

# Genetic Fuzzy Systems: A Tutorial\*

Francisco Herrera<sup>1</sup> and Luis Magdalena<sup>2</sup>

<sup>1</sup>Dept. of Computer Science and A.I., E.T.S. Ingeniería Informática  
University of Granada, 18071 - Granada, Spain

<sup>2</sup>Dept. of Applied Mathematics, E.T.S.I. de Telecomunicación  
Universidad Politécnica de Madrid, 28040 - Madrid, Spain  
e-mail: herrera@decsai.ugr.es, llayos@mat.upm.es

## Abstract

The automatic definition of a fuzzy system can be considered in many cases as an optimization or search process. Genetic Algorithms (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. GAs are known to be capable of finding near optimal solutions in complex search spaces. A priori knowledge of a fuzzy system may come in the form of known linguistic variables, fuzzy membership functions parameters, fuzzy rules, number of rules, etc. The generic code structure and independent performance features of GAs make them suitable candidates for incorporating a priori knowledge.

The search capabilities and ability for incorporating a priori knowledge have extended the use of GAs in the development of a wide range of methods for designing fuzzy systems over the last few years. Systems applying these design approaches have received the general name of Genetic Fuzzy Systems (GFSs).

In this tutorial we summarize different GFSs approaches, focusing our presentation on genetic fuzzy rule-based systems.

**Keywords:** Fuzzy systems, fuzzy rule-based systems, genetic algorithms, tuning, learning.

**Subject Classifications:** AMS: 03E72, 94B70, 92D15.

---

\*This work has been partially supported by CICYT (Spanish Comisión Interministerial de Ciencia y Tecnología) (Projects TIC96-0778 and TAP94-0115).

## 1 Introduction

In a very broad sense, a Fuzzy System (FS) is any Fuzzy Logic-Based System, where Fuzzy Logic can be used either as the basis for the representation of different forms of system knowledge, or to model the interactions and relationships among the system variables. FFSs have proven to be an important tool for modeling complex systems, in which, due to the complexity or the imprecision, classical tools are unsuccessful ([44, 51]).

Genetic Algorithms (GAs) are search algorithms that use operations found in natural genetics to guide the trek through a search space. GAs are theoretically and empirically proven to provide robust search capabilities in complex spaces, offering a valid approach to problems requiring efficient and effective searching ([16, 28]).

Recently, numerous papers and applications combining fuzzy concepts and GAs have appeared, and there is 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).

The automatic definition of an FS can be considered in many cases as an optimization or search process. 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. GAs are known to be capable of finding near optimal solutions in complex search spaces. A priori knowledge may be in the form of linguistic variables, fuzzy membership function parameters, fuzzy rules, number of rules, etc. The generic code structure and independ-

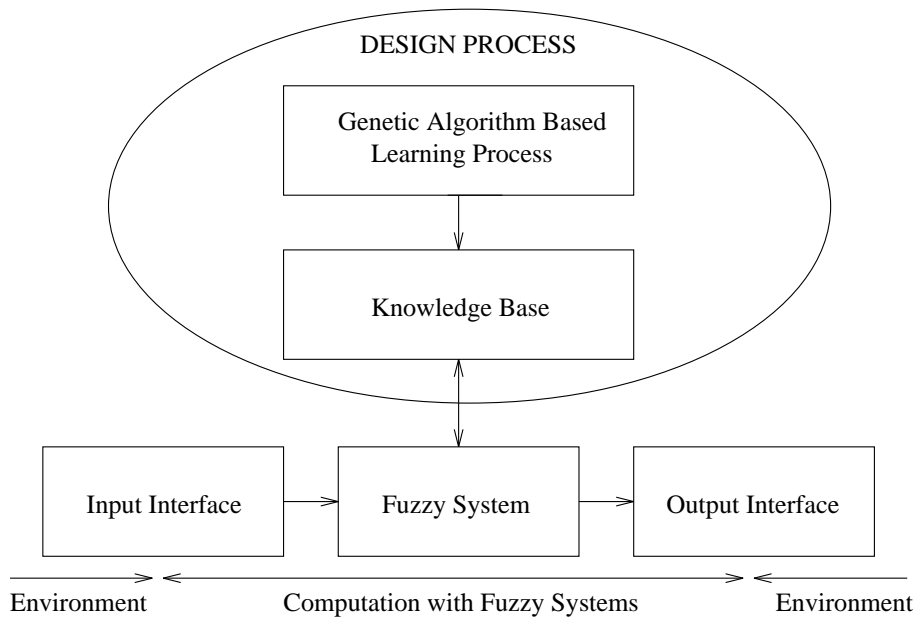


Figure 1: Genetic Fuzzy Systems

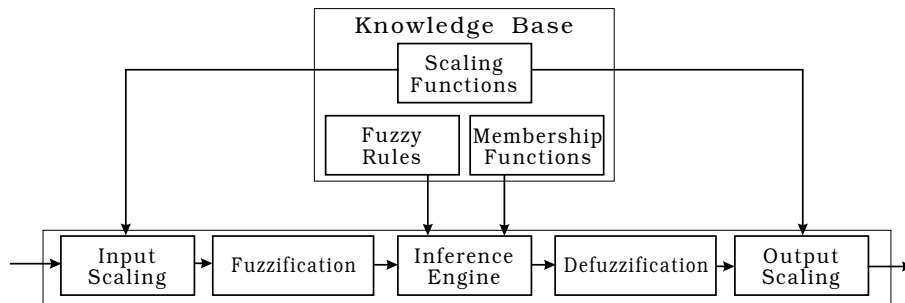


Figure 2: Structured Knowledge Base

ent performance features of GAs make them suitable candidates for incorporating a priori knowledge. These advantages have extended the use of GAs in the development of a wide range of approaches for designing fuzzy systems over the last few years. Figure 1 shows this idea.

We shall center our presentation on Fuzzy Rule Based Systems (FRBSs), [1], the most extended FS model to which the most successful application of FSs belong, the fuzzy logic controllers (FLCs), which have been and are used in many real-world control problems ([12]).

As is well known, the Knowledge Base (KB) of an FRBS is comprised of two components, a *Data Base* (DB), containing the definitions of the scaling factors and the membership functions of the fuzzy sets specifying the meaning of the linguistic terms, and a *Rule Base* (RB), constituted by the collection of fuzzy rules. Figure 2 shows the structure of a KB integrated in an FS with fuzzification and defuzzification modules, as used in fuzzy control.

GAs may be applied to adapting/learning the DB and/or the RB of an FRBS. This tutorial will summarize and analyze the GFSS,

paying a special attention to FRBSs incorporating tuning/learning through GAs.

With this aim in mind, the paper is divided into 7 Sections, the first being this introduction. Section 2 introduces GAs and learning with GAs, Section 3 presents some characteristics of genetic fuzzy rule based systems, Section 4 explores the topic of genetic tuning in FRBSs, Section 5 overviews the genetic learning processes in FRBSs, Section 6 presents an example of a GFS, and finally, some concluding remarks are commented upon in Section 7.

## 2 Genetic Algorithms

GAs are general purpose search algorithms which use principles inspired by natural genetics to evolve solutions to problems ([16, 28]). The basic idea is to maintain a population of chromosomes, which represents candidate solutions to the concrete problem being solved, that evolves over time through a process of competition and controlled variation. Each chromosome in the population has an associated *fitness* to determine (*selection*) which chromosomes are used to form new ones in the competition process. The new ones 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 for a great part of this success is their ability to exploit the information accumulated about an initially unknown search space in order to bias subsequent searches into useful subspaces, i.e., *their adaptation*. This is their key feature, particularly in large, complex, and poorly understood search spaces, where classical search tools (enumerative, heuristic . . .) are inappropriate, offering a valid approach to problems requiring efficient and effective search techniques.

A GA starts off with a population of randomly generated *chromosomes*, and advances toward better *chromosomes* by applying genetic operators modeled on the genetic processes occurring in nature. The population undergoes evolution in a form of natural selection. During successive iterations, called *generations*, chromosomes in the population are rated for their adaptation as solutions, and on the basis of these evaluations, a new population of chromosomes is formed using a selec-

tion mechanism and specific genetic operators such as crossover and mutation. An *evaluation* or *fitness function* ( $f$ ) must be devised for each problem to be solved. Given a particular chromosome, a possible solution, the fitness function returns a single numerical fitness, which is supposed to be proportional to the utility or adaptation of the solution represented by that chromosome.

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

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

Figure 3 shows the structure of a basic GA, where  $P(t)$  denotes the population at generation  $t$ .

### 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 3: Structure of a GA

### 2.1 Main characteristics of GAs

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

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

The basic principles of GAs were first laid down rigorously by Holland ([28]), and are well described in many books, such as [16, 40]. It is generally accepted that the application of a GA to solve a problem must take into account the following five components:

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

Some of these components will be described or analyzed below based on the ideas contained in previous paragraphs.

### **2.1.1 Representation and evaluation of solutions**

Representation is a key issue when applying GAs because they directly manipulate a coded representation of the problem and, consequently, the representation schema can severely limit the window through which a genetic system observes its world.

Additionally, the search process involved when applying GAs to solve a problem is driven or biased by the concept of utility or adaptation of the individuals as solutions to that problem. A fitness function must be devised for each problem in such a way that given a particular chromosome, a solution, the fitness function returns a single numerical fitness, which is supposed to be proportional to (to evaluate) the utility or adaptation of the individual which that chromosome represents.

### **2.1.2 Selection Mechanism**

Let's consider  $P$ , a population with chromosomes  $C_1, \dots, C_N$ . The selection mechanism produces an intermediate population,  $P'$ , with copies of chromosomes in  $P$ . The number of copies received from each chromosome depends on its fitness (the evaluation described in the previous paragraph), chromosomes with higher fitness usually have a greater chance of contributing copies to  $P'$ . There are a number of ways of making this selection. We might view the population as mapping onto a roulette wheel, where each chromosome is represented by a space that proportionally corresponds to its fitness. By repeatedly spinning the roulette wheel, chromosomes are chosen using "stochastic sampling with replacement" to fill the intermediate population. The selection procedure called *stochastic universal sampling* is one of the most efficient, where the number of offspring of any structure is bound by the floor and ceiling of the expected number of offspring.

### **2.1.3 Recombination through Crossover and Mutation**

After selection has been carried out the construction of the intermediate population is complete, then the genetic operators, crossover and mutation, can be applied.

**Recombination through Crossover.** The crossover operator is a method for sharing information between chromosomes; it combines the features of two parent chromosomes to form two offsprings, with the possibility that the offsprings generated through recombination are better adapted than their parents. The crossover operator is not usually applied to all pairs of chromosomes in the intermediate population. A random choice is made, where the likelihood of crossover being applied depends on probability defined by a crossover rate, the crossover probability ( $p_c$ ). The crossover operator plays a central role in GAs, in fact it may be considered to be one of the algorithm's defining characteristics, and it is one of the components to be borne in mind to improve the GA behavior. Definitions for this operator (and the next one) are highly dependent on the particular representation chosen.

**Mutation.** A 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 ( $p_m$ ).

We should point out that after crossover and mutation, an additional selection strategy, called *elitist strategy*, may be adopted to make sure that the best performing chromosome always survives intact from one generation to the next. This is necessary since it is possible that the best chromosome disappears thanks to crossover or mutation.

## 2.2 Learning with GAs

Although GAs are not learning algorithms, they may offer a powerful and domain-independent search method for a variety of learning tasks. In fact, there has been a good deal of interest in using GAs for machine learning problems ([21]).

Three alternative approaches, in which GAs have been applied to learning processes, have been proposed, the Michigan, the Pittsburgh, and the Iterative Rule Learning (IRL) approaches. In the first one, the chromosomes correspond to classifier rules which 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 learning, but contrary to the first, only the best individual is considered as the solution, discarding the remaining chromosomes in the population.

Below, we will describe them briefly.

### 2.2.1 The Michigan approach

The chromosomes are individual rules and a rule set is represented by the entire population. The collection of rules are modified over time via interaction with the environment. This model maintains the population of classifiers with credit assignment, rule discovery and genetic operations applied at the level of the individual rule.

There is a considerable variety in the structural and functional details of this model. The prototype organization is composed of three parts:

1. the *performance system* that interacts with the environment and contains the rule base and the production system,
2. the *credit assignment system* or credit apportionment system, developing learning by the modification and adjustment of conflict-resolution parameters of the classifier (rule) set, their strengths; Holland's Bucket Brigade is one example of this ([29]), and
3. the *classifier discovery system* that generates new classifiers, rules, from a classifier set by means of GAs.

A genetic learning process based on the Michigan approach receives the name of Classifier System (CS). The prototypical organization of a CS is illustrated on Figure 4. A complete description is to be found in [5].

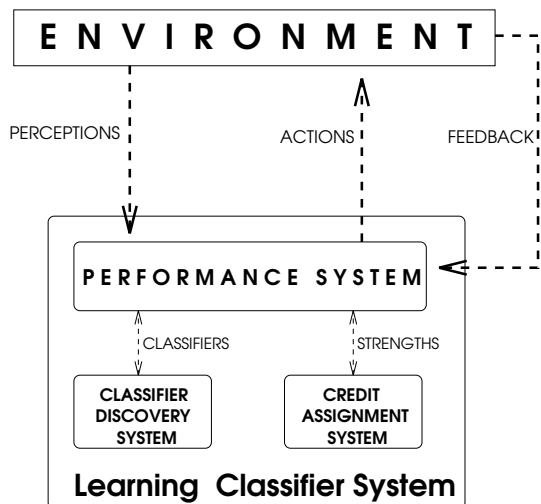


Figure 4: Organization of a classifier system

### 2.2.2 The Pittsburgh approach

Each chromosome encodes a whole RB or KB. Crossover serves to provide a new combination of rules and mutation provides new rules. In some cases, variable-length rule bases are used, employing modified genetic operators for dealing with these variable-length and position independent genomes.

Recent instances of this approach are to be found in [21].

### 2.2.3 Iterative Rule Learning approach

In this latter model, as in the Michigan one, each chromosome in the population represents a single rule, but contrary to the Michigan one, only the best individual is considered to form part of the solution, discarding the remaining chromosomes in the population. Therefore, in the iterative model, the GA provides a partial solution to the problem of learning. In order to obtain a set of rules, which will be a true solution to the problem, the GA (with a structure similar to the one described in Figure 3) has to be placed within an iterative scheme similar to the following:

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

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

This way of learning is to allow "niches" and "species" formation. Species formation seems particularly appealing for concept learning, considering the process as the learning of multimodal concepts.

The main difference with respect to the Michigan approach is that the fitness of each chromosome is computed individually, without taking into account cooperation with other ones. This substantially reduces the search space, because in each sequence of iterations only one rule is searched.

A more detailed description of this approach may be found in [20].

### 2.2.4 Conclusions

The Michigan approach will prove to be the most useful in an on-line process. It is more flexible to handle incremental-mode learning (training instances arrive over time)

and dynamically changing domains, whereas the Pittsburgh and the IRL approaches seem to be better suited to batch-mode learning, where all training instances are available before learning is initiated, and for static domains.

The major problem in the Michigan approach is that of resolving the conflict between the individual and collective interests of classifiers within the system. The ultimate aim of a learning classifier system is to evolve a set of co-adapted rules which act together in solving some problems. In a Michigan style system, with selection and replacement at the level of the individual rule, rules which cooperate to effect good actions and receive payoff also compete with each other under the action of the GA.

This conflict between individual and collective interests of individual classifiers does not arise with Pittsburgh-style classifier systems, since reproductive competition occurs between complete rule sets rather than individual rules. However, maintenance and evaluation of a population of complete rule-sets in Pittsburgh-style systems can often lead to a much greater computational burden (in terms of both memory and processing time). Therefore, problems with the Pittsburgh approach have proven to be, at least, equally as challenging. Although the approach avoids the problem of explicit competition between classifiers, large amounts of computing resources are required to evaluate a complete population of rule-sets.

When compared with the latter, the advantage of the IRL approach is that, in the first stage space it considerably reduces the search because it looks for only one rule in each sequence of iterations, although this approach also implies a great computational burden.

On the other hand, GAs are also used for refining parameters in other learning approaches, as is done using GAs for determining weights in a neural network.

## 3 Genetic Fuzzy Rule Based Systems

The idea of a Genetic FRBS is that of a genetic FRBS design process which incorporates genetic techniques to achieve the automatic

generation or modification of its KB (or a part of it). This generation or modification usually involves a tuning/learning process, and consequently this process plays a central role in GFSs. The objective of this tuning/learning process is optimization, i.e., maximizing or minimizing a certain function representing or describing the behavior of the system.

It is possible to define two different groups of optimization problems in FRBSs. The first group contains those problems where optimization only involves the behavior of the FRBS, while the second one refers to those problems where optimization involves the global behavior of the FRBS and an additional system. The first group contains problems such as modeling, classification, prediction and, in general, identification problems. In this case, the optimization process searches for an FRBS able to reproduce the behavior of a certain target system. The most representative problem in the second group is control, where the objective is to add a FRBS to a controlled system in order to obtain a certain overall behavior. Next, we analyze some aspects of the Genetic FRBSs.

### 3.1 Obtaining the knowledge for an FRBS

As a first step, it is interesting to distinguish between tuning and learning problems. In tuning problems, a predefined RB is used and the objective is to find a set of parameters defining the DB. In learning problems, a more elaborate process including the modification of the RB is performed.

We can distinguish between three different groups of GFSs depending on the KB components included in the genetic learning process.

For an extensive bibliography see [8] (section 3.13), some approaches may be found in [25].

#### 3.1.1 Genetic tuning of the DB

The tuning of the scaling functions and fuzzy membership functions is an important task in the design of fuzzy systems. It is possible to parameterize the scaling functions or the membership functions and adapt them using GAs to deal with their parameters according to a fitness function.

As regards to the tuning of membership functions, several methods have been pro-

posed in order to define the DB using GAs. Each chromosome involved in the evolution process represents different DB definitions, i.e., each chromosome contains a coding of the whole set of membership functions giving meaning to the linguistic terms. Two possibilities can be considered depending on whether the fuzzy model nature is descriptive or approximate, either to code the fuzzy partition maintaining a linguistic description of the system, or to code the rule membership functions tuning the parameters of a label locally for every rule, thereby obtaining a fuzzy approximate model.

#### 3.1.2 Genetic learning of the RB

All the methods belonging to this family involve the existence of a predefined collection of fuzzy membership functions giving meaning to the linguistic labels contained in the rules, a DB. On this basis GAs are applied to obtain a suitable rule base, using chromosomes that code single rules or complete rule bases.

#### 3.1.3 Genetic learning of the KB

There are many approaches for the genetic learning of a complete KB (RB and DB). We may find approaches presenting variable chromosome lengths, others coding a fixed number of rules and their membership functions, several working with chromosomes encoding single control rules instead of a complete KBs, etc.

### 3.2 The keys to the tuning/learning process

Regardless of the kind of optimization problem, i.e., given a system to be modeled/controlled (hereafter we use this notation), the involved tuning/learning process will be based on evolution. Three points are the keys to an evolutionary based tuning/learning process. These three points are: the population of potential solutions, the set of evolution operators and the performance index.

**The population of potential solutions.** The learning process works on a population of potential solutions to the problem. In this case, the potential solution is an FRBS. From

this point of view, the learning process will work on a population of FRBSs, but considering that all the systems use an identical processing structure, the individuals in the population will be reduced to DB/RB or KBs. In some cases the process starts off with an initial population obtained from available knowledge, while in other cases the initial population is randomly generated.

**The set of evolution operators.** The second question is the definition of a set of evolution operators that search for new and/or better potential solutions (KBs). The search reveals two different aspects: the exploitation of the best solution and the exploration of the search space. The success of evolutionary learning is specifically related to obtaining an adequate balance between exploration and exploitation, that finally depends on the selected set of evolution operators.

The new potential solutions are obtained by applying the evolution operators to the members of the population of knowledge bases, each one of these members is referred to as an individual in the population. The evolution operators, that work with a code (called a chromosome) representing the KB, are basically three: selection, crossover and mutation. These evolution operators are in depth analyzed in subsections 2.1.2 and 2.1.3.

Since these evolution operators work in a coded representation of the KBs, a certain compatibility between the operators and the structure of the chromosomes is required. This compatibility is stated in two different ways: work with chromosomes coded as binary strings (adapting the problem solutions to binary code) using a set of *classical* genetic operators, or adapt the operators to obtain compatible evolution operators using chromosomes with a non-binary code. Consequently, the question of defining a set of evolution operators involves defining a compatible couple of evolution operators and chromosome coding.

**The performance index.** Finally, the third question is that of designing an evaluation system capable of generating an appropriate performance index related to each individual in the population, in such a way that a better solution will obtain a higher performance index. This performance index will drive the optimization process.

In identification problems, the performance index will usually be based on error measures that characterize the difference between the desired output and the actual output of the system. In control problems there are two different sources of information to be used when defining the performance index: information describing the desired behavior of the controlled system, or describing the desired behavior of the controller (FRBS) itself. The second situation is closely related to identification problems. The definition of a performance index is usually more complex for the first situation, where the objective is to find a controller that gives the desired behavior in the controlled system. A possible method is illustrated in subsection 3.3.

**The process.** Summarizing the points that characterize a specific learning process, these are: the initial population of solutions (obtained randomly or from some initial knowledge), the coding scheme for KBs (chromosomes), the set of evolution operators and the evaluation function. The initial population and the evaluation function are related to the specific problem while the coding scheme and the evolution operators could be generic. In addition to these four points, each evolutionary learning process is characterized by a set of parameters such as the dimension of the population (fixed or variable), the parameters regulating the activity of the operators or even their effect, and the parameters or conditions defining the end of the process or the time when a qualitative change in the process occurs.

### 3.3 A learning process of FLCs

FLCs represent a particular and widely applied kind of FRBSs. A genetic process using a Pittsburgh approach and working on an FLC is illustrated in Figure 5.

The process described in Figure 3 may be rewritten as follows in such a situation:

1. Start with an initial population of solutions that constitutes the first generation ( $P(0)$ ).
2. Evaluate  $P(0)$ :
  - (a) take each chromosome (KB) from the population and introduce it into the FLC,

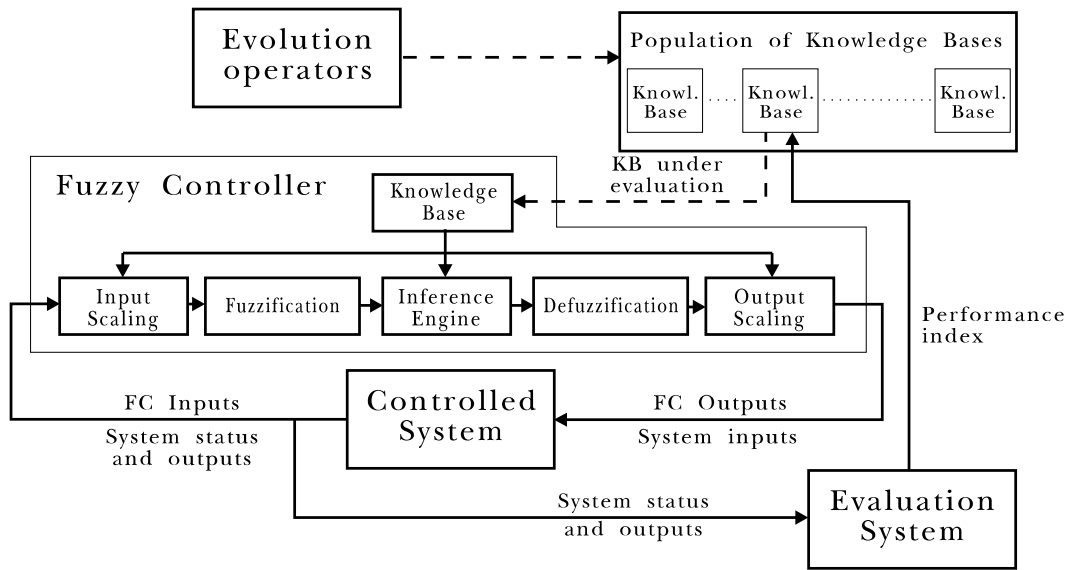


Figure 5: Evolutionary learning, with Pittsburgh approach, of the KB of an FLC

- (b) apply the FLC to the controlled system for an adequate evaluation period (a single control cycle, several control cycles or even several times, starting out from different initial conditions) and
  - (c) evaluate the behavior of the controlled system by producing a performance index related to the KB.
3. While the Termination Condition is not met, do
    - (a) create a new generation ( $P(t+1)$ ) by applying the evolution operators to the individuals in  $P(t)$ ,
    - (b) evaluate  $P(t+1)$  and
    - (c)  $t = t + 1$ .
  4. Stop.

### 3.4 The cooperation vs. competition problem

A GFS combines the main aspects of the system to be obtained, an FS, and the design technique used to obtain it, a GA, with the aim of improving as far as possible the accuracy of the final FS generated.

One of the most interesting features of an FS is the interpolative reasoning it develops.

This characteristic plays a key role in the high performance of FSs and is a consequence of the *cooperation between the fuzzy rules composing the KB*. As is known, the output obtained from an FS 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 between 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 an FS, because the different members of the population compete with one another to provide a final solution presenting the best cooperation between 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 (CCP)* ([2]).

The difficulty of solving the introduced problem depends directly on the genetic learning approach followed by the GFS (Michigan,

Pittsburgh or IRL approaches). Below we briefly analyze them.

### 3.4.1 Michigan approach

It is difficult to solve the CCP when working with the Michigan approach. In this case, the evolution is performed at the level of fuzzy rules instead of at the level of KBs and it is not easy to obtain an adequate cooperation between fuzzy rules that are competing with one another. To do this, we need a fitness function able to measure both the goodness of a single fuzzy rule and the quality of its cooperation with the other fuzzy rules in the population to give the best action as output. As mentioned in [2], the design of a fitness function of this kind is not an easy task.

### 3.4.2 Pittsburgh approach

This approach is able to solve adequately the CCP. When using this approach, the GFS evolves populations of KBs and the fitness function associated to each individual is computed taking into account the real action that the FS encoded into the chromosome should give as output when it receives a concrete input. Thus, each time an individual is evaluated, the cooperation between the fuzzy rules composing the KB is measured, so the GFS is able to evolve adequately the population to obtain the FS presenting the best possible cooperation between the fuzzy rules composing its KB.

Unfortunately, this approach presents the drawback of having to deal with very large search spaces, which makes it difficult to find optimal solutions. This drawback is usual when designing GFSs belonging to the third family, i.e., when the generation of the whole KB is considered in the genetic learning process. In this case, a large quantity of KB parameters have to be included in the genetic representation, which therefore becomes larger. This fact will be more pronounced if an approximate fuzzy model is considered, the use of different membership function definitions for each rule makes the number of KB parameters increase, and then the search space becomes more complex, making the problem computationally hard.

### 3.4.3 IRL approach

Finally, GFSs based on the IRL approach try to solve the CCP at the same time reducing the search space by encoding a single fuzzy rule in each chromosome. To put this into effect, these processes follow the usual problem partitioning working way and divide the genetic learning process into, at least, two stages. Therefore, the CCP is solved in two steps acting at two different levels, with the competition between fuzzy rules in the first one, the genetic generation stage, and with the cooperation between these generated fuzzy rules in the second one, the post-processing stage.

## 4 Genetic Tuning of DB

As mentioned earlier (subsection 3.1.1) the use of GAs for the tuning of DBs may be developed in two areas, the adaptation of contexts using scaling functions and the tuning of fuzzy membership functions. Below we shall present briefly them.

### 4.1 Adapting the context

The use of scaling functions that are applied to the input and output variables of an FRBS, allows us to work with normalized universes of discourse where the fuzzy membership functions are defined. These scaling functions could be interpreted as gains associated with the variables (from a control engineering point of view) or as context information that translates relative semantics into absolute ones (from a knowledge engineering point of view). If using scaling functions, it is possible to fix them or to parameterize the scaling functions and adapt them. Linear and non-linear contexts have been used.

**Linear context.** It is the simplest scaling. The parameterized function is defined by means of two parameters (one, if used as a scaling factor). The effect of scaling is that of linearly mapping the real interval  $[a,b]$  into a reference interval (e.g.,  $[0,1]$ ). The use of a scaling factor maps the interval  $[-a,a]$  in a symmetrical reference interval (e.g.,  $[-1,1]$ ). This kind of context is the most broadly applied one. Genetic techniques have been applied

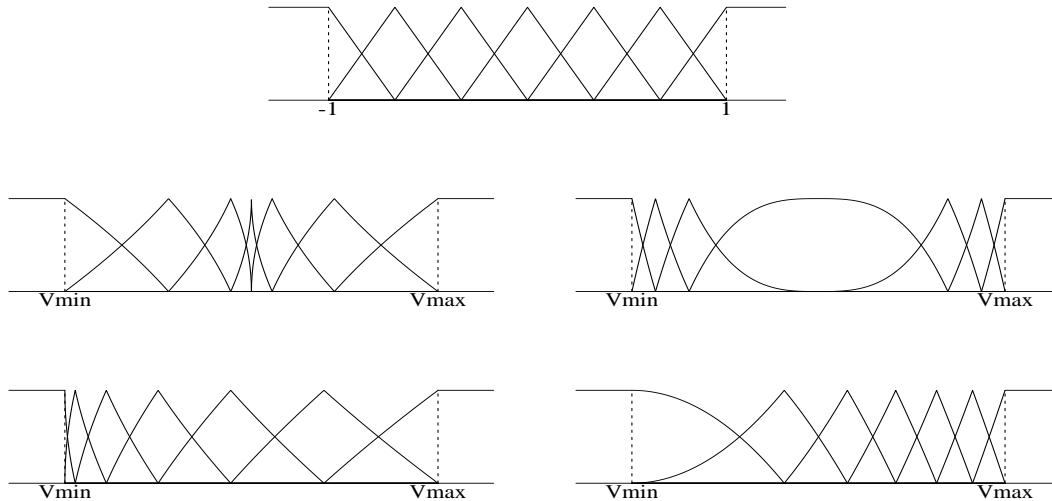


Figure 6: Nonlinear contexts adaption

to adapting the parameters defining the scaling factors ([41]) and linear scaling functions ([38]).

**Nonlinear context.** The main disadvantage of linear scaling is the fixed relative distribution of the membership functions (uniformly distributed or not) once they have been generated. To solve this problem nonlinear scaling is used allowing us to obtain a modified relative distribution and a change in the shape of the membership functions. The definition of parameterized nonlinear scaling functions is more complex than in the linear case and a larger number of parameters are needed. The process actually requires two steps: previous scaling (linear) and nonlinear mapping. Parameterized potential ([36]) and sigmoidal ([22]) functions have been used when applying GAs to adapt the nonlinear context. Usually, the parameters (real numbers) constitute the genes of the chromosomes without binary representation.

Figure 6 shows a normalized fuzzy partition (top), a nonlinear adaption with lower granularity for middle or for extreme values (center) and lower granularity for lowest or for highest values (bottom).

## 4.2 Tuning the membership functions

Another element of the KB is the set of membership functions. This is a second point where GAs could be applied with a tuning purpose. As in the previous case of scaling functions, the main idea is the definition of parameterized functions and the subsequent adaptation of parameters. Some approaches are found to be in [4, 3, 24, 30, 48]. The different proposals differ in the coding scheme and the management of the solutions (fitness functions, ...).

### 4.2.1 Shape of the membership functions

Two main groups of parameterized membership functions have been proposed and applied: piecewise linear functions and differentiable functions.

**Piecewise linear functions.** The most broadly used parameterized membership functions in the field of GFSs are triangles, in some cases these are isosceles ([7, 13, 31, 42]) and other times they are irregular ([33]). A second possibility are trapezoidal membership functions ([32]).

Each parameter of the function consti-



are the fuzzy rule membership functions, i. e., the DB definition.

Taking into account a parametric representation with triangular-shaped membership functions based on a 3-tuple of real values, each rule

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

$$\text{THEN } y \text{ is } B_i,$$

of a certain KB ( $KB_l$ ), is encoded in a piece of chromosome  $C_{li}$ :

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

where  $A_{ij}$ ,  $B_i$  have the parametric representation  $(a_{ij}, b_{ij}, c_{ij})$ ,  $(a_i, b_i, c_i)$ ,  $i = 1, \dots, m$  ( $m$  represents the number of rules),  $j = 1, \dots, n$  ( $n$  is the number of input variables).

Therefore the complete RB with its associated DB is represented by a complete chromosome  $C_l$ :

$$C_l = C_{l1} C_{l2} \dots C_{lm}$$

This chromosome may be a binary or a real coded individual.

#### 4.2.4 The descriptive genetic tuning process

In this second genetic tuning process each chromosome encodes a different DB definition based on the fuzzy domain partitions. A primary fuzzy partition is represented as an array composed by  $3 \cdot N$  real values, with  $N$  being the number of terms forming the linguistic variable term set. The complete DB for a problem, in which  $m$  linguistic variables are involved, is encoded into a fixed length real coded chromosome  $C_j$  built up by joining the partial representations of each one of the variable fuzzy partitions,

$$C_{ji} = (a_{i1}, b_{i1}, c_{i1}, \dots, a_{iN_i}, b_{iN_i}, c_{iN_i})$$

$$C_j = C_{j1} C_{j2} \dots C_{jm}$$

where  $C_{ji}$  represents the fuzzy partition corresponding to the  $i$ -th variable.

## 5 Learning with GFSs

In this Section, we study the learning of FRBSs by means of genetic learning processes, dividing it into two subsections, the learning

of RB and the learning of KB, and presenting some notes on the application of the different genetic approaches to these problems.

We must denote that we present a more wide study of the Pittsburgh approach, the most used approach for learning fuzzy systems.

### 5.1 Genetic learning of RB

The third element of the KB is the RB. It is possible to represent the RB of a fuzzy controller with three different representations and all of them are used in evolutionary fuzzy controllers. These representations are: relational matrix, decision table and list or set of rules.

Genetic learning of RB makes sense only when working with a descriptive approach, since in the approximate approach, modifying the rules implies the modification of membership functions.

#### 5.1.1 Michigan approach

A Michigan learning algorithm of fuzzy rules merges the credit assignment mechanisms of CSs and fuzzy systems, integrating a fuzzy rule base (each classifier represents a fuzzy rule and the population represents the RB) and a fuzzy inference system instead of the rule base and production system in a classical CS. This learning process receives the name of Fuzzy Classifier Systems (FCSs).

Some peculiarities of this model are that a fuzzification process is defined in the input interface, which fuzzifies inputs into fuzzy messages by creating minimal messages, one for each fuzzy set defined over the variable. Then each message has an associated activity level which measures the degree of belonging to the input variable defined by the fuzzy sets represented by the message. For each rule, a previously fixed credit is given. As a result of actions performed in the environment by the fuzzy inference system, the credit assignment system receives the payoff of that action from the environment, and in accordance with the degree of conformity of the rules, the payoff is apportioned to each rule by increasing or decreasing its credit. Therefore, the system learns fuzzy relations between the fixed fuzzy sets.

Valenzuela-Rendón gave the first description of an FCS in [50]. Figure 8 ([14]) shows

the organization of an FCSs. Details about this approach may be found in [2, 14, 15, 50].

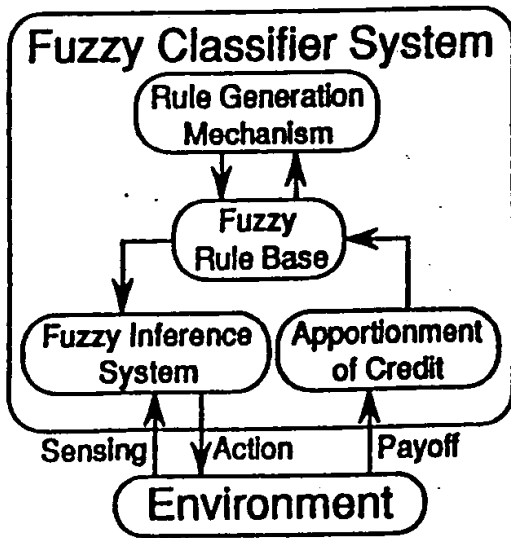


Figure 8: Organization of a fuzzy classifier system

### 5.1.2 Pittsburgh approach

The Pittsburgh approach has been applied to learn rule bases in two different situations. The first situation refers to those systems using a complete rule base represented by means of a decision table or a relational matrix. The second situation is that of FRBSs, whose RB is represented using a list or set of fuzzy rules.

**Using a complete RB.** A tabular representation guarantees the completeness of the knowledge of the FRBS in the sense that the coverage of the input space (the Cartesian product of universes of the input variables) is only related to the level of coverage of each input variable (the corresponding fuzzy partitions), and not to the rules.

**Decision tables.** A possible representation for the RB of an FS is a decision table. It is a classical representation used in different GFSs. A chromosome is obtained from the decision table by going row-wise and coding each output fuzzy set as an integer or any other kind of label. It is possible to include the “no output” definition in a certain position, using a “null” label ([41, 49]).

**Relational matrices.** Occasionally GAs are used to modify the fuzzy relational matrix (R) of a Fuzzy System with one input and one output. The chromosome is obtained by concatenating the  $m \times n$  elements of R, where  $m$  and  $n$  are the number of fuzzy sets associated with the input and output variables respectively. The elements of R that will make up the genes may be represented by binary codes [45] or real numbers.

**Using a partial RB.** Neither the relational nor the tabular representations are adaptable to systems with more than two or three input variables because of the dimension of a complete RB for these situations. This fact stimulated the idea of working with sets of rules. In a *set of rules* representation the absence of applicable rules for a certain input that was perfectly covered by the fuzzy partitions of individual input variables is possible. As a counterpart to the loss of completeness, this representation allows *compressing* several rules with identical outputs into a singular rule and this is a really important question as the dimension of the system grows.

There are many different methods for coding the rule base in this kind of evolutionary system. The code of the rule base is usually obtained by concatenating rules codes.

**Rules of fixed length.** A first approach is to represent a rule with a code of fixed length and position dependent meaning. The code will have as many elements as the number of variables in the system. A possible content of these elements is: a label pointing to a certain fuzzy set in the fuzzy partition of the variable or a binary string with a bit per fuzzy set in the fuzzy partition of the variable coding the presence or absence of the fuzzy set in the rule [39].

**Rules of variable length.** Codes with position independent meaning and based on pairs  $\{variable, membership\ function\}$  (the membership functions is described using a label) are used in [27].

### 5.1.3 Learning an RB with the IRL approach

Using this approach a chromosome represents a fuzzy rule and the whole rule base is ob-

tained through an iterative process where individual rules are obtained at each iteration based on a Genetic process.

From the description given in subsection 2.2.3, we may see that in order to learn rules using an algorithm based on GAs with an IRL approach, we need, at least, the following:

1. a criterion for selecting the best rule at each iteration,
2. a penalization criterion, and
3. a criterion for determining when enough rules are available to have a solution to the problem.

The first criterion is normally associated with one or several characteristics that are desirable so as to determine good rules. Usually criteria about the rule strength have been proposed (number of examples covered), criteria of consistency of the rule or criteria of simplicity.

The second criterion is often associated, although it is not necessary, with the elimination of the examples covered by the previous rules.

Finally, the third criterion is associated with the completeness of the set of rules and must be taken into account when we can say that all the examples in the training set are sufficiently covered and no more rules are needed to represent them.

The IRL approach does not analyze any relationship between the rules that are obtained. That is why, once the rule base has been obtained, it may be improved either because there are rules that may be refined or redundant rules if high degrees of coverage are used. Therefore, after this is done, some post-processing methods are used for improving the accuracy of the rule base.

An inductive learning algorithm for RB, called SLAVE has been designed based on this approach [17, 18, 19]. SLAVE selects a rule covering the maximum number of positive examples and simultaneously verifies a soft consistency condition to a high degree [18]. SLAVE uses a GA in this process. The details of the GA are to be found in [19].

## 5.2 Genetic learning of KB

The simultaneous use as genetic material of the DB and the RB of an FS has produced different and interesting results.

### 5.2.1 Michigan approach

Parodi and Bonelli, [43], present an FCS that learns KBs (RB and DB) and output weights. The two main points of the system are the rule syntax and semantics, and the determination of output weights.

Each rule  $R_k$  contains the actual description of the membership functions that correspond to each variable, considering symmetrical fuzzy sets and having two parameters, the center position and the width, where the center position is determined through a GA search and the width for each variable is a system parameter. A real coding for rules is used.

The classifier population consists of a fixed size list of such rules. Each rule has associated with it an output weight which is set equal to the strength (fitness) of the individuals (rules) in the population. The rule strength performs a dual function: firstly it forms the basis of selection and replacement for the GA and secondly it allows stronger rules to take a greater part than weaker ones in decision making.

The "measure of goodness" function is designed to not only determine the quality of the rule's conclusion fuzzy set, but also its accuracy in predicting through the matching between the input vector and the rules how well the rule's condition intervals are set for this particular input.

### 5.2.2 Pittsburgh approach

The most general approach is the use of a set of parameterized membership functions and a list of fuzzy rules that are jointly coded to generate a chromosome, then applying a Pittsburgh-type GA to evolve a population of such chromosomes. This kind of GFSs use chromosomes containing two subchromosomes that encode separately, but not independently, the DB and the RB.

It is possible to maintain, at this point, the same division that was stated when talking about genetic learning of RBs with a Pittsburgh approach: learning complete rule bases or partial rule bases.

**Using a complete RB.** In [42] the rule base is represented as a fuzzy relation matrix (R), and the GA modifies R or the fuzzy membership functions (triangular) or both of

them simultaneously, on an FLC with one input and one output variables. Each gene is a real number. When generating the optimal fuzzy relation matrix this real number corresponds to a fuzzy relation degree whose value is between 0 and 1. The genetic string is obtained by concatenating the  $m \times n$  real numbers that constitute R. When finding simultaneously the optimal rule base and the fuzzy membership functions, each chromosome allocates two subchromosomes: the genes of the rule base and the genes of the fuzzy membership functions. Both subchromosomes are treated as independent entities as far as crossover and mutation are concerned but as a single entity as far as reproduction is concerned.

A slightly different approach is to use a TSK-type rule base, structuring its genetic code as if it came from a decision table. In this case, the contents of the code of a rule base is an ordered and complete list containing the consequents of all possible rules, where the antecedents are implicitly defined as a function of the position the consequent occupies in the list.

The fuzzy membership functions constitute a first subchromosome while the coefficients of the consequents for a TSK fuzzy model constitute the second subchromosome. One gene is used to code each coefficient of a TSK-type in [33], in [47] a single coefficient is considered for the output.

**Using a partial RB.** Liska and Melsheimer ([34]) use a rule base defined as a set of a fixed number of rules, and code each rule with integer numbers that define the membership function related with a certain input or output variable that is applied by the rule (membership functions for every variable are ordered). The systems use radial membership functions coded through two real numbers (two genes). The genetic string is obtained by concatenating the two genes in each membership function.

There are many different methods for coding the rule base in this kind of evolutionary system. The code of the rule base is usually obtained by concatenating rule codes. To represent a single rule, it is possible to use a position dependent code with as many elements as the number of variables of the system. A possible content in these elements is: a label

pointing to a certain fuzzy set in the fuzzy partition of the variable ([46]) or a binary string with a bit per fuzzy set in the fuzzy partition of the variable ([35]).

Using an approximate approach, [6, 7] include the definition of the membership functions into the rules, coding each rule through the corresponding set of membership functions.

### 5.2.3 Learning KB process based in the IRL approach

Below we introduce a multi-stage GFS structure for learning KB based on this approach. This is designed according to three stages that we now introduce.

**a) A fuzzy rule generation process.** This process presents two components:

- A *fuzzy rule generating method*, where a chromosome represents a fuzzy rule, and the GA finds the best rule in every running  $i$  from the examples, according to different features that are included in the fitness function. These features allow the adequate membership functions to be determined.
- An *iterative covering method* of the system behavior example set, which penalizes each rule generated with the fuzzy rule generating method by considering its covering over the examples in the training set and removes the ones already covered by it. This process allows us to obtain a set of fuzzy rules with a concrete semantic covering the training set in an adequate form.

**b) A genetic simplification process** for selecting rules, based on a binary coded GA with a phenotypic sharing function and a measure of the FRBS accuracy in the problem being solved. It will save the overlearning that the previous component may cause due to the existence of redundant rules, with the aim of obtaining a simplified KB presenting the best possible cooperation among the fuzzy rules that compose it.

**c) A genetic tuning process.** It will give the final KB as output by adjusting the mem-

bership functions for each fuzzy rule in each possible KB obtained from the stage above.

A more complete description of the multi-stage GFSs may be found in [20]. Different multi-stage GFSs for learning KB following these ideas may be found in [10, 11, 23, 26].

## 6 An example of GFS

This section will describe, in a few lines, one of the GFSs previously cited, specifically a GFS learning RBs using a Pittsburgh approach and representing the rule base with a decision table. This method was proposed by Philip Thrift ([49]). This example will be analyzed according to the keys of the learning process described in subsection 3.2.

Given a single output FRBS with  $n$  input variables, a fuzzy partition is defined for each variable ( $n + 1$  fuzzy partitions). In this case each fuzzy partition contains five or seven fuzzy sets. An  $n$ -dimensional decision table is then made up by placing the consequents of each rule in the place corresponding to its premise. Entries in the table can be either one of the labels representing a fuzzy set of the output variable partition, or a blank representing no fuzzy set output for the corresponding rule.

### **The population of potential solutions.**

The population of potential solutions will be made up of RBs applied by a common processing structure to solve a specific problem. Because the learning process is centered on rules and all the KBs will contain an identical DB, consequently the population of solutions can be reduced to a population of RBs. Each RB is represented by a decision table, and these decision tables must be coded to constitute suitable genetic material.

Each position in the decision table will represent a gene of the chromosome coded with an integer in  $\{0, 1, \dots, 5\}$ , with its 6 possible values corresponding to the 5 components of the fuzzy partition and the blank output. A chromosome is obtained by going rowwise through the table and producing a string with the integers found at each place in it. For a system with two input variables and five fuzzy sets per partition, the decision table will contain  $5 \times 5$  places and consequently will generate a chromosome with 25 genes.

The population where the genetic process will be applied is a number of chromosomes (31 in the example described in the paper) coded as strings with 25 integers in  $\{0, 1, \dots, 5\}$ .

**The set of evolution operators.** The system uses a standard two point crossover ([40]) and a mutation operator that changes a fuzzy code either up one level or down one level, or to the blank code. When the mutation operator acts on a blank code, a non-blank code is generated at random. An elite strategy allows the best solution at a given generation to be directly promoted to the next.

**The performance index.** The system described is applied to center a cart by applying a force on it. The objective is to move the cart to the zero position and velocity in a minimum time. Each RB is tested by applying the FRBS to control the cart starting at 25 equally spaced starting points and over 500 steps (0.02 sc. for each step). The performance index assigned to an RB is  $500 - T$  where  $T$  is the average time (number of steps) required to place the cart sufficiently close to the center ( $\max(|x|, |v|) < 0.5$ ). If, for a certain starting point, more than 500 steps are required, the process times out and 500 steps are recorded.

With this performance index the learning process becomes a minimization problem since the best solution is the one with the lowest average time to center the cart (the highest performance index).

## 7 Concluding Remarks

In this tutorial we have dealt with several issues relating to GFSs, focusing on the use of GAs for designing FRBSs. We have presented the most important keys of the tuning/learning processes. The two genetic tuning approaches (adapting the context or the membership functions) and the three different modes to cope with the problem of designing RB or KBs with GAs (Michigan, Pittsburgh and IRL approaches) have been attached, and different proposals developing each one of them have been analyzed.

Finally, we should point out that although the application of GAs for designing fuzzy systems is recent, it has seen of increasing interest

over the last few years and will allow to fruitful research to be carried out in the building of fuzzy logic-based intelligent systems.

## References

- [1] A. Bárdossy, L. Duckstein. Fuzzy Rule-Based Modeling with Applications to Geophysical, Biological and Engineering Systems. CRC Press, 1995.
- [2] A. Bonarini. Evolutionary learning of fuzzy rules: competition and cooperation. In: W. Pedrycz (Ed.), Fuzzy Modeling: Paradigms and Practice. Kluwer Academic Press, 1996, 265-283.
- [3] P.P. Bonissone, P.S. Khedkar, Y. Chen. Genetic Algorithms for Automated Tuning of Fuzzy Controllers: A Train Handling Application. Proc. Fifth IEEE International Conference on Fuzzy Systems (FUZZ-IEEE'96), New Orleans, 1996, 675-680.
- [4] F. Bolata, A. Nowé. From fuzzy linguistic specifications to fuzzy controllers using evolution strategies. Proc. Fourth IEEE International Conference on Fuzzy Systems (FUZZ-IEEE'95), Yokohama (Japan, 1995) 1089-1094.
- [5] L.B. Booker, D.E. Goldberg, J.H. Holland. Classifier systems and genetic algorithms. Artificial Intelligence 40 (1989) 235-282.
- [6] B. Carse, T.C. Fogarty, A. Munro. Evolving fuzzy rule based controllers using genetic algorithms. Fuzzy Sets and Systems 80 (1996) 273-294.
- [7] M.G. Cooper, J.J. Vidal. Genetic design of fuzzy controllers. Proceedings 2nd International Conference on Fuzzy Theory and Technology, October 1993.
- [8] O. Cordón, F. Herrera, M. Lozano. A classified review on the combination fuzzy logic genetic algorithms bibliography. Tech. Report #*DECSAI* – 95129, Dept. of Computer Science and A.I., Univ. of Granada, 1995. Available at the URL address: <http://decsai.ugr.es/~herrera/flga.html>
- [9] O. Cordón, F. Herrera, M. Lozano. A three-stage method for designing genetic fuzzy systems by learning from examples. In: H.M. Voight, W. Ebeling, E. Rechemberg, H.P. Schwefel (Eds.), L.N.C.S. 1141, Proc. 4th International Conference on Parallel Problem Solving from Nature (PPSN IV), Berlin, 1996, 720-729.
- [10] O. Cordón, F. Herrera. A hybrid genetic algorithm-evolution strategy process for learning fuzzy logic controller knowledge bases. In: F. Herrera and J.L. Verdegay, (Eds.), Genetic Algorithms and Soft Computing, Physica Verlag, 1996, 251-278.
- [11] O. Cordón, F. Herrera. A three-stage evolutionary process for learning descriptive and approximate fuzzy logic controller knowledge bases from examples. International Journal of Approximate Reasoning (1997) To appear.
- [12] D. Driankov, H. Hellendoorn, M. Reinfrank. An Introduction to Fuzzy Control. Springer-Verlag, 1993.
- [13] B. Filipič, D. Juričić. A genetic algorithm to support learning fuzzy control rules from examples. In: F. Herrera and J.L. Verdegay (Eds.), Genetic Algorithms and Soft Computing, Physica-Verlag, 1996, 403-418.
- [14] T. Furuhashi, K. Nakaoka, K. Morikawa, H. Maeda, Y. Uchikawa. A study on knowledge finding using fuzzy classifier system. Japanese Journal of Fuzzy Theory and Systems 7 (1995) 555-567.
- [15] A. Geyer-Schulz. Fuzzy Rule-Based Expert Systems and Genetic Machine Learning. Physica-Verlag, 1995.
- [16] D.E. Goldberg. Genetic Algorithms in search, optimization & machine learning. Addison Wesley, 1989.
- [17] A. González, R. Pérez, J.L. Verdegay. Learning the structure of a fuzzy rule: a genetic approach. Fuzzy System and Artificial Intelligence 3 (1994) 57-70.
- [18] A. González, R. Pérez. Completeness and consistency conditions for learning fuzzy rules. Fuzzy Sets and Systems (1997) To appear.

- [19] A. González, R. Pérez. A learning system of fuzzy control rules. In: F. Herrera, J.L. Verdegay (Eds.), *Genetic Algorithms and Soft Computing*, Physica-Verlag, 1996, 202-225.
- [20] A. González, F. Herrera. Multi-stage genetic fuzzy systems based on the iterative rule learning approach, *Mathware & Soft Computing* (1997) To appear.
- [21] J.J. Grefenstette (Ed.). *Genetic Algorithms for Machine Learning*. Kluwer Academic, 1994.
- [22] R.R. Gudwin, F. Gomide, W. Pedrycz. Nonlinear context adaptation with genetic algorithms. *Proceedings 7th International Fuzzy Systems Association World Congress*, June 1997.
- [23] F. Herrera, M. Lozano, J.L. Verdegay. Generating fuzzy rules from examples using genetic algorithms. In: Bouchon-Meunier B., Yager R. R., and Zadeh L. A. (Eds.), *Fuzzy Logic and Soft Computing*, World Scientific, 1995, 11-20.
- [24] F. Herrera, M. Lozano, J.L. Verdegay. Tuning fuzzy logic controllers by genetic algorithms. *International Journal of Approximate Reasoning* 12 (1995) 299-315.
- [25] F. Herrera, J.L. Verdegay (Eds.). *Genetic Algorithms and Soft Computing*. Physica-Verlag, 1996.
- [26] F. Herrera, M. Lozano, J.L. Verdegay. A Learning process for fuzzy control rules using genetic algorithms. *Fuzzy Sets and Systems* (1997) To appear.
- [27] F. Hoffmann, G. Pfister. Learning of a fuzzy control rule base using messy genetic algorithms. In: F. Herrera and J.L. Verdegay (Eds.), *Genetic Algorithms and Soft Computing*, Physica-Verlag, 1996, 279-305.
- [28] J.H. Holland. *Adaptation in Natural and Artificial Systems*. Ann Arbor, 1975. [MIT Press (1992)].
- [29] J.H. Holland. Escaping brittleness: The possibilities of general purpose learning algorithms applied to parallel rule-based systems. In: R. Michalski, J. Carbonell, T. Michel (Eds.), *Machine Learning: An AI Approach*, Vol. II, Morgan-Kaufmann, 1986, 593-623.
- [30] C.L. Karr. Genetic algorithms for fuzzy controllers, *AI Expert* (1991) 26-33.
- [31] C.L. Karr. Design of an adaptive fuzzy logic controller using a genetic algorithm. *Proceedings 4th. International Conference on Genetic Algorithms*, Morgan Kaufmann, 1991, 450-457.
- [32] C.L. Karr, E.J. Gentry. Fuzzy control of pH using genetic algorithms. *IEEE Transactions on Fuzzy Systems* 1 (1993) 46-53.
- [33] M.A. Lee, H. Takagi. Integrating design stages of fuzzy systems using genetic algorithms. *Proceedings 2nd IEEE International Conference on Fuzzy Systems, FUZZ-IEEE'93*, 1993, volume 1, 612-617.
- [34] J. Liska, S. Melsheimer. Complete design of fuzzy logic systems using genetic algorithms. *Proceedings 3rd IEEE International Conference on Fuzzy Systems, FUZZ-IEEE'94*, 1994, volume II, 1377-1382.
- [35] L. Magdalena. *Estudio de la coordinación inteligente en robots bípedos: aplicación de lógica borrosa y algoritmos genéticos*. Doctoral dissertation, Universidad Politécnica de Madrid (Spain), 1994.
- [36] L. Magdalena. Adapting gain and sensibility of FLCs with genetic algorithms. *Sixth International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*, 1996, volume 2, 739-744.
- [37] L. Magdalena. A first approach to a taxonomy of fuzzy-neural systems. In: R. Sun and F. Alexandre (Eds.), *Connectionist Symbolic Integration*, chapter 5. Lawrence Erlbaum Associates, 1996.
- [38] L. Magdalena, F. Monasterio. Evolutionary based learning applied to fuzzy controllers. *Proceedings 4th IEEE International Conference on Fuzzy Systems and the Second International Fuzzy Engineering Symposium, FUZZ-IEEE/IFES'95*, 1995, volume III, 1111-1118.

- [39] L. Magdalena. Adapting the gain of an FLC with genetic algorithms. *International Journal of Approximate Reasoning* (1997) To appear.
- [40] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, 1992.
- [41] K.C. Ng, Y. Li. Design of sophisticated fuzzy logic controllers using genetic algorithms. *Proceedings 3rd IEEE International Conference on Fuzzy Systems, FUZZ-IEEE'94, 1994, volume III, 1708-1712.*
- [42] D. Park, A. Kandel, G. Langholz. Genetic-based new fuzzy reasoning models with application to fuzzy control. *IEEE Transactions on Systems, Man and Cybernetics* 24 (1994) 39–47.
- [43] A. Parodi, P. Bonelli. A new approach of fuzzy classifier systems. *Proc. Fifth Int. Conf. on Genetic Algorithms, Morgan Kaufmann, 1993, 223-230.*
- [44] W. Pedrycz. *Fuzzy Control and Fuzzy Systems*, Wiley, 1989.
- [45] D.T. Pham and D. Karaboga. Optimum design of fuzzy logic controllers using genetic algorithms. *Journal of Systems Engineering* 1 (1991) 114–118.
- [46] A. Satyadas, K. KrishnaKumar. EFM-based controllers for space attitude control: applications and analysis. In: F. Herrera and J.L. Verdegay (Eds.) *Genetic Algorithms and Soft Computing*, Physica-Verlag, 1996, 152–171.
- [47] K. Shimojima, T. Fukuda, Y. Hasegawa. RBF- fuzzy system with GA based unsupervised/supervised learning method. *Proceedings 4th IEEE International Conference on Fuzzy Systems and the Second International Fuzzy Engineering Symposium, FUZZ-IEEE/IFES'95, 1995, volume I, 253-258.*
- [48] H. Surmann, A. Kanstein, K. Goser. Self-organizing and genetic algorithms for an automatic design of fuzzy control and decision systems. *Proc. First European Congress on Fuzzy and Intelligent Technologies (EUFIT'93), Aachen, 1993, 1097-1104.*
- [49] P. Thrift. Fuzzy logic synthesis with genetic algorithms. *Proceedings 4th. International Conference on Genetic Algorithms, Morgan Kaufmann, 1991, 509-513.*
- [50] M. Valenzuela-Rendón. The fuzzy classifier system: A classifier system for continuously varying variables. *Proc. Fourth International Conference on Genetic Algorithms (ICGA'91), San Diego, 1991, 346-353.*
- [51] R.R. Yager. *Essentials of Fuzzy Modeling and Control*. John Wiley, 1995.