Techniques for Learning and Tuning Fuzzy Rule-Based Systems for Linguistic Modeling and their Application *

R. Alcalá*, J. Casillas*, O. Cordón*, F. Herrera*, S. J. I. Zwir†

* Department of Computer Science and Artificial Intelligence.
E.T.S. de Ingeniería Informática, University of Granada. 18071 – Granada, Spain
e-mails: {alcala,casillas,ocordon,herrera}@decsai.ugr.es

† Department of Computer Science.
University of Buenos Aires, Buenos Aires, Argentina
e-mail: zwir@dc.uba.ar

Abstract

Nowadays, Linguistic Modeling is considered to be one of the most important areas of application for Fuzzy Logic. Linguistic Mamdani-type Fuzzy Rule-Based Systems (FRBSs), the ones used to perform this task, provide a human-readable description of the model in the form of linguistic rules, which is a desirable characteristic in many problems.

In this Chapter we are going to accomplish a short revision of the FRBSs where we shall see the different types that currently exist, along with their structures and characteristics, centering our attention on linguistic Mamdani-type FRBS. The performance of a linguistic FRBS depends on its Rule Base and the membership functions associated to the fuzzy partitions. Due to the complexity in the design of these components, a large quantity of automatic techniques has been proposed to put it into effect.

Thereafter, we are going to review several learning (when it sets the Rule Base and sometimes the Data Base as well) and tuning (when it only sets the Data Base) methods. These methods are inspired in the three most well known approaches: *ad hoc* data covering, neural networks, and genetic algorithms. We shall introduce a brief description of these techniques and their synergy with FRBSs.

The accuracy of the reviewed methods will be compared when solving two real-world applications. Some interesting conclusions will be obtained about the behavior of the methods, approaches, and techniques.

I Introduction

At present, the most important application of Fuzzy Set Theory as developed by Zadeh in 1965 [1] are Fuzzy Rule-Based Systems (FRBSs). These kinds of systems constitute an extension of classical Rule-Based Systems, because they deal with fuzzy rules instead of classical logic rules. Thanks to this, they have been successfully applied to a wide range of problems from different areas presenting uncertainty and vagueness in different ways [2, 3, 4, 5].

An FRBS presents two main components: 1) the Inference System, which puts into effect the fuzzy inference process needed to obtain an output from the FRBS when an input is specified, and 2) the Knowledge Base (KB) representing the knowledge known about the problem being solved, constituted by a collection of fuzzy rules.

*This research has been supported by CICYT TIC96-0778

There are two different kinds of FRBSs in the literature, the Mamdani and TSK ones, depending on the expression of the consequent of the fuzzy rules composing the KB. While Mamdani-type fuzzy rules consider a linguistic variable in the consequent, TSK fuzzy rules are based on representing the consequent as a polynomial function of the inputs.

Linguistic Modeling based on Fuzzy Logic is considered as a system model constituting a linguistic description, being put into effect by means of a linguistic Mamdani-type FRBS. Thereby the concept of linguistic variable plays a central role. A crucial reason why the linguistic fuzzy rule-based approach is worth considering is that it may remain verbally interpretable. These FRBSs have been widely used and have obtained very good results in many different applications.

The main of this Chapter is to present a short description of linguistic FRBSs and a review of different design techniques for them. Many different approaches have been presented taking different learning algorithms as a base, such as, *ad hoc* methods, neural networks (NNs), genetic algorithms (GAs), clustering, etc. We shall present the most well known approaches to learn and tune linguistic FRBSs, apply them to some examples, analyze their results, and give finally some guidelines for their application.

In this Chapter, we use the terms learning and tuning of FRBSs refering to the design of an FRBS by means of a learning process, deriving the Rule Base and sometimes defining the Data Base as well, or by a tuning process, which only defines the Data Base. Therefore, we shall use the first expressions to make easy the reading of this Chapter.

We arrange this Chapter as follows. Section II presents some preliminaries by briefly introducing the different types and structures of FRBSs. In Sections III and IV, several linguistic FRBSs learning and tuning methods will be presented, covering the different types of techniques. Section V shows two experiments developed to evaluate the behavior of the methods. Finally, in Section VI, some concluding remarks are pointed out.

II Fuzzy Rule-Based Systems

In a very wide sense, an FRBS is a Rule-Based System where Fuzzy Logic may be used as a tool for representing different forms of knowledge about the problem being solved, as well as for modeling the interactions and relationships existing between its variables. Thanks to this, they have been successfully applied to a wide range of problems from different areas presenting uncertainty and vagueness in different ways [2, 3, 4, 5].

In this section, the basic aspects of FRBSs will be introduced: the different existing types, their composition and functioning will be described. Nevertheless, we shall not focus on the basic principles of Fuzzy Logic, that are to be found in texts like [6, 7].

II.A Framework: Fuzzy Logic and Fuzzy Systems

As it is known, Rule-Based Systems (production rule systems) have been successfully used to model human problem-solving activity and adaptive behavior, where a classical way to represent the human knowledge is the use of *"IF-THEN"* rules. The satisfaction of the rule antecedent gives rise to the execution of the consequent, i.e., one action is performed. Conventional approaches to knowledge representation are based on bivalent logic, which makes them have a serious shortcoming associated: their inability to deal with the issue of uncertainty and imprecision. As a consequence, conventional approaches do not provide an adequate model for modes of reasoning and all commonsense reasoning fall into this category.

Fuzzy Logic may be viewed as an extension of classical logical systems, providing an effective conceptual framework for dealing with the problem of knowledge representation in an environment

of uncertainty and imprecision. Fuzzy Logic, as its name suggests, is the logic underlying modes of reasoning which are approximate rather than exact. The importance of Fuzzy Logic derives from the fact that most modes of human reasoning —and especially commonsense reasoning— are approximate in nature. Fuzzy Logic is concerned in the main with imprecision and Approximate Reasoning [8].

In a wide interpretation of Fuzzy Logic (in terms of which it is coextensive with Fuzzy Set Theory, that is, classes of objects in which the transition from membership to nonmembership is gradual rather than abrupt), Fuzzy Logic and fuzzy sets have placed modeling into a new and broader perspective by providing innovative tools to cope with complex and ill-defined systems. The area of fuzzy sets has emerged following some pioneer works of Zadeh [1, 8] where the first fundamentals of Fuzzy Systems were established.

From a very broad point of view, a Fuzzy System is any Fuzzy Logic-Based System, where Fuzzy Logic may be used either as the basis for the representation of different forms of knowledge, or to model the interactions and relationships among the system variables. Fuzzy Systems have proven to be an important tool for modeling complex systems in which, due to the complexity or the imprecision, classical tools are unsuccessful [2, 3, 4, 5].

Concretely, the application of Fuzzy Logic to Rule-Based Systems leads us to FRBSs. These systems consider rules with the *"IF-THEN"* form whose antecedents and consequents are composed of Fuzzy Logic statements, thus presenting two essential advantages with respect to classical Rule-Based Systems:

- the key features of knowledge captured by fuzzy sets involve handling uncertainty, and
- inference methods become more robust and flexible with approximate reasoning methods of Fuzzy Logic.

Knowledge representation is enhanced with the use of linguistic variables and their linguistic values, that are defined by context-dependent fuzzy sets whose meanings are specified by graded membership functions [8]. On the other hand, inference methods such as generalized modus ponens, tollens, etc., which are based on Fuzzy Logic, form the basis of Approximate Reasoning with pattern matching scores of similarity [8]. Hence, Fuzzy Logic provides a unique computational framework for inference in Rule-Based Systems. Unlike traditional logical systems, Fuzzy Logic is aimed at providing modes of reasoning which are approximate and analog rather than exact.

II.B Types of Fuzzy Rule-Based Systems

Two different types of FRBSs are usually distinguished in the specialized literature according to the form of the fuzzy rules considered and to the types of inputs and outputs used [9]. We shall introduce them in the next subsections.

II.B.1 Mamdani Fuzzy Rule-Based Systems

This type of FRBS was proposed by Mamdani [10], who was able to translate Zadeh's preliminary assumptions to the first FRBS applied to a control problem. These kinds of Fuzzy Systems, the ones most used since then, are also referred to as FRBSs with Fuzzifier and Defuzzifier or, more commonly, as *fuzzy logic controllers* (name proposed by Mamdani in his pioneer papers [11]), due to the fact that their main application has been system control historically.

Mamdani FRBSs are composed of four main components: a *Knowledge Base* (KB), an *Inference System* and the *Fuzzification* and *Defuzzification Interfaces*. The KB —composed of Data Base (DB)

and Rule Base (RB)— stores the available knowledge about the problem in the form of linguistic *"IF-THEN"* rules. The Inference System puts into effect the inference process on the system inputs making use of the information stored in the KB. The generic structure of Mamdani FRBSs is shown in Figure 1.

Figure 1: Basic structure of a Mamdani Fuzzy Rule-Based System

The Mamdani-type FRBS has several very interesting characteristics:

- On the one hand, it may be used in real-world applications due to the fact that it is able to deal with real-valued inputs and outputs.
- On the other hand, it provides a natural framework to include expert knowledge in the form of linguistic rules and allows us to combine, in a very simple way, the latter with rules automatically generated from data sets representing the behavior of the system.
- Finally, it has more freedom in the choice of the Fuzzification and Defuzzification Interface components, as well as the Inference System ones, thus allowing us to design a more suitable FRBS for a specific problem. We shall analyze these aspects in the next subsection.

  As we shall see in Section II.C, where each of the components of this system kind will be analyzed in deep, the Fuzzification Interface establishes a mapping between crisp values in the input domain $\mathbf{U}$ of the system outputs and fuzzy sets defined in the same universe of discourse. On the other hand, the Defuzzification Interface develops the opposite operation by defining a mapping between fuzzy sets defined in the output domain $\mathbf{V}$ and crisp values defined in the same universe.

These systems present the maximum description level. The fuzzy rules are composed of input and output linguistic variables taking values from a linguistic term set with a real-world meaning. Therefore, each rule is a description of a condition-action statement that may be clearly interpreted by human beings. This fact makes Mamdani FRBSs appropriate for Linguistic Modeling, subarea of Fuzzy Logic Modeling in which the main characteristic is the model interpretability [12], and System Control [13] problems.

There exists an extension of Mamdani FRBSs based on changing the linguistic rule structure making it more flexible. This new type of FRBS makes use of DNF (Disjunctive Normal Form) fuzzy rules with the following form [14, 15, 16, 17]:

$$IF\ X_1\ is\ \widetilde{A_1}\ and\ \ldots\ and\ X_n\ is\ \widetilde{A_n}\ THEN\ Y\ is\ B\ ,$$

where each variable X_i has a referential set $\mathbf{U_i}$ and takes values in a finite domain (term set) D_i, $i = 1, \ldots, n$. The referential set for Y is $\mathbf{V}$ 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 $\widetilde{A_i}$, where $\widetilde{A_i} \in P(D_i)$ and $P(D_i)$ denotes the set of subsets of D_i. Thus, the complete syntax for the antecedent of the rule is

$$X_1 \; is \; \widetilde{A_1} = \{A_{11} \; or \; \ldots \; or \; A_{1l_1}\} \; and \; \ldots \; and \; X_n \; is \; \widetilde{A_n} = \{A_{n1} \; or \; \ldots \; or \; A_{nl_n}\} \; .$$

An example of this kind of rule is shown as follows. Let us suppose we have three input variables, X_1, X_2, and X_3, and one output variable, Y, such that the linguistic term set associated with each one is

$$D_1 = \{A_{11}, A_{12}, A_{13}\} \; , \; D_2 = \{A_{21}, A_{22}, A_{23}, A_{24}, A_{25}\} \; , \; D_3 = \{A_{31}, A_{32}\} \; , \; F = \{B_1, B_2, B_3\} \; .$$

In this case, a possible DNF rule may be

$$IF \; X_1 \; is \; \{A_{11} \; or \; A_{13}\} \; and \; X_2 \; is \; \{A_{23} \; or \; A_{25}\} \; and \; X_3 \; is \; \{A_{31} \; or \; A_{32}\} \; THEN \; Y \; is \; B_2 \; .$$

Recently a new more flexible KB structure that allows us to improve the accuracy of Linguistic Models without losing their interpretability has been proposed [18]. Now, it is allowed the linguistic rule to have two different consequents associated. This will allow the FRBS to locally improve its accuracy in complex space zones, because the final rule output lie in an intermediate zone between the both main labels. This transformation has a linguistic interpretation:

$$IF \; X_1 \; is \; A_1 \; and \; \ldots \; and \; X_n \; is \; A_n \; THEN \; Y \; is \; between \; B_1 \; and \; B_2 \; .$$

On the other hand, in the last few years it has been proposed a new variant of these kinds of systems, in which the system accuracy is more preferable than its interpretability. These systems are usually called *approximate Mamdani-type FRBSs* [2, 19, 20, 21, 22], the opposite to the ones following the classical structure presented that are usually named as *linguistic or descriptive Mamdani-type FRBSs*. The main application of the former is the area of Fuzzy Modeling [2], where the model accuracy is the main requirement instead of its description ability.

The structure of an approximate FRBS is the same that the descriptive one, which was shown in Figure 1. The only difference is related to the type of rule considered in the KB and, consequently, to the composition of this component. In this case, the rules do not use linguistic variables but, directly, fuzzy variables. Therefore, the structure of the fuzzy rules considered in approximate systems is the following:

$$IF \; X_1 \; is \; A_1 \; and \; \ldots \; and \; X_n \; is \; A_n \; THEN \; Y \; is \; B \; ,$$

where A_i and B are fuzzy sets without a direct interpretation instead of linguistic labels.

Hence, in these systems a DB is not found storing the existing linguistic terms and the fuzzy sets associated defining their semantics. In this case, the KB used in FRBSs of the previous types, that was composed of the said DB and of the RB, becomes a single Fuzzy Rule Base formed by a set of rules presenting the latter structure shown, where each individual rule directly contains the meaning describing it. Figure 2 graphically shows this.

II.B.2 Takagi-Sugeno-Kang Fuzzy Rule-Based Systems

Instead of working with linguistic rules of the kind introduced in the previous subsection, Takagi, Sugeno, and Kang [23, 24] proposed a new model based on rules where the antecedent was composed of linguistic variables and the consequent was represented by a function of the input variables. The

Figure 2: Graphical comparison between a linguistic KB and an approximate Fuzzy Rule Base

most usual form of these kinds of rules is the one shown in the following, in which the consequent constitutes a linear combination of the variables involved in the antecedent:

$$IF\ X_1\ is\ A_1\ and\ \ldots\ and\ X_n\ is\ A_n\ THEN\ Y = p_1 \cdot X_1 + \ldots + p_n \cdot X_n + p_0\ ,$$

with X_i being the system input variables, Y being the output variable, and p_i being real parameters. As regards A_i, they may be either linguistic labels with a meaning associated in the form of fuzzy sets, or fuzzy sets, in the case in which they are directly fuzzy variables. These kinds of rules are usually called *TSK fuzzy rules*, in reference to their creators.

The output of a TSK FRBS using a KB composed of m rules is obtained as the weighted average of the individual outputs provided by each rule, Y_i, $i = 1, \ldots, m$, as follows:

$$\frac{\sum_{i=1}^{m} h_i \cdot Y_i}{\sum_{i=1}^{m} Y_i}\ ,$$

with $h_i = T(A_1^i(x_1), \ldots, A_n^i(x_n))$ being the matching degree between the antecedent part of the i-th rule and the current inputs to the system, $x = (x_1, \ldots, x_n)$. T stands for a conjunctive operator modeled by a t-norm.

Therefore, and as it was enunciated by its creators in [23], this FRBS is based on dividing the input space in several fuzzy subspaces and defining a linear input-output relationship in each one of these subspaces. In the inference process, these partial relationships are combined in the said way for obtaining the global input-output relationship, taking into account the dominance of the partial relationships in their respective areas of application and the conflict existing in the overlapped zones. A graphical representation of this second kind of FRBSs is shown in Figure 3.

The main advantage of these systems is the fact that they present a compact system equation that allows us to estimate the parameters p_i by means of classical methods, which makes its design easier. Nevertheless, the main drawback relates to the form of the rule consequents as well, which causes the system not to constitute a natural framework for representing expert knowledge. It is possible to integrate expert knowledge in these FRBSs by performing a small modification on the rule consequent: when a linguistic rule with consequent $Y\ is\ B$ is provided by an expert, this

Figure 3: Basic structure of a TSK FRBS

consequent is substituted by $Y = p_0$, with p_0 standing for a characteristic value of the fuzzy set associated to the label B. These kinds of rules are usually called *simplified TSK rules* or *zero order TSK rules*.

II.C Components of a Linguistic Mamdani-type Fuzzy Rule-Based System

Summarizing the concepts introduced previously, a Mamdani-type FRBS is composed of the following components:

- A *Knowledge Base*, that contains the linguistic rules which guide the system behavior.
- A *Fuzzification Interface*, that takes care on transforming the crisp input data in values that may be handled in the fuzzy reasoning process, i.e., in fuzzy sets of any kind.
- An *Inference System*, that makes use of these values and of the information stored in the base to put into effect the inference process.
- A *Defuzzification Interface*, that transforms the fuzzy action obtained from the inference process in a crisp action that constitutes the global output of the FRBS.

We shall analyze in deep each one of these components in the following subsections.

II.C.1 The Knowledge Base

As we have mentioned, the component that stores these rules is the KB that is composed of two different components:

- RB: Formed by a set of linguistic *"IF-THEN"* rules that, in the case of multiple input-single output FRBSs, present the following structure:

  $$IF\ X_1\ is\ A_1\ and\ \ldots\ and\ X_n\ is\ A_n\ THEN\ Y\ is\ B\ ,$$

  with X_i and Y being input and output linguistic variables, respectively, and with A_i and B being linguistic labels. The RB is comprised by a collection of rules of this kind joined by the *also* operator, which means that, as we shall see in the following subsection, all of them may be fired when a specific input is given to the system.
- DB: Containing the definitions of the fuzzy sets associated to the linguistic terms used in the rules collected in the RB, that is, the membership function parameters.

II.C.2 The Fuzzification Interface

The Fuzzification Interface is one of the two components that allow Mamdani-type FRBSs to deal with real inputs and outputs. Its aim is to define a mapping that establishes a correspondence between each value in the crisp input space and a fuzzy set defined in the universe of discourse of this input, obtaining the membership function associated to each one of the system inputs. Symbolically, this component works as follows:

$$A' = F(x_0) \ ,$$

with x_0 being a crisp input value for the FRBS defined in the universe of discourse $\mathbf{U}$, A' being a fuzzy set defined in the same domain, and F being a fuzzification operator.

There are two main possibilities to choose the operator F:

1. *Punctual fuzzification*: A' is built as a punctual fuzzy set (a singleton) with support x_0, i.e., it presents the following membership function:

$$A'(x) = \begin{cases} 1, & \text{if } x = x_0 \\ 0, & \text{otherwise} \end{cases} \ .$$

2. *Non punctual or approximate fuzzification*: In this case, $A'(x_0) = 1$ and the membership degree of the remaining values decreases when moving away from x_0. This second kind of operator allows us to deal with different types of membership functions. For example, in the case of triangular-shaped fuzzy sets, it is possible to use the following one:

$$A'(x) = \begin{cases} 1 - \frac{|x-x_0|}{\sigma}, & \text{if } |x - x_0| \leq \sigma \\ 0, & \text{otherwise} \end{cases} \ .$$

The former is the one most used due to its simplicity.

II.C.3 The Inference System

The Inference System is the component that puts into effect the fuzzy inference process. To do so, it makes use of Fuzzy Logic principles for establishing a mapping between fuzzy sets defined in $\mathbf{U} = \mathbf{U_1} \times \mathbf{U_2} \times \ldots \times \mathbf{U_n}$ and fuzzy sets defined in $\mathbf{V}$ (with $\mathbf{U_1}, \ldots, \mathbf{U_n}$ and $\mathbf{V}$ being the domains where the input variables $X_1, \ldots, X_n$, and the output one Y are defined, respectively).

The fuzzy inference process is based on the application of the Generalized Modus Ponens, an extension of the classical Modus Ponens, proposed by Zadeh in the way [8]:

$$\begin{array}{l} IF\ X\ is\ A \quad THEN \quad Y\ is\ B \\ X\ is\ A' \\ \hline \qquad\qquad\qquad\qquad\qquad Y\ is\ B' \ . \end{array}$$

A fuzzy conditional statement with the form *"IF X is A THEN Y is B"* represents a fuzzy relation between A and B defined in $\mathbf{U} \times \mathbf{V}$. This fuzzy relation is expressed by a fuzzy set R whose membership function $\mu_R(x, y)$ is given by:

$$\mu_R(x, y) = I(\mu_A(x), \mu_B(y)), \forall x \in \mathbf{U}, y \in \mathbf{V} \ ,$$

with $\mu_A(x)$ and $\mu_B(y)$ being the membership functions of the fuzzy sets A and B, respectively, and I being a fuzzy implication operator modeling the existing fuzzy relation.

The membership function of the fuzzy set B' is obtained as a result of the application of the *Compositional Rule of Inference* (CRI) introduced by Zadeh in [8] as follows: *"If R is a fuzzy relation defined in* **U** *and* **V** *and A' is a fuzzy set defined in* **U***, then the fuzzy set B', induced by A', is obtained from the composition of R and A'"*, that is:

$$B' = A' \circ R \ .$$

Therefore, when the CRI is applied on rules whose antecedent is formed by n input variables and whose consequent has only one output variable, it presents the following expression:

$$\mu_{B'}(y) = Sup_{x \in \mathbf{U}} \left\{ T'(\mu_{A'}(x), I(\mu_A(x), \mu_B(y))) \right\} \ ,$$

where $\mu_{A'}(x) = T(\mu_{A'_1}(x), \ldots, \mu_{A'_n}(x))$, $\mu_A(x) = T(\mu_{A_1}(x), \ldots, \mu_{A_n}(x))$, with T and T' being fuzzy conjunctive operators (t-norms) and with I being a fuzzy implication operator.

Since, as we mentioned in the previous subsection, in the great majority of the cases the Fuzzification Interface transforms the input $x_0 = (x_1, \ldots, x_n)$ received by the system in several punctual fuzzy sets $A'_1, \ldots, A'_n$, and due to the application of the properties $T(1,1) = 1$ and $T(x,1) = x$, satisfied by t-norms [25, 26], the CRI is finally reduced to the form:

$$\mu_{B'}(y) = I(\mu_A(x_0), \mu_B(y)) \ .$$

II.C.4 The Defuzzification Interface

From the operation mode of the linguistic Mamdani-type FRBS Inference System introduced in the previous subsection, it may be clearly drawn the fact that the fuzzy inference process is applied at the level of individual rules. Thus, once the CRI has been applied on the m rules composing the KB, m fuzzy sets B'_i are obtained representing the fuzzy actions deduced by the FRBS from the inputs received.

Since the system must give a crisp output, the Defuzzification Interface has to develop the task of aggregating the information provided by each one of the fuzzy sets and transform it in a single crisp value. There are two different operation modes to do this aggregation [27]:

1. *Mode A: Aggregation first, defuzzification after:* In this first case, the Defuzzification Interface performs the following tasks:

   - Aggregates the individual fuzzy sets inferred, B'_i, for obtaining a final fuzzy set B' by means of a fuzzy aggregation operator, G, which models the *also* operator that relates the rules in the base:

     $$\mu_{B'}(y) = G\left\{ \mu_{B'_1}(y), \mu_{B'_2}(y), \ldots, \mu_{B'_n}(y) \right\} \ .$$

   - By using a defuzzification method, D, transforms the fuzzy set B' obtained in a crisp value, y_0, that will be given as system global output:

     $$y_0 = D(\mu_{B'}(y)) \ .$$

2. *Mode B: Defuzzification first, aggregation after:* In this second operation mode, the contribution of each fuzzy set inferred is individually considered by means of a characteristic value (center of gravity, mean of maximum value, etc.) and the final crisp value is obtained by means of an aggregation operator (an average, a weighted average, or the selection of one of

them, among others) performed on a crisp characteristic value of each one of the individual fuzzy sets.

This operation mode constitutes a different approach of the concept represented by the *also* operator.

Historically, the first mode proposed was the Mode A, that was already used by Mamdani in his first approach to Fuzzy Control [10]. In the last few years, the second approach is becoming more used [27, 28].

III Learning of Linguistic FRBSs

As we have mentioned, two main tasks have to be performed to design an intelligent system of this kind for a specific application: to select the fuzzy operators involved in the Inference System —i.e., to define the way in which the fuzzy inference process will be performed—, and to derive an adequate KB about the problem being solved. The accuracy of the FRBS in solving this problem will directly depend on both components.

The first design task has been widely analyzed in the specialized literature, and a big amount of theoretical and comparative studies have been carried out in order to deal with the problem of the selection of the best possible fuzzy operators in the Inference System [27, 29, 30].

As regards the second design task, it seems to be a more difficult decision because the composition of the KB depends directly on the problem being solved. Due to the complexity of the KB derivation, a large quantity of automatic techniques has been proposed to put it into effect.

Many of these techniques are collected under the name of Soft Computing (SC) [31, 32]. SC is a new field of Computer Science that deals with the integration of problem-solving techniques such as Fuzzy Logic, NNs, or GAs. Each of these techniques provides us with complementary reasoning and search methods to solve complex problems. Among all the possible combinations, we are interested on how NNs [33] or GAs [34] can help us to design an FRBS by defining the KB, acting as it is shown in Figure 4.

First of all, it is interesting to distinguish between tuning and learning problems. In tuning problems, a predefined RB and a preliminary DB are used and the objective is to find a better set of parameters defining the DB. In learning problems, a process including the learning of the RB or of the whole KB is performed.

In this section we shall focus on the learning of linguistic FRBSs. As we have mentioned in the previous section, Linguistic Models are characterized by offering a very appropriate interpretability that have repercussions on an expensive efficiency loss. Therefore, the methods for learning Linguistic Models are adopted in problems that require a high degree of interpretability.

We are going to introduce several learning algorithms for linguistic FRBSs. These are grouped in three approaches according to the three most well know techniques: *ad hoc* data covering approaches, NNs, and GAs. The methods that we shall see are collected in Table 1.

III.A *Ad Hoc* Data Covering Methods

With this name, "*Ad Hoc* Data Covering Methods", we are collecting those methods that are based on processes in which the learning of the fuzzy rules is guided by covering criteria of the data in the example set. Generally, they are characterized by being methods based on a short time-consuming iterative procedure.

Figure 4: Design of FRBSs

III.A.1 Wang and Mendel's Learning Method

The *ad hoc* data covering RB generation process proposed by Wang and Mendel in [35] has been widely known because of its simplicity and good performance. It is based on working with an input-output data set representing the behavior of the problem being solved, using a previous definition of the DB composed of the input and output primary fuzzy partitions.

The generation of the RB is put into effect by means of the following steps:

1. *Consider a fuzzy partition of the input variable spaces*: It may be obtained from the expert information (if it is available) or by a normalization process. If the latter is the case, perform a fuzzy partition of the input variable spaces dividing each universe of discourse into a number of equal or unequal partitions, select a kind of membership function and assign one fuzzy set to each subspace. In our case, we shall work with symmetrical fuzzy partitions of triangular membership functions (see Figure 5).

Figure 5: Graphical representation of a uniform fuzzy partition

2. *Generate a preliminary linguistic rule set*: This set will be formed by the rule best covering

Table 1: Learning and Tuning Methods used

Reference	Author(s) of the Method	Technique used
[35]	Wang and Mendel	*Ad hoc* Data Covering
[36]	Nozaki, Ishibuchi, and Tanaka	*Ad hoc* Data Covering
[37]	Shann and Fu	Neural Network
[38]	Jang (ANFIS)	Neural Network
[39]	Thrift	Genetic Algorithm
[40]	Liska and Melsheimer	Genetic Algorithm
[15]	González and Pérez (SLAVE)	Genetic Algorithm
[20]	Cordón and Herrera (D-MOGUL)	*Ad hoc* DC + Genetic Algorithm[a]
[18]	Cordón and Herrera (WM-ALM)	*Ad hoc* DC + Genetic Algorithm[a]
[38]	Jang (ANFIS for tuning)	Neural Network
[20]	Cordón and Herrera (tuning)	Genetic Algorithm

[a] They are composed of two stages: the first (Generation Process) is *Ad Hoc* and the second (Simplification Process) is GA-based.

each example (input-output data pair) contained in the input-output data set. The structure of these rules is obtained by taking a specific example, i.e., an $n+1$-dimensional real array (n input and 1 output values), and setting each one of the variables to the linguistic label (associated fuzzy set) best covering every array component.

3. *Give an importance degree to each rule*: Let R_l = *IF* x_1 *is* A_1 *and* ... *and* x_n *is* A_n *THEN* y *is* C be the linguistic rule generated from the example $e_l = (x_1^l, \ldots, x_n^l, y^l)$. The importance degree associated to it will be obtained as follows:

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

4. *Obtain a final RB from the preliminary fuzzy rule set*: The rule with the highest importance degree is chosen for each combination of antecedents.

III.A.2 Nozaki, Ishibuchi, and Tanaka's Learning Method

Nozaki *et al.* try to obtain a Linguistic Model that, maintaining the level of interpretability, presents a higher accuracy. To achieve it, they design a fuzzy model based on simplified TSK-type rules to, thereinafter, transform it into a Linguistic Model (having linguistic terms in the consequent) with the same performance. As we have said, the method consists of two phases. In a first step, a simplified TSK KB is learnt. The rule consequents are composed of a single real value that can be considered as the independent term of the linear combination in a TSK representation (introduced in Section II.B.2). In the second phase, each TSK rule is associated to two descriptive Mamdani-type rules defined in the same input subspace. To do so, we shall have two RBs: a main and a secondary ones. The method works as follows:

1. *First phase: Generation of TSK Rules.* To determine the real consequent of each rule, a weight is defined for each training data array as the result of raising the membership function of the input to the power of α (a parameter that defines a non-linear scaling function) and of getting the product of the membership function values of each input. The real consequent will

be obtained as the weighted average of the known output value of each array of input values. If the weight is zero, the rule will not be considered. In this way, we can quickly obtain a base of TSK rules with a linguistic antecedent and whose consequent is a real number.

2. *Second phase: Derivation of Mamdani rules from TSK rules.* To obtain better results taking the advantage offered by the TSK representation, the authors use two Mamdani-type RBs: the main and secondary ones. In the first place it will be necessary to accomplish a fuzzy partition of the universe of discourse of the consequent in some linguistic labels. They consider uniformly distributed partitions with triangular membership functions (see Figure 5).

   For each TSK rule, the greatest value of the membership function applied to the consequent is searched. The real value (x_0) will be substituted for the corresponding label obtaining a Mamdani rule. This rule will be inserted in the main RB. The corresponding label to the second best value will form the rule that is included in the secondary RB. Figure 6 graphically clarifies the operation mode.

Figure 6: Derivation of Mamdani rules from TSK rules in the Nozaki *et al.*'s Method

To draw profit from the two RBs, the authors present an Inference System capable of obtaining a combined result from them. To do so, a real value between 0 and 1 that acts as a certainty factor is associated to each rule. This value is the result of evaluating the TSK consequent on the corresponding membership functions, according to the RB (in the drawing, it would be h_3 and h_2).

III.B Neural Learning FRBS

A short introduction to Neural Networks and Neuro-Fuzzy Systems is presented in Appendix A. In the next subsections we introduce two specific methods with different characteristics: the first one learns fuzzy rules through the weights setting while the second one learns the membership function shapes.

III.B.1 Shann and Fu's Learning Method

The method proposed by Shann and Fu [37] use a layer-structured Neuro-Fuzzy System (NFS) for learning fuzzy rules by considering weights. The learning is produced adjusting the certainty factors of each rule.

The method is a two-phase learning procedure. The first phase is an error-backpropagation (EBP) training, and the second one is a rule-pruning algorithm.

1. *First phase: EBP training.* The NFS is composed of five layers (as shown in Figure 7).

Figure 7: Neural Network Structure proposed by Shann and Fu

- *Layer 1 (input layer)*: Each node in this layer represents an input linguistic variable of the network and it is used as a buffer to broadcast the input to the next layer. The domain of each input linguistic variable is determined by the application and it does not need to be constrained within [0,1].
- *Layer 2 (membership-function layer)*: Each node in this layer represents the membership function of a linguistic value associated to an input linguistic variable. Therefore, a membership function node is a fuzzifier. We can use any shape of membership function such as the triangle, trapezoid or bell-shaped ones. The function of the nodes in this layer are determined and formulated by applications.
- *Layer 3 (AND layer)*: Each node in layer 3 represents a possible IF-part for the fuzzy rules. In fuzzy set theory, there are many different operators for fuzzy intersection [6, 7, 27, 29, 30]. The authors choose the most commonly used one, i.e., the min-operator, which is simple and effective and has strong characteristics of competition.
- *Layer 4 (OR layer)*: Each node in layer 4 represents a possible THEN-part for the fuzzy rules. The operation performed by an OR node is to combine fuzzy rules with the same consequent. In this first phase, the links between layers 3 and 4 are fully connected so that all the possible fuzzy rules are embedded in the structure of the network. As in the previous layer, the authors choose the most commonly used one, i.e., the max-operator suggested by Zadeh [1], as the function of an OR node.

  The links of this layer have weights associated. For example, the weight w_{kj} of an input link in layer 4 represents the certainty factor of a fuzzy rule, which comprises the AND node j in layer 3 as the IF-part and the OR node k in layer 4 as the THEN-part. Hence, these weights are adjustable while learning the knowledge of fuzzy rules and they have to be nonnegative real numbers.

- *Layer 5 (defuzzification layer)*: Each node in this layer represents an output linguistic variable and performs defuzzification, taking into account the effects of all membership functions of the output linguistic values. Suppose that the correlation-product inference and the fuzzy centroid defuzzification scheme [41] are used. Then, the function of node l in layer 5 is defined as follows:

$$z_l = \frac{\sum_{k \in P_l}(x_{lk} a_{lk} c_{lk})}{\sum_{k \in P_l}(x_{lk} a_{lk})} \ ,$$

with P_l being the set of indices of the nodes in layer 4 that have an output link connected to node l, $x_{lk} = z_k$, and with a_{lk} and c_{lk} being the area and centroid of the membership function of the output linguistic value represented by node k in layer 4, respectively.

Since it is assumed that the membership functions of the output linguistic values are known, the areas and centroids can be calculated before learning.

2. *Second phase: Pruning.* The RB directly obtained right after the training phase will contain all the rules with nonzero weight, i.e., certainty factor. This causes the size of the RB to be very large in most cases. With this second phase, a concise RB is obtained. At the meantime, the structure of the NFS will be reduced as well. The technique involves calculating the centroids among the rules with the same antecedent. The linguistic variable where each centroid lies will determine the rule selected for each set of rules.

III.B.2 Jang's Learning Method (ANFIS)

In [38], Jang presents a new algorithm called ANFIS (Adaptive-Network-Based Fuzzy Inference System) that defines the composition of the DB. The inference system is implemented in an NN that uses an hybrid method to adjust the parameters in its nodes, a gradient method and the least squares estimate (LSE) to identify the parameters.

The proposed ANFIS can construct an input-output mapping based on both expert knowledge (in the form of linguistic rules) and specified input-output data pairs.

In spite of the fact that the author talks about several types of inference systems, he focuses on a TSK system to describe ANFIS, but he notes that it is possible to implement a Mamdani system as well. In the following, it will be first explained the TSK system and then the modifications to transform to Mamdani.

An adaptive network is a multilayer feedforward network in which each node performs a particular function (node function) on the incoming signals as well as a set of parameters pertaining to this node of which its output depends. These parameters can be fixed or variable, and it is through the change of the last ones that the network is tuned. ANFIS has nodes with variable parameters, called square nodes, which will represent the membership functions of the antecedents, and the membership function for the Mamdani-type consequent or the linear functions for the TSK-type consequent. The nodes in the intermediate layers connect the antecedents with the consequent. Their parameters are fixed and they are called circular nodes.

The node functions in the same layer belong the same function family as described below:

Layer 1: Parameters in this layer are referred to as premise parameters. In fact, any differentiable function, such as bell and triangular-shaped membership functions are valid for the nodes in this layer. Every node i in this layer is a square node with a node function. From now on, we consider that bell-shaped membership functions are used:

$$O_i^1 = \mu_{A_i}(x) \ , \quad \mu_{A_i}(x) = \frac{1}{1 + [(\frac{x - c_i}{a_i})^2]^{b_i}} \ .$$

Bell-shaped membership functions are used.

Layer 2: Each node represents the firing strength of a rule ω_i through a conjunction operator. The function considered is the minimum t-norm. They are circular nodes with a node function:

$$\omega_i = \begin{cases} \mu_{A_i}(x), & \text{if } \mu_{A_i}(x) \leq \mu_{B_i}(x) \\ \mu_{B_i}(x), & \text{if } \mu_{A_i}(x) > \mu_{B_i}(x) \end{cases} , \ i = 1, 2 \ .$$

Layer 3: It calculates the ratio of ith rule's firing strength with respect to the sum of all the rule's firing strengths.

$$\omega_i = \frac{\omega_i}{\omega_1 + \omega_2}, \ i = 1, 2$$

Layer 4: Every node i in this layer is a square node with a node function:

$$O_i^4 = \bar{\omega_i} f_i = \bar{\omega_i} \ (p_i x + q_i y + r_i) \ .$$

Layer 5: The single node in this layer computes the overall output as the sum of all incoming signals.

$$O_i^5 = \sum_i \bar{\omega_i} f_i = \frac{\sum_i \omega_i f_i}{\sum_i \omega_i}$$

The initial values of the antecedent and consequent parameters are set through a partition of the input universe of each variable.

After the network parameters are set with their initial values, the consequent is adjusted through the LSE [42]. This method is used because the structure of the TSK consequent, a linear combination of the inputs, makes it suitable to learn the weights composing the consequent. On the other hand, this is preferred to the strict gradient descent due to the former is quicker than the latter.

In order to achieve a desired input-output mapping, the network parameters are updated according to the given training data and the gradient descent method [43], that propagates the error rates from the output end towards the input end as a linear combination of the error in the nodes of the following layer which is multiplied by the derivative from the function of these nodes with respect to the output of this. By using this measure, it will be able to calculate the error that is produced by each parameter and to correct it. Let us suppose an adaptive network with L layers and $\#(k)$ nodes in layer k, we can denote by (k, i) the node at position i in layer k, and its membership function by

$$O_i^k = O_i^k(O_i^{k-1}, \ldots, O_{\#(k-1)}^{k-1}, a, b, c, \ldots) \ ,$$

with a, b, c, ... being the parameters pertaining to this node.

Assuming that the given training data set has P entries, taking $T_{m,p}$ as the mth component of the pth target output vector and $O_{lm,p}$ as the mth component of the actual output vector produced by the presentation of the pth input vector, the error measure for the pth entry of training data entry will be:

$$E_p = \sum_{m=1}^{\#(L)} (T_{m,p} - O_{m,p}^L)^2 \Longrightarrow E = \sum_{p=1}^{P} E_p \ .$$

The error rate for the output node at (L, i) can be calculated as

$$\frac{\partial E_p}{\partial O_{i,p}^L} = -2(T_{i,p} - O_{i,p}^L) \ ,$$

and for the internal node (k, i), the error rate can be derived by the chain rule as

$$\frac{\partial E_p}{\partial O_{i,p}^L} = \sum_{m=1}^{\#(k+1)} \frac{\partial E_p}{\partial O_{m,p}^{k+1}} \frac{\partial O_{m,p}^{k+1}}{\partial O_{i,p}^k} \quad .$$

Therefore, by means of these two expressions we can find $\frac{\partial E_p}{\partial O_{i,p}^k}$ for each node in the network. Then, the derivative of the overall error measure E with respect to α is

$$\frac{\partial E}{\partial \alpha} = \sum_{p=1}^{P} \frac{\partial E_p}{\partial \alpha} \quad .$$

Thus, the update formula for the generic parameter α is

$$\triangle \alpha = -\eta \frac{\partial E}{\partial \alpha} \quad ,$$

in which η is a learning rate which can be further expresed as

$$\eta = \frac{s}{\sqrt{\sum_\alpha (\frac{\partial E}{\partial \alpha})^2}} \quad ,$$

with s being the step size, the length of each gradient transition in the parameter space. Usually, we can change the value of s to vary the speed of convergence.

Some modifications have been made over the original system proposed by Jang in order to transform it in a Mamdani system and to consider triangular-shaped membership functions. These changes let the method vary the number of rules and tune the rules learnt previously.

In order to construct a Mamdani system, the expression of node function in the layer 4 must be changed: Every node in this layer calculates the rule consequent multiplied by the output of its corresponding node in layer 3. The consequent of the rule is obtained through the center of gravity weighted by the matching degree,

$$O_i^4 = \bar{\omega_i} f_i = \bar{\omega_i} \ (\frac{\int_{\mathbf{V}} y \cdot \mu_{C_i}(y) \cdot dy}{\int_{\mathbf{V}} \mu_{C_i}(y) \cdot dy}) \quad ,$$

whith C_i being the consequent membership function for the i rule.

The network architecture for a Mamdani system in ANFIS is shown in Figure 8.

It will not be permitted the use of the LSE technique in a Mamdani system with linguistic variables in the consequent. In Mamdani systems, the complete network will be adjusted by the gradient method.

To provide well formed triangular-shaped membership functions using the gradient method, each triangle must be represented by three parameters, its center, the square root of the distance to its left extreme, and the square root of the distance to its extreme right, allowing any type of triangle to be obtained. Thus, if the updated parameter has a negative value the triangle will still be correct.

Fuzzy partitions considering triangular-shaped membership functions do not necessarily have to satisfy the epsilon-completeness, which means that given a value x of one of the inputs in the universe of discourse, a linguistic label A can always be found so that $\mu_A(x) \geq \epsilon$, being

$$\mu_{A_i}(x) = \begin{cases} \frac{x-a}{b-a}, & \text{if } a \leq x \leq b \\ \frac{c-x}{c-b}, & \text{if } b > x \geq c \\ 0, & \text{if } x < a \text{ or } x > c \end{cases} \quad .$$

Figure 8: Structure of a Mamdani-type ANFIS, Jang's Method

Using the gradient method, the membership functions could fall apart from the domain causing the corresponding rules to be eliminated. If too many rules are eliminated, the system will have a wrong behavior. The solution is to insert in the universe of discourse, according to a probability defined beforehand, some of those membership functions, so that they can be tuned again. This is done for the antecedents as well as for the consequent.

III.C Genetic Learning FRBS

A short introduction to GAs and Genetic Fuzzy Systems is presented in Appendix B. In the following we shall review several methods based on GAs under the Pittsburgh and IRL approaches.

III.C.1 Thrift's Learning Method

This method is based on encoding all the cells of the complete decision table in the chromosomes. In this way, Thrift [39] establishes a mapping between the label set associated to the system output variable and an ordered integer set (containing one more element and taking 0 as its first element) representing the allele set. An example is shown to clarify the concept. Let $\{NB, NS, ZR, PS, PB\}$ be the term set associated to the output variable, and let us note the absence of value for the output variable by the symbol "–". The complete set formed joining this symbol to the term set is mapped into the set $\{0, 1, 2, 3, 4, 5\}$. Hence the label NB is associated with the value 0, NS with 1, ..., PB with 4 and the blank symbol "–" with 5.

Therefore, the GA employs an integer coding. Each one of the chromosomes is constituted by joining the partial coding associated to each one of the linguistic labels contained in the decision table cells. A gene presenting the alelle "–" will represent the absence of the fuzzy rule contained in the corresponding cell in the RB.

The GA proposed employs an elitist selection scheme and the genetic operators used are of different nature. While the crossover operator is the standard two-point crossover, the mutation operator is specifically designed for the process. When it is applied over an allele different from the blank symbol, changes it one level either up or down or to the blank code. When the previous gene value is the blank symbol, it selects a new value at random.

Finally, the fitness function is based on an application specific measure. The fitness of an individual is determined by applying the FRBS considering the RB coded in its genotype to the controlled system with several different starting points and computing the convergence to the desired equilibrium point.

III.C.2 Liska and Melsheimer's Learning Method

Liska and Melsheimer [40] present a general method that covers all the areas of KB learning: number of fuzzy rules, structure of the rules, and membership function parameters. The power of their method lies on learning optimizing all the three parts simultaneously, against doing it separately, which could result in a suboptimal solution from the authors' point of view.

The KB is represented as a chromosome composed of three substrings:

- The first substring of real numbers encodes the membership functions associated to all system variables. Each membership function is represented by two parameters —center and width— that allow to use bell-shaped or isosceles triangular-shaped membership functions.
- The second substring of integer numbers encodes the structure of each rule in the RB so that one integer number represents one membership function in the space of an input variable. Membership functions are numbered in ascending order according to their centers, i.e., a number '1' refers to the membership function with the lowest value of the membership function center in a particular input variable. The value '0' in the second substring indicates the "null" membership function, i.e., the input variable is not involved in the rule.
- The third substring of integer numbers encodes membership functions in the rule consequents. The integer numbers refer to membership functions in the same way as in the second substring. A value '0' in the third substring means that the rule is deleted from the FRBS RB.

The inclusion of '0' in the second and third substrings allows both the number of input variables involved in each rule and the number of rules to change dynamically during the GA search. The number of rules is constrained by an upper limit specified by the designer.

As regards the genetic operators (crossover and mutation), they act in a different way depending on what substring of the chromosome they are applied on.

On the one hand, if we are working with the first substring it is necesary that the operator acts in the way that the relative order of membership functions in each variable is preserved, e.g., a center of the membership function with a linguistic meaning "low" can take any value in the variable range but always lower than a center of the membership function "high".

On the other hand, with the second and the third substring we shall use three specific operators that are not applied at the level of genes but of fields in the second substring. In this case, a field is considered to be the antecedents of a rule.

- *Uniform crossover* creates two offspring from two parents by deciding randomly, field by field, which offspring receives the field from which parent.
- *Mutation* randomly replaces selected fields in a parent by a random value between the minimum and maximum allowed values.
- *Creep* creates one offspring from one parent by randomly altering its field within a specified range.

The performance of each chromosome in a population is evaluated by an exponential ranking technique based on the error function. This technique assigns the highest fitness $fitness(1) = 1000$ to the chromosome with the lowest value of the error function, the chromosome with the next lowest value of the error function receives $fitness(2) = \alpha * fitness(1)$, $\alpha \in [0, 1]$, or generally, $fitness(N + 1) = \alpha * fitness(N)$ with the exception that no string was given a fitness lesser than

1. The higher the fitness a string receives, the higher the probability it has to be selected for reproduction. Liska and Melsheimer obtained the best results when α was set constant to 0.96. In their GA implementation they also do not allow duplicates, i.e., a new offspring is introduced into the current population only if it differs from every member of the current population at least in one field.

Once the genetic learning process has finished, the method applies a second phase of tuning of membership functions for ensuring the optimum adjustment. The objective is to avoid the stagnation of the GA. This fine-tuning is developed with a multilayer feedforward network [40]. In contrast to the feedforward NNs, the tuned parameters in this realization are the centers and widths of membership functions in the fuzzification and defuzzification interfaces, and all the interconnections are assigned a constant weight 1.

III.C.3 González and Pérez's Learning Method (SLAVE)

SLAVE (Structural Learning Algorithms in Vague Environments) is a inductive learning algorithm, that was initially proposed in [14] and later developed in [15, 44]. The basic element of the SLAVE learning algorithm is its model of rule

$$IF\ X_1\ is\ \widetilde{A_1}\ and\ \ldots\ and\ X_n\ is\ \widetilde{A_n}\ THEN\ Y\ is\ B\ ,$$

where each variable X_i has a referential set $\mathbf{U_i}$ and takes values in a finite domain (term set) D_i, $i = 1, \ldots, n$. The referential set for Y is $\mathbf{V}$ 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 $\widetilde{A_i}$, where $\widetilde{A_i} \in P(D_i)$ and $P(D_i)$ denotes the set of subsets of D_i. As we saw in Section II.B.1, this type of FRBS are called DNF (Disjunctive Normal Form). It has been used by other authors [16, 17] as well.

The key to this rule model is that each variable can take as a value an element or a subset of elements from its domain. The concept may be clarified with the following example:

Let X_i be a variable whose domain is shown in Figure 5(a). An antecedent like

$$\ldots\ and\ X_i = \{ZR, PS, PM\}\ and\ \ldots\ ,$$

is equivalent to

$$\ldots\ and\ \{X_i\ is\ ZR\ or\ X_i\ is\ PS\ or\ X_i\ is\ PM\}\ and\ \ldots$$

Using the previous model of rule, the set of all possible rules is

$$\Delta = P(D_1) \times P(D_2) \times \ldots \times P(D_n) \times F\ .$$

Another important characteristic of this learning algorithm is that it uses the genetic IRL approach for generating the rule set (see *Genetic Fuzzy Systems* in Appendix B). The iterative approach is presented as an alternative to the well-known Michigan and Pittsburgh approaches for genetic learning. This approach involves including a GA in a iterative scheme similar to the following [45]:

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 sufficient to represent the examples in the training set, the system will return this set of rules as the solution. Otherwise return to step 1.

The GA obtains a rule that is a partial solution of the learning problem in each step. The true solution is obtained by appending each rule to the rule set. In the recent literature we may find different algorithms that use this new approach, such as [14, 20, 46]. A study about the problems generated by the use of the IRL approaches can be seen in [45].

The SLAVE parameters are described in [15]. For measuring the goodness of the rule are introduced two definitions that are based on the classical consistency and completeness conditions. These conditions provide the logical foundation of the algorithms for concept learning from examples.

Definition 1. The completeness condition states that every example in some class must verify some rule from this class.

Definition 2. The consistency condition states that if an example satisfies a description of some class, then it cannot be a member of a training set of any other class.

These definitions are associated to the whole set of rules, but SLAVE obtains the set of rules that describes the system by extracting one rule in each iteration of the learning process. For this reason, the authors need to define these concepts on each rule. Moreover, they are not interested in proposing hard definitions on fuzzy problems, thus, they propose a degree of completeness and a degree of consistency, both definitions using the concepts proposed in [15].

Definition 3. 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 is the number of examples of the B class and $n^+(R_B(A))$ is the number of positive examples covered by $R_B(A)$.

The soft consistency degree is based on the possibility of admitting some noise in the rules. Thus, in order to define the soft consistency degree the authors use the following set:

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

that represents the set of rules having a number of negative examples ($n^-(R_B(A))$) strictly lesser than a percentage (depending on k) of the positive examples ($n^+(R_B(A))$).

Definition 4. The degree in which a rule R satisfies the soft consistency condition is

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

where $k_1, k_2 \in [0, 1]$ and $k_1 < k_2$, and $n^-(R)$, $n^+(R)$ 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.

These definitions are based on the use of the cardinality of two fuzzy sets (the positive and negative example sets). We can say that an example is positive for a rule when it matches the antecedent and consequent of the rule. On the contrary, we consider it as a negative example when it matches the antecedent but not the consequent. In any other case, we can say that it has no influence on this rule.

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

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

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

Figure 9 shows the different elements of the SLAVE learning algorithm and the relations among them.

Figure 9: Description of SLAVE Learning Process

III.C.4 Cordón and Herrera's Learning Method (D-MOGUL)

In this subsection we are going to introduce the Descriptive-MOGUL approach [20] based in the MOGUL paradigm presented in [47]. Making use of this method it will be possible to automatically generate a complete KB when a example training set is available. It consists of the following three steps:

1. An *iterative RB generation process* of desirable fuzzy rules able to include the complete knowledge of the set of examples.
2. A *genetic simplification process*, which finds the final RB able to approximate the input-output behavior of the real system. It is based on eliminating some unnecessary rules from the rule set obtained in the previous stage, avoiding thus the possible overlearning, by selecting the subset of rules best cooperating.
3. A *genetic tuning process* of the DB used in order to improve as far as possible the accuracy of the final KB.

The Rule Base Generation Process

The first stage consists of an increasing RB generation process based on an iterative exploration of the problem search space. Apart from the training data set E_p, a previously defined DB, constituted by uniform fuzzy partitions with triangular membership functions crossing at height 0.5, is considered. The FRBS designer can specify the number of linguistic terms forming each one of them in order to obtain the desired granularity level. Figure 5 shows the generic structure of a fuzzy partition with seven linguistic labels.

This component allows us to obtain a set of Mamdani-type fuzzy rules B^g describing the system behavior. In order to do that, it is necessary to establish a condition for it. This is the requirement of covering all possible situation-action pairs, $e_l \in E_p$, the *completeness property* [15, 48]. This may be formalized for a constant $\tau \in [0, 1]$, it requires the non-zero union of fuzzy sets $A_i(\cdot)$, $B_i(\cdot)$, $i = 1, ..., T$, $T = |B^g|$, and is formulated by the following expressions:

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

$$R_i : \text{If } x_1 \text{ is } A_{i1} \text{ and } ... \text{ and } x_n \text{ is } A_{in} \text{ then } y \text{ is } B$$

$$e_l = (ex_1^l, \ldots, ex_n^l, ey^l)$$

$$R_i(e_l) = *(A_i(ex^l), B_i(ey^l))$$

$$A_i(ex^l) = *(A_{i1}(ex_1^l), ..., A_{in}(ex_n^l)) \ ;$$

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 a set of rules R, the *covering value* of an example e_l is defined as

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

and we require the following condition

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

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

The RB is derived rule by rule, selecting the most accurate one at each step in the algorithm. Once this rule is obtained, its covering over the training set examples is taken into account. Those examples covered in a degree higher than a value ϵ specified by the FRBS designer are removed from the training set. Hence, the increasing example covering guides the search to other promising space zones at each step.

Each time the best rule has to be selected in the generation process, the accuracy of the candidates is measured by using a multicriteria fitness function. This function is designed taking into account the following three criteria allowing us to ensure the completeness and consistency of the final KB generated:

a) *High frequency value* [48]

The frequency of a fuzzy rule, R_i, through the set of examples, E_p, is defined as:

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

b) *High average covering degree over positive examples* [48]

The set of positive examples to R_i with compatibility degree greater than or equal to ω is defined as:

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

The *average covering degree* on $E_\omega^+(R_i)$ can be defined as:

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

where $n_\omega^+(R_i) = |E_\omega^+(R_i)|$.

c) *Small negative examples set* [15]

The set of the negative examples to R_i is defined as:

$$E^-(R_i) = \{e_l \in E_p / R_i(e_l) = 0 \text{ and } A_i(ex^l) > 0\} \ .$$

An example is considered negative for a rule when it better matches some another rule that has the same antecedent but a different consequent. The negative examples are always considered over the complete training set.

With $n_{R_i}^- = |E^-(R_i)|$ being the number of negative examples, the *penalty function on the negative examples set* will be:

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

where we permit up to a percentage of the number of positive examples, $k \cdot n_\omega^+(R_i)$, of negative examples per rule without any penalty. This percentage is determined by the parameter $k \in [0, 1]$.

Therefore these three criteria are combined into a fitness function using any aggregation function increasing in the three variables. The authors propose the product:

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

Rules getting a higher value in this function will be more accurate.

Taking the previous comments into account, the generation method is developed in the following steps:

1. Initialization:

   (a) To introduce the k, ω, and ϵ parameter values.

(b) To set the example covering degree $CV[l] \leftarrow 0$, $l = 1, ..., p$.

(c) To initialize the final Rule Base B^g to empty.

2. To initialize the candidate fuzzy rule set B^c to empty.

3. For every $e_l \in E_p$, generate the fuzzy rule R_c best covering it by taking the linguistic label matching best with the e_l component value for each variable. If $R_c \notin B^c$, add it to B^c.

4. To evaluate all the fuzzy rules contained in B^c and to select the one getting a higher value in the fitness function, R_r.

5. To introduce R_r into the set of rules B^g.

6. For every $e_l \in E_p$ do

   (a) $CV[l] \leftarrow CV[l] + R_r(e_l)$.

   (b) If $CV[l] \geq \epsilon$ then remove e_l from E_p.

7. If $E_p = \emptyset$ then Stop else return to Step 2.

The Genetic Simplification Process

Due to the iterative nature of the generation process, an overlearning phenomenon may appear. This occurs when some examples are covered at a higher degree than the desired one and it makes the RB obtained perform worse. In order to solve this problem and improve its accuracy, it is necessary to simplify the rule set obtained from the previous process, removing the redundant rules and selecting the rule subset with best cooperation for deriving the final RB solving the problem.

The simplification process used was proposed in [48]. It is based on a binary coded GA, in which the selection of the individuals is developed using the stochastic universal sampling procedure proposed by Baker in [49] together with an elitist selection scheme, and the recombination 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 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 RB 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 every $c_i = 1$. The remaining chromosomes are selected at random.

As regards to the fitness function, $E(\cdot)$ it is based on an application specific measure usually employed in the design of FRBSs, the mean square error over a training data set, E_{TDS}, which is represented by the following expression:

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

whith $S(ex^l)$ being the output value obtained from the FRBS using the RB coded in C_j, $R(C_j)$ when the input variable values are ex^l, and ey^l is the known desired value.

Anyway, there is a need to keep the *control rule completeness* property considered in the previous stage. An FRBS must always be able to infer a proper output for every system input. We shall ensure this condition by forcing every example contained in the training set to be covered by the encoded RB in a degree greater than or equal to τ,

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

where τ is the minimal training set completeness degree acepted 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_{TDS} as

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

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

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

The Genetic Tuning Process

Finally, a modified version of the genetic tuning method presented in [50] is applied by the authors. Since we are going to use this third phase as a independent algorithm to adjust the KB previously learned with some of the reviewed methods, it will be presented in Section IV.B.

III.C.5 Cordón and Herrera's Learning Method (WM-ALM)

In [18], the authors propose a methodology to design Linguistic Models with high accuracy and a good description level, called Accurate Linguistic Modeling (ALM). In the same paper, two different learning methods based on the paradigm are proposed. The WM-ALM method is composed of two stages, which will be described in the following:

1. A *linguistic rule generation method* from examples based on a modification made on a *ad hoc* data covering fuzzy rule generation processes, the Wang and Mendel's method [35] (showed in Section III.A.1). The modification involves generating the two most important consequents for each combination of antecedents (instead of only the most important one, as this kind of methods usually do). The authors remark that an input subspace will only have two consequents associated (and thus, two linguistic rules) when there is a need to do so, i.e., when there is any data in it or when two different consequents may be generated.

2. A *rule selection genetic process*, that removes the redundant or unnecesary rules from the fuzzy rule set generated in the previous step to select the subset of them cooperating best. This process will be implemented by means of a binary-coded Genetic Algorithm.

As mentioned, the Wang and Mendel's method is modified to allow to have (if is necessary) two consequent for a specific combination of antecedents. The fourth step (see Section III.A.1) is the only one that is different from the original Wang and Mendel's algorithm. Whilst in that method the rule with the highest importance degree is the only one chosen for each combination of antecedents, now two different rules, the two most important ones in each input subspace (if they exist), are allowed to form part of the RB.

Of course, a combination of antecedents may not have rules associated (if there are no examples in that input subspace) or only one rule (if all the examples in that subspace generated the same rule). Therefore, the generation of rules with double consequent is only addressed when the problem complexity, represented by the example set, shows that it is necessary.

The operation mode of the generation method means that, in each input subspace, the rules are created individually from the examples in the input-output data set without taking into account the cooperation existing among them to give the final model output. That is, no information about the neighbor rules is considered in order to generate them.

Because of this, the generated RB may present redundant or unnecessary rules making the model using this KB less accurate. In order to avoid this fact and to achieve that more than a single rule is used only in those zones where it is really necessary, the authors use a genetic rule selection process with the aim of simplifying the initial linguistic rule set by removing the unnecessary rules from it and generating a KB with good cooperation.

The selection of the subset of linguistic rules best cooperating is a combinatorial optimization problem [12, 51]. Since the number of variables involved in it, i.e., the number of preliminary rules, may be very large, they consider an approximate algorithm to solve it, a GA.

This selection algorithm was presented in the second subsection of the Section III.C.4 as a simplification process.

IV Tuning of Linguistic FRBSs

The performance of an FRBS depends on its RB and on the membership functions of associated to the fuzzy partition, i.e., the DB. Hence, it is very important to tune these parameters to the process to be modeled. The tuning methods fit the membership functions of the fuzzy rules obtaining high-performance membership functions by minimizing an error function defined by means of the system behavior or the evaluation of a training example set.

Recent work has centered on the use of mathematical and heuristic optimization techniques as gradient descent, GAs, etc. In the following, we shall present two approaches for tuning the parameters of the membership functions, the first one that represents the KB by means of an NN, being tuned by the EBP, and the second one that uses GAs.

IV.A Jang's Tuning Method

The method proposed by Jang [38] is more a tuning method than a learning method. Jang initially determines the labels of the input variables through a uniform partition of the universe of each variable, and the parameters of the consequent are initiated to zero. A fixed rule base is created from the one that it begins to adjust. A first assumption for using this approach as a tuning method is that with a finer initialization, the method will work better. We shall allow it to start from a KB already learnt through another method. If the initial KB is simplified, the nodes that do not participate are disconnected. In this sense will be used ANFIS as a tuning method. The operation of the method can be consulted in Section III.B.2.

IV.B Cordón and Herrera's Tuning Method

This process, introduced in [20], is based on the existence of a previous complete Mamdani KB, that is, an initial DB definition and an RB constituted by m Mamdani-type rules.

Each chromosome forming the genetic population will encode a complete DB definition that will be combined with the existing RB in order to evaluate the individual adaptation.

The GA designed for the tuning process presents real coding issue [52], uses the stochastic universal sampling [49] as a selection procedure and Michalewicz's non-uniform mutation operator. As regards to the crossover operator, the *Max-Min-Arithmetical* [50], which makes use of fuzzy tools in order to improve the GA behavior, is employed.

As it has been commented in Section III.C.4, the fuzzy partitions are triangular-shaped (see Figure 5). Thus, each one of the membership functions has an associated parametric representation based on a 3-tuple of real values and a primary fuzzy partition can be represented by an array composed by $3 \cdot N$ real values, with N being the number of terms forming the linguistic variable term set. The DB is encoded into a fixed length real coded chromosome C_r built by joining the partial representations of each one of the variable fuzzy partitions as is shown in the following:

$$C_{ri} = (a_{i1}, b_{i1}, c_{i1}, \ldots, a_{iN_i}, b_{iN_i}, c_{iN_i})$$
$$C_r = C_{r1}\ C_{r2}\ \ldots\ C_{rm}\ ,$$

where C_{ri} codes the fuzzy partition of the i-th variable.

The initial gene pool is created making use of the initial DB definition. This is encoded directly in 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, $h = 1 \ldots \sum_{i=1}^{m} N_i \cdot 3$. Each interval of performance will be the interval of adjustment for the corresponding gene, $c_h \in [c_h^l, c_h^r]$.

If (t mod 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:

$$c_t \in [c_t^l, c_t^r] = [c_t - \tfrac{c_{t+1}-c_t}{2}, c_t + \tfrac{c_{t+1}-c_t}{2}]\ ,$$
$$c_{t+1} \in [c_{t+1}^l, c_{t+1}^r] = [c_{t+1} - \tfrac{c_{t+1}-c_t}{2}, c_{t+1} + \tfrac{c_{t+2}-c_{t+1}}{2}]\ ,$$
$$c_{t+2} \in [c_{t+2}^l, c_{t+2}^r] = [c_{t+2} - \tfrac{c_{t+2}-c_{t+1}}{2}, c_{t+2} + \tfrac{c_{t+2}-c_{t+1}}{2}]\ .$$

Figure 10 shows a graphical representation of these intervals.

Figure 10: Intervals of performance

Therefore the authors 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 $E(\cdot)$ presented in the genetic simplification process introduced in Section III.C.4 is used for evaluating the adaptation of each individual of the population in the genetic tuning process.

V Examples of Application: Experiments Developed and Results Obtained

To illustrate the performance of the seen methods, two applications with different features have been chosen: the problem of rice taste evaluation [36] and a real-world Spanish electrical distribution network problem [53, 54, 55].

We are going to develop a comparative study between the eleven methods for each problem. Afterwards, in Section V.C, we shall carry out a global analysis where we shall look at the general behavior of the different methods.

With the aim of making easier the identification of the methods, the abbreviations shown in Table 2 will be considered.

Table 2: Notations considered in this section

Section	**Author(s) of the Method**	**Abbreviation-Name**
III.A.1	Wang and Mendel	WM
III.A.2	Nozaki, Ishibuchi, and Tanaka	NIT
III.B.1	Shann and Fu	SF
III.B.2	Jang (ANFIS)	$J\text{-}ANFIS$
III.C.1	Thrift	T
III.C.2	Liska and Melsheimer	LM
III.C.3	González and Pérez (SLAVE)	$GP\text{-}SLAVE$
III.C.4	Cordón and Herrera (D-MOGUL)	$CH\text{-}D\text{-}MOGUL$
III.C.5	Cordón and Herrera (WM-ALM)	$CH\text{-}WM\text{-}ALM$
IV.A	Jang (ANFIS for tuning)	$J\text{-}ANFIS\text{-}tuning$
IV.B	Cordón and Herrera (tuning)	$CH\text{-}tuning$

We have considered the parameters showed in Table 3 for every method in every experiment developed. With these values we have tried to select standard parameters that work well in most cases instead of searching very specific values for each problem.

Each table of results will present three groups of data associated to each method: the results obtained by the method itself (noted by Generation) and two columns showing those obtained after the application of each one of the introduced tuning processes (noted by $J\text{-}ANFIS\text{-}tuning$ and $CH\text{-}tuning$ respectively).

The NIT method uses a scale factor (α) that deforms the membership function shape, which can make it to be considered as a tuning process. For this reason, we do not apply any additional tuning to this method. In the case of the $CH\text{-}D\text{-}MOGUL$ method, the generation results showed are the ones obtained after the application of the Simplification Process (second stage).

To evaluate the quality of the results we are going to use an error function called *mean square error* (MSE). We have defined MSE as

$$MSE = \frac{1}{2 \cdot N} \sum_{i=1}^{N} (y'_i - y_i)^2 \ .$$

with y' being the output obtained from the FRBS and with y being the known desired output. The more closer to zero the measure is, the greater the efficiency is.

Table 3: Parameters values considered for the methods

Method	Parameters	Decision
WM	*Without Parameters*	
NIT	Parameter α	5
SF	Maximum number of epochs	1000
	Learning rate β	0.01
J-ANFIS	Maximum number of epochs	1000
and	Step size	0.1
J-ANFIS-tuning	Decrease rate of step size	0.9
	Increase rate of step size	1.1
	Label insertion percentage	60%
T	*Genetic Parameters explained lower down*	
LM	Creep probability	0.1
	Maximum number of rules	*Max allow*
	Genetic Parameters explained lower down	
GP-SLAVE	Maximum number of iterations	1000
	Parameter λ	0.8
	Parameter K1	0
	Parameter K2	1
	Adaptation Degree	0
CH-D-MOGUL	**Generation:**	
	Example covering value ϵ	1.5
	Positive example degree ω	0.05
	k-consistency property parameter	0.1
	Simplification:	
	Completeness property parameter τ	0.1
	Number of simplified KBs to generate	3
	Niche radius r	10% of initial KB rule no.
	Power factor β	0.5
	Genetic Parameters explained lower down	
CH-WM-ALM	*Genetic Parameters explained lower down but with 500 generations*	
CH-tuning	Non-uniform mutation parameter b	5
	Max-min-arithmetical parameter a	0.35
	Completeness property parameter τ	0.1
	Genetic Parameters explained lower down	
Genetic Parameters	Maximum number of generations	1000
	Population size	61
	Crossover probability	0.6
	Mutation probability	0.1

V.A Rice Taste Evaluation

V.A.1 The Rice Taste Evaluation Problem

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* [36].

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 performed 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 linguistic model to solve the said problem.

In order to do so, we are going to use the data set presented in [36]. 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 set mentioned, composed by 75 pieces of data in the training set and 30 in the test one, for generating ten qualitative models in each experiment. We have worked with fuzzy partitions obtained from a normalization process in which the universe of discourse of each one of the six variables has been equally divided into 2 parts, and a triangular fuzzy set has been associated to each one of them. Figure 5 shows an example of a fuzzy partition with seven linguistic labels.

V.A.2 Experiments and Results in the Rice Taste Evaluation Problem

The results obtained in the experiments developed are collected in Table 4. The values shown in columns MSE_{tra} and MSE_{tst} have been computed as an average of the MSE values obtained in the approximation of the training and test data sets, respectively, by the ten linguistic models generated in each case. The column $\#R$ stands for the average of the number of linguistic rules in the KBs of the models generated from each process.

Table 4: Results obtained in the rice taste evaluation problem

	Generation			*J-ANFIS-tuning*			*CH-tuning*	
Method	#R	MSE_{tra}	MSE_{tst}	#R	MSE_{tra}	MSE_{tst}	MSE_{tra}	MSE_{tst}
WM	15	0.01328	0.01312	15	0.00363	0.00372	0.00111	**0.00214**
NIT	64	0.00300	0.00352	–	—	—	—	—
SF	32	0.01940	0.02137	32	0.00397	0.00481	0.00183	0.00331
J-ANFIS	32	0.00503	0.00563	–	—	—	0.00277	0.00387
T	15.9	0.00495	0.00600	15.9	0.00348	0.00483	0.00115	0.00293
LM	30.6	0.00128	0.00236	30.6	0.00128	0.00236	**0.00081**	0.00234
GP-SLAVE	24	0.01803	0.02049	24	0.01771	0.02290	0.00791	0.01325
CH-D-MOGUL	6	0.00486	0.00370	**4**	0.01018	0.01015	0.00108	0.00233
CH-WM-ALM	5	0.00341	0.00398	5	0.00247	0.00323	0.00103	0.00274

It is very remarkable the excellent treatment of the complexity by *CH-D-MOGUL* and *CH-WM-ALM* that obtain bases with a very small number of rules (an average of 6 and 5 respectively). The number of rules is extremely important in problems that need to be as highly interpretable as this. Therefore, the work carried out in the phase of Simplification (used in both methods) is very valuable.

On the other hand, *LM* obtains results of excellent accuracy but at the expense of to generate a large number of rules that, as said, is not desirable in this case.

As regards the tuning processes, the adjustment of *CH-tuning* improves all the results and equalizes the efficiency grades of the methods. The adjustment of *J-ANFIS-tuning* also improves the results but less significantly than *CH-tuning*. When the models obtained by *CH-D-MOGUL* are tuned, the results makes worse a lot. This is due to the fact that, upon to have a low number of

rules, when *J-ANFIS-tuning* eliminates some of them, it increases the error. This newly indicates that the selection realized in *CH-D-MOGUL* is very appropriate and the number of rules is optimal.

Although *NIT* generate good values, all of the methods produce less rules than it. We must not forget that in this problem the complexity is very important. Furthermore, if *CH-tuning* is applied, all except *J-ANFIS* and *GP-SLAVE* obtain also better accuracy.

V.B Electrical Distribution Networks

V.B.1 The Electrical Distribution Networks Problem

Sometimes, there is a need to measure the amount of electricity lines that an electric company owns. This measurement may be useful for several aspects such us the estimation of the maintenance costs of the network, which was the main goal of the problem presented here in Spain [53, 54, 55]. 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 used to be very convoluted and, in some cases, one company may serve more than 10,000 small nuclei. An indirect method for determining the length of line is needed.

The problem involves finding a model that relates the total length of low voltage line installed in a rural town with the number of inhabitants in the town and the mean of the distances from the center of the town to the three furthest clients in it [54, 55]. This model will be used to estimate the total length of line being maintained.

We shall limit ourselves to the estimation of the length of line in a town, given the inputs mentioned before. Hence, our objective is to relate the first variable (line length) with the other two ones (population, radius of village).

In this problem, it would be preferable that the solutions obtained verify the following requirement: they have not only to be numerically accurate in the problem-solving, but must be able to explain how a specific value is computed for a certain village. That is, it is interesting that *these solutions are interpretable by human beings to some degree.*

To compare the methods, we have randomly divided the sample, composed of 495 pieces of real data obtained from direct measures in this number of villages [53], into two sets comprising 386 and 99 samples, labeled training and test. In this case, the linguistic variable fuzzy partitions are divided into 7 fuzzy sets in the experiments developed.

V.B.2 Experiments and Results in the Electrical Distribution Networks Problem

The results obtained with the different linguistic modeling methods considered are shown in Table 5.

The significant differences existing between *WM* and *CH-WM-ALM* are because of the fact that the RB obtained with *WM* is insufficient since in this application is necessary to deep in certain zones. *CH-WM-ALM* allows us to use two different rules defined in the same input subspace but presenting a different consequent in the case in which it is necessary.

In the generation phase, the genetic methods stand out against the remaining ones. Especially, *LM* offers a good result since it generates a model accurate which is in both learning and prediction thus not presenting overlearning.

SF obtains very bad results because of it has a big dependency of the value assigned to the weights to accomplish the learning process. Let us suppose that an important rule begins with a low weight associated, its influence on the NN evolution will be nonvalued and it will be eliminated in the pruning phase. However, the authors propose to randomly initialize these weights, provoking with this action the convergence to a local optimum.

Table 5: Results obtained in the electrical application problem

	Generation			*J-ANFIS-tuning*			*CH-tuning*	
Method	#R	MSE_{tra}	MSE_{tst}	#R	MSE_{tra}	MSE_{tst}	MSE_{tra}	MSE_{tst}
WM	24	222623	240018	24	161889	**160370**	144510	173167
NIT	64	173230	190808	–	—	—	—	—
SF	45	1281547	1067993	45	247696	296587	233133	252569
J-ANFIS	49	256605	268451	–	—	—	244286	263940
T	47	185204	168060	47	184804	168318	169689	175985
LM	49	167014	167383	49	165650	296804	150224	290461
GP-SLAVE	61	346921	379076	39	184650	256420	176145	191721
CH-D-MOGUL	35	167621	207598	35	165633	208693	142503	181288
CH-WM-ALM	20	155866	178601	**17**	209259	256890	**136437**	210236

The adjustment of *CH-tuning* only causes overlearning precisely in the three methods that obtained the best value of prediction in the generation phase, *LM*, *T*, and *CH-WM-ALM*. This also occurs when *J-ANFIS-tuning* are applied. The adjustment of *J-ANFIS-tuning* works well with the *WM* method, which has wrong results in the generation phase, but generally it does not obtain significant improvement.

V.C Global Analysis

In the following, some conclusions about the methods' behavior and about the different techniques used are presented.

V.C.1 Learning Methods

We should underline the good behavior presented by the methods based on GAs on the generation phase.

The bad results of *J-ANFIS* are due to the lack of the LSE technique in the Mamdani approach. The power of this method is based on this technique, but it is only applicable when the consequents are not linguistic terms (TSK approach). This also affects to the tuning method.

As regards the number of rules in the KBs, the models designed by means of process with Genetic Simplification Phase (*CH-D-MOGUL* and *CH-WM-ALM*) should be highlighted as well, due to the fact that they always present less rules than the ones in the models generated from remaining methods.

The number of rules used by *GP-SLAVE* is only 2 and 9 for the rice and the electrical problems respectively if we represent them in DNF form. This aspect provides a good interpretation to the RB generated and, although its accuracy is worse than most methods, it is very interesting in specific problems. For example, one of the RBs derived by *GP-SLAVE* in the rice test problem is as simple as:

IF *Taste* **is** *Bad* **and** *Stickiness* **is** *Not sticky* **THEN** *Overall Evaluation* **is** *Low*
IF *Taste* **is** *Good* **THEN** *Overall Evaluation* **is** *High*

V.C.2 Tuning Methods

We should remark the good performance of the tuning processes (especially the genetic approach, *CH-tuning*), that broadly improve the accuracy of the preliminary RB definitions. However, some problems of high complexity appear: the tuning applied to initial KBs with good accuracy does not obtain beneficial results since it increases the overlearning and makes worse the generalization ability. This is due to that the tuning tends to overfit on the training data in these cases.

A clear inconvenient of *J-ANFIS-tuning* is that it is more restrictive than the GA-based one regarding the structure of the KBs to adapt. It does not allow initial KBs with several rules with the same antecedents and different consequents. In this case, we must consider only one of them in each input subspace, but this will provoke the loss of part of the input information.

V.C.3 Techniques Used

Next, we remark some comments on the techniques used by the different methods:

- The *ad hoc* approaches are generally based in data covering, i.e., in an attempt to cover properly the example set. This implies a big dependency on the data training that may be imprudent.
- The principal advantages of NNs is that they can learn by generalization and many NNs are resistive against noise or partial damage after learning. As disadvantages we found that the convergence of the learning process cannot be guaranteed but has to be gained by often-extensive experimentation. The NNs have a serious problem when the number of variables or the number of labels is increased, since the complexity grows geometrically. This is due to the way considered to represent the rules by means of connections between nodes. Another important weakness is that NNs are not flexible with the KB representation.
- Whereas GAs have a guaranteed local convergence and are easy to implement, there is no guarantee to find the optimal solution, and the high amount of individuals to be processed makes the algorithm relatively time consuming. Pittsburgh GA approaches, as NNs, also have problems when they work with a lot of variables since the search space will be huge. A good advantage is that GAs can represent different types of fuzzy rules as approximate Mamdani or DNF ones.

VI Concluding Remarks

In this Chapter we have accomplished a short revision of FRBSs and we have seen the different types that currently exist.

Thereinafter, we have focused on linguistic Mamdani FRBSs, which obtain better legibility in exchange for offering worse accuracy. We have seen *ad hoc* learning methods and two techniques, NNs and GAs, that can be applied to the learning of FRBSs. Several forms of facing the learning have been presented.

Finally, through the application of the reviewed methods, two real applications have been considered to analyze their behavior. In this study, we have obtained some interesting conclusions. Among other things, we have proven that the GAs work best and that the tuning phase improves significantly the accuracy. We have also been able to see the limitations associated with the linguistic representation when the problem is complex.

Appendix A: Neural Networks

A.1. Brief Description of Neural Networks

Artificial Neural Networks (ANNs) are computational models that emerged as mathematical formalization attempts of the structure of the brain [56, 57, 58, 59]. They are outlined as calculation models for those which exist algorithms that permit to develop cognition tasks such as learning, classification or optimization. Furthermore, they are characterized by a massively parallel operation that makes them very efficient. The ANN model can be understood as another way of representing knowledge. It differs of the high-level mappings (rules, semantic networks, frames, ...) in which works at a subsymbolic level.

An ANN is formed by a set of elemental processing units that are communicated mutually being shipped signals through weighted connections. The main characteristics of an NN model are:

- A set of processing units, called *neurons*, cell process elements, etc.
- Each unit, i, receives an input series (from the environment or from other units), that can take values in various domains. If the set is $\{0, 1\}$, the neuron is called binary. If values outlet in $\{-1, 1\}$ the name of the neuron is bipolar. But also there are very used the cases in which the set is a continuous interval as $[0, 1]$ or $[-1, 1]$.
- The neurons are connected among them through links, also called connections or *synapses*, that tend to be one-way, though there are models with symmetrical connections as well. Each connection communicates two units and have associated a weighting or weight, w_{ij}, that determines the efficiency with which the signals of the neuron i affects the j one.
- An aggregation rule of the inputs, that determines the excitement level of a unit i.
- An activation function, f, that indicates the state or activation level, a_i, according to the excitement level, that can be propagated as input to other neurons.
- In many cases, an external input or trend for each unit, θ_i, is employed as well. Its performance is equivalent to a threshold that the total signal that arrives to the neuron must exceed to enable it.

Processing units. Each unit accomplishes a relatively simple role: it takes the inputs of the neurons connected to it and the external one and obtains its new state or excitement level from them. From a new state, and applying the activation function, the output value of the neuron is obtained to be transmitted to the neurons connected to its output. This output value is the activation state.

We can build networks by connecting neurons. In them, we distinguish three types of neurons: input neurons, by which the network downloads from the environment; output neurons that provide the result of the processing to the environment and hidden neurons whose input and output signals remain within the system.

The network is inherently parallel in the sense of the fact that many units can accomplish their calculations at the same time. During the processing, the units can update their state of synchronous or asynchronous mode. In the synchronous mode, all the neurons are updated simultaneously. In the asynchronous one, each unit has a variable probability, time-sensitive, of being updated and usually only one unit is updated in a given moment. In some instances, the second mode is more advantageous.

Connections among the units and aggregation rule. In most cases, it is supposed that the contributions of the input units are additive. Thus, it is established that the excitement of the unit j is the weighted sum of the inputs, result to which the threshold is added:

$$u_j(t) = \sum_i w_{ij}(t)x_i(t) + \theta_j(t) \ .$$

When a connection has a positive weight w_{ij}, it is said that the connection is *exciting* and when it is negative, it is said it is *inhibiting*.

Activation rules and output rules. The effect of the excitement level on the activation state of the neuron is established through the activation function, f. This function takes the current state of the neuron and the excitement level as inputs and returns the new activation state:

$$a_j(t+1) = f(a_j(t), u_j(t)) \ .$$

Usually, the activation function is a non-diminishing function of the excitement level of the neuron. The more employed functions are the step ones: the sign function, the linear function or S elongated functions such as the logistics and the hyperbolic tangent.

In some instances, the output of a unit can be a stochastic function of the inputs. In this case, the new activation state is a dependent random variable of the input.

Connection topologies. A relevant aspect of an ANN is the mode in which the units are connected. According to the type of connection topology, the models of ANN are split into:

- *Feed-forward networks.* The neurons are arranged in sets called *layers*. The layers are ordered, beginning in the input layer (the one which contains the input units) and terminating in the output layer, so that the links only exist among the neurons of a layer and those of the immediately next one and only in that sense. There are no links among neurons of a same layer neither among neurons of non-consecutive layers. In this way, the signals flow from the input to the output.

- *Feed-back networks.* Connections toward back and intralayer are allowed, disappearing basically the layer idea as a functional neurons group.

This distinction is important because the analysis of the network operation (above all, its dynamic behavior) is very different according to if it spreads toward forward or with feedback.

Once the neurons which a given one is connected to are established, it can be represented the set of connections with a weight vector. The ordered arrangement of the arrays permits us to build a matrix that is designated *weight matrix* of the network. When there is no connection between two neurons, it is understood that the associated weight is zero.

Training of an ANN. An ANN must be trained so that the application of a set of inputs produces, well directly or through a learning process, the waited output. The learning involves modifying the free parameters of the network, i.e., the weights of the connections, so that the assignment target error to be made zero.

There are several methods to accomplish this. If there is enough knowledge, the weights can be beforehand calculated. But the more usual method of training the network is based on showing examples to it and modifying its weights in view of the error produced according to some learning rule.

The learning methods are classified in two groups:

- *Supervised.* Examples that include the desired input and output are shown to the network.
- *Unsupervised.* The network is trained with examples including only the inputs (the waited output is not indicated to it). The network must be capable of discovering statistically the main characteristics of the input samples and to employ them to classify the input patterns. It does not exist any set of categories in which the patterns should be classified *a priori*; the system must develop its own mapping of the input stimuli.

Learning rules. All the learning methods give the setting of the weights of the connections among units as a result according to a rule.

Nearly all these learning rules are variations of the *Hebb rule* [60]. The basic idea of this rule is that if two units i and j are activated simultaneously, their interconnections should be reinforced. If i is connected to the input of j, the simplest formulation of the Hebb rule prescribes the following modification:

$$\Delta w_{ij} = \gamma a_i a_j \ ,$$

with γ being a constant of proportion that represents the *learning rate*. Another common rule employs the discrepancy among the current and the desired output instead of the output of the unit j:

$$\Delta w_{ij} = \gamma a_i (d_j - a_j) \ ,$$

with d_j being the value expected for the activation level, supplied by a teacher. This rule is known as the *delta rule* [61].

Both the Hebb and the delta rules are adapted for the training of feed-forward propagation networks in which there are no hidden neurons. However, these kinds of networks have a very limited area of application since they are only capable of solving linearly separable problems. Hence, simple problems as the representation of or-exclusive (XOR) function can not be solved employing them. This limitation was demonstrated by Minsky and Papert [62], who also proved that it could be solved employing hidden neurons. The problem was to find appropriate training procedures. Rumelhart *et al.* [63] settled the problem with the presentation in 1986 of the *error-backpropagation algorithm* that has turned out to be fruitful and very used.

The error-backpropagation algorithm (EBP). The EBP is a generalization of the delta rule. The main idea is again to quantify the existing error through the difference among the system output and the real output. The difficulty lies in that there is now a great number of connections that increase the error, which do not only relate to the neurons of the output layer. Particularly, since there is not information on the role that the hidden units play, their contribution to the total error can not be directly calculated and it is not known how to correctly modify the weights of the connections associated to them.

The problem can be tackled if it is admitted that the error observed in the output layer should be primarily due to the action of the hidden units located in the immediately previous layer, that at the same time are seen as influenced by the elements of the previous layer, and so on. The adaptive process remains defined in two stages that are going to be repeated until the learning is considered to be accomplished. In the first the activity level of all the neurons of the network is evaluated maintaining fixed the values of the connections, which permits to determine the magnitude of the existing error. In a second stage this error is propagated back layer to layer, successfully modifying the weights that in the next stage will serve to calculate the new error. This process is the one which gives name to the learning algorithm.

Figure 11: Multilayer Neural Network

We consider a network with n input neurons, h hidden neurons and m output neurons (see Figure 11). The activation function is a differential function of the actual input to the neuron:

$$a_j^p = F_j(e_j^p) \ ,$$

where

$$e_j^p = \sum_i w_{ij} a_i^p + \theta_j \ ,$$

that is the actual input to the j-th unit for the p-th example.

After the first epoch in which the assignment error by the network is evaluated, each weight is adjusted according to:

$$\Delta_p w_{ij} = \gamma \delta_j^p a_i^p \ .$$

The parameter δ_j^p adopts different values for the case of output and internal units. It is started making the setting for the weights that bind the units of the hidden layer and the output ones. In this case:

$$\delta_j^p = (d_j^p - a_j^p) F_j'(e_j^p)$$

for any output unit j. If a hidden unit is considered, the values of δ_j^p are calculated according to:

$$\delta_j^p = F_j'(e_j^p) \sum_{h=1}^{N_o} \delta_h^p w_{hj} \ .$$

The latter two equations give us a recursive procedure to calculate the deltas for all the units in the network. This procedure constitutes the generalized delta rule or the EBP.

The expressions of the algorithm for the case where the activation function is the logistics is defined as:

$$F(e_j^p) = \frac{1}{1 + \exp(-e_j^p)} \ ,$$

and its derivative is:

$$F'(e_j^p) = a_j^p (1 - a_j^p) \ .$$

The deltas for the output units result:

$$\delta_j^p = (d_j^p - a_j^p) a_j^p (1 - a_j^p) \ ,$$

and for the units of the hidden layer:

$$\delta_j^p = a_j^p(1 - a_j^p) \sum_{h=1}^{N_o} \delta_h^p w_{hj} \ .$$

The gradient descent algorithm requires that the setting is accomplished through minimal steps, but for practical effects it is interesting to consider a large learning rate γ that permits a quick learning. The problem is that oscillations can appear. A way to avoid the oscillations for large values is to accomplish changes in the dependent weights of the previous changes appending a term moment:

$$\Delta w_{ij}(t+1) = \gamma \delta_i^p a_j^p + \alpha \Delta w_{ij}(t) \ ,$$

with α being a constant of the moment, measuring the importance of the previous setting in the current one.

A.2. Neuro-Fuzzy Systems

Fuzzy Logic and NNs are two disciplines that efficiently deal with two different areas of information processing. Fuzzy Logic deals with various aspects of uncertain knowledge representation and processing, and it allows approximate reasoning in the field of knowledge based systems. NNs are efficient computation structures capable of learning and adapting from examples.

Both techniques have their advantages and their weaknesses. In some sense, they become complementary since some features lacking in one approach are dominant in the other. Fuzzy Systems are not powerful in learning, adaptation, and parallel computation, where NNs offer a good performance in these aspects. NNs lacks in flexibility, human interaction or knowledge representation, where Fuzzy Logic is a powerful tool. In the last few years, a large research effort has been made to synthesize Fuzzy Logic and NNs to produce hybrid Fuzzy-Neural systems.

We prefer to call *Neuro-Fuzzy Systems* (NFS) to these approaches where NNs are used to provide inputs for a Fuzzy System, or to change the output of a Fuzzy System to remark that the parameters of a Fuzzy System are not changed by a learning process in these approaches. If the creation of an NN is the main target, it is possible to apply fuzzy techniques to speed up the learning process, or to fuzzify an NN by the extension principle to be able to process fuzzy inputs. These approaches could be called *Fuzzy Neural Networks* to stress that fuzzy techniques are used to create or enhance NNs. We are interested in the first approach.

An NFS has the following five characteristics as in [64] is described:

- An NFS is a Fuzzy System that is trained by a learning algorithm usually derived from NN theory. The learning procedure operates on local information, and causes only local modifications in the underlying Fuzzy System. The learning process is not knowledge-based, but data driven.

- It can be viewed as a special 3-layer feedforward NN. The units in this network use t-norms or t-conorms instead of the activation functions common in NNs. The first layer represents input variables, the middle or hidden layer represents fuzzy rules, and the third layer represents output variables. Fuzzy sets are encoded as connection weights. This view of a Fuzzy System illustrates the data flow within the system, and its parallel nature. However, this neural network view is not a prerequisite for applying a learning procedure, but merely a convenience.

- An NFS can always —i.e., before, during, and after the learning— be interpreted as a FRBS. It is both possible to create the system out of training data from scratch, and to initialize it by prior knowledge in the form of fuzzy rules.

- The learning procedure of an NFS takes the semantic properties of the underlying Fuzzy System into account. This results in constraints on the possible modifications applicable to the system parameters.
- An NFS approximates an n-dimensional unknown function that is partially given by the training data. An NFS should not be seen as a kind of fuzzy expert system, and it has nothing to do with Fuzzy Logic in the narrow sense.

NFSs can be considered as a technique to derive a Fuzzy System from data, or to enhance it by learning from examples. It is possible to use an NN to learn certain parameters of a Fuzzy System, like using a self-organizing feature map to find fuzzy rules [65] (cooperative models), or to view a Fuzzy System as a special NN, and directly apply a learning algorithm [66] (hybrid models).

For more information, refer to [41, 64, 67, 68].

Appendix B: Genetic Algorithms

B.1. Brief Description of Genetic Algorithms

GAs are general-purpose search algorithms that use principles inspired by natural population genetics to evolve solutions to problems [69]. The basic idea is to maintain a population of knowledge structures that evolves over time through a process of competition and controlled variation. Each structure in the population represents a candidate solution to the specific problem and has an associated *fitness* to determine which structures are used to form new ones in the process of competition. The new individuals are created using genetic operators such as crossover and mutation. GAs have had a great measure of success in search and optimization problems. The reason of great part of this success is their ability to exploit accumulative information about an initially unknown search space in order to bias subsequent search into useful subspaces, i.e., *their robustness*. This is their key feature, especially in large, complex and poorly understood search spaces, where the classical search tools (enumerative, heuristic, ...) are inappropriate, offering a valid approach to problems requiring efficient and effective search.

A 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. As mentioned in these algorithms we maintain a population of solutions for a given problem; this population undergoes evolution in a 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 one constitutes one generation in the execution of a genetic algorithm.

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

(1) evaluation of individual fitness,

(2) formation of a gene pool (intermediate population), and

(3) recombination and mutation.

Figure 12 shows the structure of a simple GA.

A fitness function must be devised for each problem to be solved. Given a particular chromosome, a solution, the fitness function returns a single numerical fitness that is supposed to be proportional to the utility or adaptation of the solution represented by this chromosome.

```
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)
```

Figure 12: Structure of a GA

There are a number of ways to do selection. We might view the population as a mapping onto a roulette wheel, where each individual is represented by a space that proportionally corresponds to its fitness. By repeatedly spinning the roulette wheel, individuals are chosen using *stochastic sampling with replacement* to fill the intermediate population. The selection procedure proposed by Baker [49], called *stochastic universal sampling*, is one of the efficient proposals for avoiding the genetic drift [70]. The number of offspring of any structure is bound by the floor and ceiling of the expected number of offspring.

After selection has been carried out, the construction of the intermediate population is completed and recombination and mutation can occur.

The crossover operator combines the features of two parent structures to form two similar offspring. Classically, it is applied at a random position with a probability of performance, the crossover probability, P_c. The mutation operator arbitrarily alters one or more components of a selected structure so as to increase the structural variability of the population. Each position of each solution vector in the population undergoes a random change according to a probability defined by a mutation rate, the mutation probability, P_m.

Figures 13, 14, and 15 illustrate the basic operations: reproduction, crossover, and mutation.

Figure 13: Evaluation and contribution to the gene pool

It is generally accepted that a GA must take into account the five following components to solve a problem:

Figure 14: Recombination. One-point crossover

Figure 15: Mutation

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 individual,*
4. *genetic operators that alter the genetic composition of children during reproduction, and*
5. *values for the parameters that the GA uses (population size, probabilities of applying genetic operators, etc.).*

The basic principles of the GAs were first laid down rigorously by Holland [69] and are well described in many texts as [70, 71].

B.2. Genetic Fuzzy Systems

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 [72].

Three alternative approaches, in which GAs have been applied to learning processes, have been proposed, the Michigan [73], the Pittsburgh [74], and the Iterative Rule Learning (IRL) [46] approaches. In the first one, the chromosomes correspond to classifier rules that are evolved as a whole, whereas in the Pittsburgh approach, each chromosome encodes a complete set of classifiers. In the IRL approach each chromosome represents only one rule, but contrary to the first, only the best individual is considered as the solution, discarding the remaining chromosomes in the population.

Recently, numerous papers and applications combining fuzzy concepts and GAs have appeared, and there is an increasing concern about the integration of these two topics. In particular, a great number of publications explore the use of GAs for designing Fuzzy Systems. These approaches receive the general name of *Genetic Fuzzy Systems* (GFSs) [19, 34].

The automatic design of Fuzzy Systems can be considered in many cases as an optimization or search process on the space of potential solutions. GAs are the best known and most widely used global search technique with an ability to explore and exploit a given operating space using available performance measures. *A priori* knowledge in the form of linguistic variables, fuzzy membership function parameters, fuzzy rules, number of rules, etc., may be easily incorporated into the genetic design process. The generic code structure and independent performance features of GAs make them suitable candidates for incorporating *a priori* knowledge. Over the last few years, these

advantages have extended the use of GAs in the development of a wide range of approaches for designing Fuzzy Systems.

As in the general case of Fuzzy Systems, the main application area of GFSs is system modeling and control. Regardless the kind of optimization problem, i.e., given a system to be modeled or controlled, the involved learning or tuning process will be based on evolution. Three points are the keys to a genetic process: the population of potential solutions, the pair evolution operators/code, and the performance index.

A GFS combines the main aspects of the system to be obtained, a Fuzzy System, and the design technique used to obtain it, a GA, with the aim of improving as far as possible the accuracy of the final Fuzzy System generated. One of the most interesting features of a Fuzzy System is the interpolated reasoning that develops. This characteristic plays a key role in the high performance of Fuzzy Systems and is a consequence of the *cooperation among the fuzzy rules composing the KB*. As is known, the output obtained from a Fuzzy System 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 to the system to some degree.

On the other hand, the main feature of a GA is the *competition among members of the population representing possible solutions to the problem* being solved. In this case, this characteristic is due to the mechanisms of natural selection on which the GA is based.

Therefore, since a GFS combines both aforementioned 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 a Fuzzy System, because the different members of the population compete with one another 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 way of working into effect. This is referred to as *Cooperation vs. Competition Problem* [75]. The difficulty of solving the introduced problem depends directly on the genetic learning approach followed by the GFS (Michigan, Pittsburgh, or IRL).

Different GFS proposals can be found in [34, 76, 77, 78].

References

[1] Zadeh, L. A. (1965). Fuzzy sets. *Information and Control* **8**, pp. 338–353.

[2] Bárdossy, A., and Duckstein, L. (1995). "Fuzzy rule-based modeling with application to geophysical, biological and engineering systems." CRC Press.

[3] Chi, Z., Yan, H., and Pham, T. (1996). "Fuzzy algorithms: with applications to image processing and pattern recognition." World Scientific.

[4] Hirota, K. (Ed.) (1993). "Industrial applications of fuzzy technology." Springer-Verlag.

[5] Pedrycz, W. (Ed.) (1996). "Fuzzy Modelling. Paradigms and Practice." Kluwer Academic Press.

[6] Klir, G. J., and Yuan, B. (1995). "Fuzzy sets and fuzzy logic." Prentice-Hall.

[7] Zimmermann, H. J. (1996). "Fuzzy sets theory and its applications." Kluwer Academic Press.

[8] Zadeh, L. A. (1973). Outline of a new approach to the analysis of complex systems and decision processes. *IEEE Transactions on Systems, Man, and Cybernetics* **3**, pp. 28–44.

[9] Wang, L. X. (1994). "Adaptive fuzzy systems and control." Prentice-Hall.

[10] Mamdani, E. H. (1974). Applications of fuzzy algorithm for control a simple dynamic plant. *Proc. of the IEEE 121*, pp. 1585–1588.

[11] Mamdani, E. H., and Assilian, S. (1975). An experiment in linguistic synthesis with a fuzzy logic controller. *Int. Journal of Man-Machine Studies* **7**, pp. 1–13.

[12] Sugeno, M., and Yasukawa, T. (1993). A fuzzy-logic-based approach to qualitative modeling. *IEEE Transactions on Fuzzy Systems* **1**, pp. 7–31.

[13] Lee, C. C. (1990). Fuzzy logic in control systems: Fuzzy logic controller – Parts I and II. *IEEE Transactions on Systems, Man, and Cybernetics* **20**, pp. 404–435.

[14] González, A., Pérez, R., and Verdegay, J. L. (1993). Learning the structure of a fuzzy rule: A genetic approach. *Proc. of the 1st European Congress on Fuzzy and Intelligent Technologies*, Aachen (Germany), pp. 814–819.

[15] González, A., and Pérez, R. (1998). Completeness and consistency conditions for learning fuzzy rules. *Fuzzy Sets and Systems* **96**, pp. 37–51.

[16] Magdalena, L., and Monasterio, F. (1997). A fuzzy logic controller with learning through the evolution of its knowledge base. *Int. Journal of Approximate Reasoning* **16**, pp. 335–358.

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

[18] Cordón, O., and Herrera, F. (1998). A proposal for improving the accuracy of linguistic modeling. *Dept. of Computer Science and Artificial Intelligence*, University of Granada, Spain. Technical Report #DECSAI-98113.

[19] Cordón, O., and Herrera, F. (1995). A general study on genetic fuzzy systems. *In* "Genetic algorithms in engineering and computer science" (J. Periaux, G. Winter, M. Galán, and P. Cuesta, Eds.), pp. 33–57. John Wiley & Sons.

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

[21] Koczy, L. (1996). Fuzzy if ... then rule models and their transformation into one another. *IEEE Transactions on Systems, Man, and Cybernetics* **26**, pp. 621–637.

[22] Cordón, O., Herrera, F. (to appear). Hibridizing genetic algorithms with sharing scheme and evolution strategies for designing approximate fuzzy rule-based systems. *Fuzzy Sets and Systems.*

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

[24] Sugeno, M., and Kang, G. T. (1988). Structure identification of fuzzy model. *Fuzzy Sets and Systems* **28**, pp. 15–33.

[25] Gupta, M. M., and Qi, J. (1991). Design of fuzzy logic controllers based on generalized T-operators. *Fuzzy Sets and Systems* **40**, pp. 473–489.

[26] Trillas, E., and Valverde, L. (1985). On implication and indistinguishability in the setting of fuzzy logic. *In* "Management Decision Support Systems Using Fuzzy Logic and Possibility Theory" (J. Kacpryzk and R. R. Yager, Eds.), pp. 198–212. Verlag TUV Rheinland.

[27] Cordón, O., Herrera, F., and Peregrín, A. (1997). Applicability of the fuzzy operators in the design of fuzzy logic controllers. *Fuzzy Sets and Systems* **86**, pp. 15–41.

[28] Driankov, D., Hellendoorn, H., and Reinfrank, M. (1993). "An introduction to fuzzy control." Springer-Verlag.

[29] Cao, Z., and Kandel, A. (1989). Applicability of some fuzzy implication operators. *Fuzzy Sets and Systems* **31**, pp. 151–186.

[30] Kiszka, J., Kochanska, M., and Sliwinska, D. (1985). The influence of some fuzzy implication operators on the accuracy of a fuzzy model — Parts I and II. *Fuzzy Sets and Systems* **15**, pp. 111–128 and 223–240.

[31] Zadeh, L. A. (1994). Fuzzy logic and soft computing: Issues, contentions and perspectives. *Proc. of the 3rd Int. Conf. on Fuzzy Logic, Neural Nets and Soft Computing*, Iizuka (Japan), pp. 1–2.

[32] Bonissone, P.P. (1997). Soft computing: The convergence of emerging reasoning technologies. *Soft Computing. A Fusion of Foundations, Methodologies, and Applications* **1**(1), pp. 6–18.

[33] Takagi, T. (1990). Fusion technology of fuzzy theory and neural networks — survey and future directions. *Int. Conf. on Fuzzy Logic and Neural Networks*, Iizuka (Japan), pp. 13–26.

[34] Cordón, O., Herrera, F., Hoffmann, F., Magdalena, L. (in preparation). Genetic fuzzy systems: Evolutionary tuning and learning of fuzzy knowledge bases. World Scientific.

[35] Wang, L. X., and Mendel, J. M. (1992). Generating fuzzy rules by learning from examples. *IEEE Transactions on Systems, Man, and Cybernetics* **22**(6), pp. 1414-1427.

[36] Nozaki, K., Ishibuchi, H., and Tanaka, H. (1997). A simple but powerful heuristic method for generating fuzzy rules from numerical data. *Fuzzy Sets and Systems* **86**, pp. 251–270.

[37] Shann, J. J., and Fu, H. C. (1995). A fuzzy neural network for rule acquiring on fuzzy control systems. *Fuzzy Sets and Systems* **71**, pp. 345–357.

[38] Jang, J.-S.R. (1993). ANFIS: Adaptive-Network-Based Fuzzy Inference System. *IEEE Transactions on Systems, Man, and Cybernetics* **23**(3), pp. 665–685.

[39] Thrift, P. (1991). Fuzzy logic synthesis with genetic algorithms. *Proc. of the 4th Int. Conf. on Genetic Algorithms*, pp. 509–513.

[40] Liska, J., and Melsheimer, S. S. (1994). Complete design of fuzzy logic systems using genetic algorithms. *Proc. of the 3rd IEEE Int. Conf. on Fuzzy Systems*, Vol. 2, pp. 1377–1382.

[41] Kosko, B. (1992). "Neural networks and fuzzy systems: A dynamic systems approach to machine intelligence." Prentice-Hall.

[42] Jang, J.-S.R. (1991). Fuzzy modeling using generalized neural networks and kalman filter algorithm. *Proc. of the 9th Nat. Conf. Artificial Intelligence*, pp. 762–767.

[43] Watrous, R. L. (1991). Learning algorithms for connectionist network: Applied gradient methods of nonlinear optimization. *Proc. of the IEEE Int. Conf. Neural Networks*, pp. 619–627.

[44] González, A., and Pérez, R. (1995). Structural learning of fuzzy rules from noisy examples. *Proc. of the 4th IEEE Int. Conf. on Fuzzy Systems/IFES'95*, Yokohama, Vol. 3, pp. 1323–1330.

[45] González, A., and Herrera, F. (1996). Multi-stage genetic fuzzy systems based on the iterative rule learning approach. *Mathware and Soft Computing* **4**, pp. 233–249.

[46] Venturini, G. (1993). SIA: A supervised inductive algorithm with genetic search for learning attribute based concepts. *Proc. of the European Conf. on Machine Learning*, Vienna, pp. 280–296.

[47] Cordón, O., del Jesus, M. J., Herrera, F., and Lozano, M. (1999). MOGUL: A methodology to obtain genetic fuzzy rule-based systems under the iterative rule learning. *Int. Journal of Intelligent Systems*, to appear.

[48] Herrera, F., Lozano, M., and Verdegay, J. L. (1998). A learning process for fuzzy control rules using genetic algorithms. *Fuzzy Sets and Systems* **100**, pp. 143–158.

[49] Baker, J. E. (1987). Reducing bias and inefficiency in the selection algorithm. *Proc. of the 2nd Int. Conf. on Genetic Algorithms*, Lawrence Erlbaum (Hillsdale, NJ), pp. 14–21.

[50] Herrera, F., Lozano, M., and Verdegay, J. L. (1995). Tuning fuzzy controllers by genetic algorithms. *Int. Journal of Approximate Reasoning* **12**, pp. 299–315.

[51] Ishibuchi, H., Nozaki, K., Yamamoto, N., and Tanaka, H. (1993). Selecting fuzzy if-then rules for classification problems using genetic algorithms. *IEEE Trans. on Fuzzy Systems* **3**(3), pp. 260–270.

[52] Herrera, F., Lozano, M., and Verdegay, J. L. (1998). Tackling real-coded genetic algorithms: Operators and tools for behavioural analysis. *Artificial Intelligence Review* **12**, pp. 265–319.

[53] Sánchez, L. (1997). Interval-valued GA-P algorithms. Technical Report, Dept. of Computer Science, University of Oviedo, Oviedo (Spain).

[54] Sánchez, L. (1997). Study of the Asturias rural and urban low voltage network. Technical Report, Hidroeléctrica del Cantábrico Research and Development Department (in spanish), Asturias (Spain).

[55] Cordón, O., Herrera, F., and Sánchez, L. (1999). Solving electrical distribution problems using hybrid evolutionary data analysis techniques. *Applied Intelligence* **10**, pp. 5–24.

[56] Haykin, S. (1994). "Neural Networks." Macmillan.

[57] Kröse, B. J. A., and van der Smagt, P. D. (1993). "An introduction to neural networks." University of Amsterdam.

[58] Müller, B., and Reinhardt, J. (1990). "Neural networks. An introduction." Springer-Verlag.

[59] Sim, K. P. (1989). "Artificial Neural Systems." Pergamon Press.

[60] Hebb, D. (1949). "Organization of Behaviour." John Wiley & Sons.

[61] Widrow, B., and Hoff, M. (1960). Adaptive switching circuits. *WESCON Convention Record* **4**, pp. 96–104.

[62] Minsky, M., and Papert, S. (1969). "Preceptrons." MIT Press.

[63] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by backpropagation errors. *Nature* **323**, pp. 533–536.

[64] Nauck, D., Klawonn, F., and Kruse, R. (1997). "Fundations of neuro-fuzzy systems." John Wiley & Sons.

[65] Pedrycz, W., Card, H.C. (1992). Linguistic interpretation of self-organizing maps. *Proc. of the 1st IEEE Int. Conf. on Fuzzy Systems*, San Diego (CA), pp. 371–378.

[66] Nauck, D., Klawonn, F., and Kruse, R. (1996). "Designing neuro-fuzzy systems through backpropagation". *In* "Fuzzy modelling: Paradigms and practice" (W. Pedrycz, Ed.). Kluwer Academic Press.

[67] Brown, M., and Harris, C. (1994). "Neurofuzzy adaptive modeling and control." Prentice-Hall.

[68] Harris, C. J., Moore, C. G., and Brown, M. (1993). "Intelligent control: Aspects of fuzzy logic and neural nets." *In* "Robotics and automated systems", Vol. 6. World Scientific.

[69] Holland, J. H. (1975). "Adaptation in natural and artificial systems." Ann arbor: The University of Michigan Press.

[70] Goldberg, D. E. (1989). "Genetic algorithms in search, optimization and machine learning." Addison Wesley.

[71] Michalewicz, Z. (1996) "Genetic algorithms + data structures = evolution programs." Springer-Verlag.

[72] Grefensette, J. J. (Ed.) (1994). "Genetic algorithms for machine learning." Kluwer Academic Press.

[73] Holland, J. H., and Reitman, S. (1978). Cognitive systems based on adaptive algorithms. *In* "Pattern-Directe Inference Systems" (D. A. Waterman and F. Hayes-Roth, Eds.). Academic Press.

[74] Smith, S. F. (1980). A learning system based on genetic adaptive algorithms. Ph. D. Thesis, University of Pittsburgh.

[75] Bonarini, A. (1996). Evolutionary learning of fuzzy rules: competition and cooperation. *In* "Fuzzy Modelling: Paradigms and Practice" (W. Pedrycz, Ed.), pp. 265–283. Kluwer Academic Press.

[76] Herrera, F., and Verdegay, J. L. (Eds.) (1997). "Genetic algorithms and soft computing." Physica-Verlag.

[77] Herrera, F. (Ed.) (1997). Special issue on genetic fuzzy systems for control and robotics. *Int. Journal of Approximate Reasoning* **17**(4).

[78] Herrera, F., Magdalena, L. (Eds.) (1998). Special issue on genetic fuzzy systems. *Int. Journal of Intelligent Systems* **13**(10).