

# Hybridizing genetic algorithms with sharing scheme and evolution strategies for designing approximate fuzzy rule-based systems<sup>1</sup>

O. Cordón\*, F. Herrera

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

Received September 1997; received in revised form July 1998

## Abstract

Genetic algorithms and evolution strategies are combined in order to build a multi-stage hybrid evolutionary algorithm for learning constrained approximate Mamdani-type knowledge bases from examples. The genetic algorithm niche concept is used in two of the three stages composing the learning process with the purpose of improving the accuracy of the designed fuzzy rule-based systems. The proposed genetic fuzzy rule-based system is used to solve an electrical engineering problem and the results obtained are compared with other methods presenting different characteristics. © 2001 Elsevier Science B.V. All rights reserved.

*Keywords:* Fuzzy rule-based systems; Approximate Mamdani-type knowledge bases; Genetic fuzzy rule-based systems; Genetic algorithms; Evolution strategies; Niching; Inductive learning

## 1. Introduction

Nowadays Fuzzy rule-based systems (FRBSs) have been successfully applied to a wide range of real-world problems from different areas [3,31,37,44]. In order to design an intelligent system of this kind for a concrete application, several tasks have to be performed. One of the most important and difficult ones is the derivation of the fuzzy rule base (FRB), which will contain the information needed to solve the problem in the form of fuzzy rules.

There exist two different kinds of FRBSs in the literature, Mamdani and TSK ones, according to the expression of the consequent of the fuzzy rules

composing the FRB. While Mamdani-type fuzzy rules consider a linguistic variable in the consequent [34], TSK fuzzy rules are based on representing the consequent as a polynomial function of the inputs [41].

Focusing on the first system type, the FRB is composed of a collection of fuzzy rules with the following structure:

$$R_i: \text{IF } x_1 \text{ is } A_{i1} \text{ and } \dots \text{ and } x_n \text{ is } A_{in} \text{ THEN } y \text{ is } B_i$$

where  $x_1, \dots, x_n$  and  $y$  are the input variables and the output variable, respectively. Depending on the characteristics of these fuzzy rules, we can consider two different Mamdani-type FRBSs:

- On the one hand, we have the usual *descriptive* approach [22,34] when  $x_1, \dots, x_n$  and  $y$  are linguistic variables that have associated a term set of possible values presenting a real-world meaning. In this

<sup>1</sup> This research has been supported by CICYT TIC96-0778.

\* Corresponding author. Tel.: +34-58-246143; fax: +34-58-243317.

E-mail address: ocordova@decsai.ugr.es (O. Cordón).

way, each  $A_{ij}$  or  $B_i$  corresponds to a linguistic term that has associated a fuzzy set defining its meaning and this mapping is uniform for all rules in the FRB. This FRBS has been widely used and it has obtained very good results in many different applications. Anyway, it suffers some limitations due to the inflexibility of the concept of the linguistic variable [4]. The homogeneous partitioning of the input and output spaces when the input–output mapping varies in complexity within the space is inefficient and does not scale to high-dimensional spaces [8]. Therefore, its performance decreases when dealing with complex problems in which small changes in the input have associated strong changes in the output [12].

- On the other hand, in the past few years a new approach, the *approximate Mamdani-type* FRBS [3,9], has been proposed to avoid these drawbacks. It is based on working directly with fuzzy variables in the fuzzy rules. In this case, each fuzzy rule presents its own semantics, i.e., the variables  $x_j$  and  $y$ , respectively, take a different fuzzy set  $A_{ij}$  and  $B_i$  as value and not a linguistic term from a global term set. Therefore, it is said that the rules present *free semantics*. According to Carse et al. [8], the advantage of the approximate representation is its expressive power for learning rules which present their own specificity in terms of the fuzzy sets involved in them. This is likely to be of benefit in tackling the curse of dimensionality when scaling to multi-dimensional systems. Anyway, its drawback with respect to the descriptive FRBS is the loss of FRB readability. The approximate approach is considered in [3,8,10–13,21,28,35].

Due to the complexity of the FRB derivation, a large quantity of automatic techniques have been proposed to put it into effect. In the last few years, many different approaches have been presented taking Evolutionary Algorithms (EAs) [1], usually Genetic Algorithms (GAs), as a base, to automatically design fuzzy systems, constituting the so-called genetic fuzzy systems (GFSs) [9]. In particular, the promising results obtained by the EAs in the learning or tuning of the FRB have extended the use of these algorithms in the design of FRBSs. These kind of GFSs, genetic fuzzy rule-based systems (GFRBSs) [9,30], are considered nowadays as an important branch of the soft computing area [7] in view of the large number of

contributions developed in the last few years (see [15, Section 3.13; 16, Section 13]).

In this paper we present a multi-stage hybrid GA-evolution strategy (GA-ES) process for designing approximate Mamdani-type FRBSs from examples. This GFRBS will allow us to derive the whole FRB, that is, the definition of the fuzzy rules themselves and of the membership function shapes considered in them. It is based on the iterative rule learning (IRL) approach [24] and is composed of three different stages, an *evolutionary generation process*, a *genetic multi-simplification process*, and a *genetic tuning process*, following the structure presented in [18,30]. The GA niche concept [19] (see the appendix) will play a main role in the learning process, due to it will be used to solve the cooperation vs. competition problem (CCP) [6], existing when designing FRBSs by means of EAs.

The GFRBS performance will be shown by using it to solve an electrical engineering problem and the evolutionary learning process proposed will be compared with classical methods, Neural Networks and other GFRBSs presenting different characteristics.

In order to put this into effect, we arrange this paper as follows. The next section presents some preliminaries by briefly introducing GFRBSs, the IRL approach and the CCP. The characteristics and the working way of the evolutionary learning process are discussed in Section 3. The three stages composing the proposed GFRBS are described in Sections 4–6. Section 7 shows the experiments developed in the solving of the commented problem, while in Section 8 some concluding remarks are pointed out. Finally, an appendix introducing briefly the GAs, niching GAs and ESSs is presented.

## 2. Preliminaries

### 2.1. Genetic fuzzy rule-based systems

EAs, specially GAs, have proven to be a powerful tool for automating the definition of the FRB, since adaptive control, learning, and self-organizational FRBSs can be considered in a lot of cases as optimization or search processes. Their advantages have extended the use of EAs in the development of a wide range of approaches for designing FRBSs over the past few years. These approaches receive the general name of *GFRBSs* [9,30].

EAs are applied to modify/learn the definition of the membership functions shapes and/or the composition of the fuzzy rules. Therefore, it is possible to distinguish three different groups of GFRBSs depending on the FRB components included in the learning process [9,30]:

1. *Genetic definition of the membership functions.*
2. *Genetic derivation of the fuzzy rules.*
3. *Genetic learning of the whole FRB.*

For a wider description of each family, see [9,30] and for an extensive bibliography; see [15, Section 3.13; 16, Section 13]. Different approaches may be found in [9,29].

Carse et al. [8] divide the third family in two different subgroups depending on the simultaneity in the learning of both FRB components. Therefore, they differentiate between learning them in a single process or in different stages. We shall refer to these latter kind of systems as multi-stage GFRBSs [24]. The process proposed in this paper belongs to this family.

## 2.2. The iterative rule learning approach

The main problem that has to be solved to design a GFRBS consists of finding a suitable representation both capable of gathering the problem characteristics and representing adequately the potential solutions to it.

Classically, two genetic learning approaches, adopted from the field of genetic based machine learning systems, have been used: the *Michigan* and *Pittsburgh* approaches [20]. In the past few years, some authors have designed several GFRBSs following a new learning model, the *IRL* approach [24].

In this latter model, as in the *Michigan* one, each chromosome in the population represents a single fuzzy rule, but only the best individual is considered to form part of the final FRB. Therefore, in this approach the EA provides a partial solution to the problem of learning, and, contrary to both previous ones, it is run several times to obtain the complete FRB. This is put into effect by including it into an iterative scheme based on obtaining the best current fuzzy rule for the system, incorporating this rule into the final FRB, and penalizing it before repeating the process. It ends up when the FRB is able to represent the system adequately.

This scheme is usually employed in GFRBSs based on inductive learning, in which the penalization of the fuzzy rules yet generated is made by removing from the training data set all those examples that are yet covered by the FRB obtained until this moment. On the other hand, as the learning processes using it do not envisage any relationship between the fuzzy rules generated, it is usual to employ postprocessing processes to simplify and/or adjust the FRB obtained, so forming a multi-stage GFRBS.

The main characteristic of the IRL is that it reduces substantially the search space, because in each iteration only one fuzzy rule is searched. This allows us to obtain good solutions in GFRBSs for off-line learning problems.

A more complete description of the IRL and a short comparison of the three genetic learning approaches is to be found in [24].

## 2.3. The cooperation vs. competition problem

One of the most interesting features of an FRBS is the interpolative reasoning it develops. This characteristic plays a key role in the high performance of FRBSs and is a consequence of the *cooperation among the fuzzy rules composing the FRB*. As it is known, the output obtained from an FRBS is not usually due to a single fuzzy rule but to the cooperative action of several fuzzy rules that have been fired because they matched the input to the system to any degree. On the other hand, the main feature of an EA is the *competition among members of the population representing possible solutions to the problem* under solving. In this case, this characteristic is due to the mechanisms of the natural selection in which the EA is based.

Therefore, since a GFRBS combines both commented features, it works by *inducing competition to get the best possible cooperation*. This seems to be a very interesting way to solve the problem of designing an FRBS, because the different members of the population compete among them to provide a final solution presenting the best cooperation among the fuzzy rules composing it. The problem is to obtain the best possible way to put this working way into effect. This is referred to as *CCP* [6].

The difficulty of solving the introduced problem depends directly on the genetic learning approach followed by the GFRBS. Multi-stage GFRBSs based on

the IRL approach try to solve adequately the CCP at the same time that reduce the search space by encoding a single fuzzy rule in each chromosome. To put this into effect, these processes divide the genetic learning process in, at least, two stages. Therefore, *the CCP is solved in two steps acting at two different levels, predominating the competition among rules in the first one, the genetic generation stage, and the cooperation among these generated fuzzy rules in the second one, the postprocessing stage.* Hence:

- the genetic generation stage forces the competition among fuzzy rules, as the genetic learning processes based on the Michigan approach, *to obtain an FRB composed of the best possible fuzzy rules*, and the cooperation among them is only smoothly addressed by means of the rule penalization criterion.
- the postprocessing stage forces the cooperation among the fuzzy rules generated in the previous stage by refining or eliminating the redundant or unnecessary fuzzy rules from the previously generated fuzzy rule set *to obtain the best possible FRB.*

As it can be observed, the iterative operation mode followed by the genetic generation stage in multi-stage GFRBSs based on the IRL induces the formation of niches and reduces substantially the dimension of the search space. The postprocessing stage deals with a simple search space as well because it only works on the FRB obtained from the previous stage.

An analysis on the way in which the CCP is solved by the other genetic learning approaches is to be found in [24].

### 3. On the evolutionary learning process

In this section we are going to introduce the basis followed by the proposed GFRBS by analyzing the following aspects in-depth.

#### 3.1. Type of fuzzy rule-based system and generation process

There exist two different variants to generate an approximate Mamdani-type FRBS, the *constrained* and *unconstrained* ones [13]:

- The fuzzy rules present a *constrained free semantics* when they are generated with a free semantics but based on an initial domain fuzzy

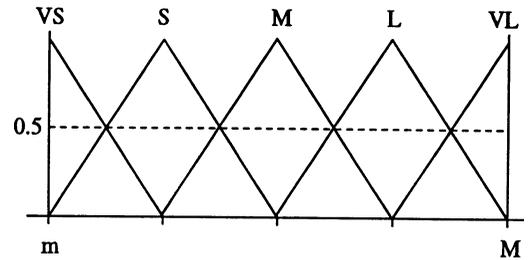


Fig. 1. Graphical representation of a possible fuzzy partition.

partition that determines the intervals in which each point defining the membership functions may take value.

- On the other hand, when the only restriction imposed on the membership function locations and shapes is to lie in a concrete interval, the fuzzy rules present a *unconstrained free semantics*. The most extreme case is when the interval associated to each fuzzy set corresponds to the whole domain of the system variable.

Both approximate approaches perform better than the descriptive one when working with complex problems, but the constrained free semantics will have better behavior than the unconstrained one when the complexity of the problem is intermediate. The unconstrained approximate FRBS is the most adequate for dealing with very hard problems.

In [13,28] is presented an unconstrained approximate GFRBS based on generating the fuzzy sets involved in each fuzzy rule by searching in the whole variable domain, while in [11,12] the intervals considered are obtained from initial fuzzy partitions of these domains. A preliminar version of the constrained approximate GFRBS presented in this paper is to be found in [10,13].

In this paper, we shall work with the constrained approximate approach, considering every fuzzy set associated to a normalized triangular membership function. We take into account a computational way to characterize it by using a parametric representation achieved by means of the 3-tuple  $(a_{ij}, b_{ij}, c_{ij})$ ,  $(a_i, b_i, c_i)$ ,  $j = 1, \dots, n$ .

Hence, due to the constrained free semantics nature of the GFRBS proposed, we need initial fuzzy partitions as the one shown in Fig. 1 for each one of the process fuzzy variables. Making use of them, an interval of performance, defined as follows, is associated

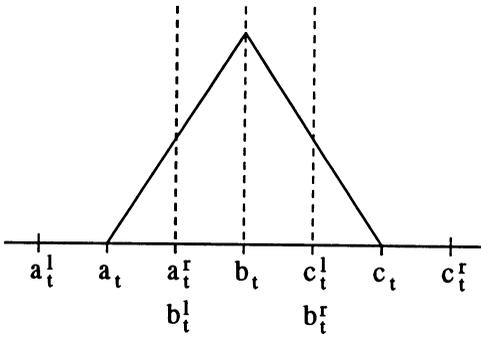


Fig. 2. Membership function and intervals of performance for the generating process.

to each one of the three points defining the membership functions  $A_i(\cdot)$ ,  $(a_i, b_i, c_i)$ :

$$[a_i^l, a_i^r] = \left[ a_i - \frac{b_i - a_i}{2}, a_i + \frac{b_i - a_i}{2} \right],$$

$$[b_i^l, b_i^r] = \left[ b_i - \frac{b_i - a_i}{2}, b_i + \frac{c_i - b_i}{2} \right],$$

$$[c_i^l, c_i^r] = \left[ c_i - \frac{c_i - b_i}{2}, c_i + \frac{c_i - b_i}{2} \right]$$

for locally adjusting their parameters during the generating process. Fig. 2 shows the intervals of performance associated to each one of the parameters.

Therefore, the generated fuzzy rules will have their semantic within the performance interval established by the fuzzy partition membership functions, so maintaining a constrained free semantics.

### 3.2. Properties required to the generated fuzzy rule base

Several important statical properties have to be verified by the FRB in order to obtain an FRBS presenting good behavior [22,34]. As in [12], two of them are going to be considered in our learning process, the *completeness* and *consistency*. Since we consider an inductive approach for designing GFRBSs, both properties will be based on the existence of a training data set,  $E_p$ , composed of  $p$  numerical input-output problem variable pairs. These examples will present the

following structure:

$$e_l = (ex_1^l, \dots, ex_n^l, ey^l), \quad l = 1, \dots, p.$$

A brief description of the said properties can be found below. For a wider description, refer to [12].

#### 3.2.1. Completeness of a fuzzy rule base

It is clear that an FRBS should always be able to infer a proper output for every possible system input. This property is called *completeness* and may be mathematically formulated using two real values  $\tau$  and  $\varepsilon$  by means of the following expressions:

$$C_R(e_l) = \bigcup_{i=1..T} R_i(e_l) \geq \tau, \quad l = 1, \dots, p,$$

$$R_i(e_l) = *(A_{i1}(ex_1^l), \dots, A_{in}(ex_n^l), B_i(ey^l)),$$

where  $*$  is a t-norm, and  $R_i(e_l)$  is the *compatibility degree* between the rule  $R_i$  and the example  $e_l$ .

Given an FRB composed of  $T$  fuzzy rules  $R_i$ , the *covering value* of an example  $e_l \in E_p$  is defined as

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

and we require the following condition:

$$CV_R(e_l) \geq \varepsilon, \quad l = 1, \dots, p.$$

A good FRB must satisfy both the conditions presented above, to verify the *completeness property* and to have an adequate final *covering value*.

#### 3.2.2. Consistency of a fuzzy rule base

A generic set of if-then rules is *consistent* if it does not contain contradictions. There is a need to relax the consistency property for considering it in FRBs. We do this by means of the *positive* and *negative example* concepts [25,28]. An example is considered positive for a fuzzy rule when it matches with its antecedent and consequent, and it will be considered a negative example when it matches with its antecedent and not with its consequent.

Let  $E^+(R_i) = \{e_l \in E_p \mid R_i(e_l) \geq 0\}$  and  $E^-(R_i) = \{e_l \in E_p \mid R_i(e_l) = 0 \text{ and } A_i(ex^l) > 0\}$  be, respectively, the positive and negative example set for the rule

$R_i$ . Let  $n_{R_i}^+ = |E^+(R_i)|$  and  $n_{R_i}^- = |E^-(R_i)|$ . Given a parameter  $k \in [0, 1]$ , it is said that

$R_i$  is  $k$ -consistent when  $n_{R_i}^- \leq k \cdot n_{R_i}^+$ .

Hence, the way to incorporate the satisfaction of this property in the proposed GFRBS is to encourage the generation of  $k$ -consistent rules. Those rules not verifying this property will be penalized to not allow them to be in the finally generated FRB.

### 3.3. Tackling the cooperation vs. competition problem

The usual way of solving the CCP associated to the evolutionary learning processes based on the IRL approach, introduced in Section 2.3, will be extended in the multi-stage GFRBS presented in this paper by considering the following three aspects. As it may be seen, the GA niching concept [19] (see the appendix) will play a main role in the multi-stage GFRBS proposed:

*Considering fuzzy rule cooperation in the first stage:* We shall improve the fuzzy rule generation process by using a criterion allowing us to generate the best possible fuzzy rule in each iteration taking into account both the goodness of this rule and the goodness of its cooperation with the previous ones generated.

To put this criterion into effect, we shall induce a second type of niching in the generating method, apart from the one induced by the IRL operation mode. It will be based on a *phenotypic niching scheme* [19] and will penalize the excessive proximity of the fuzzy rule being generated to the previously obtained ones, so obtaining a better cooperation level in the rule set. In [10–13] we presented a niche sharing function working in this way, the *Low Niche Interaction Rate*, which showed good results. This will be the one considered in this paper.

*Tackling both refinement possibilities in the post-processing stage:* The postprocessing stage usually forces the cooperation among the fuzzy rules generated by refining them or eliminating the redundant or unnecessary ones. In the multi-stage GFRBS proposed in this paper, we shall consider both tasks in order to improve the performance of the final FRBS designed. Following this idea and the purpose of dealing with

a more simple search space, we shall divide the post-processing stage into two different processes [18,30], each one developing each task:

- First, a *genetic simplification process* will simplify the previous rule set by removing the fuzzy rules not cooperating adequately with the others, but not modifying the membership function definitions. It will allow us to obtain the best possible FRB composed of the best combination of the fuzzy rules generated in the first stage.
- Then, once an FRB with the optimal number of rules have been obtained from the previous two stages, a *genetic tuning stage* will adjust the definitions of the membership functions with the aim of obtaining a better cooperation among its fuzzy rules. In this case, the search process works only over the membership function parameter space and not over the fuzzy rule one.

*Generating different FRB definitions:* The other existing type of niching, the *genotypic sharing scheme* [19], will be considered to obtain not only a single FRB definition as output from the process but different ones presenting the best possible behavior, i.e., the best possible cooperation among the fuzzy rules composing them. In this way, we shall extend the genetic simplification process presented in [28] by inducing a genotypic niching in it by means of the *sequential niche technique* [5]. This process is a multimodal optimization method based on iterating a GA to find a solution at each run, and on penalizing the search space zone in which this solution is located for subsequent runs. The new *genetic multisimplification process* so developed iterates the genetic simplification process commented and modifies the fitness landscape each time an FRB is obtained from it, allowing us to obtain different FRBs representing the system known knowledge existing in the training data set.

The idea of using the niche concept in this stage is based on the genetic learning process presented by Satyadas and Krishnakumar in [33]. The authors make use of a GA with a phenotypic sharing scheme for generating different optimal FRBs for a concrete problem. Then they apply a clustering method in order to select the best FRB from the set of candidates.

Since in our multi-stage GFRBs, the genetic tuning process will be applied on the different FRBs obtained from the multisimplification process, and the most accurate will be the one given as output, we solve

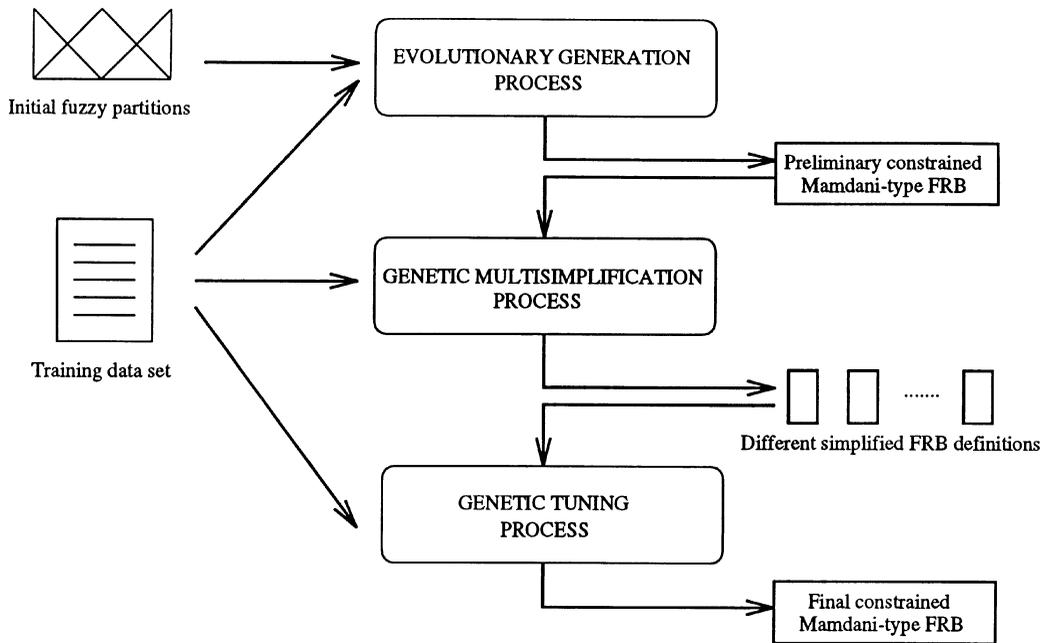


Fig. 3. Multi-stage GFRBS structure and operation mode.

the problem of the choice of the best FRBS existing in [33].

### 3.4. Composition of the multi-stage genetic fuzzy rule-based system

Therefore, the multi-stage GFRBS based on the IRL proposed will consist of the following three stages:

1. An *evolutionary generation process* for generating fuzzy rules with constrained free semantics. The process has two components: a *fuzzy rule generating method* composed of a hybrid GA-ES process which uses a phenotypic niche criterion to obtain the best possible cooperation among the fuzzy rules generated, and an *iterative covering method* of the system behaviour example set, which penalizes each rule generated by the fuzzy rule generating method by considering its covering over the examples in the training set and removes the ones yet covered from it. This process allows us to obtain a set of fuzzy rules with constrained free semantics covering the training set in an adequate form.
2. A *genetic multisimplification process* for selecting rules, based on a binary coded GA with a genotypic

sharing function and a measure of the FRBS performance. It will remove the redundant rules generated by the previous component with the aim of obtaining different simplified FRBs presenting the best possible cooperation among the fuzzy rules composing them.

3. A *genetic tuning process*, based on a real coded GA and a measure of the FRBS performance. It will give the final FRB as output by adjusting the membership functions for each fuzzy rule in each possible FRB derived from the genetic multisimplification process. The more accurate FRB obtained in this stage will constitute the final output of the whole evolutionary learning process.

These processes will be analyzed in Sections 4, 5, and 6, respectively. Fig. 3 illustrates the introduced GFRBS structure.

## 4. The evolutionary generation process

As has been said, the first stage consists of two processes, a *generating method* of desirable fuzzy rules from examples and a *covering method* of the set of examples.

1. The fuzzy rule generating method is developed by means of a special GA encoding a single fuzzy rule in each chromosome. The GA finds the best rule in every run over the set of examples according to the features included in the fitness function. A sharing function implemented as a fitness function criterion, which induces a phenotypic niching in the GA population, allows us to obtain a suitable cooperation among the generated fuzzy rules. An ES is used for locally tuning the best fuzzy rules obtained in the genetic search iterations.
2. The covering method allows a set of fuzzy rules to be obtained covering the set of examples. It is developed as an iterative process. In each iteration, it runs the generating method choosing the best fuzzy rule, considers the relative covering value that this rule provokes over the example set and removes the examples with a covering value greater than a value  $\varepsilon$  provided by the FRBS designer.

The following subsections present both methods in-depth.

#### 4.1. The fuzzy rule generating method

The structure of the generating method for fuzzy rules was presented in [10]. It is developed by means of a special GA, where a chromosome encodes a fuzzy rule and an ES locally tunes the fuzzy rules. We describe the hybrid EA components below.

##### 4.1.1. Representation

A chromosome  $C$  encoding a candidate rule is composed of two different parts,  $C_1$  and  $C_2$ , each one corresponding to each one of the FRB components. The first part of the chromosome encodes the composition of the fuzzy rule and the second one the membership functions involved in it.

In order to represent the first part there is a need to number the primary fuzzy sets belonging to each one of the variable fuzzy partitions considered. A fuzzy variable  $x_i$  taking values in a primary set  $T(x_i) = \{L_1(x_i), \dots, L_{n_i}(x_i)\}$  has associated the ordered set  $T'(x_i) = \{1, \dots, n_i\}$ .

On the other hand, the second part has the representation introduced in Section 3.1. Each one of the triangular membership functions composing the rule,  $L_i(x_j)$ , is encoded by means of its associated 3-tuple  $(a_{L_i(x_j)}, b_{L_i(x_j)}, c_{L_i(x_j)})$ .

Hence, the following fuzzy rule,

IF  $x_1$  is  $L_{i_1}(x_1) \dots$  and  $x_n$  is  $L_{i_n}(x_n)$

THEN  $y$  is  $L_{i_{n+1}}(y)$

is encoded into a chromosome  $C$  with the form

$$C_1 = (i_1, \dots, i_n, i_{n+1}),$$

$$C_2 = (a_{L_{i_1}(x_1)}, b_{L_{i_1}(x_1)}, c_{L_{i_1}(x_1)}, \dots, a_{L_{i_n}(x_n)},$$

$$b_{L_{i_n}(x_n)}, c_{L_{i_n}(x_n)}, a_{L_{i_{n+1}}(y)}, b_{L_{i_{n+1}}(y)}, c_{L_{i_{n+1}}(y)}),$$

$$C = C_1 C_2.$$

Now, the fundamental underlying mechanisms of a GA, formation of an initial gene pool, fitness function, and genetic operators are developed.

##### 4.1.2. Initial gene pool

A third of the initial gene pool is created making use of the examples contained in the training set,  $E_p$ , and other third is initiated totally at random. The initialization of the individuals belonging to the remaining third takes common characteristics of the other two. The first part of them is initiated from the examples, and the second one at random.

With  $M$  being the GA population size and  $t = \min\{|E_p|, M/3\}$ , let  $t$  examples be selected at random from  $E_p$ . Then, the initial population generation process is performed in three steps as follows:

1. Making use of the existing linguistic variable primary fuzzy partitions, generate  $t$  individuals by taking the rule best covering each one of the  $t$  randomly obtained examples. Initiate  $C_1$  and  $C_2$  by coding, respectively, the rule primary fuzzy sets and their meaning in the said way.
2. Generate another  $t$  individuals initiating  $C_1$  in the same way that in the previous step, and computing the values of  $C_2$  at random, each gene varying in its respective interval.
3. Generate the remaining  $M - 2 \cdot t$  individuals by computing at random the values of the first part,  $C_1$ , and making use of these for randomly generating the  $C_2$  part, each gene varying in its respective interval.

### 4.1.3. Evaluation of individual fitness

The fitness function measuring the adaptation of each rule of the population is a multiobjective function based on the following criteria:

*High-frequency value* [28]: The frequency of a fuzzy rule,  $R_i$ , through the set of examples,  $E_p$ , is defined as

$$\Psi_{E_p}(R_i) = \frac{\sum_{l=1}^p R_i(e_l)}{p}.$$

*High average covering degree over positive examples* [28]: The set of positive examples to  $R_i$  with a compatibility degree greater than or equal to  $\omega$  is defined as

$$E_{\omega}^+(R_i) = \{e_l \in E_p \mid R_i(e_l) \geq \omega\}$$

with  $n_{\omega}^+(R_i)$  being equal to  $|E_{\omega}^+(R_i)|$ . The *average covering degree* on  $E_{\omega}^+(R_i)$  can be defined as

$$G_{\omega}(R_i) = \sum_{e_l \in E_{\omega}^+(R_i)} R_i(e_l) \mid n_{\omega}^+(R_i).$$

*Penalization associated to the no satisfaction of the k-consistency property* [14]: This last criterion penalizes those fuzzy rules with many negative examples with respect to the number of positive examples with a compatibility degree greater than or equal to  $\omega$ . In this way, it penalizes the no satisfaction of the  $k$ -consistency property. The *penalty function on the negative examples set of the rule  $R_i$*  will be

$$g_n(R_i^-) = \begin{cases} 1 & \text{if } n_{R_i}^- \leq k \cdot n_{\omega}^+(R_i), \\ \frac{1}{n_{R_i}^- - k n_{\omega}^+(R_i) + \exp(1)} & \text{otherwise.} \end{cases}$$

We should note that the negative example set is always computed over the whole training data set  $E_p$ .

*Low niche interaction rate* [10]: With  $N_i = (N_i x, N_i y)$  being the centers of the rules (niches) determined until now ( $i = 1, \dots, d$ , where  $d$  is the number of generating process runs developed), and  $C$  being the individual encoding the fuzzy rule being adapted,  $R_i$ , the *low niche interaction rate* penalizes the fitness associated to  $C$  in the following way:

$$LNIR(R_i) = 1 - NIR(R_i)$$

$$NIR(R_i) = \text{Max}_i \{h_i\}$$

$$h_i = * (A(N_i x), B(N_i y)), \quad i = 1, \dots, d$$

$$A(N_i x) = * (A_1(N_i x_1), \dots, A_n(N_i x_n))$$

$C \sim R_i$ : IF  $x_1$  is  $A_1$  and ... and  $x_n$  is  $A_n$

THEN  $y$  is  $B$ .

Hence  $LNIR(C)$  penalizes the excessive interaction among the fuzzy rules, which leads to a bad cooperation among them. It is defined in  $[0, 1]$  and gives the maximum value (no penalization) when  $R_i$  does not interact with any of the rules generated until now. The minimum value (maximum penalization) is obtained when this rule is equal to one of those generated previously.

These four criteria are combined into a fitness function using any aggregation function increasing in the four variables. In this paper, we work with the product in the following way:

$$F(R_i) = \Psi_{E_p}(R_i) \cdot G_{\omega}(R_i) \cdot g_n(R_i^-) \cdot LNIR(R_i).$$

Rules obtaining higher values in this function will be more accurate.

### 4.1.4. Genetic operators

Due to the special nature of the chromosomes involved in this generation process, the design of the genetic operators able to deal with it become a main task. As there exists a strong relationship between both chromosome parts, operators working cooperatively in  $C_1$  and  $C_2$  are required in order to make best use of the representation used.

It can be clearly observed that the existing relationship will present several problems if not handled adequately. For example, modifications in the first chromosome part have to be automatically reflected in the second one. It makes no sense to modify the primary fuzzy set and continue working with the previous membership function. On the other hand, there is a need to develop the recombination in a correct way in order to obtain meaningful offsprings.

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

**Mutation:** Two different operators are used, each one of them acting on a different chromosome part.

A short 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 [42]. 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 current primary fuzzy set to the immediately preceding or subsequent one (the decision is made at random). When the primary fuzzy set to be changed is the first or last one in the primary fuzzy partition, the only possible change is developed. As it has been commented, a mutation in  $C_1$  provokes a change in  $C_2$ . When a fuzzy variable changes its value from one primary fuzzy set to another, the membership function associated to it is automatically updated in the second chromosome part to the default values in the corresponding primary fuzzy partition.
- **Mutation on  $C_2$ :** As  $C_2$  is based on a real-coding scheme, Michalewicz’s non-uniform mutation operator is employed [36].

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

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

where  $a$  is a random number that 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:

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

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

**Crossover:** As regards the recombination process, two different crossover operators are employed depending on the two parents’ scope:

- **Crossover when both parents encode the same rule:** If this is the case, then 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 crossover operator in  $C_2$  and obviously by maintaining the parent  $C_1$  values in the offspring. This crossover operator is proposed in [27] and works in the way shown below.

If  $C_v^t = (c_1, \dots, c_k, \dots, c_H)$  and  $C_w^t = (c'_1, \dots, c'_k, \dots, c'_H)$  are to be crossed, the following four offsprings are generated:

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

This operator can use a parameter  $a$  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 rules:** In this second case, it makes no sense to apply the previous operator because it will provoke the obtaining of disrupted descendents. This fact is due to the combination of two membership functions associated to different primary fuzzy sets makes the obtaining of two new fuzzy sets not belonging to the intervals of performance determined by the initial fuzzy partition. 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. In this way, an standard crossover operator is applied over both parts of the chromosomes. This operator performs as follows: a crossover point  $cp$  is randomly generated in  $C_1$  and the two parents are crossed at the  $cp$ -th and  $n + 1 + 3 \cdot cp$  genes. The crossover is developed this way in both 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. Since  $C_l = (c_1, \dots, c_{cp}, c_{cp+1}, \dots, c_{n+1}, a_{c_1}, b_{c_1}, c_{c_1}, \dots, a_{c_{cp}}, b_{c_{cp}}, c_{c_{cp}}, a_{c_{cp+1}}, b_{c_{cp+1}}, c_{c_{cp+1}}, \dots, a_{c_{n+1}}, b_{c_{n+1}}, c_{c_{n+1}})$  and

$C'_t = (c'_1, \dots, c'_{cp}, c'_{cp+1}, \dots, c'_{n+1}, a_{c'_1}, b_{c'_1}, c_{c'_1}, \dots, a_{c'_{cp}}, b_{c'_{cp}}, c_{c'_{cp}}, a_{c'_{cp+1}}, b_{c'_{cp+1}}, c_{c'_{cp+1}}, \dots, a_{c'_{n+1}}, b_{c'_{n+1}}, c_{c'_{n+1}})$  the individuals to be crossed at point  $cp$ , the two resulting offsprings are

$$C_{t+1} = (c_1, \dots, c_{cp}, c'_{cp+1}, \dots, c'_{n+1}, a_{c_1}, b_{c_1}, c_{c_1}, \dots, a_{c_{cp}}, b_{c_{cp}}, c_{c_{cp}}, a_{c'_{cp+1}}, b_{c'_{cp+1}}, c_{c'_{cp+1}}, \dots, a_{c'_{n+1}}, b_{c'_{n+1}}, c_{c'_{n+1}}),$$

$$C'_{t+1} = (c'_1, \dots, c'_{cp}, c_{cp+1}, \dots, c_{n+1}, a_{c'_1}, b_{c'_1}, c_{c'_1}, \dots, a_{c'_{cp}}, b_{c'_{cp}}, c_{c'_{cp}}, a_{c_{cp}}, b_{c_{cp}}, c_{c_{cp}}, \dots, a_{c_{n+1}}, b_{c_{n+1}}, c_{c_{n+1}}).$$

Hence, the complete recombination process will allow 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 number of max–min–arithmetical 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 begins, characterized by the increasing of the exploitation of these zones and the decreasing of the space exploration. Therefore the number of max–min–arithmetical crossovers rises a lot and the application of the standard crossover decreases. An example of this behavior is shown in Section 7 (Fig. 7).

*Evolution strategy:* The last genetic operator to be applied consists of an (1+1)-ES. This optimization technique has been selected and integrated into the genetic recombination process in order to perform a local tuning of the best population individuals (rules) in each generation, so obtaining a genetic local search process [45]. Each time a GA generation is performed, the ES will be applied over a percentage  $\alpha$  of the best population individuals existing in the current genetic population. In this way, it allows us to develop again a strong exploitation over the promising space zones found in each generation by adjusting the  $C_2$  part values of the chromosomes located at them.

The basis of the ES employed is briefly presented in the appendix. Now we are going to describe the

adaptation of this algorithm to our problem. In the (1+1)-ES, the mutation strength depends directly on the value of the parameter  $\sigma$ , which determines the standard deviation of the normally distributed random variable  $z_i$ . In our case, the step size  $\sigma$  cannot be a single value because each one of the membership functions encoded in the second part of the chromosome is defined over different universes and so requires different order mutations. Therefore, an step size  $\sigma_i = \sigma \cdot s_i$  for each component has already been used in the (1+1)-ES. Anyway the relations of all  $\sigma_i$  were fixed by the values  $s_i$  and only the common factor  $\sigma$  is adapted following the assumptions presented in [2].

Each parent component  $c_i$  varying in the interval of performance  $[c_i^l, c_i^r]$  will have its own associated step size  $\sigma_i$  with  $s_i = (c_i^r - c_i^l)/4$ . Hence when  $\sigma$  takes value 1 at the first ES generation, the obtaining of a large quantity of  $z_i$  normal values in the interval  $[-(c_i^r - c_i^l)/4, (c_i^r - c_i^l)/4]$  is ensured. All these values, as the ones remaining in the intervals  $[-(c_i^r - c_i^l)/2, -(c_i^r - c_i^l)/4]$  and  $[(c_i^r - c_i^l)/4, (c_i^r - c_i^l)/2]$ , perform a successful  $c_i$  mutation (that is, the corresponding  $c_i + z_i$  lies in the  $c_i$  interval of performance). When this value does not belong to the commented interval, the mutated value  $c'_i$  is equal to the interval extent,  $c_i^l$  or  $c_i^r$ , closer to  $c_i + z_i$ .

This ES has been designed in order to maintain the constrained free semantic approach followed by the fuzzy rule generation process. Other variant of the same operator, performing an unconstrained free semantic rule generation, was used in [11,12].

Fig. 4 summarizes the application scope of the genetic operators proposed.

Finally, the following algorithm summarizes the whole process:

1. Compute the value  $n_c$  of pairs of individuals belonging to  $P(t)$  to be crossed taking as a base the value of the crossover rate  $P_c$ .
2. While ( $n_c > 0$ ) do
  - (a) Select at random the parents to be crossed.
  - (b) If ( $C_1(\text{father}) = C_1(\text{mother})$ ) then maintain  $C_1$  and perform max–min–arithmetical crossover on  $C_2$  for obtaining the two descendents else perform standard crossover on  $C_1$  and  $C_2$ .
  - (c)  $n_c \leftarrow n_c - 1$ .
3. Compute the value  $n_m$  of genes to be mutated taking as base the value of the mutation rate  $P_m$ .

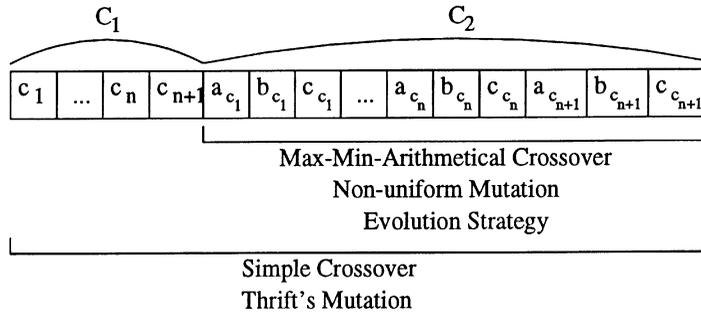


Fig. 4. Generating process genetic representation and operators' application scope.

4. While ( $n_m > 0$ ) do
  - (a) Select at random the parent and gene to be mutated.
  - (b) If (the gene belongs to  $C_1$ ) then perform Thrift's mutation on it at  $C_1$  and update the corresponding  $C_2$  piece of chromosome to represent adequately the initial mutated gene meaning else perform non-uniform mutation on the  $C_2$  gene.
  - (c)  $n_m \leftarrow n_m - 1$ .
5. Compute the value  $n_{es} \leftarrow \alpha \cdot M$  of individuals to be applied the ES and sort descendently the current population, taking into account only the different chromosomes.
6. While ( $n_{es} > 0$ ) do
  - (a) Select the next parent to be locally tuned, beginning at the population head.
  - (b) Perform ES on  $C_2$ .
  - (c)  $n_{es} \leftarrow n_{es} - 1$ .

With regards to the selection procedure, it is Baker's stochastic universal sampling, 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. The covering method

The covering method is developed as an iterative process that allows us to obtain a set of fuzzy rules covering the example set. In each iteration, it runs the generating method, obtaining the best fuzzy rule according to the current state of the training set, considers the relative covering value this rule provokes over it, and removes from it the examples with a covering

value greater than  $\varepsilon$ . The covering method is developed as follows:

1. Initialization:
  - (a) Introduce  $k$ ,  $\omega$  and  $\varepsilon$ .
  - (b) Set the example covering degree  $CV[l] \leftarrow 0$ ,  $l = 1, \dots, p$ .
  - (c) Initialize the final set of rules  $B^g$  to empty.
2. Over the set of examples  $E_p$ , apply the generating method, obtaining as output the best fuzzy rule  $R_r$  according to the current state of  $E_p$ .
3. Introduce  $R_r$  in  $B^g$ .
4. For every  $e_l \in E_p$  do
  - (a)  $CV[l] \leftarrow CV[l] + R_r(e_l)$ ,
  - (b) If  $CV[l] \geq \varepsilon$  then remove it from  $E_p$ .
5. If  $E_p = \emptyset$  then Stop else return to Step 2.

### 5. The genetic multisimplification process

As commented, in this process, we follow the idea of obtaining different simplified FRBs from the fuzzy rule set generated in the previous stage presenting the best possible cooperation among the fuzzy rules composing them. The sequential niche technique [5] is used to induce niches in this GFRBS stage, with the genetic simplification process proposed in [28] being the basic optimization technique iterated at each run of the multisimplification process. Next subsections introduce respectively the basic simplification algorithm and the particular aspects of the multisimplification one.

#### 5.1. The basic genetic simplification process

It is based on a binary coded GA, in which the selection of the individuals is developed using the

stochastic universal sampling procedure together with an elitist selection scheme, and the generation of the offspring population is put into effect by using the classical binary multipoint crossover (performed at two points) and uniform mutation operators.

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

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

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

As regards the fitness function,  $E(\cdot)$ , it is based on the mean square error (SE) over a training data set,  $E_{TDS}$ , which is represented by the following expression:

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

where  $S(ex^\ell)$  is the output value obtained from the FRBS using the FRB coded in  $C_j$ ,  $R(C_j)$ , when the input variable values are  $ex^\ell = (ex_1^\ell, \dots, ex_n^\ell)$ , and  $ey^\ell$  is the known desired value.

There is a need to keep the *completeness property* considered in the previous stage. We shall ensure this condition by forcing every example contained in the training set to be covered by the encoded FRB to a degree greater than or equal to  $\tau$ ,

$$C_{R(C_j)}(e_\ell) = \bigcup_{j=1..T} R_j(e_\ell) \geq \tau, \\ \forall e_\ell \in E_{TDS} \text{ and } R_j \in R(C_j)$$

where  $\tau$  is the minimal training set completeness degree accepted in the simplification process. Usually,  $\tau$  is less than or equal to  $\omega$ , the compatibility degree used in the generation process.

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

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

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

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

### 5.2. The genetic multisimplification process

In order to induce niching in the sequential niche algorithm, there is a need to define any kind of *distance metric* which, given two individuals, returns a value of how close they are [5]. We use a *genotypic sharing* due to the metric considered is the Hamming distance measured on the binary coding space. With  $A = (a_1, \dots, a_m)$  and  $B = (b_1, \dots, b_m)$  being two individuals, it is defined as follows:

$$H(A, B) = \sum_{i=1}^m |a_i - b_i|.$$

Making use of this metric, the *modified fitness function* guiding the search on the multisimplification process is based on modifying the value associated to an individual by the basic algorithm fitness function, multiplying it by a *derating function* penalizing the closeness of this individual to the solutions previously obtained. Hence, the modified fitness function used by the multisimplification process is the following:

$$F'(C_j) = F(C_j) \cdot G(C_j, S),$$

where  $F$  is the basic genetic simplification process fitness function,  $S = \{s_1, \dots, s_k\}$  is the set containing the  $k$  solutions yet found, and  $G$  is a kind of *derating function*. We consider the following taking into account the fact that the problem we deal with is a minimization one:

$$G(C_j, S) = \begin{cases} \infty & \text{if } d = 0, \\ 2 - \left(\frac{d}{r}\right)^\beta & \text{if } d < r \text{ and } d \neq 0, \\ 1 & \text{if } d \geq r, \end{cases}$$

where  $d$  is the minimum value of the Hamming distance between  $C_j$  and the solutions  $s_i$  included in  $S$ , i.e.,  $d = \text{Min}_i \{H(C_j, s_i)\}$ , and the penalization is considered over the most close solution,  $r$  is the *niche*

radius, and  $\beta$  is the power factor determining how concave ( $\beta > 1$ ) or convex ( $\beta < 1$ ) the derating curve is. Therefore, the penalization given by the derating function takes its maximum value when the individual  $C_j$  encodes one of the solutions yet found. There is no penalization when the  $C_j$  is far away from  $S$  in a value greater than or equal to the niche radius.

The algorithm of the genetic multisimplification process is shown below:

1. Initialization: Equate the multisimplification modified fitness function to the basic simplification fitness function:  $F'(C_j) \leftarrow F(C_j)$ .
2. Run the basic genetic simplification process, using the modified fitness function, keeping a record of the best individual found in the run.
3. Update the modified fitness function to give a depression in the region near the best individual, producing a new modified fitness function.
4. If not all the simplified FRBs desired have been obtained, return to step 2.

Hence, the number of runs of the sequential algorithm performed is the number of solutions that are desired to obtain. We allow the FRBS designer to decide this number as well as the values of the parameters  $r$  and  $\beta$ .

### 6. The genetic tuning process

The genetic tuning process was presented in-depth in [26]. It is based on the existence of a previous complete FRB constituted by  $m$  fuzzy rules.

Each chromosome forming the genetic population will encode a complete FRB definition and the genetic process will adapt the membership functions of each individual fuzzy rule.

The GA designed for the tuning process presents a real coding issue, uses the stochastic universal sampling as selection procedure and Michaeliewicz's non-uniform mutation operator. As regards the crossover operator, the max–min–arithmetical is employed again.

As we have said, the membership functions are triangular-shaped. Thus, each one of them has an associated parametric representation based on a 3-tuple of real values. Each one of the fuzzy rules in the FRB will be encoded in pieces of chromosome  $C_{ri}$ ,  $i = 1, \dots, m$ , in the following way:

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

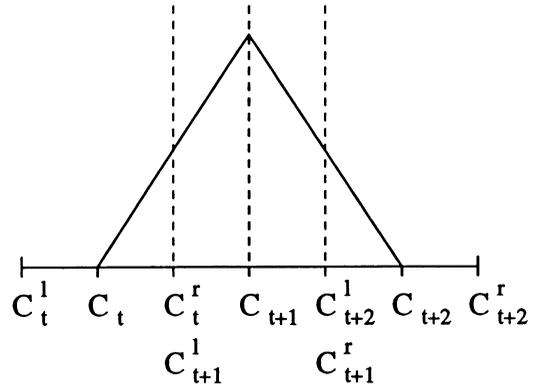


Fig. 5. Membership function and intervals of performance for the tuning process.

Therefore, the complete FRB definition is represented by a complete chromosome  $C_r$ :

$$C_r = C_{r1} C_{r2} \dots C_{rm}$$

The initial gene pool is created from the initial FRB. This FRB is encoded directly into a chromosome, denoted as  $C_1$ . The remaining individuals are generated by associating an interval of performance,  $[c_h^l, c_h^r]$  to every gene  $c_h$  in  $C_1$ ,  $h = 1 \dots (n+1) \cdot m \cdot 3$ . Each interval of performance will be the interval of adjustment for the correspondent variable,  $c_h \in [c_h^l, c_h^r]$ .

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

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

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

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

Fig. 5 shows these intervals.

Therefore, we create a population of chromosomes containing  $C_1$  as its first individual and the remaining

ones initiated randomly, with each gene being in its respective interval of performance.

The fitness function used will be the same considered in the previous stage, the genetic multisimplification process, as it is done in [12].

## 7. Practical application of the genetic fuzzy rule-based system to a real-world electrical engineering problem

In order to analyze the accuracy of the method proposed, we are going to use it to solve a real-world electrical engineering problem consisting on obtaining a model relating the length of line in a rural population with its characteristics [38,39]. We shall compare the behaviour of our multi-stage GFRBS in the solving of the problem with the one presented by classical methods, neural networks and other multi-stage GFRBSs presenting different characteristics.

To do so, first we shall introduce the commented application in the next subsection. Then, we analyze the use of different techniques to solve it: classical regression methods, neural modeling and GFRBS fuzzy modeling will be analyzed. Finally, the results obtained by them will be compared.

### 7.1. The electrical engineering problem considered

In Spain, electrical industries do not charge the energy bill directly to the final user, but they share the ownership of an enterprise (called REE, Red Eléctrica de España) which gets all payments and then distributes them according to some complex criteria (amount of power generation of every company, number of customers, etc.)

Recently, some of these companies have asked to redistribute the maintenance costs of the network. Since maintenance costs depend on the total length of electrical line each company owns, and on their kind (high, medium, urban low and rural low voltage) it was necessary to know the exact length of every kind of line each company was maintaining.

High and medium voltage lines can be easily measured. But low voltage line is contained in cities and villages, and it would be very expensive to measure it. This kind of line uses to be very convoluted and, in some cases, one company may serve more than 10 000

Table 1  
Notation considered for the problem variables

Symbol	Meaning
$A_i$	Number of clients in population $i$
$R_i$	Radius of $i$ population in the sample
$n$	Number of populations in the sample
$l_i$	Line length, population $i$
$\tilde{l}_i$	Estimation of $l_i$

small nuclei. An indirect method for determining the length of line is needed.

Therefore, there is a need to find a relationship between the population and size of a certain area and the length of line in it, making use of some known data, that may be employed to predict the real length of line in any other village.

We shall try to solve this problem by generating different kind of models determining the unknown relationship. To do so, we were provided with the measured line length, the number of inhabitants and the mean distance from the center of the town to the three furthest clients in a sample of 495 rural nuclei [38,39]. Our variables are named as shown in Table 1.

### 7.2. Application of classical methods

In order to apply classical methods, we needed to make some hypothesis [39]. In the populations that are being studied, electrical networks are star-shaped and arranged in sectors. A main line passes near all clients inside them, and clients are connected to these main lines by small segments (see Fig. 6).

To build a theoretical simplified model we have admitted that:

- A population comprises  $s_i$  sectors. Each sector covers an angle  $2\theta_i$ . All sectors in the same population cover the same angle. Each sector is served by one output of the only transformation center in the village.
- All sectors in a population have the same radius,  $R_i$ .
- Clients are uniformly distributed inside every sector.
- Inside a sector, the electrical line comprises a main nerve of length  $R_i$  and so many branches as consumers.

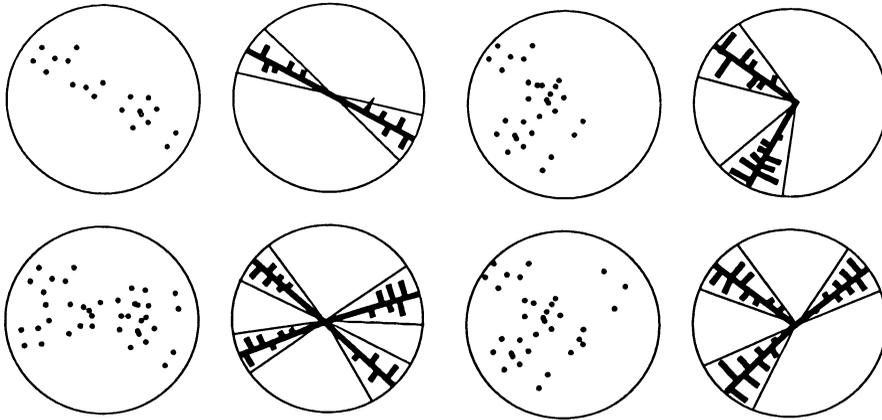


Fig. 6. Models of some kind of nuclei.

If we admit that customers are uniformly distributed, we can approximate the total length by multiplying the mean distance between one of them and the nerve by the number of inhabitants. Let us name this mean distance  $d_i$  for population  $i$ , and let the sector be  $2\theta_i$  wide. Then

$$d_i = \frac{2(1 - \cos \theta_i)}{3\theta_i} R_i$$

so the cable length will be

$$\tilde{l}_i = s_i \left( R_i + \frac{A_i}{s_i} d_i \right) = s_i R_i + A_i \frac{2(1 - \cos \theta_i)}{3\theta_i} R_i.$$

### 7.3. Classical regression adjust

If the angles  $\theta_i$  and the numbers  $s_i$  were similar enough between them, we could regard them as constants and estimate them by the parameters  $\bar{\theta}_i = \theta$  and  $\bar{s}_i = s$  of a least-squares linear regression

$$\tilde{l}_i/R_i = s + k(\theta)A_i$$

to a set of pairs  $(x, y) = (A_i, \tilde{l}_i/R_i)$ .

We can get a better adjustment by allowing a certain dependence between the number of sectors, their angles and the number of inhabitants. This can be done by dividing the sample into classes or by means of a change of variables. Both cases were studied, and the best adjust was obtained with the model

$$\frac{\tilde{l}_i}{R_i} = k_1 A_i^{k_2}.$$

### 7.4. GFRBS fuzzy modeling

Different ways of developing a fuzzy modeling of the introduced problem are going to be compared by using several FRBS design methods following different types of fuzzy models.

As regards the descriptive fuzzy model, the following three processes are considered:

- D1.** A two-stage GFRBS based on obtaining a complete knowledge base by deriving the rule base by means of the Wang and Mendel's (WM) method [43] in the first stage, and defining the data base by means of the descriptive genetic tuning process presented in [12,14] in the second.
- D2.** A two-stage GFRBS based on obtaining a complete knowledge base by deriving the rule base by means of the Thrift's genetic learning process [42] in the first stage, and defining the data base by means of the same descriptive genetic tuning process used above.
- D3.** The three-stage descriptive GFRBS design method proposed in [12,14].

On the other hand, when working with the approximate one, the following three multi-stage GFRBSs are employed:

- A1.** A two-stage GFRBS based on obtaining a complete FRB by generating a preliminary definition by means of the weighted counting algorithm [3], and refining it by adjusting the membership function definitions using the approximate genetic tuning process used in this paper (see Section 6).

- A2.** The three-stage unconstrained approximate GFRBS proposed in [11,12].
- A3.** The three-stage constrained approximate GFRBS proposed in this paper.

The initial data base definition used in the generating processes of the descriptive GFRBS is constituted by three primary fuzzy partitions (two corresponding to the input variables and one associated to the output one) formed by *five fuzzy sets* (as shown in Fig. 1), and the adequate scaling factors to translate the generic universe of discourse into the one associated with each problem variable. The same primary fuzzy partitions have been considered to define the performance intervals for the approximate GFRBS generating processes.

Since GFRBS **D3**, **A2** and **A3** presents the same structure, following the assumptions presented in [18], they share many parameters. The values considered for these parameters are the following:  $\varepsilon = 1.5$ ,  $\omega = 0.05$ ,  $k = 0.1$ , and  $\tau = 0.1$ . For the experiments developed with GFRBSs **D3** and **A2**, we have worked with the parameter values giving best results in [11–14].

With respect to the remaining parameters of GFRBS **A3**, the t-norm \* used in the fuzzy rule generation process is the Minimum, the generating process GA runs over 100 generations, the ES is applied until there is no improvement in 25 generations over a percentage  $\alpha = 20\%$  of the individuals of the population (the parameter  $c$  of the  $\frac{1}{5}$ -success rule is equal to 0.9). The genetic multisimplification process generates three different FRBs for run (each time, the basic GA runs over 500 generations), the niche radius  $r$  is equal to a 10 percent of the number of rules in the initial FRB, and the power factor  $\beta$  is equal to 0.5. The genetic tuning processes run over 1000 generations. In all cases, the population is formed by 61 individuals, the value of the non-uniform mutation parameter  $b$  is 5.0, and the crossover and mutation rates are, respectively,  $P_c = 0.6$  and  $P_m = 0.1$  (this last one per individual). The max–min–arithmetic crossover parameter  $a$  takes the value 0.35.

As regards GFRBSs **D1**, **D2** and **A1**, the parameters considered in the second stage, the genetic tuning process, are the same shown above. While WM generating process does not consider any parameter, the values associated to the Thrift one in **D2** and the weighted counting algorithm one in **D3** are the following:

- *Thrift*: Population size: 61,  $P_c = 0.6$ ,  $P_m = 0.1$  and number of generations: 1000.
- *Weighted counting algorithm*:  $\alpha = 0.5$ .

Finally, the FRBS reasoning method used in all the processes have been the same. We have selected the *minimum t-norm* playing the role of the implication and conjunctive operators, and the *center of gravity weighted by the matching* strategy acting as the defuzzification operator [17].

### 7.5. Comparison between methods

To compare classical methods, GFRBS fuzzy modeling and neural modeling we have randomly divided the sample into two sets comprising 396 and 99 samples. The SE values over these two sets are labeled *training* and *test*. In this case, we define SE as

$$\frac{1}{2 \cdot N} \sum_{i=1}^N (\tilde{l}_i - l_i)^2.$$

The results obtained in the different experiments developed with the GFRBSs considered are collected in Table 2 where  $\#R$  stands for the number of rules of the corresponding FRB, and  $SE_{tra}$  and  $SE_{tst}$  for the values obtained in the SE measure computed over the training and test data sets, respectively.

In view of the results obtained, we have to remark the good performance of the genetic multisimplification process. In the second iteration, it allows us to generate a fuzzy model with better approximative and predictive behavior, i.e., less value in the SE over both data sets, than the first one obtained. Nevertheless, the fuzzy model generated in the third iteration presents worse behaviour than both previous ones.

A graphical representation of the behavior of the crossover operators used is shown in Fig. 7. As was commented in Section 4.1.4, the expected behavior consists of an initial phase where a high number of standard crossovers and a very small of max–min–arithmetic ones are developed, and a second phase where the number of max–min–arithmetic crossovers rises a lot and the application of the standard crossover decreases. The figure, drawn making use of the data collected in the first iteration of the experiment developed, shows clearly this behavior.

Once we have analyzed individually the behaviour presented by the proposed GFRBS, we are going to

Table 2  
Results obtained by the multi-stage GFRBSs in the problem being solved

Generation GFRBS	#R	SE <sub>tra</sub>	SE <sub>1st</sub>	Multisimplification			Tuning	
				#R	SE <sub>tra</sub>	SE <sub>1st</sub>	SE <sub>1st</sub>	EC <sub>prue</sub>
<b>D1</b>	13	298446.0	282058.1				175337.9	180102.7
<b>D2</b>	25	218591.9	204426.8				154314.0	199551.3
<b>D3</b>	47	286794.8	285869.0	22	168603.2	174246.6	143921.9	177790.0
<b>A1</b>	20	356434.3	311195.0				175887.2	180211.4
<b>A2</b>	31	431904.0	435649.5	19	226403.9	222550.9	148036.9	191339.5
<b>A3</b>	57	185577.6	228294.2	36	135893.7	203569.7	111250.0	177106.8
				33	132738.0	194710.8	108203.1	166186.7
				33	135871.8	204459.0	111742.9	185395.6

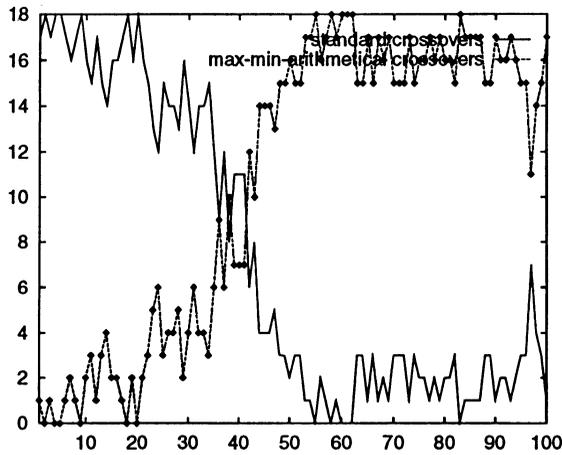


Fig. 7. Number of crossovers per generation in the first generation of the experiment developed.

compare its accuracy with the remaining techniques considered. Table 3 shows the results obtained by all of them in the problem. To obtain them, the parameters of the polynomial models were fitted by Levenberg–Marquardt, while exponential and linear models were fitted by linear least squares. The multilayer perceptron was trained with the QuickPropagation algorithm. The number of neurons in the hidden layer was chosen to minimize the test error.

In view of the results shown, the constrained approximate GFRBS proposed in this paper has presented the best behaviour. It outperforms the other techniques considered by obtaining the best values in the SE computed over both data sets, the training and test ones. Therefore, the constrained approximate FRBS generated is the model that best approximates

Table 3  
Results obtained in the problem being solved

Method	SE <sub>tra</sub>	SE <sub>1st</sub>
Linear	287 775	209 656
Exponential	232 743	197 004
Second-order polynomial	235 948	203 232
Third-order polynomial	235 934	202 991
Three-layer perceptron 2-25-1	169 399	167 092
<b>D1</b>	175 337	180 102
<b>D2</b>	154 314	199 551
<b>D3</b>	143 921	177 790
<b>A1</b>	175 887	180 211
<b>A2</b>	148 036	191 339
<b>A3</b>	108 203	166 186

the real system and that presents best generalization capabilities.

We have to note that the fuzzy model obtained is more accurate to a high degree than the neural one, which is the second best model in view of its generalization level. Although the results does not differ too much in this characteristic (166 186 vs. 167 092), the value obtained by the constrained approximate FRBS in the SE over the training data set shows a large performance advantage for it over the neural network (108 203 vs. 169 399).

Hence, we have been able to generate a model that is more accurate and more interpretable at the same time. This is due to, although approximate Mamdani-type FRBs are less readable than descriptive ones, approximate FRBSs are more interpretable than neural networks because of the following two main reasons:

- The approximate fuzzy model is locally interpretable. We are always able to know which fuzzy

rules in the FRB are fired when the system receives a specific input.

- The parameters involved in an approximate fuzzy model have a real-world meaning understandable by a human since they define membership functions. However, it is difficult to interpret the meaning of the neural network weights.

## 8. Concluding remarks

A multi-stage GFRBS has been presented for designing constrained approximate FRBSs by learning the FRB from examples combining a hybrid GA-ES generation process based on the iterative rule learning approach and two GA-based simplification and tuning processes. Its performance in a real-world electrical engineering problem has been shown and compared with classical methods, neural networks and other descriptive and unconstrained approximate GFRBSs. The proposed evolutionary learning process has obtained the best results.

## Acknowledgements

We would like to thank Luciano Sánchez, from Oviedo University, for the Electrical Engineering application from Hidroeléctrica del Cantábrico and for solving it by means of classical and neural techniques.

## Appendix A. Genetic algorithms and evolution strategies

### A.1. Genetic algorithms

GAs are general-purpose search algorithms that use principles inspired by natural population genetics to evolve solutions to problems. They were first proposed by Holland [32] and are well analyzed in some books such as [23]. GAs are theoretically and empirically proven to provide a robust search in complex spaces, thereby offering a valid approach to problems requiring efficient and effective searches.

Any GA starts with a population of randomly generated solutions, chromosomes, and advances toward better solutions by applying genetic operators, modeled on the genetic processes occurring in nature. In

these algorithms we maintain a population of solutions for a given problem; this population undergoes evolution in the form of natural selection. In each generation, relatively good solutions reproduce to give offspring that replace the relatively bad solutions which die. An evaluation or fitness function plays the role of the environment to distinguish between good and bad solutions. The process of going from the current population to the next population constitutes one generation in the execution of a GA.

### A.2. Niching genetic algorithms

GAs are known to be a powerful tool for performing search in complex spaces. Anyway, one drawback they present is that when dealing with multimodal functions with peaks of unequal value, simple GAs are characterized by converging to the best peak of the space (or to a space zone containing several of the best peaks) and to lose an adequate individual sampling over other peaks in other space zones. This phenomenon is called *genetic drift* [19] and is not a correct behavior for several kinds of problems in which one may be interested in knowing the location of other function optima.

The *niche* and *species* concepts were introduced in order to overcome this behavior [19,23]. As the great majority of the GA concepts, they are based on translating natural notions to the field of GAs. In nature, a niche is viewed as an organism's task in the environment and a species is a collection of individuals with similar features. In this way, the formation of stable subpopulations of organisms surrounding separate niches by forcing similar individuals to share the available resources is induced.

One of the most usually employed methods for introducing niche and species in GAs is based on the *individual fitness sharing* [19,23]. In this scheme, the population is divided in different subpopulations (species) according to the similarity of the individuals. These subpopulations form niches in two possible solution spaces: the gene and the decoded parameter ones, *genotypic* and *phenotypic sharing* respectively. Acting as in nature, the individuals belonging to each niche share the associated payoff among them. A *sharing function* is defined to determine the neighbourhood and degree of sharing for each string in the population.

### A.3. Evolution strategies

ESs [2,40] were initially developed by Rechenberg and Schwefel in 1964 with a strong focus on building systems capable of solving difficult real-valued parameter optimization problems. The natural representation was a vector or real-valued genes which were manipulated primarily by mutation operators designed to perturb the real-valued parameters in useful ways.

The first ES algorithm, the so-called (1 + 1)-ES, was based on working with only two individuals per generation, one parent and one descendent. It is based on evolving the parent string by applying a mutation operator to each one of its components. The mutation strength is determined by a value  $\sigma$ , a standard deviation of a normally distributed random variable. This parameter is associated to the parent and it is evolved in each process step as well. If the evolution has been performed successfully, then the descendent substitutes the parent in the next generation. The individual adaptation is measured by using a fitness function. The process is iterated until a determined finishing condition is satisfied.

The mutation operator **mut** is composed of two components. The first one, **mu $_{\sigma}$** , evolves the value of the standard deviation  $\sigma$  using Rechenberg's 1/5-success rule:

$$\sigma' = \mathbf{mu}_{\sigma}(\sigma) = \begin{cases} \frac{\sigma}{\sqrt[5]{c}}, & \text{if } p > \frac{1}{5}, \\ \sigma \cdot \sqrt[5]{c}, & \text{if } p < \frac{1}{5}, \\ \sigma, & \text{if } p = \frac{1}{5}, \end{cases}$$

where  $p$  is the relative frequency of successful mutations and  $c$  is a constant determining the updating amount of  $\sigma$ .

The second one, **mu $_x$** , mutates each component of the real coded string by adding normally distributed variations with standard deviation  $\sigma'$  to it:

$$x' = \mathbf{mu}_x(x) = (x_1 + z_1, \dots, x_n + z_n),$$

where  $z_i \sim N_i(0, \sigma'^2)$ .

### References

- [1] T. Bäck, *Evolutionary Algorithms in Theory and Practice*, Oxford University Press, Oxford, 1996.
- [2] T. Bäck, H.-P. Schwefel, Evolution strategies I: variants and their computational implementation, in: J. Periaux, G.

- Winter, M. Galán, P. Cuesta (Eds.), *Genetic Algorithms in Engineering and Computer Science*, Wiley, New York, 1995, pp. 111–126.
- [3] A. Bardossy, L. Duckstein, *Fuzzy Rule-Based Modeling with Application to Geophysical, Biological and Engineering Systems*, CRC Press, Boca Raton, FL, 1995.
- [4] A. Bastian, How to handle the flexibility of linguistic variables with applications, *Internat. J. Uncertainty, Fuzziness Knowledge-Based Systems* 2 (4) (1994) 463–484.
- [5] D. Beasley, D.R. Bull, R.R. Martin, A sequential niche technique for multimodal function optimization, *Evolutionary Comput.* 1 (2) (1993) 101–125.
- [6] A. Bonarini, Evolutionary learning of fuzzy rules: competition and cooperation, in: W. Pedrycz (Ed.), *Fuzzy Modelling: Paradigms and Practice*, Kluwer Academic Press, Dordrecht, 1996, pp. 265–283.
- [7] P.P. Bonissone, Soft computing: the convergence of emerging reasoning technologies, *Soft Computing* 1 (1) (1997) 6–18.
- [8] B. Carse, T.C. Fogarty, A. Munro, Evolving fuzzy rule based controllers using genetic algorithms, *Fuzzy Sets and Systems* 80 (1996) 273–294.
- [9] O. Cordón, F. Herrera, A general study on genetic fuzzy systems, in: J. Periaux, G. Winter, M. Galán, P. Cuesta (Eds.), *Genetic Algorithms in Engineering and Computer Science*, Wiley, New York, 1995, pp. 33–57.
- [10] O. Cordón, F. Herrera, A hybrid genetic algorithm-evolution strategy process for learning fuzzy logic controller knowledge bases, in: F. Herrera, J.L. Verdegay (Eds.), *Fuzzy Logic and Soft Computing*, Physica-Verlag, Wierzburg, 1996, pp. 251–278.
- [11] O. Cordón, F. Herrera, Generating and selecting fuzzy control rules using evolution strategies and genetic algorithms, *Proc. Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU'96)*, Granada, Spain, 1996, pp. 733–738.
- [12] O. Cordón, F. Herrera, A three-stage evolutionary process for learning descriptive and approximate fuzzy logic controller knowledge bases, *Internat. J. Approx. Reasoning* 17(4) (1997) 369–407.
- [13] O. Cordón, F. Herrera, Identification of linguistic fuzzy models by means of genetic algorithms, in: D. Driankov, H. Hellendoorn (Eds.), *Fuzzy Model Identification. Selected Approaches*, Springer, Berlin, 1997, pp. 215–250.
- [14] O. Cordón, F. Herrera, M. Lozano, A three-stage method for designing genetic fuzzy systems by learning from examples, in: H.M. Voight, W. Ebeling, E. Rechemberg, H.P. Schwefel (Eds.), *Proc. 4th Internat. Conf. on Parallel Problem Solving from Nature (PPSN IV)*, Berlin, Germany, Lecture Notes in Computer Science, vol. 1141, Springer, Berlin, 1996, pp. 720–729.
- [15] O. Cordón, F. Herrera, M. Lozano, A classified review on the combination fuzzy logic-genetic algorithms bibliography: 1989–1995, in: E. Sanchez, T. Shibata, L. Zadeh (Eds.), *Genetic Algorithms and Fuzzy Logic Systems. Soft Computing Perspectives*, World Scientific, Singapore, 1997, pp. 209–241.
- [16] O. Cordón, F. Herrera, M. Lozano, On the combination of fuzzy logic and evolutionary computation: a short review

- and bibliography, in: W. Pedrycz (Ed.), *Fuzzy Evolutionary Computation*, Kluwer Academic Press, Dordrecht, 1997, pp. 57–77.
- [17] O. Cordón, F. Herrera, A. Peregrín, Applicability of the fuzzy operators in the design of fuzzy logic controllers, *Fuzzy Sets and Systems* 86 (1997) 15–41.
- [18] O. Cordón, M.J. del Jesus, F. Herrera, M. Lozano, An evolutionary paradigm for designing fuzzy rule-based systems from examples, *Proc. 2nd IEE/IEEE Internat. Conf. on Genetic Algorithms and Engineering Systems: Innovations and Applications (GALESIA'97)*, Glasgow, UK, 1997, pp. 139–144.
- [19] K. Deb, D.E. Goldberg, An investigation of niche and species formation in genetic function optimization, *Proc. 2nd Internat. Conf. on Genetic Algorithms*, Lawrence Erlbaum, Hillsdale, NJ, 1989, pp. 42–50.
- [20] K.A. De Jong, Learning with genetic algorithms: an overview, *Machine Learning* 3 (1988) 121–138.
- [21] M. Delgado, A.F. Gómez Skarmeta, F. Martín, A fuzzy clustering based rapid-prototyping for fuzzy rule-based modeling, *IEEE Trans. Fuzzy Systems* 5 (2) (1997) 223–233.
- [22] D. Driankov, H. Hellendoorn, M. Reinfrank, *An Introduction to Fuzzy Control*, Springer, Berlin, 1993.
- [23] D.E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, New York, 1989.
- [24] A. González, F. Herrera, Multi-stage genetic fuzzy systems based on the iterative rule learning approach, *Mathware & Soft Computing* 4 (3) (1997) 233–249.
- [25] A. González, R. Pérez, Completeness and consistency conditions for learning fuzzy rules, *Fuzzy Sets and Systems* 96 (1998) 37–51.
- [26] F. Herrera, M. Lozano, J.L. Verdegay, Tuning fuzzy controllers by genetic algorithms, *Internat. J. Approx. Reasoning* 12 (1995) 299–315.
- [27] 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.
- [28] F. Herrera, M. Lozano, J.L. Verdegay, A learning process for fuzzy control rules using genetic algorithms *Fuzzy Sets and Systems* 100 (1998) 143–158.
- [29] F. Herrera, J.L. Verdegay (Eds.), *Genetic Algorithms and Soft Computing*, Physica-Verlag, Wurzburg, 1996.
- [30] F. Herrera, L. Magdalena, Genetic fuzzy systems, in: R. Mesiar, B. Riccan (Eds.), *Tatra Mountains Mathematical Publications*, vol. 13: “Fuzzy Structures. Current Trends”, Lecture Notes of the Tutorial: Genetic Fuzzy Systems, Seventh IFSA World Congress (IFSA'97), 1997, pp. 93–121.
- [31] K. Hirota (Ed.), *Industrial Applications of Fuzzy Technology*, Springer, Berlin, 1993.
- [32] J.H. Holland, *Adaptation in Natural and Artificial Systems*, The University of Michigan Press, Ann Arbor: 1975; The MIT Press, London, 1992.
- [33] K. Krishnakumar, A. Satyadas, Evolving multiple fuzzy models and its application to an aircraft control problem, in: J. Periaux, G. Winter, M. Galán, P. Cuesta (Eds.), *Genetic Algorithms in Engineering and Computer Science*, Wiley, New York, 1995, pp. 305–320.
- [34] C.C. Lee, Fuzzy logic in control systems: fuzzy logic controller – parts I and II, *IEEE Trans. Systems Man Cybernet.* 20 (1990) 404–435.
- [35] L. Magdalena, J.R. Velasco, Fuzzy rules-based controllers that learn by evolving its knowledge base, in: F. Herrera, J.L. Verdegay (Eds.), *Fuzzy Logic and Soft Computing*, Physica-Verlag, Wurzburg, 1996, pp. 172–201.
- [36] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, Springer, Berlin, 1996.
- [37] W. Pedrycz (Ed.), *Fuzzy Modelling. Paradigms and Practice*, Kluwer Academic Press, Dordrecht, 1996.
- [38] L. Sánchez, Interval-valued GA-P algorithms, Technical Report, Computer Science Department, Oviedo University, Oviedo, Spain, 1997.
- [39] L. Sanchez, Estudio de la red asturiana de baja tensión rural y urbana, Confidential Report Hidroeléctrica del Cantábrico Research and Development Department, 1997.
- [40] H.-P. Schwefel, *Evolution and Optimum Seeking*, Sixth-Generation Computer Technology Series, Wiley, New York, 1995.
- [41] T. Takagi, M. Sugeno, Fuzzy identification of systems and its application to modeling and control, *IEEE Trans. Systems Man Cybernet.* 15 (1) (1985) 116–132.
- [42] P. Thrift, Fuzzy logic synthesis with genetic algorithms, *Proc. 4th Internat. Conf. on Genetic Algorithms (ICGA'91)*, 1991, pp. 509–513.
- [43] L.X. Wang, J.M. Mendel, Generating fuzzy rules by learning from examples, *IEEE Trans. Systems Man Cybernet.* 22 (1992) 1414–1427.
- [44] R.R. Yager, L.A. Zadeh (Eds.), *An Introduction to Fuzzy Logic Applications in Intelligent Systems*, Kluwer Academic Press, Dordrecht, 1992.
- [45] M. Yagiura, T. Ibaraki, Genetic and local search algorithms as robust and simple optimization tools, in: I.H. Osman, J.P. Kelly (Eds.), *Meta-heuristics: Theory and Applications*, Kluwer Academic Press, Dordrecht, 1996, pp. 63–82.