Identification of Linguistic Fuzzy Models by Means of Genetic Algorithms*

Oscar Cordón and Francisco Herrera

Dept. of Computer Science and Artificial Intelligence. E.T.S.I. Informatica. University of Granada. 18071 - Granada. Spain.

1. Introduction

In this chapter, we deal with the identification of linguistic fuzzy models (or Mamdani fuzzy models) for multiple-input/single-output (MISO) systems. We consider a variant of the classical linguistic fuzzy model in which there does not exist a pre-determined relationship between the linguistic values of the input and output variables and the membership functions used to define the meaning (semantics) of these linguistic values. We call this type of a linguistic fuzzy model *an approximative linguistic fuzzy model* [4, 10].

Once an approximative linguistic fuzzy model has been chosen to represent a MISO system, the next step is to determine its structure and estimate its parameters. This is done in three steps. First we obtain a initial model structure and parameters. That is, we generate an initial set of fuzzy rules (or initial fuzzy rule base) and initial membership functions for the antecedent and consequent parts of the fuzzy rules. Second, the initial fuzzy rule base is simplified by removing redundant fuzzy rules and thus, the final structure of the approximative linguistic fuzzy model is determined. Third, we determine the final membership functions so that to maximize the accuracy of the approximative linguistic fuzzy model. Therefore, the proposed identification technique deals with both *structure* and *parameter identification*, and is based on learning from available input-output data.

The learning techniques used for the purpose of the identification of fuzzy models normally deal with the problem of designing and optimizing a fuzzy rule base and/or the parameters of membership functions using on- and/or off-line input-output data. These techniques include inductive learning [13, 38], descent methods [31], neural networks [26, 28], clustering techniques [39], genetic algorithms [6] (section 3.13), etc. On the other hand, a large class of methods known under the name of *evolutionary computation* (EC) use computational models of evolutionary processes as key elements in the design and implementation of identification and optimization algorithms. There is a variety of evolutionary computational models referred to as *evolutionary algorithms* (EAs). There are three well-defined EAs which serve as the basis for much of the activity in the field of EC: *Genetic Algorithms*

* This paper has been partially supported by CICYT PB96-0778

(GAs), *Evolution Strategies* (ESs), *Evolutionary Programming* (EP), and *Genetic Programming* (GP). In this chapter, we make use the first two types of EAs.

The most well known EAs are the GAs, i.e., search algorithms that use operations found in natural genetics to guide the search in complex search spaces. GAs are have been theoretically and empirically proven to have robust and computationally efficient search capabilities. They also have been demonstrated to be a powerful tool for automating the construction of the fuzzy rule bases, since learning and self-organization may be considered in a lot of cases as optimization and/or efficient search problems. The GAs based approaches used in the context of automating and optimizing the construction of fuzzy rule bases are known under the name of *genetic fuzzy systems* (GFSs) [4]. A short description of these approaches is included in Sect. 2. Using the more general term *evolutionary* instead of *genetic* they are also called *evolutionary fuzzy systems*.

The identification method presented in this chapter uses GAs, and can be described as three-stage inductive learning process. These three stages are the following:

1. GAs based generation of approximative fuzzy rules, based either on prior fuzzy partions of the domains of the input and output variables, or no prior fuzzy partitions at all. In the first case, ESs are used for a local tuning of the fuzzy rules. At this stage the initial fuzzy model structure and parameters are obtained.
2. GAs based simplification of the initial fuzzy rule base, thereby avoiding possible overlearning, and removing redundant fuzzy rules. At this stage the final model structure is obtained, i.e., all the fuzzy rules constituting the approximative linguistic fuzzy model.
3. GAs based tuning for adjusting the membership functions in the fuzzy rules using fitness criteria. At this last stage, the final model parameters are estimated.

In addition, the *genetic fuzzy identification method* (GFIM) presented in this chapter, permits the incorporation of prior, qualitative knowledge, and is able of blending this knowledge with the available input-output data.

The remaining part of this chapter is structured as follows. Section 2 serves as a brief introduction to EAs and GFSs. Section 3 considers the fuzzy model identification problem. Section 4 is devoted to the genetic fuzzy identification method. Section 5 contains an example illustrating the application of GFIM. Some practical aspects and concluding remarks are presented in Section 6.

2. Evolutionary Algorithms and Genetic Fuzzy Systems

In the following we briefly review the GAs and the ESs, both of which shall be used in this contribution.

2.1 Genetic Algorithms

GAs are both theoretically and empirically proven to provide the means for efficient search in complex spaces [17].

Any GA starts with a population of randomly generated solutions, chromosomes, and advances towards better solutions by applying genetic operators such as crossover (recombination) and point mutation. These algorithms maintain a population of solutions for a given problem. The population undergoes "evolution" in a form that resembles natural selection. In each generation, relatively good solutions reproduce to give "birth" to offsprings that replace relatively bad solutions which in turn eventually "die". Fitness criteria play the role of the environment to distinguish between good and bad solutions. The process of going from the current population to the next population constitutes one generation in the execution of a GA.

Although there are many possible variants of the basic GA, its fundamental underlying mechanism operates on a population of chromosomes (individuals) representing possible solutions to a problem, and consists of three basic operations:

1. Evaluation of chromosome's fitness.
2. Formation of a chromosome pool (intermediate population).
3. Recombination and mutation.

The structure of a GA is the following:

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)

A fitness function must be constructed for each particular problem to be solved. Given a particular chromosome, the fitness function returns a single numerical fitness score which is proportional to the utility, or the adaptation ability, of this same chromosome.

There are a number of ways to perform selection. One might view the population as a mapping onto a roulette wheel, where each chromosome is

represented by a space that is proportional to its fitness. By repeatedly spinning the roulette wheel, chromosomes are chosen using *stochastic sampling with replacement* to form the intermediate population. The selection procedure proposed in [3], and called *stochastic universal sampling* is one of the most efficient. Here the number of offsprings of any population is bound by the floor and ceiling of the expected number of offsprings. After selection has been carried out, the construction of the intermediate population is complete and recombination and mutation can occur.

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

The next figures illustrate the basic operations: reproduction, crossover, and mutation.

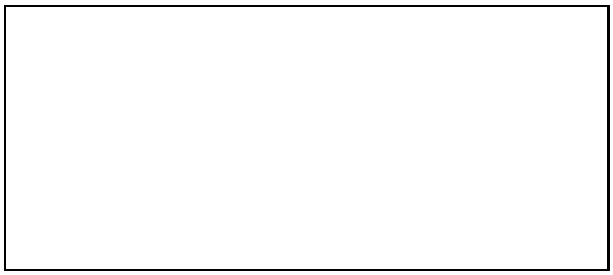

Fig. 2.1. Evaluation and contribution to the chromosome pool.

Fig. 2.2. Recombination (one-point crossover).

Fig. 2.3. Mutation.

The basic principles of the GAs were first laid down rigorously by Holland [25], and are well described in many texts such as [17, 30].

Binary coded strings as solutions for the representation problem have been extensively used. But GAs are not solely dependent on the use of bit strings. Nonbinary representations which are more suitable for a variety of application problems have emerged. One of the most important nonbinary representations is the real numbers representation which seems particularly natural when dealing with optimization problems with variables in continuous search spaces. In this context, a chromosome is a vector of floating point numbers whose size is kept the same as the length of the vector. GAs based on real numbers representation are called *real-coded GAs* (RCGAs). RCGAs have been mainly used for numerical optimization in continuous domains. Using real coding, the representation of the chromosomes is very close to the natural formulation of many problems, e.g., there are no differences between the *genotype* (coding) and the *phenotype* (search space). The use of real coding also makes easier the design of other operators incorporating problem specific knowledge. RCGAs provide greater precision especially in the case of large domains where binary coding would require prohibitively long representation [24, 30].

It is generally accepted that a GA must take into account the five following components in solving any given problem:

1. A genetic representation of the problem solutions.
2. A way to create an initial population of solutions.
3. An evaluation function which computes the fitness of each chromosome.
4. Genetic operators that alter the genetic composition of offsprings during reproduction.
5. Values for the parameters that the GA uses (population size, probabilities for applying genetic operators, etc.).

Numerous GA applications have been presented over the last years. Some of these can be classified as numerical function optimization, combinatorial optimization, image processing, fuzzy control and classification, engineering processes, biology, artificial life, machine learning, etc. There is an exceptionally large number of applications of GAs for the design of learning systems [18] and learning fuzzy systems [4, 20]. The interested reader can find free GAs software in [16].

2.2 Evolution Strategies

ESs were developed with a strong focus on building systems capable of solving difficult real-valued parameter optimization problems. The natural representation is a vector or real-valued chromosomes which are manipulated primarily by mutation operators designed to perturb their real-valued parameters in a purposeful way.

ESs were initially developed by Rechenberg and Schwefel in 1964 as experimental optimization techniques. The first ES algorithm, the so-called *(1+1)-ES*, was based on working with only two individuals per generation, one parent and one descendent (ofspring). Other more complex versions, based on considering higher number of parents ($\mu > 1$) and descendents ($\lambda > 1$), have appeared in last few years years. These constitute the so called $(\mu + \lambda)$-*ES* and (μ, λ)-*ES* ESs algorithms. Also several new generalized ESs have been successfully developed [2, 33].

Without lack of generality, we will use in this chapter *(1+1)-ES*, the most simple ES algorithm. In the following we briefly describe this particular ES algorithm [2, 33]

(1+1)-ES is based on representing the possible optimization problem solution as a real coded string. This parent string is evolved by applying a mutation operator over each one of its components. The mutation strength is determined by a parameter σ, the standard deviation of a normally distributed random variable. This parameter is associated with the parent and is evolved in each step of the optimization process. If the evolution has been successfull, the offspring obtained by mutation is better adapted than its parent. Then the descendent substitutes the parent in the next generation. The individual adaptation is measured by using a fitness function. The process is iterated until a finishing condition is satisfied.

The main component of the ES algorithm is the mutation operator, **mut**. It is composed of two components: $\mathbf{mu}_\sigma$, which updates the value of the parameter σ, and $\mathbf{mu}_x$, which evolves the real coded string. The first component is based on Rechenberg's 1/5-success rule, which evolves the standard deviation according to the current value of the relative frequency p of successful mutations in the following way

$$\sigma' = \mathbf{mu}_\sigma(\sigma) = \begin{cases} \dfrac{\sigma}{\sqrt[n]{c}}, & \text{if } p > \frac{1}{5} \\ \sigma \cdot \sqrt[n]{c}, & \text{if } p < \frac{1}{5} \\ \sigma, & \text{if } p = \frac{1}{5}. \end{cases} \quad . \tag{2.1}$$

The second component mutates each element of the real coded string by adding normally distributed variations with standard deviation σ' to it

$$x' = \mathbf{mu}_x(x) = (x_1 + z_1, \ldots, x_n + z_n) \tag{2.2}$$

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

The final algorithm structure is as follows

Procedure Evolution Strategy (1+1)
begin (1)
  $t = 0$;
  initialize $P(t) \leftarrow (x, \sigma)$;
  evaluate $f(x)$;

While (**Not** *termination-condition*) **do**
begin (2)
$t = t + 1$;
$(x', \sigma') \leftarrow \mathbf{mut}(\mathbf{x}, \sigma)$;
evaluate $f(x')$;
If Better $(f(x'), f(x))$
then $P(t+1) \leftarrow (x', \sigma')$
else $P(t+1) \leftarrow P(t)$.
end (2)
end (1)

ES software is provided in [33].

2.3 Genetic Fuzzy Systems

In many cases, the identification of a fuzzy model may be considered as an optimization or a search process. GAs have the ability to explore and exploit a given complex search space using an available performance measure and are known to be capable of finding near optimal solutions in such a search space. The prior knowledge one may wish to use, in addition to input-output data, may be in the form of known linguistic variables, fuzzy membership function parameters, fuzzy rules, number of rules, etc. The generic code structure and independent performance features of GA make them suitable candidates for incorporating this type of prior knowledge.

These properties of GAs make them suitable candidates for the design and optimization of fuzzy rule bases. In particular, the design, learning, and tuning of fuzzy rule have produced quite promising results. Figure 2.4 illustrates this idea.

GAs are applied to modify/learn the model parameters, i.e., the shapes of the membership functions stored in the fuzzy data base, and/or the model structure, i.e., the fuzzy rules composing the fuzzy rule base. It is possible to distinguish three different groups of *genetic fuzzy model design approaches* according to the type of fuzzy identification performed in the learning process. They are briefly described in the following subsections. For a detailed description see [4] and for an extensive bibliography see [6] (section 3.13). Different approaches may be found in [20].

2.4 Genetic Estimation of the Fuzzy Model Parameters (DB)

A fuzzy model has a number of parameters, such as the shapes of the membership functions, the scaling factors, the number of the linguistic values in the term sets associated with the linguistic variables from the fuzzy rules. All these fuzzy model parameters consitute the fuzzy data base (DB) of the fuzzy model and the fuzzy model quality is highly dependent on all of them

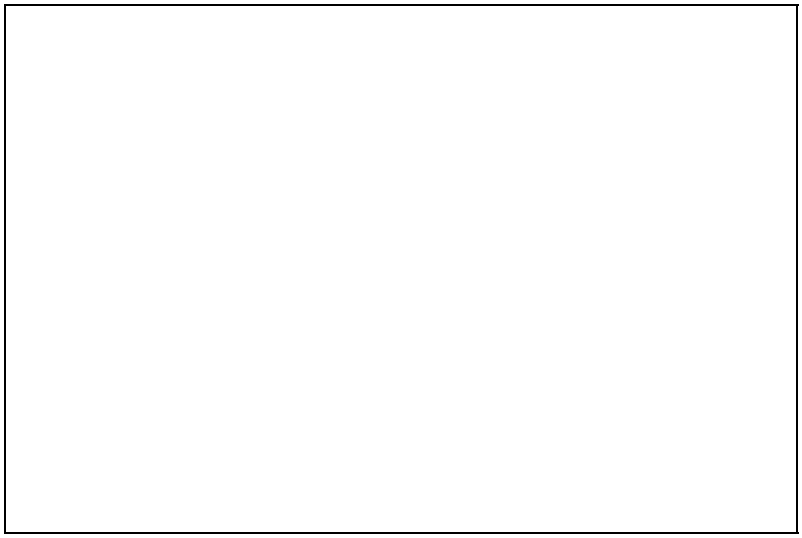

Fig. 2.4. Genetic Fuzzy Systems.

[14, 19, 27, 37]. Therefore, the proper definition of the membership functions is an important task in fuzzy model identification. The parameter estimation method using GAs tunes the membership functions by adjusting their parameters according to a given fitness function.

Several methods have been proposed in order to construct the DB using GAs. All of them are based on the existence of a given set of fuzzy rules (or rule base (RB) defining the fuzzy model structure) and an initial definition of the model parameters. Each chromosome involved in the evolution process represents a different DB definition, i.e., each chromosome will contain a coding of the membership functions. A chromosome's degree of adaptation is measured using a fitness function. This fitness function is based on the quality of the fuzzy model, represented by the given RB, and the model parameters encoded in the chromosome.

2.5 Genetic Derivation of the Fuzzy Model Structure (RB)

All the methods belonging to this family assume that the model parameters are known in advance, i.e., they suppose the existence of a DB. Different GAs -based methods for the derivation of the fuzzy model structure exist, depending on the representation chosen for RB: a set of fuzzy rules, a decision table, or a relational matrix. Most of these methods consider an RB represented in the form of a *decision table* (also called *look-up table*). As it is well known, a RB consisting of fuzzy rules with n input variables and a single output variable may be represented by using an n-dimensional decision table, where each dimension corresponds to one input variable. Table 2.1 shows an example of a

decision table for the control of an inverted pendulum. Every dimension has associated an array containing the linguistic values of the particular input variable. A cell in the decision table contains the linguistic value which the output variable takes for the combination of the linguistic values of the input variables corresponding to this cell. Therefore, each cell represents a fuzzy rule that may belong to the final model structure.

Table 2.1. Decision table for the control of inverted pendulum.

Angle	NL	NM	NS	ZR	PS	PM	PL
Change of Angle	NL						
	NM						
	NS			NS		ZR	
	ZR		NM		ZR		PM
	PS			ZR		PS	
	PM						
	PL						

The above structure is encoded in the individuals (chromosomes) forming the GA population. If there are empty cells in the decision table, then it is not possible to derive a fuzzy model structure with an optimal number of rules because for this purpose, all the possible fuzzy rules have to be considered.

2.6 Genetic Learning of the Fuzzy Model Structure and Parameters (RB and DB)

There is a multiplicity of approaches in genetic learning all aimed at the identification of the fuzzy model structure and parameters.

Amongst these some use variable chromosomal length, others use fixed chromosomal lenght encoding a fixed number of fuzzy rules together with the membership functions defining the linguistic values of the input and output variables, others use chromosomes each encoding a single fuzzy rule and its corresponding membership functionparameters etc.

Many of these approaches define the fitness function simply as an error measure, whereas others include a variety of objectives to be optimized in order to obtain more robust fuzzy models.

3. The Fuzzy Model Identification Problem

As we have mentioned earlier, in this chapter we focus on the linguistic type fuzzy model for MISO systems, where the structure of the fuzzy model consists of a collection of Mamdani-type of fuzzy rules (with the logical connective ALSO between the fuzzy rules). Thus each fuzzy rule is of the form

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

where $x_1, \ldots, x_n$ are input variables , y is the output variable and $A_{i1}, \ldots, A_{in}$, B are the linguistic values of the input and output variables in the i-th fuzzy rule The input and output variables take their values in the universes of discourse $U_1, \ldots, U_n$, and V respectively. The meaning (semantics) of the linguistic values is characterized by the membership functions $\mu_{A_{ij}}(x_j)$ and $\mu_{B_i}(y)$ defined on the universes of discourse U and V respectively. In this chapter we consider triangular membership functions. A computationally efficient way to characterize this type of membership functions is by using a parametric representation achieved by means of the 3-tuple (a_{ij}, b_{ij}, c_{ij}), (a_i, b_i, c_i), $j = 1, \ldots, n$.

The number of linguistic values for each input and output variable, the scaling factors, and the shapes of the membership functions constitute the fuzzy model parameters or DB. In [1] the following can be found about the linguistic fuzzy model:

"This representation is suitable for incorporating a priori knowledge by formulating the typical input-output situations in terms of rules. Since there is no structure assumed, virtually any system can be represented by the linguistic model. For this flexibility one has to pay by exponentially increasing model complexity, i.e., many rules may be needed to approximate a system to a given degree of accuracy, especially with many input variables. Also, the identification of the linguistic model from numerical data is not straightforward because one has to estimate both the membership functions and the relation between them (the rules). It is not trivial to estimate the membership functions from the data, since without any prior information one does not know where the 'important points' lie. Once the membership functions are found, rules can be identified quite easily."

The above difficulties can be said to motivate our particular identification strategy: once initial fuzzy model parameters have been derived and then the corresponding initial fuzzy model structure has been identified, the quality of the so obtained initial fuzzy model may be improved by deriving final fuzzy model parameters using the already identified initial fuzzy model structure.

In applying this identification strategy we consider an input-output data set without noise. This data set E_p is composed of p numerical input-output tuples $e_\ell \in E_p$, called *examples*, each example having the form

$$e_\ell = (ex_1^\ell, \ldots, ex_n^\ell, ey^\ell) \quad , \quad \ell = 1, \ldots, p. \tag{3.1}$$

In a conventional linguistic fuzzy model the set of linguistic values taken by the input and output variables is defined in advance. Furthermore, the meaning (semantics) of each linguistic value A_{ij} is determined by the membership function $\mu_{A_{ij}}(x_j)$ and one and the same linguistic value may appear in a number of fuzzy rules. However, in every fuzzy rule in which this linguistic value appears it has the same semantics, i.e., the same membership function. We call this type of linguistic fuzzy model a *descriptive* linguistic fuzzy model since a given membership function describes the semantics of an

a priori defined linguistic value and furthermore, one and the same linguistic value has the same semantics in all fuzzy rules in the RB in which it is encountered.

In this chapter we consider a linguistic fuzzy model in which the input and output variables variables do not take a priori defined linguistic values. This type of a linguistic fuzzy model is called an *approximative* linguistic fuzzy model [4, 10]. When considering this type of fuzy model we say that the fuzzy rules have *free semantics*. The difference between the descriptive and the approximative linguistic fuzzy models is illustrated is in Figure 3.1.

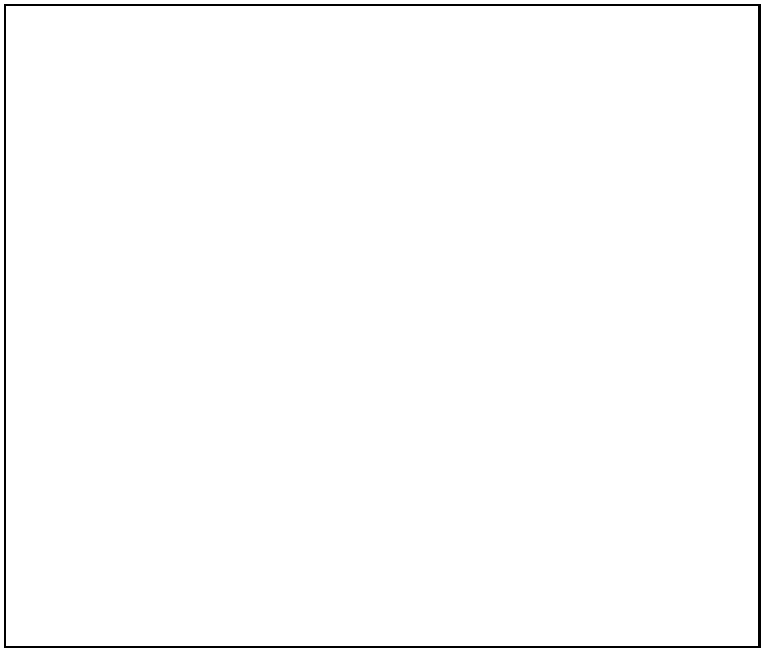

Fig. 3.1. Descriptive versus approximative linguistic fuzzy models.

For the purpose of identification we consider fuzzy rules with free semantics, i.e., no a priori defined linguistic values are associated with the input and output variables from the fuzzy rules. However, we further consider two types of free semantics:

1. *Unconstrained free semantics.* The identification method proceeds by learning the fuzzy rules and the initial shapes of the membership functions associated with these rules. This is done without any prior fuzzy partitioning being available. That is, no restrictions are placed on the membership functions locations and shapes.
2. *Constrained free semantics.* The identification method uses an initial fuzzy partitioning of the domains of the input and output variables par-

tition and the initial membership function parameters are locally (on a rule by rule basis) adjusted during the identification process.

In the case of constrained free semantics, each universe of discourse, U, is partitioned into a finite number of overlapping regions each region labeled by a linguistic value. For instance, if X is a variable taking its values in U and denoting temperature, then one may define A_1 as "*low temperature*," $A_i (1 < i < r)$ as "*medium temperature*," and A_r as "*high temperature*," etc.

These referential linguistic values are characterized by their membership functions $\mu_{A_i}(u) : U \rightarrow [0,1], i = 1, \ldots, r$. To ensure good performance of the fuzzy model it is essential that all the referential membership functions are normal and convex ones, and should satisfy the following completeness condition

$$\forall u \in U \ \exists j, \ 1 \leq j \leq r, \text{ such that } \mu_{A_j}(u) \geq \delta$$

where δ is a fixed threshold, called the *completeness degree* of a universe of discourse. Figure 3.2 shows an example of a fuzzy partitioning with $\delta = 0.5$.

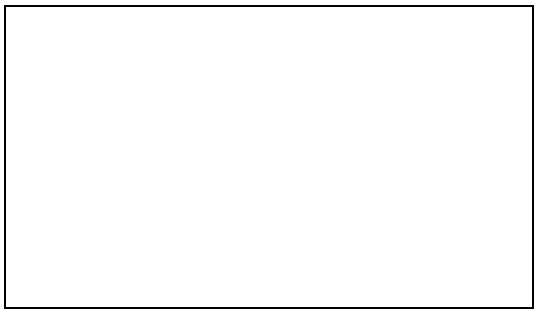

Fig. 3.2. Graphical representation of a fuzzy partitioning.

Based on this type of fuzzy partitioning , an interval of performance is associated with each one of the three parameters (a_t, b_t, c_t) defining the membership functions $\mu_{A_t}(\cdot)$

$$[a_t^\ell, a_t^r] = [a_t - \frac{b_t - a_t}{2}, a_t + \frac{b_t - a_t}{2}] \tag{3.2}$$

$$[b_t^\ell, b_t^r] = [b_t - \frac{b_t - a_t}{2}, b_t + \frac{c_t - b_t}{2}] \tag{3.3}$$

$$[c_t^\ell, c_t^r] = [c_t - \frac{c_t - b_t}{2}, c_t + \frac{c_t - b_t}{2}] \tag{3.4}$$

These intervals of performance are then used for locally adjusting the membership functions parameters during identification. Figure 3.3 shows the intervals of performance associated with each one of the parameters.

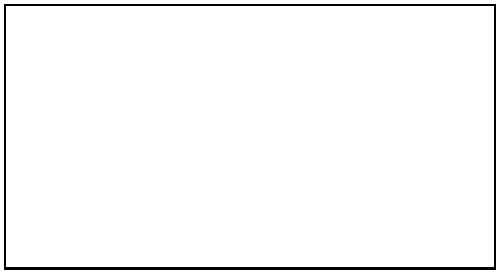

Fig. 3.3. Membership function and intervals of performance.

Therefore, the fuzzy rules identified will have their semantics within the performance intervals established by the initial fuzzy partitioning. As already mentioned, the identification procedure is concerned with

- the *fuzzy model structure identification* which allows us to obtain the RB, and
- the *fuzzy model parameters estimation* which allows us to obtain the DB.

The phases of genetic simplification and tuning involved in the identification of the approximative linguistic fuzzy model use the following fuzzy logic operations and an inference procedure [37]. Consider an input vector $\mathbf{x} = (x_1, \ldots, x_n)$ and a fuzzy RB constituted by m linguistic fuzzy rules R_i, $i = 1, \ldots, m$,

If X_{11} is A_{11} **and** ... **and** X_{1n} is A_{1n} **then** Y is B_1
...
If X_{m1} is A_{m1} **and** ... **and** X_{mn} is A_{mn} **then** Y is B_m

1 The logical *and* connective is used to connect the antecedents in each individual rule. This connective is interpreted by the *min* operator

$$\mu_{A_i}(\mathbf{x}) = min(\mu_{A_{i1}}(x_1), \mu_{A_{i2}}(x_2), \ldots, \mu_{A_{in}}(x_n)). \tag{3.5}$$

2 The *if-then* fuzzy implication is defined as a *conjunction* of the membership functions from the antecedent and consequent part of an if-then fuzzy rule. It is interpreted by the *min* operator

$$\mu_{B_i'}(\mathbf{x}, y) = min(\mu_{A_i}(\mathbf{x}), \mu_{B_i}(y). \tag{3.6}$$

3 The logical connective *also* aggregating the fuzzy rules is given as the *weighted sum* of the defuzzified values of the output $\mu_{B_i'}(y)$ of each individual fuzzy rule. The final defuzzified output, y, generated by the fuzzy model, is computed as [11]

$$y = \frac{\sum_{i=1}^{m} \mu_{A_i}(\mathbf{x}) \cdot CG(\mu_{B_i'}(y))}{\sum_{i=1}^{m} \mu_{A_i}(\mathbf{x})} \tag{3.7}$$

where $CG(\mu_{B_i'}(y))$ is the center of gravity of the fuzzy set B_i'.

4. The Genetic Fuzzy Identification Method

The GFIM is based on work presented in [5, 21, 22, 23]. It is composed by the following three parts:

I A genetic fuzzy rules construction method based on EAs, and a covering method for the input-output example set. This part results in an initial fuzzy model structure and parameters, i.e., a set of approximative fuzzy rules with their associated fuzzy sets covering the training input-output data set in an adequate manner.

II A genetic simplification of fuzzy rules, based on a binary coded GA and a fuzzy model performance measure. This part has the purpose of avoiding overlearning and removing redundant fuzzy rules.

III A genetic tuning of fuzzy model parameters, based on a RCGA and a fuzzy model performance measure. This part results in a final fuzzy model by tuning the membership functions for each fuzzy rule, i.e., it will estimate the final fuzzy model parameters taking into account the fuzzy model structure identified in the previous part.

Therefore, the first two parts form the genetic structure identification part of GIFM and provide the fuzzy model structure or RB together with initial fuzzy model parameters. The third part of GFIM adjusts the DB parameters giving us the final fuzzy model parameters.

The GIFM may be used in a number of ways according to the information available about the system under identification:

1. *Available example set:* The GFIM is applied for identifying a fuzzy model with unconstrained free semantics. No restrictions are imposed on the membership functions of inputs and outputs.
2. *Available example set and a fuzzy partition:* When initial fuzzy partitions (initial fuzzy model parameters) are available for all inputs and outputs, the GFIM is applied for constructing a RB using the initial fuzzy partitions and then locally tuning the membership functions associated with the identified fuzzy rules by means of an ES. Hence, the initial fuzzy partitions are used to guide the genetic search, restricting the shapes and the domains in which each one of the membership functions involved can be locally adjusted.
3. *Available partial RB and an example set:* In this case a fuzzy partitioning may or may not be available. Furthermore, an incomplete linguistic fuzzy model has been derived by a human expert. In this case the GIFM augments the partial RB with fuzzy rules learned from the example set, and then genetic simplification is applied to obtain the final fuzzy model structure. Finally, genetic tuning is applied for adjusting the membership functions of the final RB, thus obtaining the final fuzzy model parameters.
4. *Available RB:* In this case a complete linguistic fuzzy model is made available i.e., the fuzzy model structure and parameters are derived from a

human expert. Then, genetic tuning process, is used to obtain a more accurate fuzzy model by adjusting the already available fuzzy model parameters.

In the next sections we describe the three parts of the GFIM. First, we will present some requirements with respect to the RB.

4.1 Requirements

For identifying a set of fuzzy rules R_i, describing the structure of a linguistic fuzzy model, one has to "cover" all possible input-output pairs, $e_\ell \in E_p$, so that the so called *completeness property* [14, 27] is achieved. The formulation of this requirement requires a constant $\tau \in [0, 1]$, the nonempty union of the membership functions $\mu_{A_i}(\cdot)$, $\mu_{B_i}(\cdot)$, and is formulated as

$$\begin{aligned} &C_R(e_\ell) = \bigcup_{i=1}^{T} R_i(e_\ell) \geq \tau \quad \ell = 1, \ldots, p, \\ &R_i(e_\ell) = *(\mu_{A_i}(ex^\ell), \mu_{B_i}(ey^\ell)), \\ &\mu_{A_i}(ex^\ell) = *(\mu_{A_{i1}}(ex_1^\ell), \ldots, \mu_{A_{in}}(ex_n^\ell)), \end{aligned} \tag{4.1}$$

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

Given a set of fuzzy rules R_i, the *covering value* of an example e_ℓ is defined as

$$CV_R(e_\ell) = \sum_{i=1}^{T} R_i(e_\ell), \tag{4.2}$$

and we require that

$$CV_R(e_\ell) \geq \epsilon \quad \ell = 1, \ldots, p. \tag{4.3}$$

The set of fuzzy rules must satisfy both of the conditions presented above, i.e., it has to have the completeness property and an adequate covering value.

4.2 Genetic Construction of Fuzzy Rules

The different parts of the genetic fuzzy rules construction method have been presented in [5, 22, 23]. In this chapter we describe their integration.

The genetic fuzzy rules construction consists of a *construction method* together with a *covering method*, both working on a given set of examples.

- The construction method is realized by means of a GA encoding of a single fuzzy rule in each chromosome. The GA finds the best fuzzy rule in every run over the set of examples according to GA fitness function. When

constrained free semantics is considered, an ES is used for locally tuning the best fuzzy rules obtained during the iterations involved in the genetic search.
- The covering method is realized as an iterative process. It allows the construction of a set of fuzzy rules such that they cover the set of examples. In each iteration, the construction method chooses the best chromosome (fuzzy rule), considers the relative covering value this fuzzy rule has with respect to the example set, and removes the examples with a covering value greater than ϵ.

The above methods were separately presented in [22] and [5]. Here we introduce a new fitness criterion for the case of free semantics. This is the so called *niche interaction rate* which defines the degree of overlapping between a newly constructed fuzzy rule and the previously constructed fuzzy rules. This fitness criterion was introduced in the case of constrained semantics in [5]. We also introduce some modifications to the fitness criteria presented in [22]. The first subsection presents these fitness criteria, the next two sections describe identification with constrained and unconstrained free semantics and covering methods, and the sections after consider the covering method, genetic simplification, and genetic tuning.

4.2.1 Fitness Criteria. The fitness functions employed in GFIM are designed according to different fitness criteria.

High frequency value. The frequency of a fuzzy model rule, R_i, on the set of examples, E_p, is defined as

$$\Psi_{E_p}(R_i) = \frac{1}{p}\sum_{\ell=1}^{p} R_i(e_\ell) \tag{4.4}$$

where $R_i(e_\ell)$ is the *compatibility degree* between R_i and e_ℓ.

High average covering degree on positive examples. The set of positive examples for R_i with compatibility degree greater than or equal to ω is defined as

$$E_\omega^+(R_i) = \{e_\ell \in E_p / R_i(e_\ell) \geq \omega\} \tag{4.5}$$

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

$$G_\omega(R_i) = \sum_{e_\ell \in E_\omega^+(R_i)} R_i(e_\ell)/n_\omega^+(R_i). \tag{4.6}$$

Small negative example set. The set of the negative examples for R_i is defined as

$$E^-(R_i) = \{e_\ell \in E_p / R_i(e_\ell) = 0 \quad \text{and} \quad A_i(ex^\ell) > 0\}. \tag{4.7}$$

An example is considered negative for a fuzzy rule when it better matches some other fuzzy rule with the same antecedent (if-part), but a different consequent (then-part). The negative examples are always considered on the complete training set of examples.

With $n^-_{R_i} = |E^-(R_i)|$ being the number of negative examples, the *penalty function on the negative examples set* is

$$g_n(R_i{}^-) = \begin{cases} 1 & \text{if } n^-_{R_i} \leq k \cdot n^+_\omega(R_i) \\ \dfrac{1}{n^-_{R_i} - k n^+_\omega(R_i) + \exp(1)} & \text{otherwise} \end{cases} \tag{4.8}$$

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

Small membership function width. The variable width (RW) of a fuzzy rule R_i is defined as

$$RW_i = \frac{1}{h} \sum_{j=1}^{h} \frac{WVR_{ij}}{DW_j} \tag{4.9}$$

where

$$WVR_{ij} = c^3_{ij} - c^1_{ij} \tag{4.10}$$

and $(c^1_{ij}, c^2_{ij}, c^3_{ij})$, $j = 1, \ldots, h = n + 1$, are the parameters associated with membership function. DW_j is the *domain interval width* per input/output variable.

We define the *membership width rate, MWR*, of the rule R_i as a function of RW_i

$$MWR(R_i) = g(RW_i) \tag{4.11}$$

where the function g represents requirements with respect to the size of the width. We require a small width by considering the function

$$g(x) = \frac{e^{1-x} - 1}{e - 1}. \tag{4.12}$$

Highly symmetrical membership functions.. The rate of symmetry is defined in order to achieve the symmetry of a membership function and to prevent the bad covering of extreme points. It is defined as

$$RS(R_i) = \frac{1}{d^i} \tag{4.13}$$

where

$$d^i = \max_{j=1,\ldots,n+1} \{d_i^j\}, \tag{4.14}$$

with

$$d_i^j = \max\left\{\frac{d_{i1}^j}{d_{i2}^j}, \frac{d_{i2}^j}{d_{i1}^j}\right\} \tag{4.15}$$

and

$$d_{i1}^j = c_{ij}^2 - c_{ij}^1, d_{i2}^j = c_{ij}^3 - c_{ij}^2. \tag{4.16}$$

Clearly, $RS \leq 1$, and if the membership function is symmetric, then $RS = 1$.

Low niche interaction rate. This criterion is introduced in [5]. Let $N_i = (N_ix, N_iy)$ be the centers of the fuzzy rules (niches) determined until now ($i = 1, \ldots, d$, where d is the number of runs of the construction method). Let C be a chromosome from the current population. Then the *niche interaction rate* penalizes the fitness score associated with C in the following way

$$LNIR(C) = 1 - NIR(C), \tag{4.17}$$

$$NIR(C) = \max_i\{h_i\}, \tag{4.18}$$

$$h_i = *(A(N_ix), B(N_iy)), i = 1, \ldots, d, \tag{4.19}$$

$$A(N_ix) = *(A_1(N_ix_1), \ldots, A_n(N_iy)), \tag{4.20}$$

$$C \sim R_i\text{: } \textbf{If } x_1 \text{ is } A_1 \textbf{ and } \ldots \textbf{and } x_n \text{ is } A_n \textbf{ then } y \text{ is } B. \tag{4.21}$$

Hence $LNIR(C)$ is defined on $[0, 1]$. It gives the maximum value (no penalization) when the fuzzy rule encoded in C does not interact with any of the fuzzy rules constructed previously. The minimum value (maximum penalization) is obtained when the fuzzy rule encoded in C is identical to a fuzzy rule constructed previously.

Therefore, the combination of the niche scheme and the covering method will allow us to verify the two following fundamental aspects of the GFIM:

– GFIM will ensure that fuzzy rules are identified for each available example. The completeness property is verified in this manner.

– GFIM will maintain an adequate rule distribution in each one of the niches existing in the space of examples. It is known that the good performance of a fuzzy model is due to its interpolatation ability. A particular input usually fires more than one fuzzy rule and the interaction between the simultaneously fired fuzzy rules is what allows the fuzzy model to determine the best output for this particular input. Hence, an adequate interaction rate between neighboring rules will improve the fuzzy model quality. Fuzzy rules too close to each other may cause an undesirable *overlearning* due to the fact that their excessive interaction makes the inferred output move away from the optimal. Remote rules make the fuzzy model lose its interpolation capability.

4.2.2 Identification with Unconstrained Free Semantics. The construction method for fuzzy rules is developed by means of a RCGA, where each chromosome represents a fuzzy rule. The RCGA obtains the best fuzzy rule according to a set of fitness criteria, which are included in the fitness function of the RCGA. We describe the RCGA components [22] below.

Representation. In the RCGA population, a candidate chromosome C_r, $r = 1, \ldots, M$, represents a fuzzy rule

If x_1 is A_{r1} **and** ... **and** x_n is A_{rn} **then** y is B_r

where the reals (a_{rj}, b_{rj}, c_{rj}, a_r), (b_r, c_r) are the parameter vectors of the membership functions of A_{rj}, $j = 1, \ldots, n$, and B_r respectively. C_r codes these vectors as

$$(a_{r1}, b_{r1}, c_{r1}, \ldots, a_{rn}, b_{rn}, c_{rn}, a_r, b_r, c_r).$$

As was justified in [21], we propose approaching the identification problem with real coded chromosomes together with special genetic operators developed for them. Then a fuzzy rule will be a chromosome vector coded as a vector of floating point numbers [22].

Finally, we represent a population of M chromosomes (fuzzy rules) by C, and it is set up as follows

$$C = (C_1, \ldots, C_M). \tag{4.22}$$

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

Initial chromosome pool. We denote the domain of every input and output variable as X_j and Y, respectively. These domains are closed intervals of reals denoted as U_j and V, respectively. Furthermore, we consider an extension of these intervals in order to define the membership functions covering their extreme values. The intervals will increase their width by 10% or more for each extreme value. This permits us to cover the extreme values of the domains in

an adequate form. In this way, the extended intervals are $U_j = [u_j^1, u_j^2]$ and $V = [v^1, v^2]$.

The initial chromosome pool is created partially from $E_t \subseteq E_p$ (t chromosomes). The remaining ($M - t$ chromosomes) are initialized randomly, as follows:

- Let $t = \min\{|E_p|, M/2\}$. We choose at random t examples from E_p and for each example we determine the chromosome (fuzzy rule) belonging to the initial chromosome pool as follows. Consider the example $e^\ell \in E_t$ and its component $ex_j^\ell \in [u_j^1, u_j^2]$, $\triangle ex_j^\ell = \min\{ex_j^\ell - u_j^1, u_j^2 - ex_j^\ell\}$. Let $\delta(ex_j^\ell)$ be a random value in the range $[0, \triangle ex_j^\ell]$. Then we construct the membership function by means of the triple

  $$(ex_j^\ell - \delta(ex_j^\ell), ex_j^\ell, ex_j^\ell + \delta(ex_j^\ell)).$$

  The procedure is the same for the remaining components of e_ℓ.
- The remaining $M - t$ chromosomes of the initial population are chosen at random, each chromosome in its respective interval,

  $$C_r = (c_{r1}, \ldots, c_{r\ell}), \tag{4.23}$$

  $\ell = 3 \cdot (n + 1)$, with requirements $c_{3s+1} \leq c_{3s+2} \leq c_{3s+3}$, $s = 0, \ldots, n$.

Evaluation of chromosome fitness. As we commented earlier, we define the fitness function either according to the five criteria employed in [22] or the six ones presented in this work.

An *evaluation function* for the fuzzy rule R_i, and therefore a fitness function for the associated chromosome C_i is defined as

$$Z_1(R_i) = \Psi_{E_p}(R_i) \cdot G_w(R_i) \cdot g_n(R_i^-) \cdot MWR(R_i) \cdot RS(R_i), \tag{4.24}$$

$$Z_2(R_i) = \Psi_{E_p}(R_i) \cdot G_w(R_i) \cdot g_n(R_i^-) \cdot MWR(R_i) \cdot RS(R_i) \cdot LNIR(R_i). \tag{4.25}$$

The objective in both of the above cases is the maximization of the fitness function.

Genetic operators. During the GA reproduction phase we use two classical genetic operators, mutation and crossover. The ones selected are the non-uniform mutation proposed by Z. Michalewicz [30], and the max-min-arithmetical crossover used in [21]. A short description of them is given below.

- *Non-uniform mutation*
  If $C_v^t = (c_1, \ldots, c_k, \ldots, c_H)$ is a chromosome and the element c_k was selected for this mutation (the domain of c_k is $[c_{k\ell}, c_{kr}]$), the result is a vector $C_v^{t+1} = (c_1, \ldots, c_k', \ldots, c_H)$, with $k \in 1, \ldots, H$, and

  $$c_k' = \begin{cases} c_k + \triangle(t, c_{kr} - c_k) & \text{if } a = 0 \\ c_k - \triangle(t, c_k - c_{k\ell}) & \text{if } a = 1 \end{cases} \tag{4.26}$$

where a is a random number that may have a value of zero or one, and the function $\triangle(t, y)$ returns a value in the interval $[0, y]$. This value is such that the probability of $\triangle(t, y)$ being close to 0 increases as t increases

$$\triangle(t, y) = y(1 - r^{(1-\frac{t}{T})^b}). \tag{4.27}$$

In the above expression, r is a random number in the interval $[0, 1]$, T is the maximum number of generations, and b is a parameter chosen by the user, which depends on the number of iterations. This property of the probability of $\triangle(t, y)$ forces the non-uniform mutation operator to perform uniform search in the initial chromosome space when t is small, and local search for larger t.

– *Max-min-arithmetical crossover*
If $C_v^t = (c_1, \ldots, c_k, \ldots, c_H)$ and $C_w^t = (c'_1, \ldots, c'_k, \ldots, c'_H)$ are to be crossed, we generate the following four offsprings,

$$\begin{aligned}
&C_1^{t+1} = aC_w^t + (1-a)C_v^t,\\
&C_2^{t+1} = aC_v^t + (1-a)C_w^t,\\
&C_3^{t+1} \text{ with } c_{3k}^{t+1} = \min\{c_k, c'_k\},\\
&C_4^{t+1} \text{ with } c_{4k}^{t+1} = \max\{c_k, c'_k\}.
\end{aligned} \tag{4.28}$$

This operator can use a parameter a which is either a constant, or a variable whose value depends on the age of the population. The resulting descendents are the two best of the four aforesaid offsprings.

With regard to the *selection procedure*, it is a stochastic universal sampling [3], in which the number of offsprings is limited by the floor and ceiling of the expected number of offsprings, together with an elitist selection.

4.2.3 Identification with Constrained Free Semantics. This construction method for fuzzy rules is developed by means of a special GA, where a chromosome encodes a fuzzy rule, and an ES that locally tunes the fuzzy rules. In the following, we describe this method. For more detail see [5].

Representation. Here, a chromosome C is composed of two different parts, C_1 and C_2, each one corresponding to each one of the fuzzy model components. The first part of the chromosome encodes the linguistic fuzzy rule (belonging to the RB), and the second one the membership functions parameters for the input and output variables involved in the fuzzy rule (belonging to the DB).

In order to represent the first part there is a need to number the linguistic values belonging to each one of the term sets for the input and output variables. A variable x_i taking linguistic values in a term set $T(x_i) = \{L_1(x_i), \ldots, L_{n_i}(x_i)\}$ has associated with it the set $T'(x_i) = \{1, \ldots, n_i\}$. On the other hand, the second part adopts the same representation as in the case of unconstrained free semantics. Hence, the fuzzy rule

If x_1 is $L_{i_1}(x_1)$ **and** ... **and** x_n is $L_{i_n}(x_n)$ **then** y is $L_{i_{n+1}}(y)$

is encoded into a chromosome C of the following form

$$\begin{aligned} C_1 &= (i_1, \ldots, i_n, i_{n+1}), \\ C_2 &= (a_{L_{i_1}(x_1)}, b_{L_{i_1}(x_1)}, c_{L_{i_1}(x_1)}, \ldots, a_{L_{i_n}(x_n)}, b_{L_{i_n}(x_n)}, c_{L_{i_n}(x_n)}, \\ &\quad a_{L_{i_{n+1}}(y)}, b_{L_{i_{n+1}}(y)}, c_{L_{i_{n+1}}(y)}) \\ C &= C_1 C_2. \end{aligned} \tag{4.29}$$

Initial chromosome pool. Part of the initial chromosome pool is obtained making use of the examples contained in the training set, E_p, and the remainder of it is generated at random. However a third possibility can be considered. With M being the GA population size and $t = \min\{|E_p|, \frac{M}{3}\}$, let t examples be selected at random from E_p. Then, the initial population is generated in three steps as follows:

1. Using fuzzy partitions, generate t chromosomes by taking the fuzzy rule which covers best each one of the t randomly selected examples. Initialize C_1 and C_2 by coding the fuzzy rule linguistic values and their semantics.
2. Generate another t chromosomes by initializing C_1 in the same way as in the previous step, and compute the values of C_2 at random, letting each chromosome vary in its respective interval.
3. Generate the remaining $M - 2 \cdot t$ chromosomes by computing at random the values of C_1, and making use of these for randomly generating the C_2 part, letting again each chromosome vary in its respective interval.

Evaluation of chromosome fitness. Due to the fact that permitted chromosome variations are restricted to being performed in intervals smaller than those considered in the case of unconstrained free semantics, several previously employed fitness criteria lose their meaning and others become necessary. Hence, the membership function symmetricity and width rates are not used anymore, but the niche interaction rate is required for obtaining an adequate fuzzy rule interaction. The fitness function is finally defined in the following way

$$Z_3(R_i) = \Psi_{E_p}(R_i) \cdot G_w(R_i) \cdot g_n(R_i^-) \cdot LNIR(R_i). \tag{4.30}$$

Genetic operators. Due to the special nature of the chromosomes involved in the case of constrained free semantics, the design of special genetic operators is required. For a more detailed description of these see [5].

With respect to mutation, two different operators are used, each one of them acting on a different chromosome part. Since C_2 corresponds to the representation employed in the case of unconstrained free semantics, the same mutation operator designed for this case is used for C_2. Thus, Michalewicz's non-uniform mutation operator is employed.

The mutation operator selected for C_1 is similar to the one proposed by Thrift in [36]. When mutation for the C_1 part of the chromosome is going to be performed, a local modification is developed by changing the current linguistic value to one of its neighboring linguistic values (the decision is made at random). When the linguistic value to be changed is the first or last one in the term set, the only possible change is to substitute it with its right or left neighbor respectively. Obviously, a mutation in C_1 provokes a change in C_2. When an input/output changes its linguistic value value from one term to another, the semantics (membership function) associated with it is automatically updated in the second part of the chromosome by the default values in the corresponding fuzzy partitioning.

With regard to recombination, two different crossover operators are employed. If the fuzzy rule encoded by both parent chromosomes is the same, then the max-min-arithmetical crossover operator is applied to C_2 and obviously the parent C_1 values are maintained in the offspring. On the other hand, when the parent chromosomes encode different rules, it makes no sense to apply this operator. Instead, a standard crossover operator is applied on both parts of the parent chromosomes. This operator performs as follows. A crossover point *cp* is randomly generated in C_1 and the two parent chromosomes are crossed at the *cp*-th and $n+1+3\cdot cp$ crossover points in each chromosome part respectively. The crossover is thus performed in both chromosome parts, C_1 and C_2, thereby producing two meaningful offsprings.

Let us consideran example in order to clarify the standard crossover operator. Since

$$\begin{aligned} C_t &= (c_1,\ldots,c_{cp},c_{cp+1},\ldots,c_{n+1},a_{c_1},b_{c_1},c_{c_1},\ldots, \\ &\quad a_{c_{cp}},b_{c_{cp}},c_{c_{cp}},a_{c_{cp+1}},b_{c_{cp+1}},c_{c_{cp+1}},\ldots,a_{c_{n+1}},b_{c_{n+1}},c_{c_{n+1}}) \end{aligned} \tag{4.31}$$

and

$$\begin{aligned} C'_t &= (c'_1,\ldots,c'_{cp},c'_{cp+1},\ldots,c'_{n+1},a_{c'_1},b_{c'_1},c_{c'_1},\ldots, \\ &\quad a_{c'_{cp}},b_{c'_{cp}},c_{c'_{cp}},a_{c'_{cp+1}},b_{c'_{cp+1}},c_{c'_{cp+1}},\ldots,a_{c'_{n+1}},b_{c'_{n+1}},c_{c'_{n+1}}) \end{aligned} \tag{4.32}$$

are the chromosomes to be crossed at the point *cp*, the two resulting offsprings are

$$\begin{aligned} C_{t+1} &= (c_1,\ldots,c_{cp},c'_{cp+1},\ldots,c'_{n+1},a_{c_1},b_{c_1},c_{c_1},\ldots, \\ &\quad a_{c_{cp}},b_{c_{cp}},c_{c_{cp}},a_{c'_{cp+1}},b_{c'_{cp+1}},c_{c'_{cp+1}},\ldots,a_{c'_{n+1}},b_{c'_{n+1}},c_{c'_{n+1}}), \end{aligned} \tag{4.33}$$

$$\begin{aligned} C'_{t+1} &= (c'_1,\ldots,c'_{cp},c_{cp+1},\ldots,c_{n+1},a_{c'_1,b_{c'_1}},c_{c'_1},\ldots, \\ &\quad a_{c'_{cp}},b_{c'_{cp}},c_{c'_{cp}},a_{c_{cp}},b_{c_{cp}},c_{c_{cp}},\ldots,a_{c_{n+1}},b_{c_{n+1}},c_{c_{n+1}}). \end{aligned} \tag{4.34}$$

The last genetic operator is based on *(1+1)-ES*. This optimization technique has been selected and integrated into the genetic recombination process in order to perform a local tuning of the best chromosomes (fuzzy rules) obtained in each run. Each time a GA is performed, the ES will be applied over

a percentage α of the best chromosomes from the current genetic population and will adjust their C_2 parts.

The ES employed was briefly presented in Section 2.2. In our case, the step size σ can not be a single value because each one of the membership functions encoded in the second part of the chromosome is defined over different universes of discourse and thus, requires mutations of a different order. Following the modus operandi presented in [5], each parent component x_i varying in the interval of performance $[x_i^\ell, x_i^r]$ will have its own associated step size $\sigma_i = \sigma \cdot s_i$ with $s_i = (x_i^r - x_i^\ell)/4$. When the mutated value $x_i' = x_i + z_i$ does not belong to the interval of performance, it is assumed equal to the interval extent, x_i^ℓ or x_i^r, closer to $x_i + z_i$.

The *selection procedure* is again based on Baker's stochastic universal sampling and elitist selection.

4.2.4 Covering method. The covering method is presented in detail in [22]. It is developed as an iterative process that allows to obtain a set of fuzzy rules covering the example set. In each iteration, it considers the relative covering value the best fuzzy rule (chromosome) has for the given training set, and removes from it the examples for which the covering value is greater than ϵ.

Let R^e be the set of fuzzy rules provided by a human expert (expert fuzzy rules). For any one of the cases of unconstrained free semantics and constrained free semantics, the covering method proceeds as follows:

1. Initialization stage:
   - Determines k, ω and ϵ .
   - Merges the fuzzy rules in R^e with the fuzzy rules , R^g, which are obtained via identification from inut-output data.
   - Assigns $CV[\ell] \leftarrow CV_{R^e}(e_\ell)$, $\ell = 1, \ldots, p$.
   - If $CV[\ell] \geq \epsilon$, e_ℓ from E_p, $\ell = 1, \ldots, p$ is removed.
2. Applies the specific construction method for the given set of examples E_p,
3. Seelects the best chromosome C_r encoding the fuzzy rule R_r.
4. Merges R_r with R^g.
5. For every $e_\ell \in E_p$ does
   $CV[\ell] \leftarrow CV[\ell] + R_r(e_\ell)$,
   If $CV[\ell] \geq \epsilon$ then remove it from E_p.
6. If $E_p = \emptyset$ then Stop else return to Step 2.

Since there may be similar fuzzy rules in R^g, or a fuzzy from R^g may be similar to a fuzzy rule in R^e, it is necessary to simplify the RB obtained .

4.3 Genetic Simplification

Due to the iterative nature of genetic identification, it may result in redundant fuzzy rules and/or overlearning . This latter occurs when some examples are

covered to degree higher than the desired one and in this case the performance of the identified RB is negatively affected.

The genetic simplification was proposed in [23]. It is based on a binary coded GA, in which the selection of chromosomes is performed by using the stochastic universal sampling procedure together with an elitist selection scheme. Also, recombination is put into effect by using the classical binary multipoint crossover (performed on two points) and uniform mutation operators.

The coding scheme generates fixed-length chromosomes. Let us now consider the fuzzy rules in the RB, constructed at the previous step and numbered from 1 to m. An m-bit string $C = (c_1, \ldots, c_m)$ represents a subset of candidate fuzzy rules that is to be obtained as the output, B^s, of genetic simplification, and such that

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

The initial population of chromosomes is obtained by introducing a chromosome representing the complete rule set R^g, that is, $c_i = 1$ for all i. The remaining chromosomes are selected at random.

The fitness function, $E(\cdot)$ is based on an application-specific measure, usually employed in the design of GFSs: the mean square error (SE) over a training data set, E_{TDS}, given as

$$E(C_j) = \frac{1}{2|E_{TDS}|} \sum_{e_\ell \in E_{\mathrm{TDS}}} (ey^\ell - S(ex^\ell))^2 \tag{4.35}$$

where $S(ex^\ell)$ is the output value obtained from the identified fuzzy model using the model structure (RB) coded in C_j, $R(C_j)$, when the input values are ex^ℓ, and ey^ℓ is the known desired output value.

During genetic simplification, there is a need to keep the completeness property considered previously: the model must always be able to infer a proper output for every system input. We will 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,

$$C_{R(C_j)}(e_\ell) = \bigcup_{j=1..T} R_j(e_\ell) \geq \tau, \quad \forall e_\ell \in E_{TDS} \text{ and } R_j \in R(C_j) \tag{4.36}$$

where τ is the degree of completeness of the minimal training set used for genetic simplification. Usually, τ is less than or equal to ω, the compatibility degree used in the identification. Therefore, we define a *training set completeness degree* of $R(C_j)$ on the set of examples E_{TDS} as

$$TSCD(R(C_j), E_{TDS}) = \bigcap_{e_\ell \in E_{\mathrm{TDS}}} C_{R(C_j)}(e_\ell). \tag{4.37}$$

The fitness function, penalizing the lack of the completeness property is

$$F(C_j) = \begin{cases} E(C_j) & \text{if } TSCD(R(C_j), E_{TDS}) \geq \tau \\ \frac{1}{2} \sum\limits_{e_\ell \in E_{\text{TDS}}} (ey^\ell)^2 & \text{otherwise.} \end{cases} \quad (4.38)$$

4.4 Genetic Tuning

Genetic tuning process is presented in-depth in [21]. It is based on the existence of an initial fuzzy model, that is, an initial estimation of the model parameters (DB), and initially identified fuzzy model structure defined by a RB and composed by m fuzzy rules.

Each chromosome forming the genetic population encodes the whole RB, R^s, with different fuzzy model parameters associated with it.

The GA designed for the tuning uses real numbers coding , stochastic universal sampling as selection procedure, and Michaelewicz's non-uniform mutation operator. The crossover operator, max-min-arithmetical, is employed again.

As we commented before, the membership functions have a triangular form. Thus, each one of them has an associated parametric representation based on a triple of real values. Each one of the fuzzy rules will be encoded in parts of the chromosome C_{ri}, $i = 1, \ldots, m$, in the following way

$$C_{ri} = (a_{i1}, b_{i1}, c_{i1}, \ldots, a_{in}, b_{in}, c_{in}, a_i, b_i, c_i). \quad (4.39)$$

Therefore the complete RB with an associated DB is represented by a complete chromosome C_r

$$C_r = C_{r1}\ C_{r2}\ \ldots\ C_{rm}. \quad (4.40)$$

Each chromosome in the population represents a complete fuzzy model i.e., both RB and DB. In particular, all of them encode the identified RB, R^s, i.e., the model structure identified in the previous two stages, and the only difference between the different chromosomes are the different membership functions, that is, the different DBs.

The initial fuzzy model is encoded directly into a chromosome, denoted as C_1. The remaining ones are generated by associating an interval of performance, $[c_h^\ell, c_h^r]$ to every c_h in C_1, $h = 1 \ldots (n+1) \cdot m \cdot 3$. Each interval of performance will be the interval of adjustment for the corresponding variable, $c_h \in [c_h^\ell, c_h^r]$.

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

$$c_t \in [c_t^\ell, c_t^r] = [c_t - \frac{c_{t+1} - c_t}{2}, c_t + \frac{c_{t+1} - c_t}{2}], \quad (4.41)$$

$$c_{t+1} \in [c_{t+1}^\ell, c_{t+1}^r] = [c_{t+1} - \frac{c_{t+1} - c_t}{2}, c_{t+1} + \frac{c_{t+2} - c_{t+1}}{2}], \quad (4.42)$$

$$c_{t+2} \in [c_{t+2}^\ell, c_{t+2}^r] = [c_{t+2} - \frac{c_{t+2} - c_{t+1}}{2}, c_{t+2} + \frac{c_{t+3} - c_{t+2}}{2}]. \quad (4.43)$$

Figure 4.1 shows these intervals of performance.

Therefore, we create a population of chromosomes containing C_1 as its first individual and where the remaining ones are initiated randomly, with each one being in its respective interval of performance.

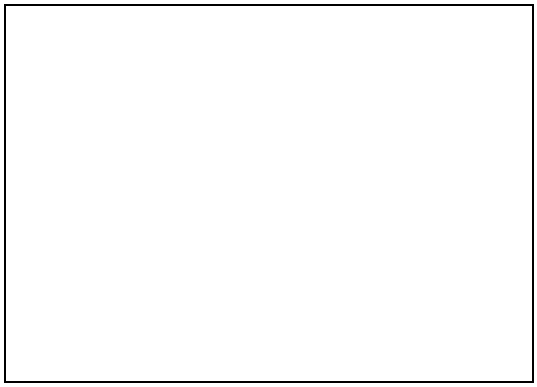

Fig. 4.1. Membership function and intervals of performance for genetic tuning.

The fitness function of a chromosome is defined by using the training input-output data set, E_{TDS}, and a specific error measure, the mean square error. In this way, the adaptation value associated with a chromosome is obtained by computing the error between the outputs given by the fuzzy model contained in the chromosome and the outputs contained in the training input-output data set. The fitness function is given as

$$E(C) = \frac{1}{2|E_{TDS}|} \sum_{e_\ell \in E_{\mathrm{TDS}}} (ey^\ell - S(ex^\ell))^2. \tag{4.44}$$

4.5 Summary of the Identification Procedure

This section summarizes the presented GFIM, showing some guidelines for its use idepending on the information available about the system under identification.

I. **Available knowledge.** Once the input-output data set is available, the existence of any other kind of prior knowledge for the purpose of identification should be studied: the possibility of defining the model parameters (in the sense of initial fuzzy partitions), the availability of a partial fuzzy model (in the sense of an incomplete RB with or without an initial definition of the DB), or a complete one (in the sense of a complete RB and an initial definition of the DB).

II. **Using the GFIM.** Once the the existing knowledge is made available, the particular identification method to be applied has to be chosen.

1. *No prior knowledge:* In this case, we only have the input-output data set for performing identification. Therefore, the unconstrained free semantics type of identification is first applied for obtaining an initial definition of the fuzzy model structure and parameters. Genetic simplification is then applied for identifying the final fuzzy model structure. Finally, genetic tuning obtains the final fuzzy model parameters.
2. *Initial fuzzy model parameters:* When initial fuzzy partitions, a very preliminary definition of the model parameters, are provided, the overall learning process is applied for identifying a fuzzy model taking the previous semantics as a base. The only change with respect to the previous case is that the genetic construction method used will be the constrained free semantics one.
3. *Partial fuzzy model:* When an incomplete linguistic fuzzy model has been derived from a human expert, the GFIM permits the incorporation of this partial RB by merging it with the one obtained from genetic identification (constrained or unconstrained free semantics). Genetic simplification and tuning are then applied for obtaining the final model structure and parameters.
4. *Complete fuzzy model:* When a complete linguistic fuzzy model has been derived from an expert, i.e., the fuzzy model structure and the fuzzy model parameters are both available, genetic tuning may be used to obtain a more accurate fuzzy model by adjusting the available fuzzy model parameters.

III. **Model validation.** The GFIM proposed takes into account some fuzzy rule base properties in order to guarantee good quality linguistic fuzzy models. Thus, the analysis of how well the fuzzy rules cover the data space should be done with care since a bad coverage decreases the quality of the linguistic fuzzy model. Anyway, there is always the need to validate the identified fuzzy model through numerical simulation and comparisons with system data in order to improve its validity.

To conclude this section we would like to note here that any other type of membership functions may be incorporated in the GFIM with minor modifications. Furthermore, in [7] a different GFIM for the case of unconstrained free semantics is presented where the construction method for fuzzy rules uses an inductive algorithm and an ES. The structure of this GFIM may be used for identifying another type of a linguistic fuzzy model, namely a descriptive linguistic fuzzy model [8, 10].

5. Example

In order to analyze the accuracy of the GFIM, we will show now how two n-dimensional functions can be used to identify three-dimensional surfaces. Three different ways of fuzzy model identification these will be compared:

1. Two GFIMs for the case of unconstrained free semantics using the fitness functions Z_1 and Z_2 respectively, and
2. The GFIM for the case of constrained free semantics using the fitness function Z_3.

The n-dimensional functions and the universes of discourse considered are shown below. The *spherical model*, F_1, is an unimodal function, while the *generalized Rastrigin function*, F_2, is a strongly multimodal one. These are illustrated in (Figures 5.1 and 5.3).

$$F_1(x_1, x_2) = x_1^2 + x_2^2, \qquad (5.1)$$
$$x_1, x_2 \in [-5, 5], \quad F_1(x_1, x_2) \in [0, 50].$$

$$F_2(x_1, x_2) = x_1^2 + x_2^2 - \cos(18x_1) - \cos(18x_2), \qquad (5.2)$$
$$x_1, x_2 \in [-1, 1], \quad F_2(x_1, x_2) \in [2, 3.5231].$$

For each function, an input-output training data set, uniformly distributed in the three-dimensional space has been obtained experimentally. In this way, two sets with 1681 values have been generated by taking 41 values for each of the two state variables considered to be uniformly distributed in their respective intervals.

Two other data sets have been generated for their use as test sets. These are to be used for evaluating the performance of the learning method thus avoiding any possible bias related to the data in the training set. The size of these data sets is ten percent of the size of the corresponding training set. The data is obtained by generating the state variables values randomly in the concrete universes of discourse for each of them, and then computing the associated output value. Hence, two test sets formed by 168 data are used to measure the accuracy of the fuzzy models identified by computing the mean square error for these fuzzy models.

The initial fuzzy model parameters used in the GFIM for the case of constrained free semantics are defined in terms of three initial fuzzy partitions (two corresponding to the input variables and one associated with the output). The fuzzy partitions have *seven linguistic values* and the semantics of these is defined via the use of triangular membership functions (as shown in Figure 3.1). Adequate scaling factors are used to translate the generic universe of discourse into the one that is associated with each system variable.

The following parameters, corresponding to the first two stages of the identification, are combined for determining the number of runs for the three different GFIMs: $\epsilon = 1.5$, $\omega = 0.05$, $k = 0.1$ and $\tau = \{0.25, 0.5\}$. This leads to an overall of 2 runs per function and GFIM. The remaining parameters used in the three GFIMs to be compared, are: the t-norm $*$ used in the fuzzy rule construction method is the *min* operator; GAs run over 50 generations and the ES is applied until there is no improvement in 25 generations over

a percentage $\alpha = 20\%$ of the population of chromosomes (the parameter c of the 1/5-success rule is equal to 0.9); genetic simplification and tuning run over 500 and 1000 generations, respectively. In all cases, the population is formed by 61 chromosomes, the value of the non-uniform mutation parameter b is 5.0, and the crossover and mutation rates are $P_c = 0.6$ and $P_m = 0.1$ (this last one per indiviual) respectively. The max-min-aritmethical crossover parameter a takes the value 0.35.

Finally, we use the *min* t-norm for representing fuzzy implication, and the center of gravity as defuzzification operator [11].

The results obtained are shown in the tables below, each table is associated with both of the functions considered. The notation $|R|_x$ stands for the number of fuzzy rules in the RB, while SE_x stands for the medium square error obtained by the current fuzzy model on the corresponding test set at each stage (x is equal to G, S, and T in the genetic construction, simplification and tuning stages, respectively).

Table 5.1. Results obtained using the three proposed GFIMs for the fuzzy model of the function F_1

GFIM	τ	$\|R\|_G$	SE_G	$\|R\|_S$	SE_S	SE_T
1	0.25	108	5.823179	82	2.794654	0.929880
1	0.5	108	5.823179	82	3.423883	1.094663
2	0.25	180	17.448940	119	2.013139	0.994133
2	0.5	180	17.448940	119	2.662488	1.177372
3	0.25	98	2.411402	67	1.779137	0.696869
3	0.5	98	2.411402	73	2.130197	1.118251

Table 5.2. Results obtained using the three proposed GFIMs for the fuzzy model of the function F_2

GFIM	τ	$\|R\|_G$	SE_G	$\|R\|_S$	SE_S	SE_T
1	0.25	250	0.398029	181	0.324123	0.290474
1	0.5	250	0.398029	196	0.344089	0.307614
2	0.25	345	0.393328	264	0.283268	0.265746
2	0.5	345	0.393328	275	0.359460	0.327285
3	0.25	346	0.268026	232	0.213960	0.195233
3	0.5	346	0.268026	253	0.232196	0.210177

To illustrate the performance of the proposed GFIM, some of the fuzzy models obtained are shown in the following figures. The two fuzzy models

depicted are the ones that best approximate each one of the functions considered, that is, the ones with the lowest mean square error. Figures 5.2 and 5.4 show the fuzzy models for the functions F_1 and F_2 which fuzzy models are obtained by means of the constrained free semantics GFIM using the fitness function Z_3 and $\tau = 0.25$.

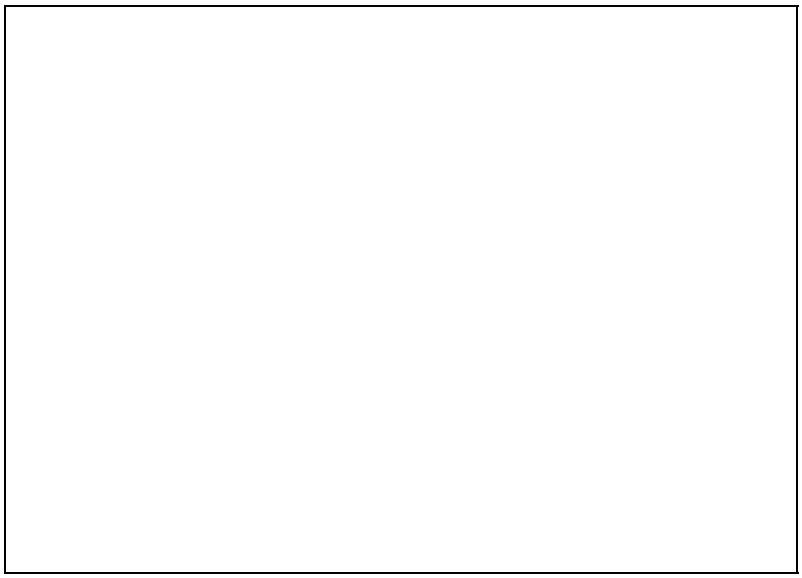

Fig. 5.1. Graphical representation of F_1.

6. Practical Considerations and Concluding Remarks

This section summarizes the practical aspects of the proposed genetic identification method, pointing out its advantages and drawbacks.

6.1 Use of Prior Knowledge

As observed in Section 4, the GFIM can be used in different modes reflecting the knowledge available about the system under identification. This offers the possibility of using both prior expert knowledge as well as numerical input-output data.

The main advantage of the fuzzy model considered is its ability to deal with complex systems with strong nonlinearities. In [10], the GFIM is used for the purpose of the identification of descriptive and approximative fuzzy models of three-dimensional functions. The approximative approach shows better

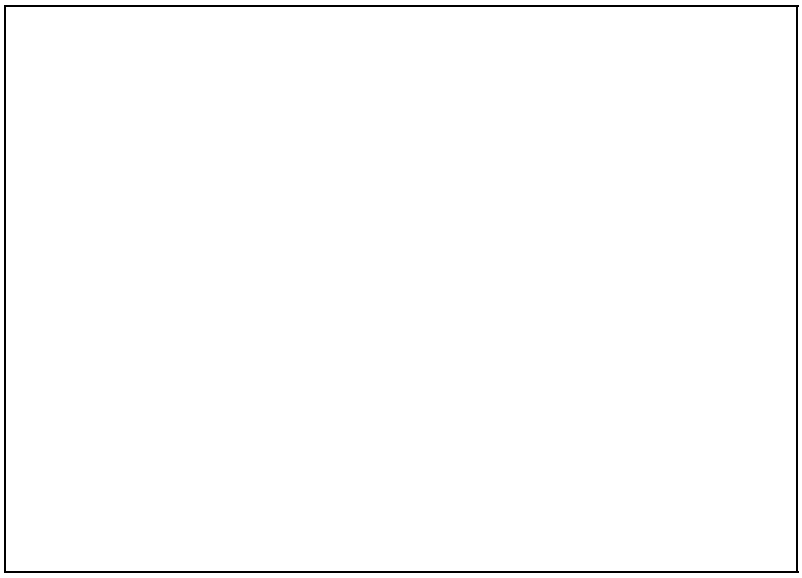

Fig. 5.2. Graphical representation of the fuzzy model obtained for F_1.

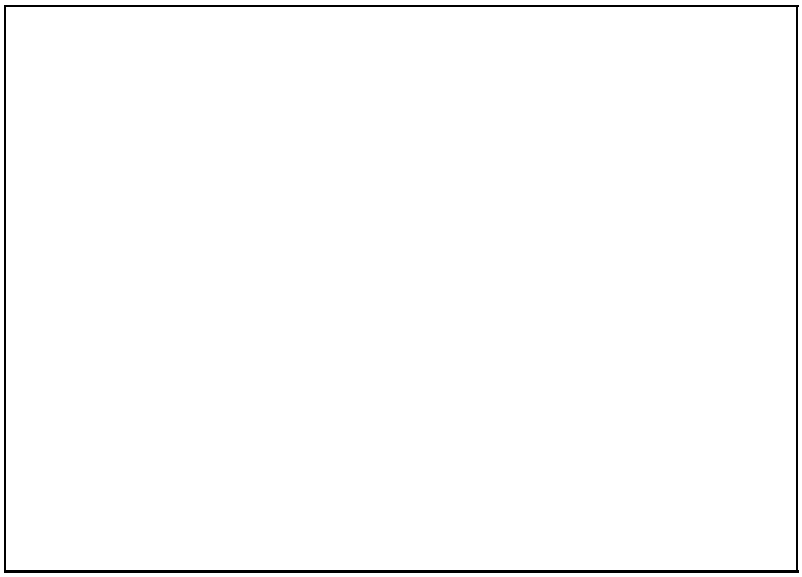

Fig. 5.3. Graphical representation of F_2.

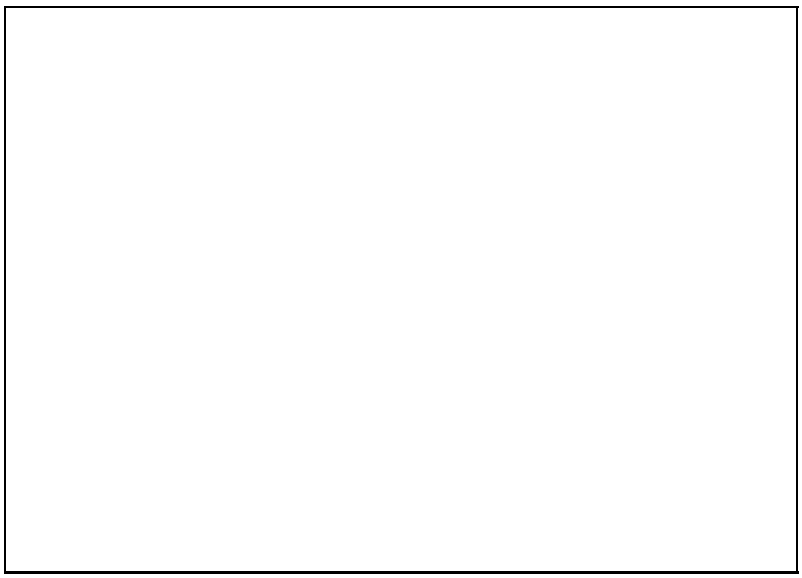

Fig. 5.4. Graphical representation of the fuzzy model obtained for F_2.

performance than the descriptive one in the case of more complex functions (such as the function F_2 considered in Section 5). The drawback that may be associated with GFIM is the loss of interpretability of the approximative fuzzy model obtained due to the lack of explicit linguistic values. However, this may be justified by the higher modeling accuracy of this type of fuzzy model.

6.2 Model Complexity

Another advantage of the GFIM presented is that the proposed learning process allows the user to obtain the desired tradeoff between fuzzy model accuracy and complexity. The number of fuzzy rules in the RB may be modified by the following factors:

1. The number of linguistic values in the initial fuzzy partitions used in the case of constrained free semantics .
2. The value of the parameter ϵ. The higher the value, the greater the number of fuzzy rules and vice versa.

On the other hand, the identification of the fuzzy model structure is directly guided by the composition of the input-output data set. Therefore, no fuzzy rules are obtained in the regions of the data space that are empty. The necessary number of fuzzy rules is determined by the values of the parameter ϵ. No redundancy may occur due to the low niche interaction rate criterion. Genetic simplification allows us to obtain an adequate interaction

rate between the fuzzy rules and avoids overlearning. Genetic simplification is a drawback in the case when an adequate input-output data set can not be obtained.

6.3 Robustness of the Identification Method

The robustness of GFIM is highly dependent on the input-output data available. Poor data results in an incomplete fuzzy model structure and a inadequate fuzzy rule interaction level. The presence of noise in the data will have the same effect on the final fuzzy model.

6.4 Real-world Applications

In [21, 23], the proposed GFIM is applied to fuzzy controller design. Recently, the method has been applied to a number of classification problems, obtaining good results on laboratory data sets such as the Iris one [9]. The prediction of economic time series is currently under investigation.

References

1. Babuška, R. (1995): Fuzzy Modeling. A Control Engineering Perspective. Proc. of Fourth IEEE International Conference on Fuzzy Systems, 1897-1902. Yokohama.
2. Bäck, T., Schwefel, H.-P. (1995): Evolution strategies I: Variants and their computational implementation. J. Periaux, G. Winter, M. Galán, P. Cuesta (Eds.), Genetic Algorithms in Engineering and Computer Science, 111-126. John Wiley and Sons.
3. Baker, J.E. (1987): Reducing Bias and Inefficiency in the Selection Algorithm. J.J. Grefenstette (Ed.), Proc. Second International Conference on Genetic Algorithms, 14-21. Lawrence Erlbaum, Hillsdale, NJ.
4. Cordón, O., Herrera, F. (1995): A General Study on Genetic Fuzzy Systems. J. Periaux, G. Winter, M. Galán, P. Cuesta (Eds.), Genetic Algorithms in Engineering and Computer Science, 33-57. John Wiley and Sons.
5. Cordón, O., Herrera, F. (1996): A Hybrid Genetic Algorithm-Evolution Strategy Process for Learning Fuzzy Logic Control Knowledge Bases. F. Herrera, J.L. Verdegay (Eds.), Fuzzy Logic and Soft Computing, 251-278. Physica-Verlag.
6. Cordón, O., Herrera, F., Lozano, M. (1996): A Classified Review on the Combination Fuzzy Logic-Genetic Algorithms Bibliography: 1989-1995. E. Sanchez, T. Shibata, L. Zadeh (Eds.) Genetic Algorithms and Fuzzy Logic Systems. Soft Computing Perspectives. World Scientific.
7. Cordón, O., Herrera, F. (1996): Generating and Selecting Fuzzy Control Rules Using Evolution Strategies and Genetic Algorithms. Proc. Sixth International Conference on Information Processing and Management of Uncertainly in Knowledge-Based Systems (IPMU'96), 733-738. Granada (Spain).

8. Cordón, O., Herrera, F., Lozano, M. (1996): A Three-Stage Method for Designing Genetic Fuzzy Systems by Learning from Examples. H. M. Voight, W. Ebeling, I. Rechenberg, H. P. Schwefel (Eds.), L.N.C.S. 1141, Proc. Fourth International Conference on Parallel Problem Solving from Nature (PPSN IV), 720-729, Berlin (Germany).
9. Cordón, O., del Jesus, M.J., Herrera, F., (1996): A Fuzzy Classification System Based on Genetic Algorithms (in spanish). Proc. Sixth Spanish Conference on Fuzzy Logic and Technologies (FLAT'96), 95-100, Oviedo (Spain).
10. Cordón, O., Herrera, F. (1996): A Three-Stage Evolutionary Process for Learning Descriptive and Approximative Fuzzy Logic Controller Knowledge Bases from Examples. To appear: International Journal of Approximate Reasoning.
11. Cordón, O., Herrera, F., Peregrin, A. (1996): Applicability of the fuzzy operators in the design of fuzzy logic controllers. To appear: Fuzzy Sets and Systems.
12. Deb, K., Goldberg, D.E. (1989): An Investigation of Niche and Species Formation in Genetic Function Optimization. Proc. Second International Conference on Genetic Algorithms, 42-50. Lawrence Erlbaum, Hillsdale, NJ.
13. Delgado, M., González, A. (1993): An Inductive Learning Procedure to Identify Fuzzy Systems. Fuzzy Sets and Systems, **55**, 121-132.
14. Driankov, D., Hellendoorn, H., Reinfrank, M. (1993): An Introduction to Fuzzy Control. Springer-Verlag, Berlin.
15. Fogel, D.B. (1995): Evolutionary Computation. Toward a New Philosophy of Machine Intelligence. IEEE Press.
16. Genetic Algorithms Archive. URL address: http://www.aic.nrl.navy.mil/galist/.
17. Goldberg, D.E. (1989): Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley.
18. Grefenstette, J.J. (Ed.) (1994): Genetic Algorithms for Machine Learning. Kluwer Academic.
19. Harris, C.J., Moore, C.G., Brown, M. (1993): Intelligent Control: Aspects of Fuzzy Logic and Neural Nets. World Scientific.
20. Herrera, F., Verdegay, J.L. (1996): Genetic Algorithms and Soft Computing. Physica-Verlag.
21. Herrera, F., Lozano, M., Verdegay, J.L. (1995): Tuning Fuzzy Logic Controllers by Genetic Algorithms. International Journal of Approximate Reasoning, **12**, 299-315.
22. Herrera, F., Lozano, M., Verdegay, J.L. (1995): Generating Fuzzy Rules from Examples using Genetic Algorithms. B. Bouchon-Meunier, R.R. Yager, L.A. Zadeh (Eds.), Fuzzy Logic and Soft Computing, 11-20. World Scientific.
23. Herrera, F., Lozano, M., Verdegay, J.L. (1995): A Learning Process for Fuzzy Control Rules using Genetic Algorithms. Technical Report #95108, Dept. of Computer Science and Artificial Intelligence, University of Granada. Spain.
24. Herrera, F., Lozano, M., Verdegay, J.L. (1996): Tackling Real Coded Genetic Algorithms. To appear: Artificial Intelligence Review.
25. Holland, J.H (1975): Adaptation in Natural and Artificial Systems. Ann Arbor. [MIT Press (1992)].
26. Jang, J.R. (1991): ANFIS: Adaptive-Network-Based Fuzzy Inference System. IEEE Transactions on Systems, Man, and Cybernetics, **23** 665-685.
27. Lee, C.C. (1990): Fuzzy Logic in Control Systems: Fuzzy Logic Controller. Parts I and II. IEEE Transactions on Systems, Man and Cybernetics, **20**, 404-435.
28. Lee, C.C. (1991): A Self-Learning Rule-Based Controller Employing Approximate Reasoning and Neural Net concepts. International Journal of Intelligent Systems, **6**, 71-93.

29. Mamdani, E.H., Assilian, S. (1975): An Experiment in Linguistic Synthesis with a Fuzzy Controler. International Journal of Man-Machine Studies, **7**, 1-13.
30. Michalewicz, Z. (1992): Genetic Algorithms + Data Structures = Evolution Programs. Springer-Verlag.
31. Nomura, H., Hayashi, I., Wakami, N. (1992): A Learning Method of Fuzzy Inference Rules by Descent Method. Proc. IEEE Conference on Fuzzy Systems, San Diego, 203-210.
32. Pedrycz, W. (1984): Identification in Fuzzy Systems. IEEE Transactions on Systems, Man, and Cybernetics, **14**, 361-368.
33. Schwefel, H.-P. (1995): Evolution and Optimum Seeking. Sixth-Generation Computer Technology Series. John Wiley and Sons.
34. Spears, W.M., De Jong, K.A., Bäck, T., Fogel, D.B., de Garis, H. (1993): An Overview of Evolutionary Computation. Proc. European Conference on Machine Learning.
35. Takagi, T., Sugeno, M. (1985): Fuzzy Identification of Systems and its Applications to Modeling and Control. IEEE Transations on Systems, Man, and Cybernetics, **15**, 116-132.
36. Thrift, P. (1991): Fuzzy Logic Synthesis with Genetic Algorithms. Proc. Fourth International Conference on Genetic Algorithms, 509-513.
37. Yager, R.R., Filev, D.P. (1994): Essentials of Fuzzy Modeling and Control. John Wiley and Sons, New York.
38. Wang, L.X., Mendel, J.M. (1992): Generating Fuzzy Rules by Learning from Examples. IEEE Transactions on Systems, Man, and Cybernetics, **22** 1414-1427.
39. Yoshinari, Y., Pedrycz, W., Hirota, K. (1993): Construction of Fuzzy Models through Clustering Techniques. Fuzzy Sets and Systems, **54**, 157-165.