Encouraging cooperation in the genetic iterative rule learning approach for qualitative modeling

O. Cordón, A. González, F. Herrera, R. Pèrez

Department of Computer Science and Artificial Intelligence
University of Granada
18071 - Granada, Spain

Abstract. Genetic Algorithms have proven to be a powerful tool for automating the Fuzzy Rule Base definition and, therefore, they have been widely used to design descriptive Fuzzy Rule-Based Systems for Qualitative Modeling. These kinds of genetic processes, called Genetic Fuzzy Rule-Based Systems, may be based on different genetic learning approaches, with the Michigan and Pittsburgh being the most well known ones.

In this contribution, we briefly review another alternative, the Iterative Rule Learning approach, based on generating a single rule in each genetic run, and dealing with the problem of obtaining the best possible cooperation among the generated fuzzy rules. Two different ways for encouraging cooperation between rules in this genetic learning approach are presented, which are used in two different Genetic Fuzzy Rule-Based Systems based on it, SLAVE and MOGUL. Finally, the behaviour of these two processes in solving a qualitative modeling problem, the rice taste analysis, is analysed, and the results obtained are compared with two other design processes with different characteristics.

Keywords. Fuzzy Logic, Fuzzy Rules, Fuzzy Rule-Based Systems, Qualitative Modeling, Genetic Algorithms, Genetic Fuzzy Rule-Based Systems.

1 Introduction

Genetic Algorithms (GAs) are search algorithms that use operations found in natural genetics to guide the trek through a search space. GAs are theoretically and empirically proven to provide robust search capabilities in complex spaces, offering a valid approach to problems requiring efficient and effective searching [14]. Although GAs are not learning algorithms, they may offer a powerful and domain-independent search method for a variety of learning tasks. In fact, there has been a good deal of interest in using GAs for machine learning problems [21].

Fuzzy Rule Based Systems (FRBSs), initiated by Mamdani applied to control problems, are now considered as one of the most important applications of fuzzy set theory. FRBSs are knowledge-based systems that make use of the known

knowledge of the process, expressed in the form of fuzzy rules collected in the fuzzy rule base (FRB). They have been applied to a wide range of areas [2]. A *descriptive FRBS* is a model that is described or expressed using linguistic terms in the framework of fuzzy logic. A crucial reason why the descriptive fuzzy rule-based approach is worth considering is that it may remain verbally interpretable. This FRBS has been widely used and has obtained very good results in many different applications [2].

Focusing on the use of GAs in the field of fuzzy modeling, particularly in FRBS, GAs have proven to be a powerful tool for automating the definition of the FRB, since adaptive control, learning and self-organizing fuzzy systems may be considered in a lot of cases as optimization or search processes. Their advantages have extended the use of GAs in the development of a wide range of approaches for designing fuzzy systems in the last few years. These approaches receive the general name of *Genetic Fuzzy Systems* (GFSs) and *Genetic Fuzzy Rule Based Systems* (GFRBSs) [7, 23]. Among the different approaches considered in the genetic learning of FRBs there is the Iterative Rule Learning (IRL) approach [15, 20] that is based on the coding of one rule per chromosome, selecting one rule per population by including the GA in an iterative scheme based on obtaining the best current rule for the system, and incorporating this rule into the final FRB.

In this contribution we deal with genetic learning processes based on the IRL approach for designing descriptive FRBSs. In particular, we shall focus on the analysis of the cooperation in the genetic learning processes for qualitative modeling using linguistic fuzzy rules. We present two alternatives used for introducing cooperation in two different GFRBSs. The first one, used in SLAVE (Structural Learning Algorithm in Vague Environments) [15, 17, 19], modifies the iterative process in order to obtain rules that cooperate with the previously learnt rules, and the second one, used in MOGUL (Methodology to Obtain GFRBSs Under the IRL approach) [10, 11], dividing the genetic learning process into, at least, two stages, thereby achiving cooperation between the fuzzy rules generated in the second stage.

In order to do this we organize the contribution as follows: Section 2 introduces some preliminaries such as descriptive FRBSs (qualitative modeling), Evolutionary and Genetic Algorithms, and GFRBSs; Section 3 studies the IRL approach and the problem of the lack of cooperation between rules; Section 4 presents two alternatives for including collaboration in the genetic learning processes based on the IRL approach; Section 5 shows some experimental results; and finally, Section 6 presents some concluding remarks.

2 Preliminaries

2.1 Qualitative modeling: Descriptive FRBSs

Qualitative modeling based on fuzzy logic is considered as a system model based on linguistic descriptions [36]. The linguistic descriptions are represented by fuzzy

membership functions, and they are used in an FRB composed of linguistic IF-THEN rules such as:

IF (a set of conditions are satisfied) THEN (a set of consequences may be inferred)

The contents of both IF- and THEN-parts are usually expressed in terms of linguistic variables. In an FRBS, the compositional rule of inference is used to draw conclusions from the set of known premises. Thereby the concept of linguistic variable [40] plays a central role.

There are different kinds of FRBSs in the literature, amongst which we should mention the Mamdani, TSK and DNF models.

1. The generic expression of the TSK rules is the following:

   IF X_1 is a_1 and ... and X_n is a_n THEN $Y = p_1 \cdot X_1 + \ldots + p_n \cdot X_n + p_0$

   where $X_1, ..., X_n$ and Y are the input variables and the output variable, respectively, a_i are linguistic variables with an associated fuzzy set defining their semantics, and p_i are real numbers.
2. In the Mamdani model, the FRB is composed of a collection of fuzzy rules with the following structure:

   IF X_1 *is* a_1 *and* ... *and* X_p *is* a_n THEN Y *is* B
3. The DNF model is an extension of the Mamdani model with the following structure:

   IF X_1 *is* A_1 *and* ... *and* X_n *is* A_n THEN Y *is* B

   where each variable X_i has a referential set U_i and takes values in a finite domain D_i, for $i \in \{1, \ldots, n\}$. The referential set for Y is V and its domain is F. The value of the variable y is B, where $B \in F$ and the value of the variable X_i is A_i, where $A_i \in P(D_i)$ and $P(D_i)$ denotes the set of subsets of D_i.

We can find some important differences between these kinds of fuzzy rules. While DNF-type and Mamdani-type fuzzy rules consider a linguistic variable in the consequent [15, 27, 28, 29], TSK fuzzy rules are based on representing the consequent as a polynomial function of the inputs [37]. The main difference between types 2 and 3 of rules is that type 3 allows subset of labels as values of a variable.

The Mamdani and DNF are linguistic models based on collections of $IF - THEN$ rules with fuzzy quantities associated with linguistic labels, and the fuzzy model is essentially a qualitative expression of the system. An FRBS in which the fuzzy sets giving meaning (semantic) to the linguistic labels are uniformly defined for all rules included in the FRB follows the *descriptive* approach since the linguistic labels take the same meaning for all the fuzzy rules contained in the FRB. In this case, the FRB is usually called Knowledge Base and it is composed of two components: the Rule Base (RB), constituted by the collection of fuzzy rules themselves, and the Data Base (DB), containing the membership functions defining their semantics.

2.2 Evolutionary and Genetic Algorithms

Evolutionary Computation (EC) uses computational models of evolutionary processes as key elements in the design and implementation of computer-based problem solving systems. There are a variety of evolutionary computational models that have been proposed and studied which are referred to as *Evolutionary Algorithms* (EAs). There have been three well-defined EAs which have served as the basis for much of the activity in the field, Genetic Algorithms, Evolution Strategies and Evolutionary Programming (EP) [1].

An EA maintains a population of trial solutions, imposes random changes to these solutions, and incorporates selection to determine which ones are going to be maintained in future generations and which will be removed from the pool of the trials. There are however important differences between them. GAs emphasize models of genetic operators as observed in nature, such as crossover (recombination) and point mutation, and apply these to abstracted chromosomes. ESs and EP emphasize mutational transformations that maintain the behavioral linkage between each parent and its offspring.

In the following, we briefly review the GAs, the most extended and most used EA.

A GA starts off with a population of randomly generated *chromosomes*, and advances toward better *chromosomes* by applying genetic operators modeled on the genetic processes occurring in nature. The population undergoes evolution in a form of natural selection. During successive iterations, called *generations*, chromosomes in the population are rated for their adaptation as solutions, and on the basis of these evaluations, a new population of chromosomes is formed using a selection mechanism and specific genetic operators such as crossover and mutation. An *evaluation* or *fitness function* (f) must be devised for each problem to be solved. Given a particular chromosome, a possible solution, the fitness function returns a single numerical fitness, which is supposed to be proportional to the utility or adaptation of the solution represented by that chromosome.

Although there are many possible variants of the basic GA, the fundamental underlying mechanism consists of three operations:

1. evaluation of individual fitness,
2. formation of a gene pool (intermediate population) through selection mechanism, and
3. recombination through crossover and mutation operators.

The next procedure shows the structure of a basic GA, where $P(t)$ denotes the population at generation t.

GAs may deal successfully with a wide range of problem areas. The main reasons for this success are: 1) GAs can solve hard problems quickly and reliably, 2) GAs are easy to interface to existing simulations and models, 3) GAs are extendible and 4) GAs are easy to hybridize. All these reasons may be summed up in only one: GAs are *robust*. GAs are more powerful in difficult environments

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

where the space is usually large, discontinuous, complex and poorly understood. They are not guaranteed to find the global optimum solution to a problem, but they are generally good at finding acceptably good solutions to problems acceptably quickly. These reasons have been behind the fact that, over the last few years, GA applications have grown enormously in many fields.

The basic principles of GAs were first laid down rigorously by Holland ([25]), and are well described in many books, such as [14, 30]. It is generally accepted that the application of a GA to solve a problem must take into account the following five components:

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

2.3 Genetic Fuzzy Rule Based Systems

EAs are applied to modify/learn the definition of the membership function shapes (DB) and/or the composition of the fuzzy rules (RB) in the way shown in Figure 1. Therefore, it is possible to distinguish three different groups of GFRBSs depending on the FRB components included in the learning process [7, 23]:

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 [7, 23] and for an extensive bibliography see [8], Section 3.13, and [9], Section 13. Different approaches may be found in [22, 33, 34].

Carse et al. [6] divide the third family into two different subgroups depending on the simultaneousness in the learning of both FRB components. Therefore, they differentiate between learning them in a single process or in different stages.

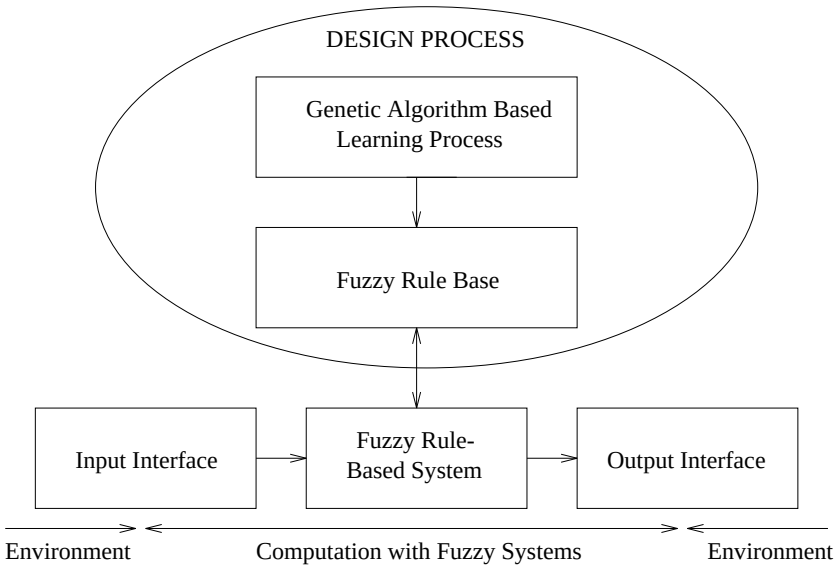

Fig. 1. Genetic Fuzzy Rule-Based Systems

3 Iterative Rule Learning Approach: Competition and Cooperation

In this Section we introduce the IRL approach and analyze the cooperation problem in the model under the cooperation versus competition versus problem.

3.1 IRL Approach

The main problem that has to be solved to design a GFRBS consists of finding a suitable representation capable of gathering the problem characteristics and representing 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* [5, 26] and *Pittsburgh* [35] approaches. In the Michigan approach, the chromosomes are individual fuzzy rules and the FRB is represented by the entire population. The collection of fuzzy rules is adapted over time using some genetic operators applied at the level of the individual rule. This evolution is guided by a credit assignment system that evaluates the adaption of each single fuzzy rule. On the other hand, in the Pittsburgh approach, each chromosome represents an entire FRB and the evolution is developed by means of genetic operators applied at the level of fuzzy rule sets. The fitness function evaluates the accuracy of the complete FRBS encoded in the chromosome.

In the last few years, the *IRL* approach has been used by some authors to obtain several GFRBSs following a new learning model [15, 20]. In the latter, 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 substantially reduces the search space, because in each sequence of iterations only one rule is searched for.

In order to obtain a set of rules, which will be a true solution to the problem, the GA has to be placed within an iterative scheme similar to the following:

1. Use a GA to obtain a rule for the system.
2. Incorporate the rule into the final set of rules.
3. Penalize this rule.
4. If the set of rules obtained is adequate to represent the examples in the training set, the system ends up returning the set of rules as the solution. Otherwise return to step 1.

A very easy way to penalize the rules already obtained, and thus be able to learn new rules, consists of eliminating from the training set all those examples that are covered by the set of rules obtained previously.

The main difference with respect to the Michigan approach is that the fitness of each chromosome is computed individually, without taking into account cooperation with other ones.

In the literature we can find some genetic learning processes that use this model such as *SLAVE* [15, 17, 20], *SIA* [38] and *MOGUL* [10, 11].

¿From the description shown above, we may see that in order to implement a learning algorithm based on GAs using the IRL approach, we need, at least, the following:

1. a criterion for selecting the best rule in each iteration, and
2. a penalization criterion, and
3. a criterion for determining when there are enough rules available to represent the examples in the training set.

The first criterion is normally associated with one or several characteristics that are desirable so as to determine good rules. Usually criteria about the rule strength have been proposed (number of examples covered), criteria of consistency and completeness of the rule or criteria of simplicity (for some examples, see [10, 17, 20, 24]).

The second criterion is often associated with the elimination of the examples covered by the previous rules.

Finally, the third criterion is associated with the completeness of the set of rules [10, 17, 24] and must be taken into account when we can say that all the examples of a concept in the training set are sufficiently covered and no more rules are needed to represent them. This criterion is often associated, although

it is not necessary, with the elimination of the examples in a concept covered by the previous rules.

This scheme is usually employed in GFRBSs based on inductive learning, in which the penalization of the fuzzy rules already generated is done by removing from the training data set all those examples that are still covered by the FRB obtained until that time. As has been previously said, a key characteristic of the IRL is that it substantially reduces 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.

3.2 Competition and cooperation in the IRL approach

Associated to the previous criteria needed in the development of an IRL algorithm, we include a natural way for competition and cooperation relations between the rules. So, the first criterion establishes the *competition between members of the population representing possible solutions to the problem*. In this case, this characteristic is due to the mechanisms of natural selection on which the EA is based. On the other hand, the second and third criteria include *cooperation relations between the rules that describe the same concept*, since, as was previously mentioned, normally the goal of the third criterion consists of trying to remove all the examples that are being learnt from the training set, when these are covered by some rules to a sufficient degree.

One of the most interesting features of an FRBS in qualitative modeling problems 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 all the fuzzy rules composing the FRB*. As 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 match the input for the system to some degree. So, a very interesting way to solve the problem of designing an FRBS consists of adding both features to the learning algorithm: competition to achieve the best rules and cooperation between rules from the same or different value of the consequent variable. This is referred to as the cooperation versus competition problem (*CCP*) [4].

However, the cooperation between rules from different concepts is not included within the IRL approach. The difficulty of solving the problem of taking into account this kind of cooperation depends directly on the genetic learning approach followed by the GFRBS. GFRBSs based on the IRL approach try to solve the CCP at the same time as reducing the search space by encoding a single fuzzy rule in each chromosome. To put this into effect, these processes can use different ways:

- adding new criteria to the evaluation of the rule for including this kind of cooperation within the IRL approach,
- dividing the genetic learning process into, at least, two stages. Therefore, the CCP is solved in two steps acting on two different levels, with the competition between fuzzy rules in the first one, the genetic generation stage, and with

the cooperation between these generated fuzzy rules in the second one, a postprocessing stage.

In the following section we present these two alternatives.

4 Alternatives for Including Cooperation in the IRL approach

In this Section, we present two alternatives used for introducing cooperation in two different GFRBSs based on the IRL approach. The first one, used in SLAVE [15, 17, 20], includes cooperation relations between rules from different values of the consequent value within the IRL approach, and the second alternative, used in MOGUL [10, 11], based on dividing the genetic learning process into, at least, two stages, managing cooperation between the generated fuzzy rules in the second one.

4.1 Cooperation within the IRL approach

In the proposal for the IRL approach the cooperation is defined between the rules from the same value in the consequent variable in a natural way, as previously has been mentioned. In many cases, this cooperation level is sufficient when the concepts are exclusive and there is no noise or inconsistency in the example set. However, with databases affected by noise and inconsistency and when the concepts are not exclusive, a higher degree of cooperation must be established that permits good collaboration between rules from different values of the consequent variable.

Normally, the degree of collaboration between rules from different concepts is measured using the inference model associated to the learning algorithm. It is not easy to establish this cooperation between rules, since the IRL approach learns the rules one by one, and the learning process does not has the whole rule set for applying the inference model.

A way for defining this cooperation level consists of applying the inference model partially on a subset of the examples and including this information in the evaluation of the rules. This subset contains the examples of the concepts that have been learnt by the learning algorithm. So, the cooperation between rules is measured by the error that produces the new rule in the outputs of the inference model, when this rule is included in the rule set.

The IRL approach, without this kind of collaboration, tries to extract the knowledge that the examples represent for each concept from the training set. The rule set obtained in this way, provides a comprehensible description of the system that we want to learn. From our point of view, the inclusion of cooperation must keep this comprehensible description of the system and furthermore it must improve the interpolative reasoning between the rules. This is important since an inappropriate use of the cooperation measure, may provoke a reduction in the quality of the rule (with respect to the interpretability of the rule set) because

the goal of the learning process is to reduce the error in the output as far as possible.

SLAVE is a learning algorithm based on the IRL approach that takes into account cooperation between the rules from different concepts in the sense previously described. SLAVE was initially proposed in [15] and later developed in [17, 18]. The algorithm begins with a simple description of the problem: the consequent variable (concept variable) and the set of all the available antecedent variables for generating the rules that describe the consequent variable. The learning algorithm, using this description and a set of examples, will decide for each value of the consequent variable and each rule which variables are needed to describe the concept (feature selection), and the rest will be eliminated.

The basic element of the SLAVE learning algorithm is its model of rules that was described in Section 2.1 (type 3). The key to this rule model is that each variable may take as a value an element or a subset of elements in its domain, i.e., we let the value of a variable be interpreted more as a disjunction of elements than just one element in its domain.

Consistency and Completeness in SLAVE In the SLAVE learning process (Figure 2), finding the best rule consists of determining the best combination of values from the antecedent variables, given a fixed value of the consequent variable and a set of examples. The best rule concept uses a simple quantitative criterion; the best rule will be the rule covering the maximum number of examples. However, there are problems with this criterion, if we do not restrict the set of possible rules.

Classical learning theory proposes conditions that must be verified for the set of rules that are obtained by a learning algorithm. These conditions, which provide the logical foundation of the algorithms for concept learning from examples, are called *consistency condition* and *completeness condition* [31].

These conditions are associated on the whole set of rules. SLAVE obtains the set of rules that describes the system, extracting one rule in each iteration of the learning process. Due to this reason, we need to define these concepts on each rule. Moreover, we are not interested in proposing hard definitions on fuzzy problems, thus we propose a degree of completeness and a degree of consistency.

Definition1. The degree of completeness of a rule $R_B(A)$ is defined as

$$\Lambda(R_B(A)) = \frac{n^+(R_B(A))}{n_B}$$

where $n_B = \sum_{i=1}^{m} U(e_i, B)$ is the number of examples of the value B of the consequent variable in the training set, m is the number of examples in the training set and $n^+(R_B(A))$ is the number of positive examples covered by the rule $R_B(A)$.

With respect to the soft consistency degree [17] it is based on the possibility of admitting some noise into the rules. Thus, in order to define the soft consistency

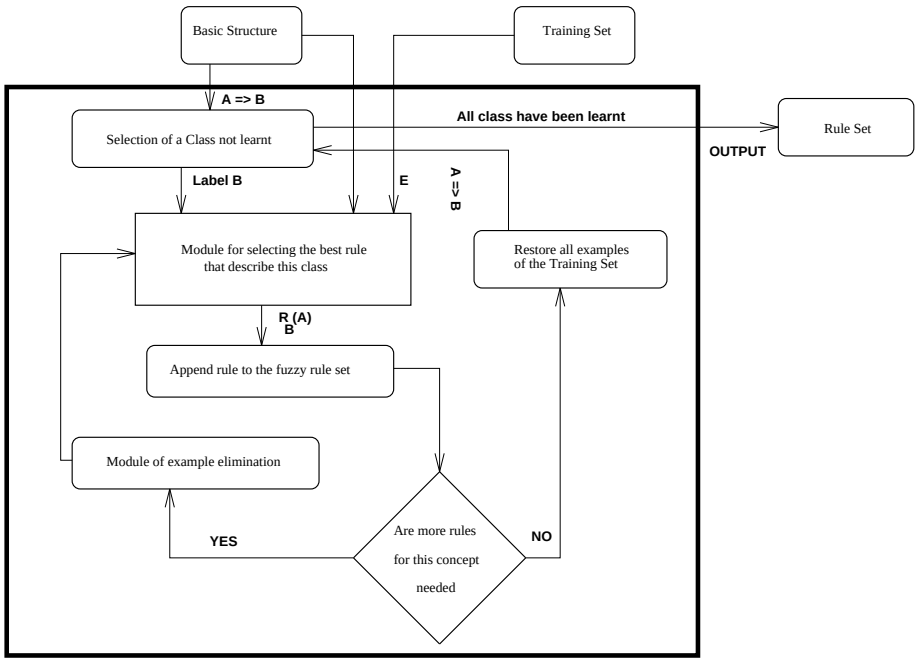

Fig. 2. SLAVE learning process

degree we use the following set:

$$\Delta^k = \{R_B(A)/n^-(R_B(A)) < k\ n^+(R_B(A))\}$$

which represents the set of rules having a number of negative examples strictly less than a percentage (depending on k) of the positive examples.

Definition 2. The degree to which a rule satisfies the soft consistency condition is

$$\Gamma_{k_1 k_2}(R) = \begin{cases} 1 & \text{if R} \in \Delta^{k_1} \\ \frac{k_2 n_E^+(R) - n_E^-(R)}{n_E^+(R)(k_2 - k_1)} & \text{if R} \notin \Delta^{k_1} \text{ y R} \in \Delta^{k_2} \\ 0 & \text{otherwise} \end{cases}$$

where $k_1, k_2 \in [0,1]$ and $k_1 < k_2$, and $n_E^-(R)$, $n_E^+(R)$ are the number of positive and negative examples to the rule, R.

This definition uses two parameters, k_1 is a lower bound of the noisy threshold and k_2 is an upper bound of the noisy threshold.

Thus, SLAVE selects the rule that simultaneously verifies the completeness and soft consistency conditions to a high degree. Therefore, rule selection in SLAVE can be solved by the following optimization problem:

$$\max_{A \in D} \{ \Lambda(R_B(A)) \ \Gamma_{k_1 k_2}(R_B(A)) \}$$

where $D = P(D_1) \times P(D_2) \times \ldots \times P(D_n)$.

Adding Cooperation between Rules The main component of SLAVE is a genetic algorithm [18]. The goal of the genetic algorithm is to find the most consistent and complete rule at each step given the available set of examples. However, we want a rule set that includes cooperation between rules from different values of the consequent variable and with understandable rules.

For this reason, it is necessary to establish measures on the simplicity of the rule and on the degree of cooperation between rules and these measures must be taken into account for evaluating the rule. So, the evaluation function for determining the best rule is defined by a multiobjective function.

The measures for evaluating the simplicity of the rule were proposed in [19]. Now, we describe briefly the way used by SLAVE for including cooperation among rules.

SLAVE distinguishes two kinds of learning problems, when the consequent variable takes values in a discrete range (Classification Problems) and when the consequent variable takes values in a continuous range (Qualitative Modeling Problems). This difference between both problems is important since the inference method associated for each one of them is different. In the first case, the inference method can be seen as a competition between the rules for determining which of them will be the most appropriate for classifying a certain example, that is, the inference method selects only one rule each time. However, in the second case, the output of the rule set is obtained by the combination of the output of each rule that can be applied for classifying the example.

In the first case, we establish competition relations between rules from different values of the consequent variable for improving the overall behaviour of the rule set. The description of this process is to be found in [19]. In the second case, where the output is obtained by interpolative reasoning, SLAVE includes cooperation relations between rules in the following form:

a) defining in a special way the concepts of the number of positive and negative examples covered by a rule, and
b) including an error measure that determines the cooperation degree of the new rule with the rest of the rules that are members of the rule set.

The definition of the number of positive and negative examples covered by a rule is related to the results that this rule will obtain in the inference process, that is, with successes and failures. Using this idea, we built the following definitions:

Definition 3. A rule $R_B(A)$ classifies correctly an example e if

$$U(e, A) > 0 \text{ and } U(e, B) > 0$$

where $U(e, A)$ is the adaptation between the example and the antecedent of the rule and $U(e, B)$ is the adaptation between the example and the consequent of the rule. The definition of this adaptation function is to be found in [17].

Definition 4. A rule $R_B(A)$ does not correctly classify an example e if

$$U(e, A) > 0 \text{ and } U(e, B) = 0$$

From the previous definitions, we can establish the definitions of the number of positive and negative examples in the following way:

$$n^+(R_B(A)) = |\{e \in E \mid U(e, A) > 0 \text{ and } U(e, B) > 0\}|$$

$$n^-(R_B(A)) = |\{e \in E \mid U(e, A) > 0 \text{ and } U(e, B) = 0\}|$$

The previous definitions allow us to determine the goodness of the rule, but a difference of the sense proposed in the crisp problems [19], in fuzzy problems it is not necessary to grade the trust of the successes, since the rules work in cooperation for obtaining the output.

However, these definitions are not sufficient for maximizing the relations of cooperation between the rules. For this reason, SLAVE includes in the evaluation of the rule, a measure of the collaboration degree. This measure is based on the error that the new rule produces in the output when it is included in the rule set.

The error measure is obtained using the inference method on the subset of the examples that are members of some of the classes from the rules learnt. So, the rule evaluation function is composed by three criteria:

Criterion 1: The degree to which the rule represents the examples from the concepts that we want to learn, i.e., the verification of the completeness and soft consistency conditions to a high degree.
Criterion 2: Influence on the output from the Knowledge Base of the inclusion of the new rule, i.e., error measure.
Criterion 3: Simplicity and understanding of the rule, i.e., simplicity measures.

Therefore, the evaluation function is a multicriteria function that contains the previous criteria. For the evaluation, a lexicographical order is applied, where the main criterion is to maximize the degree for representing the examples of the concepts that we want to learn (criterion 1). In a tie situation among them the rule is selected that produces less error in the output (criterion 2). If a new tie situation is produced, then it selects the simpler rule (criterion 3) among them.

4.2 Cooperation in Stages

As the generation process does not envisage any relationship between the fuzzy rules generated, it is necessary to employ any other postprocessing to simplify and/or adjust the FRB obtained, thereby forming a multi-stage GFRBS. Therefore, the CCP is solved in two steps acting on two different levels:

- *the genetic generation stage forces competition between fuzzy rules*, as the genetic learning processes based on the Michigan approach, *to obtain an FRB composed of the best possible fuzzy rules*. The cooperation between them is only smoothly addressed by means of the rule penalization criterion.
  This generation stage uses a covering method which 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 creates over it, and removes the examples from it with a covering value greater than ϵ, provided by the system designer. It ends up when the training set is left empty.
  Each time the generating method is run, it produces a set of candidate fuzzy rules by generating the fuzzy rule best covering every example from the training set. The accuracy of the candidates is measured by using a multicriteria fitness function, composed of three different criteria measuring the covering that each rule creates over the training set. Their expressions are to be found in [10]. Finally, the best fuzzy rule is selected from the set of candidates and given as the method output.
  The designer is allowed to build the generation stage by using different kinds of algorithms and not only a GA as in the previous existing processes following the IRL approach. It is possible to employ a non-evolutionary inductive algorithm or an Evolution Strategy [1] instead of the usual GA [10]. The way of working is still the same but the difference is the speed of the generation process, which is higher in the latter cases.
- *the postprocessing stage forces cooperation between 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 *in order to obtain the best possible FRB*.

The postprocessing stage will present two important characteristics. On the one hand, it will be designed by means of a GA based on the Pittsburgh learning approach, but significatively reducing the solution space by working only over the FRB generated in the first stage, i. e., not modifying the membership function definitions. In this way, it will simplify the FRB obtained until now by removing the redundant or unnecessary fuzzy rules not cooperating adequately with the others. This operation mode will allow us to obtain the best possible FRB composed of the best combination of the fuzzy rules generated in the first stage.

On the other hand, a genotypic sharing function [13] will be considered to obtain not only a single FRB as output from the process but different ones presenting the best possible cooperation between the fuzzy rules composing them,

and thereby the best possible behavior. Due to this, we will refer to this second stage as the *multisimplification process*.

The *Sequential Niche Technique* [3] is used to induce niches in this GFRBS stage, with the genetic simplification process proposed in [24] being the basic optimization technique iterated in each run of the multisimplification process. The following subsections introduce the basic simplification algorithm and the particular aspects of the multisimplification one, respectively.

The Basic Genetic Simplification Process As mentioned earlier, the basic genetic simplification process was first proposed in [24]. It is based on a binary coded GA, in which the selection of the individuals is performed 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 $C = (c_1, ..., c_m)$ represents a subset of candidate rules to form the FRB finally obtained as this stage output, B^s, such that,

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

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

As regards the fitness function, $F(C_j)$, it is based on two different criteria:

- On the one hand, we have a global error measure that determines the accuracy of the FRBS encoded in the chromosome. We usually work with the mean square error (SE), although other measures may be used. SE over a training data set, E_{TDS}, is represented by the following expression:

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

  where $S(ex^l)$ is the output value obtained from the FRBS using the FRB coded in C_j, $R(C_j)$, when the input variable values are $ex^l = (ex^l_1, \ldots, ex^l_n)$, and ey^l is the known desired value.
- On the other hand, since there is a need to keep the *τ-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 τ,

$$C_{R(C_j)}(e_l) = \bigcup_{j=1..T} R_j(e_l) \geq \tau, \quad \forall e_l \in E_p \text{ and } R_j \in R(C_j)$$

where τ is the minimal training set completeness degree accepted in the simplification process. Usually, τ is less than or equal to ω, 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_p as

$$TSCD(R(C_j), E_p) = \bigcap_{e_l \in E_p} C_{R(C_j)}(e_l)$$

The final expression of the fitness function is:

$$F(C_j) = \begin{cases} E(C_j), \text{ if } TSCD(R(C_j), E_p) \geq \tau, \\ \infty, \quad \text{otherwise} \end{cases}$$

The Genetic Multisimplification Process In order to induce niching in the sequential niche algorithm, there is a need to define some kind of *distance metric* which, given two individuals, returns a value of how close they are [3]. We use a *genotypic sharing* [13] due to the fact that the metric considered is the Hamming distance measured on the binary coding space. With $A = (a_1, ..., a_m)$ and $B = (b_1, ..., b_m)$ being two individuals, it is defined as follows:

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

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 previously obtained solutions. 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, $S = \{s_1, \ldots, s_k\}$ is the set containing the k solutions already 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:

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

where d is the minimum value of the Hamming distance between C_j and the solutions s_i included in S, i. e., $d = Min_i\{H(C_j, s_i)\}$, and the penalization is considered over the closest solution, r is the *niche radius*, and β 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 already found. There

is no penalization when the C_j is far away from S with a value greater than or equal to the niche radius r.

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 all the simplified FRBs desired have not been obtained, return to step 2.*

Hence, the number of runs for the sequential algorithm performed is the number of solutions desired to be obtained. We allow the FRBS designer to decide this number as well as the values of parameters r and β.

5 Example: Rice Taste Analysis

Subjective qualification of food taste is a very important but difficult problem. In the case of the rice taste qualification, it is usually put into effect by means of a subjective evaluation called the *sensory test*. In this test, a group of experts, usually composed of 24 persons, evaluate the rice according to a set of characteristics associated to it. These factors are: *flavor*, *appearance*, *taste*, *stickiness*, and *toughness* [32].

Because of the large quantity of relevant variables, the problem of rice taste analysis becomes very complex, thus leading to solve it by means of modeling techniques capable of obtaining a model representing the non-linear relationships existing in it. Moreover, the problem-solving goal is not only to obtain an accurate model, but to obtain a user-interpretable model as well, capable of putting some light on the reasoning process made by the expert for evaluating a kind of rice in a specific way. Due to all these reasons, in this Section we deal with obtaining a qualitative model to solve the said problem.

In order to do so, we are going to use the data set presented in [32]. This set is composed of 105 data arrays collecting subjective evaluations of the six variables in question (the five mentioned and the overall evaluation of the kind of rice), made up by experts on this number of kinds of rice grown in Japan (for example, Sasanishiki, Akita-Komachi, etc.). The six variables are normalized, thus taking values in the real interval $[0, 1]$.

With the aim of not biasing the learning, we have randomly obtained ten different partitions of the mentioned set, composed by 75 pieces of data in the training set and 30 in the test one, for generating ten qualitative models in each experiment. To solve the problem, we use the two GFRBSs based on the IRL approach introduced in this paper, and two qualitative modeling processes with different characteristics as well:

D1. The inductive learning process proposed by Nozaki et al. in [32].

D2. The inductive learning process proposed by Wang and Mendel (WM) in [39].
D3. The SLAVE GFRBS introduced in Section 4.1.
D4. The GFRBS obtained from MOGUL introduced in Section 4.2.

As was done in [32], we have worked with fuzzy partitions composed by a different number of linguistic labels for the six variables considered. These fuzzy partitions have been obtained from a normalization process in which the universe of discourse of each variable has been equally divided into 2 and 3 parts, and a triangular fuzzy set has been associated to each one of them. Figure 3 shows an example of a fuzzy partition with five linguistic labels. The reason why we have not considered fuzzy partitions with a higher number of labels is that there is a need to obtain simple qualitative models with FRBs composed by a small number of rules in order to make them interpretable.

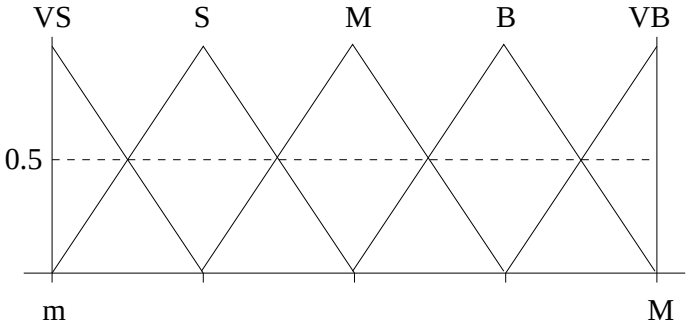

Fig. 3. Example of fuzzy partition with five linguistic labels

The results obtained in the experiments developed are collected in Table 1. The values shown in columns SE_{tra} and SE_{tst} have been computed as an average of the mean square error values obtained in the approximation of the training and test data sets, respectively, by the ten qualitative models generated in each case. The column $\#R$ stands for the average number of fuzzy rules in the FRBs of the models generated from each process. We should remember that these fuzzy rules are simple in GFRBSs **D1**, **D2** and **D4**, and present the disjunctive normal form in the case of GFRBS **D3**. Finally, as may be observed, GFRBS **D4**, the one based on MOGUL, has two rows associated, one for each one of the learning stages: generation and postprocessing.

In view of these results, many interesting conclusions may be drawn. On the one hand, as regards the accuracy in the problem solving, the qualitative model generated by the GFRBS obtained from SLAVE, **D3**, has obtained the best training result and MOGUL, **D4**, has obtained the best test result when working with 2 labels, with the second best result being obtained by SLAVE. With respect to MOGUL, the good behaviour of the cooperation encouraged by

	2 labels			*3 labels*		
Process	$\#R$	SE_{tra}	SE_{tst}	$\#R$	SE_{tst}	SE_{tst}
D1	64	0.00862	0.00985	364.8	0.00251	0.00322
D2	15	0.013284	0.013118	23	0.003339	0.003758
D3	2.0	0.004111	0.007288	3.0	0.003866	0.006533
D4 (gen.)	10.1	0.013601	0.012508	21.2	0.005157	0.006579
D4 (post.)	6	0.004861	0.0037	10.6	0.00281	0.004169

Table 1. Results obtained in the rice taste analysis problem by the different learning processes

the postprocessing stage is demonstrated in the light of the results obtained: it not only significatively reduces the number of rules in the models generated but improves their accuracy to a high degree (see rows **D4** (gen.) and **D4** (post.)). With respect to SLAVE the results are very close to the previous system and the main difference corresponds to the simplicity of the final model.

In the case of 3 labels, the best results are obtained by GFRBS **D1**, which obtains the best overall global results in this case as well. The problem is that the qualitative models generated from this GFRBS are not useful in practice due to the number of rules in their FRBs (364.8 in average) is excessively high in order to interpret them. We should remember that the goal is not only to obtain accurate fuzzy models solving the problem but to make sure these models can be interpreted by human beings.

Focusing on this second aspect, it may be observed that the simplest models are obtained by the SLAVE GFRBS, **D3**, presenting an average of 2.0 and 3.0 rules when considering 2 and 3 labels, respectively. This is a key aspect due to the fact that the interpretability of the model will depend directly on the number of fuzzy rules in the FRB. It will be very easy for an expert to interpret a qualitative model composed by only two rules. The cooperation induced in the rule generation in this GFRBS (see Section 4.1) allows it to generate very simple qualitative models with an adequate accuracy according to its simplicity. On the other hand, the behaviour of the GFRBS obtained from MOGUL, **D4**, is also good in this aspect. It allows us to design qualitative models with a very good balance between interpretability (generates simple models with 6 rules in the case of 2 labels, and 10.6 in the case of 3 labels) and accuracy.

In Tables 2 and 3, as an example, the composition of the FRB is shown for one of the models generated from each one of the GFRBSs based in the IRL introduced in this contribution. The specific values obtained by them in both measures are $SE_{tra} = 0.0041$ and $SE_{tst} = 0.0061$ for the GFRBS **D3** (SLAVE), and $SE_{tra} = 0.005681$ and $SE_{tst} = 0.001782$ for the GFRBS **D4** (MOGUL).

[ht]

R_1:	**IF** *Taste* is Bad **THEN** *Overall Evaluation* is Low
R_2:	**IF** *Appearance* is Good **AND** *Taste* is Good **THEN** *Overall Evaluation* is High

Table 2. FRB of one of the qualitative models generated from the GFRBS **D3** using 2 labels

[ht]

	Flavor	*Appearance*	*Taste*	*Stickness*	*Toughness*	**Overall Evaluation**
R_1:	Good	Good	Good	Sticky	Tender	High
R_2:	Good	Good	Good	Sticky	Tender	High
R_3:	Good	Good	Good	Sticky	Tender	High
R_4:	Good	Bad	Bad	Not sticky	Tough	Low
R_5:	Bad	Bad	Bad	Not sticky	Tough	Low
R_6:	Bad	Good	Good	Sticky	Tender	Low

Table 3. FRB of one of the qualitative models generated from the GFRBS **D4** using 2 labels

6 Concluding Remarks

In this contribution, the IRL approach, one of the existing genetic learning approaches to design GFRBSs, has been briefly reviewed, and the problem of obtaining the best possible cooperation between the generated fuzzy rules in the genetic learning processes based on it has been analyzed. Two different ways for encouraging cooperation between rules in them have been presented: the inclusion of the cooperation between rules inside of the iterative rule learning, used in SLAVE, and the creation of a postprocessing stage obtaining different simplified FRB definitions with good cooperation from the rules generated in the genetic learning stage, used in MOGUL.

Both GFRBSs have been applied to solve a qualitative modeling problem, rice taste analysis, and the results obtained by them have been compared with two other design processes with different characteristics. They have performed well, showing the good behaviour of both alternatives presented to encourage the cooperation between rules in GFRBSs based on the IRL approach.

On the other hand, we should note that these GFRBSs have been applied to solve other real-world problems as well. In [16], SLAVE was used in a medical

application, the diagnosis of myocardial infarction, while a real-world Spanish electrical engineering problem was solved by the GFRBS obtained from MOGUL in [12].

References

[1] T. Bäck, Evolutionary Algorithms in Theory and Practice (Oxford University Press, 1996).

[2] A. Bardossy, L. Duckstein, Fuzzy Rule-Based Modeling With Application to Geophysical, Biological and Engineering Systems (CRC Press, 1995).

[3] D. Beasly, D.R. Bull, R.R. Martin, A sequential niche technique for multimodal function optimization, Evolutionary Computation 1:2 (1993) 101-125.

[4] A. Bonarini, Evolutionary learning of fuzzy rules: competition and cooperation, in: W. Pedrycz, Ed., Fuzzy Modelling: Paradigms and Practice (Kluwer Academic Press, 1996) 265-283.

[5] L. Booker, Intelligent Behaviour as an Adaption to the Task Environment, PhD thesis, University of Michigan (1982).

[6] B. Carse, T.C. Fogarty, A. Munro, Evolving fuzzy rule based controllers using genetic algorithms, Fuzzy Sets and Systems 80 (1996) 273-294.

[7] Cordón, O., Herrera, F., A General Study on Genetic Fuzzy Systems. In: G. Winter, J. Periaux, M. Galan, P. Cuesta (Eds.), Genetic Algorithms in Engineering and Computer Science, Wiley and Sons, (1995), 33-57.

[8] 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, 1997) 209-241.

[9] 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, 1997) 57-77.

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

[11] O. Cordón, M.J. del Jesus, F. Herrera, M. Lozano, MOGUL: A Methodology to Obtain Genetic fuzzy rule-based systems Under the iterative rule Learning approach. Technical Report DECSAI-98101, Dept. Computer Science and Artificial Intelligence, University of Granada, Granada (Spain, January 1998).

[12] O. Cordón, F. Herrera, L. Sánchez, Computing the Spanish Medium Electrical Line Maintenance Costs by means of Evolution-Based Learning Processes, Eleventh International Conference on Industrial & Engineering Applications of Artificial Intelligence & Expert Systems (IEA-98-AIE) Castellón (Spain, 1998).

[13] K. Deb, D.E. Goldberg, An investigation of niche and species formation in genetic function optimization, Proc. of the Second International Conference on Genetic Algorithms, Lawrence Erlbaum (Hillsdale, NJ, 1989) 42-50.

[14] D.E. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley (1989).

[15] A. González, R. Pérez, J.L. Verdegay, Learning the structure of a fuzzy rule: a genetic approach. Proc. EUFIT'93 vol. 2 (1993) 814-819. Also in Fuzzy System and Artificial Intelligence 3(1) (1994) 57-70.

[16] A. González, R. Pérez, A. Valenzuela, Diagnosis of myocardial infarction through fuzzy learning techniques, Proceedings of IFSA'95 vol.I, Sao Paulo (1995) 273-276.

[17] A. González, R. Pérez, Completeness and Consistency Conditions for Learning Fuzzy Rules. Fuzzy Sets and Systems (1998, to appear).

[18] A. González, R. Pérez, A Learning System of Fuzzy Control Rules. In: F. Herrera, J.L. Verdegay (Eds.), Genetic Algorithms and Soft Computing, Physica-Verlag (1996) 202-225.

[19] A. González, R. Pérez, SLAVE: a genetic learning system based on an iterative approach, Technical Report #DECSAI-97111 (1997).

[20] A. González, F. Herrera, Multi-stage genetic fuzzy systems based on the iterative rule learning approach, Mathware & Soft Computing 4 (1997) 233-249.

[21] J.J. Grefenstette, (Ed.), Genetic Algorithms for Machine Learning. Kluwer Academic, (1994).

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

[23] F. Herrera, L. Magdalena, Genetic fuzzy systems, in: R. Mesiar, B. Riecan, Eds., Tatra Mountains Mathematical Publications 13 (1997) 93-121. Vol. "Fuzzy Structures. Current Trends". Lecture Notes of the Tutorial: Genetic Fuzzy Systems. Seventh IFSA World Congress (IFSA'97).

[24] F. Herrera, M. Lozano, J.L. Verdegay, A learning process for fuzzy control rules using genetic algorithms, Fuzzy Sets and Systems (1998, to appear).

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

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

[27] C.C. Lee, Fuzzy logic in control systems: fuzzy logic controller - parts I and II, IEEE Transactions on Systems, Man, and Cybernetics 20 (1990) 404-435.

[28] L. Magdalena, F. Monasterio, A Fuzzy Logic Controller with Learning Through the Evolution of its Knowledge Base, International Journal of Approximate Reasoning, 16 (1997) 335-358.

[29] L. Magdalena, Adapting the Gain of an FLC with Genetic Algorithms, International Journal of Approximate Reasoning, 17 (1997) 327-349.

[30] Z. Michalewicz. Genetic Algorithms + Data Structures = Evolution Programs. Springer-Verlag, (1992).

[31] R.S. Michalski, Understanding the nature of learning, Machine Learning: An artificial intelligence approach (Vol II). San Mateo, CA: Morgan Kaufmann (1986).

[32] K. Nozaki, H. Ishibuchi, H. Tanaka, *A Simple but Powerful Heuristic Method for Generating Fuzzy Rules from Numerical Data*, Fuzzy Sets and Systems 86 (1997) 251-270.

[33] W. Pedrycz (Ed.), Fuzzy Evolutionary Computation (Kluwer Academic Press, 1997).

[34] E. Sanchez, T. Shibata, L. Zadeh (Eds.), Genetic Algorithms and Fuzzy Logic Systems. Soft Computing Perspectives (World Scientific, 1997)

[35] S.F. Smith, A Learning System Based on Genetic Adaptive Algorithms, PhD thesis, University of Pittsburgh (1980).

[36] M. Sugeno, T. Yasukawa, *A Fuzzy-logic-based Approach to Qualitative Modeling*, IEEE Transactions on Fuzzy Systems 1(1) (1993) 7-31.

[37] T. Takagi, M. Sugeno, Fuzzy identification of systems and its application to modeling and control, IEEE Transactions on Systems, Man, and Cybernetics 15(1) (1985) 116-132.

[38] G. Venturini, SIA: a Supervised Inductive Algorithm with Genetic Search for Learning Attribute Based Concepts. Proc. European Conference on Machine Learning, Vienna, (1993), 280-296.

[39] L.X. Wang, J.M. Mendel, *Generating Fuzzy Rules by Learning from Examples*, IEEE Transactions on Systems, Man, and Cybernetics 22(6) (1992) 1414-1427.

[40] L. Zadeh, *The Concept of a Linguistic Variable and its Applications to Approximate Reasoning.* (1975) Part I, Information Sciences 8, 199-249, Part II, Information Sciences 8, 301-357, Part III, Information Sciences 9 43-80.