# A Genetic Learning Process for the Scaling Factors, Granularity and Contexts of the Fuzzy Rule-Based System Data Base

**O. Cordón, F. Herrera**
Dept. Computer Science & AI
E.T.S. Ing. Informática
Universidad de Granada
18071 - Granada, Spain
ocordon,herrera@decsai.ugr.es

**L. Magdalena**
Dept. Matemática Aplicada
E.T.S.I. Telecomunicación
Universidad Politécnica de Madrid
28040 Madrid, Spain
llayos@mat.upm.es

**P. Villar**
Dept. Leng. y Sist. Informáticos.
E.S. de Ing. Informática
Universidade de Vigo
32004 - Ourense, Spain
pvillar@uvigo.es

## Abstract

Fuzzy Rule-Based Systems are knowledge-based systems, incorporating human knowledge into their knowledge base through fuzzy rules (Rule Base) and fuzzy membership functions (Data Base). In these kinds of systems, the Data Base is usually defined by choosing a specific membership function type, uniformly partitioning the variable domains into a number of linguistic labels and assigning a fuzzy set to each partition. This operation mode can significantly decrease the FRBS performance. To solve this problem, in this contribution, we propose a genetic process to automatically learn the whole Data Base definition from examples, using an ad-hoc data covering learning method to obtain the Rule Base. Our process learns an appropiate number of labels for each variable primary fuzzy partition and a good distribution for the membership functions (using a non-linear scaling function to define the fuzzy partition contexts). Moreover, it tries to improve the final performance of the FRBS by changing the extents of the universe of discourse of the linguistic variables.

**Keywords:** Fuzzy Rule-Based Systems, Data Base, Learning, Genetic Algorithms.

[1]This research has been supported by CICYT under project PB98-1319.

## 1 Introduction

A Fuzzy Rule-Based System (FRBS) presents two main components: 1) the Inference System, which puts into effect the fuzzy inference process needed to obtain an output from the FRBS when an input is specified, and 2) the Knowledge Base (KB) representing the knowledge about the problem being solved, composed of the Rule Base (RB) constituted by the collection of fuzzy rules, and of the Data Base (DB), containing the membership functions of the fuzzy partitions associated to the linguistic variables.

The composition of the KB of an FRBS directly depends on the problem being solved. The best situation is when there is a human expert able to express its knowledge in the form of fuzzy rules, thus providing the definitions for the DB (the relevant input and output linguistic variables for the system, the term sets for all of them and the membership functions of the fuzzy sets defining their meaning) and for the RB (the fuzzy rules themselves). Unfortunately, this fact is not very common and either the expert is not usually able to provide all this information or there is no expert information about the problem under solving.

Although, there is a large quantity of RB learning methods proposed in the specialized literature [5, 13, 18], there is not much information about the way to derive the DB and most of these RB learning methods need of the existence of a previous definition for it (although some of them are able to learn both the definitions of the DB and the RB). In this contribution we propose a genetic method to learn an appropiate DB from examples (including its three usual components: number of labels per variable, membership function definitions —obtained from a non-linear scaling function that defines the fuzzy partition contexts— and scaling factors). This genetic learning method works cooperatively with an RB derivation method, considered to validate the quality of each DB definition generated. In this paper we will use two different ones: *Wang and Mendel rule generation method* (WM) [18] and *Cordón and Herrera generation method* (CH) [4].

## 2 Previous approaches for the definition of the Data Base

As said, the majority of RB learning methods need a previous definition of the DB to operate. A very common way to proceed to design it involves considering uniform fuzzy partitions with the same number of terms (usually an odd number between three and seven) for all the linguistic variables existing in the problem. However, this operation mode makes the DB have a significant influence on the FRBS performance. In [8], the influence of the fuzzy partition granularity (number of labels per variable) on the FRBS accuracy is analysed.

On the other hand, there are some approaches that try to improve the preliminary DB definition considered once the RB has been derived. To put this into effect, a tuning process considering the whole KB obtained (the preliminary DB and the subsequently derived RB) is used a posteriori to adjust the membership function parameters to improve the FRBS behaviour (for some examples of these kinds of methods, based on Neural Networks and Genetic Algorithms, refer to [2, 3, 9, 13]).

On the other hand, there are some contributions that consider the definition of the DB as an important task prior to the FRBS design. For example, a clustering method that obtains a good fuzzy partition for problems with a single input variable variable can be found in [17]. In [8], a method to learn a good uniform fuzzy partition granularity using Simulated Annealing is proposed. Moreover, in [14], a learning mechanism for different parameters of the DB (scaling factors and sensibility of the fuzzy partition in the variable working range, i.e., fuzzy partition contexts) is proposed included into a global KB learning method.

In this paper we combine the latter two approaches with the aim of learning the whole DB definition, i.e., the fuzzy partition granularity, the scaling factors and the sensibility of the membership functions for each linguistic variable. Our objective is to improve the FRBS performance by finding an appropiate DB for a determinated problem using a simple RB learning method.

## 3 Learning the DB of an FRBS using Genetic Algorithms

Genetic Algoritmhs (GAs) [15] are search and optimization techniques that are 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 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 $G(t)$.

Reviews of different systems that use GAs in order to design FRBS are contained in [5, 10]. GAs have been basically applied to the learning of the different components of the KB (RB in isolation or both DB and RB) and to adjust a preliminary DB definition maintaining fixed an RB previously derived [5].

The important questions when using GAs are: how to code each solution (in this case, the DB of an FRBS), 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 genetic learning method for the DB of a Mamdani FRBS that allows us to define:

- The number of labels for each linguistic variable.

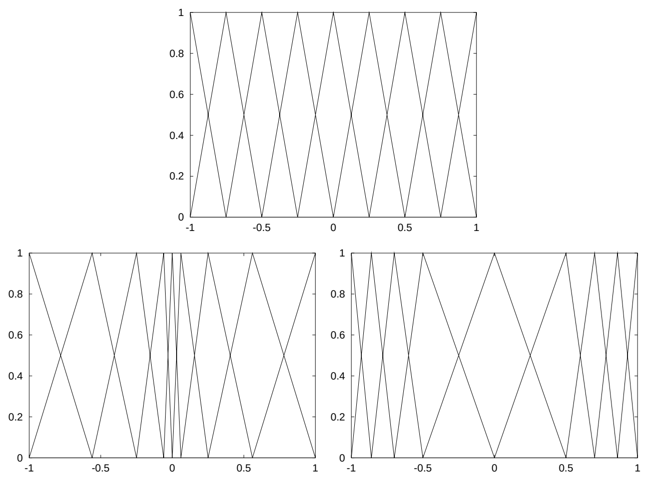
Figure 1: Fuzzy partitions with $a = 1$ (at the top), $a > 1$ (down left), and $a < 1$ (down right)

- 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.

All of that elements 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. In this paper we consider the scaling funtion proposed in [14], that has a single sensibility parameter called $a$ ($a \in \mathbb{R}$). The function used is ($f : [-1, 1] \to [-1, 1]$)

$$f(x) = sign(x) \times |x|^a, \quad with \;\; a > 0$$

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 ($a < 1$). Fig. 1 shows a graphical representation for these three possibilities.

Triangular membership functions are considered due to their simplicity. Moreover, the non-linear scaling function will only be applied on the three definition points of the membership function, in order to make easier the structure of the generated DB and to simplify the defuzzification process.

Each chromosome codifies a complete DB definition by encoding the said parameters. To evaluate a cromosome, we use an ad-hoc data covering method to learn the RB considering the DB contained in it, obtaining a complete KB, and measure the accuracy of the FRBS obtained on a training data set.

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

## 3.1 Encoding the DB

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, usually considered as a fixed part of the problem. As described in [14] for the case of Fuzzy Logic Controllers, an enlargement of the working range produces a change on the gain of the controller related to the corresponding variable. We will directly translate this idea to the Fuzzy Modeling field.

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 previously described on the three definition points of each label. Thus, the number of labels and the sensibility parameter 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 input and output variables), the number of labels per variable is encoded into an integer array of lenght $N$. In this contribution, the possible values considered are the set $\{3, \ldots, 9\}$.
- Sensibility parameters ($C_2$): A real array of lenght $N$, where the sensibility parameter ($a$) for each variable is stored. In our case, the range considered for this parameter is the interval $(0, 10)$.

- Working ranges ($C_3$): An array of $N \times 2$ real values stores the variable working range ($[v_{min}, v_{max}]$). If the initial domain of a variable is $[v^i_{min}, v^i_{max}]$, and $d$ is the interval dimension ($d = v^i_{max} - v^i_{min}$), the range considered for the variable domain lower limit is $[v^i_{min} - (1/4 * d), v^i_{min}]$, and the range for the upper limit is $[v^i_{max}, v^i_{max} + (1/4 * d)]$.

A graphical representation of the chromosome is showed next:

$$C_1 = (l_1, \ldots, l_N)$$

$$C_2 = (a_1, \ldots, a_N)$$

$$C_3 = (r_1^{inf}, r_1^{sup}, \ldots, r_N^{inf}, r_N^{sup})$$

$$C = C_1 C_2 C_3$$

## 3.2 Initial Gene Pool

The initial population is composed of three parts, the first two having $\#val \times 3$ chromosomes, being $\#val$ 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 $\#val \times 6$. The generation of the initial gene pool is described next:

- The first $\#val \times 3$ chromosomes will have the same number of labels and the initial working range in all its variables. For each possible number of labels, three individuals with the three main possibilities for the sensibility parameter will be created: one with $a = 1$, another with $a < 1$ and the other with $a > 1$ (the latter two values are generated at random).

- The second $\#val \times 3$ 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, three individuals are created as in the first part of the population (one with $a = 1$, another with $a < 1$ and the other 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 $M - (\#val \times 6)$ chromosomes, all the components are selected at random. In our case, this part has 22 chromosomes, so, the total population lenght is 64.

## 3.3 Evaluating the DB

There are three steps that must be done to evaluate each chromosome:

- Generate the fuzzy partitions for all the linguistic variables using the information contained in the chromosome. First, each variable is linearly mapped from its working range $[r_i^{inf}, r_i^{sup}]$, $i = 1, \ldots, N$ (third part of the chromosome) to $[-1, 1]$. In a second step, uniform fuzzy partitions for all the variables are created considering the number of labels per variable ($l_1, \ldots, l_N$, first part of the chromosome). Finally, the non-linear scaling function with its sensibility parameter ($a_1, \ldots, a_N$, second part of the chromosome) is applied to the definition points of the membership functions obtained in the previous step, obtaining the whole DB definition.

- Generate the RB, by running a fuzzy rule learning method considering the DB obtained.

- Calculate the Mean Square Error over the training set using the KB obtained (DB + RB). This value will be used as the fitness of the chromosome.

## 3.4 Genetic operators

A set of genetic operators is applied to the genetic code of the DB 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 used.

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

### 3.4.1 Selection

The reproduction operator is the 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.

### 3.4.2 Mutation

Two 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 [16]. 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 inmediately 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 $C_2$ and $C_3$:* Since both parts are based on a real-coding scheme, Michalewicz's non-uniform mutation operator is employed [15].

  If $C_v^t = (c_1, ..., c_k, ..., c_H)$ 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 $C_v^{t+1} = (c_1, ..., c_k', ..., c_H)$, with $k \in 1, ..., H$, and

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

  where $e$ is 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:

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

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

### 3.4.3 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), then the genetic search has located a promising space zone that has to be adequatelly exploitated. This task is developed by applying the max-min-arithmetical (MMA) crossover operator in $C_2$ and $C_3$ and obviously by maintaining the parent $C_1$ values in the offspring. This crossover operator is proposed in [11] and works in the way shown below.

  If $C_v^t = (c_1, ..., c_k, ..., c_H)$ and $C_w^t = (c_1', ..., c_k', ..., c_H')$ are to be crossed, the following four offspring are generated

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

  This operator can use 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 aforesaid offspring.

- *Crossover when the parents encode different granularity levels:* This second case highly recommends the use of the information encoded by the parents for explorating the search space in order to discover new promising zones, considering that a good sensibility parameter or working range for a variable probably causes worse behaviour if it is used with a different number of labels. 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 three 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 three chromosome parts, $C_1$, $C_2$ and $C_3$, thereby producing two meaningful descendents.

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

$$C_t = (l_1, \ldots, l_p, l_{p+1}, \ldots, l_N, a_1, \ldots, a_p, a_{p+1}, \ldots, a_N, \\ r_1^{inf}, r_1^{sup}, \ldots, r_p^{inf}, r_p^{sup}, r_{p+1}^{inf}, r_{p+1}^{sup}, \ldots, r_N^{inf}, r_N^{sup})$$

$$C_t' = (l_1', \ldots, l_p', l_{p+1}', \ldots, l_N', a_1', \ldots, a_p', a_{p+1}', \ldots, a_N', \\ r_1^{inf'}, r_1^{sup'}, \ldots, r_p^{inf'}, r_p^{sup'}, r_{p+1}^{inf'}, r_{p+1}^{sup'}, \ldots, r_N^{inf'}, r_N^{sup'})$$

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

$$C_{t+1} = (l_1, \ldots, l_p, l_{p+1}', \ldots, l_N', a_1, \ldots, a_p, a_{p+1}', \ldots, a_N', \\ r_1^{inf}, r_1^{sup}, \ldots, r_p^{inf}, r_p^{sup}, r_{p+1}^{inf'}, r_{p+1}^{sup'}, \ldots, r_N^{inf'}, r_N^{sup'})$$

$$C_{t+1}' = (l_1', \ldots, l_p', l_{p+1}, \ldots, l_N, a_1', \ldots, a_p', a_{p+1}, \ldots, a_N, \\ r_1^{inf'}, r_1^{sup'}, \ldots, r_p^{inf'}, r_p^{sup'}, r_{p+1}^{inf}, r_{p+1}^{sup}, \ldots, r_N^{inf}, r_N^{sup})$$

Hence the complete recombination process will allow the GA to follow an adequate exploration-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 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 appropiate exploration-exploitation balance in the search was succesfully applied in [7].

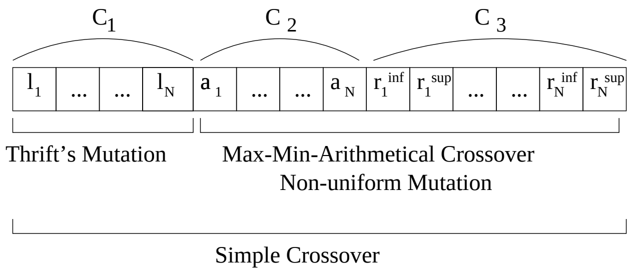

Figure 2: Genetic representation and operators' application scope

## 4 Application examples

A problem with estimations of minimum maintenance costs which are based on a model of the optimal electrical network for a spanish town [6] will be considered to validate the DB genetic learning process proposed. 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 four different characteristics of the towns and their minimum maintenance costs in a sample of 1059 simulated towns. In this case, our objective was to relate the last variable (maintenance costs) with the other four ones. The training set contains 847 elements and the test set contains 212 elements.

We are going to consider two RB automatic learning methods:

- The WM fuzzy rule generation method [18].
- An adaptation of the simplified TSK fuzzy rule generation method presented in [12], that makes the process able to deal with rules with fuzzy consequent, which can be found in [4]. This method considers the n-dimensional table representation for the RB to generate.

On every cell of this table, the subset of the input-output data pairs belonging to the corresponding input fuzzy subspace is considered. The consequent associated to the rule will be the output variable label that maximizes some covering criterion over the training set. This method will be denoted CH.

The genetic parameters used in the experiments are showed in Table 1.

Table 1: Genetic parameter values

| Parameter | Value |
|---|---|
| Population size | 64 |
| Crossover probability | 0.6 |
| Mutation probability | 0.1 |
| Parameter $b$ (non-uniform mutation) | 0.5 |
| Parameter $d$ (MMA crossover) | 0.35 |
| Number of generations | 1000 |

Four experiments were run for each one of the two RB learning methods considered with different initial seeds, and the main results ($MSE_{tra}$) are compared with the best results obtained when running these methods considering uniform fuzzy partitions and initial working ranges, obtained from an exhaustive search, as developed in [8]. The results obtained considering uniform fuzzy partitions are showed in Table 2 whilst the results obtained by our method are presented in Table 3, where the last two columns show the improvement percentage of our method over the best result considering uniform fuzzy partitions with the same number of labels per variable (1) and with any number of labels per variable (2).

## 5 Concluding Remarks

Two main conclusions can be drawn from the results obtained:

- Our method obtains a significative improvement with respect to the classical uniform fuzzy partitions considered in both RB learning methods.
- Our method is very robust, because it does not cause over-learning. The $MSE_{tra}$ and the $MSE_{tst}$ are very similar in all the experiments.

Our future work will be focused on validating the DB definitions obtained for our genetic process combined with simple and quick RB generations methods when considered to design FRBSs by means of more sophisticated RB generation processes, and to modify the process in order to apply other kinds of non-linear functions to define the fuzzy partition contexts, i.e., to change the sensibility of the membership functions into the variable working range.

## References

[1] Baker, J.E., Reducing bias and inefficiency in the selection algoritmh, *Proc. Second Int. Conference on Genetic Algorithms (ICGA'87)*, Hillsdale, 1987, pp. 14-21.

[2] Bonissone, P.P., Khedkar, P.S., Chen, Y-T., Genetic algorithms for automated tuning of fuzzy controllers, a transportation aplication, *Proc. Fifth IEEE Int. Conference on Fuzzy Systems (FUZZ-IEEE'96)*, New Orleans, 1996, pp. 674-680.

[3] Cordón, O., Herrera, F., 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, pp. 369-407.

[4] Cordón, O., Herrera, F., A proposal for improving the Accuracy of Linguistic Modeling, *IEEE Transactions on Fuzzy Systems*, 2000. To appear.

[5] Cordón, O., Herrera, F., Hoffmann, F., Magdalena, L., Genetic Fuzzy Systems. Evolutionary Tuning and Learning of Fuzzy Knowledge Bases, *World Scientific*, in preparation.

[6] Cordón, O., Herrera, F., Sánchez, A., Solving electrical distribution problems using hybrid evolutionary data analysis techniques, *Applied Intelligence* 10, 1999, pp. 5-24.

[7] Cordón, O., Herrera, F., Hybridizing genetic algorithms with sharing scheme and evolution strategies for designing approximate

Table 2: Best results considering uniform fuzzy partitions

| Method | Granularity | # Rul. | $MSE_{tra}$ | $MSE_{tst}$ |
|---|---|---|---|---|
| WM | 9 9 9 9 9 | 130 | 32337.4 (1) | 33505.9 |
|  | 5 7 7 7 9 | 95 | **24867.7** (2) | **26964.1** |
| CH | 8 8 8 8 8 | 543 | 42735.8 (1) | 53596.9 |
|  | 5 6 9 9 7 | 589 | **27698.0** (2) | **26134.3** |

Table 3: Results considering our DB learning method

| Method | Granularity | Parameter $a$ | # Rul. | $MSE_{tra}$ | $MSE_{tst}$ | %imp. (1) | %imp. (2) |
|---|---|---|---|---|---|---|---|
| WM | 4 4 9 9 8 | 8.6 7.7 0.8 0.4 0.9 | 82 | 11542.2 | 13573.5 | 64.3% | 53.5% |
|  | 8 4 9 9 9 | 0.4 4.2 1.0 0.8 0.9 | 92 | 18547.4 | 18863.4 | 42.6% | 25.4% |
|  | 6 5 9 9 9 | 0.9 0.4 0.7 0.4 1.2 | 89 | 10347.4 | 11286.2 | 68.0% | 58.3% |
|  | 3 3 9 9 9 | 1.6 2.7 0.7 0.4 1.1 | 71 | **9859.6** | **9559.4** | 69.5% | 60.3% |
| CH | 5 4 9 9 9 | 0.5 0.2 0.7 0.4 1.0 | 348 | **10972.2** | **10232.1** | 74.3% | 60.3% |
|  | 6 5 9 9 8 | 0.8 0.9 0.9 0.5 0.9 | 548 | 11282.5 | 12950.4 | 73.5% | 59.2% |
|  | 5 4 9 9 9 | 0.7 0.3 0.7 0.5 1.1 | 399 | 12086.1 | 13305.8 | 71.7% | 56.3% |
|  | 8 6 9 9 9 | 1.3 0.9 0.9 0.4 1.1 | 653 | 11206.3 | 13722.4 | 72.7% | 59.5% |

fuzzy rule-based systems, *Fuzzy Sets and Systems*, 2000. To appear.

[8] Cordón, O., Herrera, F., Villar, P., Analysis and guidelines to obtain a good uniform fuzzy partition granularity for fuzzy rule-based systems using simulated annealing, *International Journal of Approximate Reasoning* 2000. To appear.

[9] Herrera, F., Lozano, M., Verdegay, J.L., Tuning fuzzy controllers by genetic algorithms, *International Journal of Approximate Reasoning* 12, 1995, pp. 299-315.

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

[11] F. Herrera, M. Lozano, J.L. Verdegay, Fuzzy connectives based crossover operators to model genetic algorihtms population diversity, Fuzzy Sets and Systems 92:1 (1997) 21-30.

[12] Ishibuchi, H., Nozaki, K., Tanaka, H., Hosaka, Y., Matsuda, M., empirical study on learning in fuzzy systems by rice taste analysis, *Fuzzy Sets and Systems* 64, 1994, pp. 129-144.

[13] Jang, J.R., ANFIS: Adaptive-network-based fuzzy inference system, *IEEE Transactions on Systems, Man, and Cybernetics* 23(3), 1993, pp. 665-684.

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

[15] Z. Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs (Springer-Verlag, 1996).

[16] P. Thrift, Fuzzy logic synthesis with genetic algorithms. Proceedings of Fourth International Conference on Genetic Algorithms (ICGA'91) (1991) 509-513

[17] Velasco, J.R., López, S., Magdalena, L., Genetic fuzzy clustering for the definition of fuzzy sets, *Proc. of the Sixth IEEE International Conference on Fuzzy Systems, Vol III*, Barcelona, 1997, pp. 1565-1670.

[18] Wang, L.X., Mendel, J.M., Generating fuzzy rules by learning from examples, *IEEE Transactions on Systems, Man, and Cybernetics* 22, 1992, pp. 1414-1427.