, "Evolutionary design of a fuzzy knowledge base for a mobile robot," *Int. J. Approx. Reason.*, vol. 17, no. 4, pp. 447–469, 1997.

[23] W. R. Hwang and W. E. Thompson, "Design of fuzzy logic controllers using genetic algorithms," in *Proc. Third IEEE Int. Conf. Fuzzy Syst. (FUZZ-IEEE'94)*, Orlando, FL, 1994, pp. 1383–1388.

[24] H. Ishibuchi, K. Nozaki, H. Tanaka, Y. Hosaka, and M. Matsuda, "Empirical study on learning in fuzzy systems by rice analysis," *Fuzzy Sets Syst.*, vol. 64, pp. 129–144, 1994.

[25] H. Ishibuchi, K. Nozaki, N. Yamamoto, and H. Tanaka, "Selecting fuzzy if-then rules for classification problems using genetic algorithms," *IEEE Trans. Fuzzy Syst.*, vol. 3, pp. 260–270, 1995.

[26] H. Ishibuchi and T. Murata, "A genetic-algorithm-based fuzzy partition method for pattern classification problems," in *Genetic Algorithms and Soft Computing*. Berlin, Germany: Physica-Verlag, 1996, pp. 555–578.

[27] C. Karr, "Applying genetics to fuzzy logic," *AI Expert*, vol. 6, no. 3, pp. 38–43, 1991.

[28] M. A. Lee and H. Takagi, "Embedding apriori knowledge into an integrated fuzzy system design method based on genetic algorithms," in *Proc. Fifth Int. Fuzzy Syst. Assoc. World Cong. (IFSA'93)*, Seoul, Korea, 1993, pp. 1293–1296.

[29] L. Magdalena and F. Monasterio, "A fuzzy logic controller with learning through the evolution of its knowledge base," *Int. J. Approx. Reason.*, vol. 16, pp. 335–358, 1997.

[30] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*. Vienna, Austria: Springer-Verlag, 1996.

[31] A. Parodi and P. Bonelli, "A new approach to fuzzy classifier systems," in *Proc. Fifth Int. Conf. Genetic Algor. (ICGA'93)*, 1993, pp. 223–230.

[32] W. Pedrycz, R. Gudwin, and F. Gomide, "Nonlinear context adaptation in the calibration of fuzzy sets," *Fuzzy Sets. Syst.*, vol. 88, pp. 91–97, 1997.

[33] P. Thrift, "Fuzzy logic synthesis with genetic algorithms," in *Proc. Fourth Int. Conf. Genetic Algor. (ICGA'91)*, 1991, pp. 509–513.

[34] M. Valenzuela-Rendón, "The fuzzy classifier system: Motivations and first results," in *Proc. First Int. Conf. Parallel Prob. Solving Nature (PPSN I)*, H. P. Schwfel and R. Manner, Eds, 1991, pp. 330–334.

[35] ——, "The fuzzy classifier system: A classifier system for continuosly varying variables," in *Proc. Fourth Int. Conf. Genetic Algori. (ICGA'91)*, 1991, pp. 346–353.

[36] ——, "Reinforcement learning in the fuzzy classifier system," *Expert Syst. Applic.*, vol. 14, pp. 237–247, 1998.

[37] J. R. Velasco, S. López, and L. Magdalena, "Genetic fuzzy clustering for the definition of fuzzy sets," *Proc. Sixth IEEE Int. Conf. Fuzzy Syst. (FUZZ-IEEE'97)*, pp. 1665–1670, 1997.

[38] J. R. Velasco, "Genetic-based on-line learning for fuzzy process control," *Int. J. Intell. Syst.*, vol. 13, no. 10–11, pp. 891–903, 1998.

[39] L. X. Wang and J. M. Mendel, "Generating fuzzy rules by learning from examples," *IEEE Trans. Syst., Man, Cybern.*, vol. 22, pp. 1414–1427, 1992.

[40] L. Zheng, "A practical guide to tune proportional and integral (PI) like fuzzy controllers," in *Proc. First IEEE Int. Conf. Fuzzy Syst. (FUZZ-IEEE'92)*, San Diego, CA, 1992, pp. 633–640.