International Journal of Systems Science
Vol. 38, No. 12, December 2007, 991–1011

Optimization of fuzzy partitions for inductive reasoning using genetic algorithms

J. ACOSTA†, A. NEBOT*‡, P. VILLAR§ and J.M. FUERTES¶

†Departamento de Instrumentación Industrial, Inst. Univ. de Tecnología "Alonso Gamero", 4101 Coro-Estado Falcón, Venezuela
‡Departament de Llenguatges i Sistemes Informàtics, Universitat Politècnica de Catalunya, 08034 Barcelona, Spain
§Departamento de Lenguajes y Sistemas Informáticos, Universidad de Granada, 18071 Granada, Spain
¶Departament d'Enginyeria de Sistemes, Automàtica i Informàtica Industrial, Universitat Politècnica de Catalunya, 08028 Barcelona, Spain

(*Received 25 May 2006; in final form 30 August 2007*)

Fuzzy Inductive Reasoning (FIR) is a data-driven methodology that uses fuzzy and pattern recognition techniques to infer system models and to predict their future behavior. It is well known that variations on fuzzy partitions have a direct effect on the performance of the fuzzy-rule-based systems. The FIR methodology is not an exception. The performance of the model identification and prediction processes of FIR is highly influenced by the discretization parameters of the system variables, i.e. the number of classes of each variable and the membership functions that define its semantics. In this work, we design two new genetic fuzzy systems (GFSs) that improve this modeling and simulation technique. The main goal of the GFSs is to learn the fuzzification parameters of the FIR methodology. The new approaches are applied to two real modeling problems, the human central nervous system and an electrical distribution problem.

Keywords: Electrical engineering; Central nervous system; Genetic algorithms; Fuzzy inductive reasoning; Genetic fuzzy systems; Machine learning

1. Introduction

Fuzzy set theory and fuzzy logic are very powerful tools for managing uncertainties inherent to complex systems. Fuzzy systems have demonstrated their ability to solve different kind of problems like control, (Driankov *et al.* 1993, Leondes 1999), modeling (Pedrycz 1996) or classification (Chi *et al.* 1996, Vapnik 1998, Kuncheva 2000), and have been successfully applied to a wide range of applications, i.e. signal and image processing (Chi *et al.* 1996, Sattar and Tay 1999, Suzuki *et al.* 2001), risk assessment (Leondes 1999), information retrieval (Miyamoto 1989, Chen *et al.* 2001), industrial applications (Hirota and Sugeno 1995, Leondes 1999, Dote and Ovaska 2001), etc. However, most of the research done in the field in the 90s did not contain learning and adaptation capabilities. In the last decade, there has been a high interest for including learning in fuzzy systems. This has been achieved by means of the development of hybrid techniques that include fuzzy systems together with complementary techniques like neural networks, evolutionary algorithms or probabilistic methods.

It is commonly established that more intelligent systems can be obtained by the hybridization of soft computing methodologies (Bonissone 1997, Cordón *et al.* 2001). Neural fuzzy systems (NFSs) and genetic fuzzy systems (GFSs) are the most notorious representatives of hybrid systems within soft computing. NFS and GFS hybridize the approximate reasoning method of fuzzy systems with the learning processes

*Corresponding author. Email: angela@lsi.upc.edu

International Journal of Systems Science
ISSN 0020–7721 print/ISSN 1464–5319 online © 2007 Taylor & Francis
http://www.tandf.co.uk/journals
DOI: 10.1080/00207720701657581

based on a neural network and an evolutionary algorithm, respectively.

There are numerous studies on both subjects. However, neuro-fuzzy systems (Jang *et al*. 1997) have been used in a larger number of applications, in particular in the industrial area. This article is focused on GFSs (Cordón *et al*. 2001) and, therefore, we only review here some of the results that are more related to the work presented in our research.

In Cordón *et al*. (2004), the authors present an excellent overview of the research done in the last 10 years in the field of GFSs. As described in this article, the most prominent types of GFSs are genetic fuzzy-rule-based systems (GFRBSs), whose genetic processes learn or tune different components of a fuzzy-rule-base system, i.e. scaling functions (Gudwin *et al*. 1998, Magdalena 1999, Hoffmann 2001), membership functions (Herrera *et al*. 1995, Gürocak 1999, Hoffmann 2001, Casillas *et al*. 2005), rule bases (Ishibuchi *et al*. 1999, González and Pérez 2001, Hoffmann 2001, Camargo *et al*. 2004, Carmona *et al*. 2004, del Jesús *et al*. 2004) or knowledge bases (Heider and Drabe 1997, Hoffmann and Pfister 1997, Camargo *et al*. 2004, Pomares *et al*. 2004). In Gudwin *et al*. (1998), the use of contextual transformation functions to adjust membership functions is introduced. The fine tuning of membership functions is critical when evaluating the effectiveness of fuzzy systems in control, modeling or classification problems. Linear context adaptation is simple and fast, but the membership functions obtained are uniformly distributed. Non-linear context adaptation is more computationally expensive, but the membership functions can be stretched or expanded to best represent concepts in real environments, e.g. higher sensitivity in extreme classes or in middle classes. In that work, a genetic algorithm (GA) was used to find a non-linear transformation function given the base membership functions and a set of data available from the application studied. In Magdalena (1999), the author proposed a GA to learn the rule base and the gain and sensitivity of fuzzy logic controllers by means of scaling functions. Hoffmann (2001) describes two applications of GFSs, an evolutionary strategy that tunes the scaling and membership functions of a fuzzy cart-pole balancing controller and a GA that learns the fuzzy control rules for an obstacle-avoidance behavior of a mobile robot. Casillas *et al*. (2005) present a genetic tuning process for jointly fitting the fuzzy rule symbolic representations and the meaning of the involved membership functions. The good performance of this proposal mainly lies in the tuning approach performed at two different levels of significance. In Herrera *et al*. (1995) and Gürocak (1999), GA-based methods are described to alter the shapes of the fuzzy sets by shifting its peak location. In these studies, it is assumed that the rule base and the fuzzy sets are already defined. The research presented in Ishibuchi *et al*. (1999) and González and Pérez (2001) deals with the automatic generation of fuzzy if-then rules by means of genetic methods. In Ishibuchi *et al*. (1999), fuzzy if-then rules are obtained for pattern classification problems. This work uses fixed membership functions and therefore, no tuning mechanism is applied to them. In González and Pérez (2001), different search strategies (GAs, simulated annealing and hill climbing) are analyzed to find the best fuzzy rules that describe the system under study. del Jesús *et al*. (2004) proposes the use of an Adaboost algorithm for the same task. Another strategy is to identify fuzzy modes from certainty degrees, as studied in Carmona *et al*. (2004). Heider and Drabe (1997) and Hoffmann and Pfister (1997) present two genetic perspectives for the learning of fuzzy knowledge bases. In Heider and Drabe (1997), a cascaded GA is introduced with the idea of splitting the fuzzy system design process into optimization of the structure and the parameters. This algorithm is tested on a fuzzy controller design task. In Pomares *et al*. (2004), a novel approach to achieve global learning in fuzzy controllers is proposed.

Although the largest number of research efforts has been reported on GFRBSs, other kinds of GFSs, like genetic fuzzy neural networks (Russo 1998, Chung *et al*. 2000, Alpaydin *et al*. 2002) and genetic fuzzy clustering algorithms (Hall *et al*. 1994, Van Le 1995, Yuan *et al*. 1995), have also been developed with successful results.

In the research presented in this article, we propose two new GFS to improve a modeling and simulation technique, *Fuzzy Inductive Reasoning* (FIR). The main goal of the GFSs is to take advantage of the potentialities of GAs in order to learn the fuzzification parameters of the FIR methodology; i.e. the number of fuzzy sets (classes) per variable and the membership functions that define its semantics. Due to the fact that it is a methodology based on fuzzy logic, FIR modeling and prediction performance is influenced by these discretization parameters.

In the last years, the FIR methodology has been applied to different kinds of applications (e.g. control, biomedicine, ecology), usually obtaining good results (Mugica *et al*. 1994, Nebot *et al*. 1996, Nebot *et al*. 2001). In these studies, default values have been used to determine the number of classes and the associated membership functions. The default value for the number of classes' parameter for each system variable is three and the equal frequency partition (EFP) is used as the default method to obtain the membership functions of the classes. The EFP method is one of the simplest classification methods available. It consists in distributing the system data into a predefined number

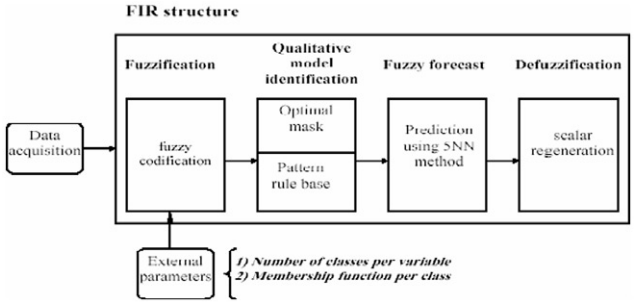

Figure 1. Fuzzy inductive reasoning (FIR) scheme.

of classes maintaining the same number of occurrences in each class.

However, experience has shown that in some applications, i.e. biomedical and ecological, the determination of the parameters needed in the discretization step becomes significant for the identification of a good model that captures systems behavior in an accurate way. Therefore, the automatic determination of good fuzzification parameters in the FIR methodology is an interesting and useful alternative to the use of heuristics and/or default values. This is, precisely, the main contribution of this article, i.e. the design and development of two GFS for the automatic determination of the FIR methodology fuzzification parameters. In our case, the fuzzy system is the FIR methodology and two GAs are developed to find the parameters of this fuzzy system. The first problem to be addressed is the learning of the optimal number of classes for each system variable. The second one is the learning of the membership functions of the classes. The reason for designing two GAs instead of only one that deals with both problems at the same time is 2-fold: 1) testing the viability of the GFS approach to fulfill each one of these goals in the context of the FIR methodology and 2) learning the number of classes and the membership functions independently, allowing the use of *a priori* expert knowledge when available or to study other methodologies for any of the problems addressed.

The improved FIR methodology is used for model identification of two real problems, i.e. modeling of the human central nervous system (CNS) control and estimating the maintenance cost of medium voltage lines in Spanish towns. The results are compared to the ones obtained by other methodologies in the same applications, i.e. NARMAX, time delay neural networks, recurrent neural networks, GFRBSs, linear models, etc.

The FIR methodology is presented in section 2. The GAs proposed are described in section 3. Sections 4 and 5 present the applications under study (i.e. medical and electrical) and the discussion of the obtained results, respectively. Finally, the conclusions of this research are given.

2. Fuzzy inductive reasoning methodology

The conceptualization of the FIR methodology arises from the General System Problem Solving approach (GSPS) proposed by Klir (1985). This methodology of modeling and qualitative simulation is based on systems behavior rather than on structural knowledge. It is able to obtain good qualitative relations between the variables that compose the system and to infer the future behavior of that system. It has the ability to describe systems that cannot easily be described by classical mathematics (e.g. differential equations), i.e. systems for which the underlying physical laws are not well understood. FIR is composed of four main processes, namely: fuzzification, qualitative model identification, fuzzy forecasting and defuzzification. Figure 1 describes the processes of the FIR methodology.

The fuzzification process converts quantitative data stemming from the system into fuzzy data, i.e. qualitative triples. The qualitative model identification process is responsible for finding causal and temporal relations between variables and therefore of obtaining the model that best represents the system.

Once the FIR model is available, the prediction system can take place using the FIR inference engine. This process is called fuzzy forecast. FIR inference engine is a specialization of the k-nearest neighbor rule, commonly used in the pattern recognition field. Defuzzification is the inverse process of fuzzification. It allows converting the qualitative predicted output into quantitative values that can then be used as input to an external quantitative model.

In order to define a useful chromosome codification and a good objective function, it is necessary to go deeply into the *fuzzification* and *model identification* processes of the FIR methodology.

2.1 *Fuzzification*

Figure 2 illustrates the process of fuzzification by means of an example. As mentioned earlier, a quantitative value is fuzzified into a qualitative triple. The first element of the triple is the class value, the second element is the fuzzy membership value and the third element is the side value. The side value indicates whether the qualitative value is to the left or to the right of the peak value of the associated membership function (figure 2).

The side value, that is not commonly used in fuzzy logic, is responsible for preserving, in the qualitative triple, the complete knowledge contained in the original quantitative value.

In figure 2, a temperature of 23°C would hence be fuzzified into the class *normal* with a side value *right* and a fuzzy membership value of 0.755.

Most fuzzy inference approaches preserve the knowledge by associating to each quantitative data value, multiple fuzzy rules consisting of tuples of class and membership values. They will thus represent the temperature of 23°C as being *normal* with likelihood 0.755 and being *warm* with likelihood 0.20. FIR accomplishes the same by associating to each quantitative data value, the described triple. Then, in the FIR methodology, the queues of the membership functions are discarded and only the part of the membership functions in the range [0.5–0.1] is used. The point where two neighboring classes match with a membership value of 0.5 is named *landmark*. Therefore, the component to be optimized by the GA is the width of the membership function of each class, specified by both landmarks. In the example of figure 2, the membership function of the class *normal* is defined by landmarks {13,27}; being this pair, the temperature values that specify the limits between the class *normal* and its adjacent classes, *fresh* and *warm*, respectively.

The result of the fuzzification process are three matrices of identical size named qualitative data matrices, one containing the class values, the second storing the membership information and the third recording the side values. Each column represents one of the observed variables and each row denotes one time point, i.e. one recording of all variables or one recorded state.

2.2 *Qualitative model identification*

A FIR model is composed of a structure, called mask, and a pattern rule base, named behavior matrix. A mask denotes a dynamic relationship among qualitative variables. An example of a mask is presented in equation (1).

t \ x	u_1	u_2	u_3	u_4	y_1
$t-2\delta t$	−1	0	0	−2	0
$t-\delta t$	0	0	0	0	−3
t	0	−4	0	0	+1

(1)

Each negative element in the mask is called an m-input (mask input). It denotes a causal relation with the output, i.e. it influences the output up to a certain degree. The enumeration of the *m*-inputs is immaterial and has no relevance. The single positive value denotes the output. The mask of equation (1) contains four m-inputs. In position notation, it can be written as (1,4,10,12,15), enumerating the mask cells from top to bottom and from left to right. In this example, the first and second *m*-inputs, i_1 and i_2, correspond to the input variables u_1 and u_4 two sampling intervals back, $(t-2\delta t)$, whereas the third *m*-input, i_3, refers to the output variable y_1 one sampling interval into the past, $(t-\delta t)$, etc.

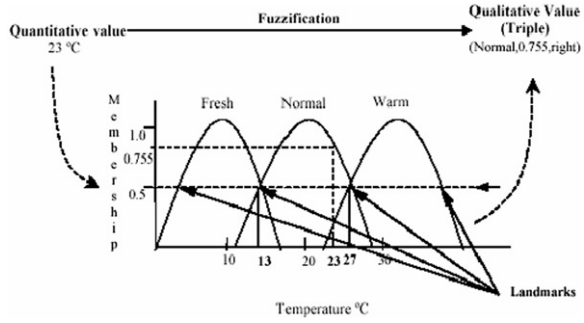

Figure 2. FIR fuzzification process of ambient temperature variable.

How is a mask found that, within the framework of all allowable masks, represents the most deterministic state transition matrix, i.e. optimizes the predictiveness of the model? In FIR, the concept of a mask candidate matrix is introduced. A mask candidate matrix is an ensemble of all possible masks from which the best is chosen by either a mechanism of exhaustive search of exponential complexity or by one of various suboptimal search strategies of polynomial complexity as described in Jerez and Nebot (1997). The mask candidate matrix contains −1 elements where the mask has a potential *m*-input, a +1 element where the mask has its *m*-output and 0 elements to denote forbidden connections. Thus, a good mask candidate matrix to start the search for the best mask shown in equation (1) might be:

t \ x	u_1	u_2	u_3	u_4	y_1
$t-2\delta t$	−1	−1	−1	−1	−1
$t-\delta t$	−1	−1	−1	−1	−1
t	−1	−1	−1	−1	+1

Each of the possible masks is compared to the others with respect to its potential merit, i.e. the degree of determinism associated with the state transition matrix constructed from it. The optimality of the mask is evaluated with respect to the maximization of its forecasting power. The Shannon entropy measure is used to determine the uncertainty associated with forecasting a particular output state, given any legal input state. The Shannon entropy relative to one input state is calculated from equation (2):

$$H_i = \sum_{\forall o} p(o|i) \cdot \log_2 p(o|i) \tag{2}$$

where $p(o|i)$ is the conditional probability of a certain m-output state o to occur, given that the *m*-input state

i has already occurred. The term probability is meant in a statistical rather than in a true probabilistic sense. It denotes the quotient of the observed frequency of a particular state divided by the highest possible frequency of that state. The overall entropy of the mask is then computed as the sum given in equation (3):

$$H_m = -\sum_{\forall i} p(i) \cdot H_i \tag{3}$$

where $p(i)$ is the probability of that input state to occur. The highest possible entropy $H_{\max}$ is obtained when all probabilities are equal, and a zero entropy is encountered for relationships that are totally deterministic. A normalized overall entropy reduction H_r is defined as:

$$H_r = 1.0 - \left(\frac{H_m}{H_{\max}}\right) \tag{4}$$

H_r is obviously a real-valued number in the range between 0.0 and 1.0, where high values usually indicate an improved forecasting power. The masks with highest entropy reduction values generate forecasts with the smallest amounts of uncertainty.

One problem still remains. The size of the pattern rule base increases as the complexity of the mask grows, and consequently, the number of legal states of the model grows fast. Since the total number of observed data records remains constant, the frequency of observation of each state shrinks rapidly, and so does the predictiveness of the model. The entropy reduction measure does not account for this problem. With increasing complexity, H_r simply keeps growing. Very soon, a situation is encountered where every state that has ever been observed has been observed precisely once. This obviously leads to a totally deterministic state transition matrix, and H_r assumes a value of 1.0. Yet the predictiveness of the model will be dismal, since in all likelihood, already the next predicted state has never before been observed, and that means the end of forecasting. Thus, this consideration must be included in the overall quality measure.

From a statistical point of view, every state should be observed at least five times (Law and Kelton 1990). Therefore, an observation ratio, O_r, is introduced as an additional contributor to the overall quality measure:

$$O_r = \frac{5 \cdot n_{5x} + 4 \cdot n_{4x} + 3 \cdot n_{3x} + 2 \cdot n_{2x} + n_{1x}}{5 \cdot n_{\text{leg}}} \tag{5}$$

where: n_{leg} is the number of legal m-input states, n_{1x} is the number of m-input states observed only once, n_{2x} is the number of m-inputs states observed twice, etc.

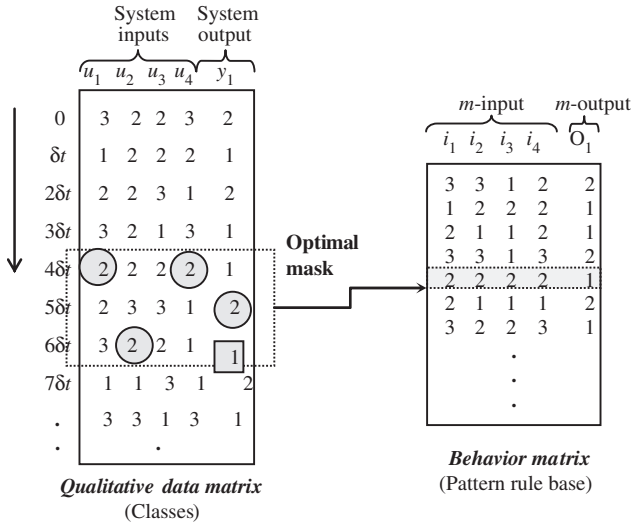

Figure 3. FIR pattern rule base obtaining.

If every legal m-input state has been observed at least five times, O_r is equal to 1.0. If no m-input state has been observed at all (no data are available), O_r is equal to 0.0. Thus, O_r can also be used as a quality measure. The overall quality of a mask, Q, is then defined as the product of its uncertainty reduction measure, H_r, and its observation ratio, O_r:

$$Q = H_r \cdot O_r \tag{6}$$

The optimal mask is the mask with the largest Q value.

Let us now address the second issue. How is the pattern rule base obtained from the mask? This process is illustrated in figure 3. The mask can be used to 'flatten' dynamic relationships into pseudo-static relationships. The left side of figure 3 shows an excerpt of the qualitative data matrix that stores the class values. It shows the numerical rather than the symbolic class values. In the example shown in figure 3, all the variables were discretized into three classes, except variable y_1, that was discretized into two classes. The dashed box symbolizes the mask that is shifted downwards along the class value matrix. The round shaded 'holes' in the mask denote the positions of the m-inputs, whereas the square shaded 'hole' indicates the position of the m-output. The class values are read out from the class value matrix through the 'holes' of the mask, and are placed next to each other in the behavior matrix that is shown on the right side of figure 3.

Here, each row represents one position of the mask along the class value matrix. It is lined up with the bottom row of the mask. Each row of the behavior matrix represents one pseudo-static qualitative state or qualitative rule (also called pattern rule). For example,

the shaded rule of figure 3 can be read as follows: 'If all the m-inputs (i_1, i_2, i_3, i_4,) have a value of 2 (corresponding to *medium*) then the m-output, O_1, assumes a value of 1 (corresponding to *high*)'.

The qualitative rules can be invoked during qualitative simulation to predict new qualitative outputs. Clearly, these rules can be written in any order, i.e. the sequencing of the rows of the behavior matrix has become irrelevant. They can be sorted alphanumerically. The sorted behavior matrix is called state transition matrix.

For a deeper and more detailed insight into the FIR methodology, the reader is referred to Nebot (1994).

3. Genetic algorithms for the optimization of fuzzy partitions

GAs are general purpose search methods used to find approximate solutions to difficult problems through the application of the principles of evolutionary biology to computer science. GAs use biologically derived techniques like inheritance, mutation, natural selection and recombination. There were initially proposed by Holland (1975) and they have been studied later in depth by other authors (Michalewicz 1996). They are very robust and highlight due to their good behavior in difficult problems, i.e. in those problems in which the search space is big, discontinuous, complex and not very well known. Although the optimal solution to the problem is not guaranteed, they usually provide acceptable solutions in a reasonable time.

The basic idea is to maintain a population of chromosomes (representing candidate solutions to the concrete problem being solved) that evolves over time through a process of competition and controlled variation that emulates the genetic processes that take place in the nature. Along successive iterations, denominated *generations*, the chromosomes in the population are ordered with regard to their degree of adaptation to the problem. Using this evaluation, a new population is built by means of a selection mechanism and a set of genetic operators like *crossover* and *mutation*. An *evaluation* or *fitness function* must be designed for each problem to be solved. Given a particular chromosome of the population (a possible solution), the fitness function returns a single numerical value, which is supposed to be proportional to the utility or adaptation of the solution represented by that chromosome. This function is responsible for guiding the GA in the search space. For this reason, it should be well designed so that it is capable, not only for distinguishing, in a clear way, the well-adapted chromosomes from those that are not, but also for ordering them with respect to their capacity to solve the problem.

The main aspects to be considered in the implementation of a GA are: (A) genetic representation, (B) fitness or objective function, (C) genetic operators and (D) genetic parameters. These points are highly important in order to achieve a good performance of the algorithm.

3.1 *Determination of the number of classes (GA1)*

In this section, the first GA proposed (GA1) is described in detail. The goal of GA1 is to determine the number of classes of each of the system variables.

3.1.1 Genetic representation. Each chromosome is composed by the number of classes associated to each variable. The number of linguistic terms for N variables is codified using a vector of N integers in the range [2–9]. The values of the genes are forced to remain in this interval, so the genetic operators must observe this requirement. Therefore, if we denote by X_i the number of classes for the variable i, a full chromosome representation for a system of N variables (including inputs and outputs), is defined by means of equation (7):

$$C = (X_1, X_2, \ldots, X_N) \tag{7}$$

3.1.2 Fitness or objective function. In order to evaluate a chromosome, the following steps are considered:

(1) Decode the information of the chromosome, building the associated fuzzy partition in the FIR structures.
(2) Execute the *qualitative model identification* process of the FIR methodology with the training data set, using the partition built in the previous step. Therefore, the mask associated to that partition with the highest quality measure is obtained.
(3) Compute an objective function. In this research, two objective functions are proposed: a) the *quality of the optimal mask* or b) the *prediction error of part of the training data set*.

As explained earlier, in the qualitative model identification process of the FIR methodology, the optimal mask (i.e. the best model structure) is identified by means of a quality measure, Q. The quality of a mask is a value between 0 and 1, where 1 indicates the highest quality. Therefore, the first cost function proposed is $1 - Q$, due to the fact that the algorithm task is to minimize the cost function.

The second cost function is defined as the prediction error of a portion of the training data set. The normalized mean square error in percentage (MSE),

given in equation (8), is used for this purpose,

$$\mathrm{MSE} = \frac{E\left[(y(t) - \hat{y}(t))^2\right]}{\mathrm{VAR}[y(t)]}.100\% \qquad (8)$$

where $\hat{y}(t)$ is the predicted output, $y(t)$ the system output and VAR denotes variance. The idea is to use a part of the training data set to identify the model and the rest of the data set to evaluate the prediction performance of that model. It is important to remember that the FIR model is composed of the optimal mask and the pattern rule base (behavior matrix). Therefore, both must be generated in the evaluation process of a certain fuzzy partition when this cost function is used. Moreover, the fuzzy forecasting process of the FIR methodology needs to be executed to obtain the cost of the evaluated chromosome. Thus, the computational cost of this evaluation function is considerably higher than the one obtained with the cost function that only depends on the quality of the mask. However, the prediction accuracy should be higher. The size of the portion of the training data set used for cost function evaluation purposes is defined with respect to the size of the whole training data set.

3.1.3 Genetic operators. In this work, the Stochastic Universal Sampling proposed by Baker (1987) is used, including an elitist selection. The genetic operators considered are:

(1) Crossover operator: operator is executed when the two parents have different granularity in one or more variables. In this case, the crossover operation is simple, a cut point is selected randomly and both parts are crossed according to the classic crossover operator.
(2) Mutation operator: Due to the nature of the values stored in the chromosome, the mutation operator proposed in Thrift (1991) is considered. In this case, the granularity associated to the gene of the selected chromosome is increased or decreased in one unit (the decision is made randomly). When the value to be changed is the minimum (2) or the maximum (9), the only possible change is done, i.e. increase or decrease by one the granularity, respectively.

3.1.4 Genetic parameters. The values of the probabilities have been established according to Grefenstette (1986). Table 1 shows the values of the parameters applied to this algorithm for each one of the two applications under study.

In the medical application, we have considered two stop criteria of the GA: to reach 40 and 64 chromosome evaluations, respectively. In the electrical application, we have considered again two stop criteria of the GA: to reach 1000 and 2000 chromosome evaluations, respectively. In this application, the search space is considerably larger.

The initial population is composed of a group of individuals with the same number of classes associated to each variable, and the rest of the chromosomes have their values chosen randomly. No repeated chromosomes are allowed.

3.2 *Determination of the membership functions (GA2)*

In this section, the second GA proposed (GA2) is described in detail. The goal of GA2 is to determine the membership function of each class.

3.2.1 Genetic representation. The genetic representation chosen takes into account the number of samples registered for each variable. A specific variable is represented by the proportion of data samples that each class contains, codified in the range [0–1]. An example of chromosome representation for a unique variable that has four classes could be $(0.4, 0.1, 0.3, 0.2)$, meaning that the membership function of the first class contains the 40% of the data samples available for this variable, and the second, third and fourth membership functions contain 10%, 30% and 20% of the data records, respectively. Of course, the sum of the proportions for each variable must be 1.

Therefore, if we denote by D_{ij} the data proportion of the class i and variable j, a full chromosome representation for a system of N variables (including inputs and outputs) with n classes per variable, is defined by:

$$C = (D_{11}, D_{n1}, D_{12}, D_{n2}, \ldots, D_{1N}, \ldots, D_{nN}) \qquad (9)$$

The minimum proportion, $V_{\min}$, is established to 0.05 and the maximum proportion, $V_{\max}$, is defined by $V_{\max} = 1 - V_{\min} \cdot (N_{\mathrm{label}} - 1)$, where N_{label} is the number of classes of the variable. Note that each time the distribution of the landmarks changes due to the action of the genetic operators, it is mandatory to recompute the proportions of the new distribution. A clear advantage of this representation is the facility to compute the landmarks from it. This is done by the following steps:

(1) The observed trajectory values of each variable are sorted in ascending order.
(2) The sorted vector is then split into segments (as many segments as classes have been determined for that variable) that contain the proportion of values determined by the GA2 solution.

Table 1. Genetic parameters of the GA1 and GA2 for the applications studied.

Parameter	GA type	Medical	Electrical
Population size (# individuals)	GA1	9	50
	GA2	50	50
Crossover probability	GA1 and GA2	0.6	0.6
Mutation probability	GA1 and GA2	0.1	0.1
Stop criteria (Chromosome evaluations)	GA1	(40, 64)	(1000, 2000)
	GA2	(3500, 7000)	(10000, 20000)

(3) Finally, the landmarks are chosen anywhere between the extreme values of neighboring segments, i.e. using the arithmetic mean values of neighboring observed data points in different segments.

3.2.2 Fitness or objective function. The same cost functions proposed for the determination of the number of classes are studied here. The evaluation of the chromosomes is done following the steps described in the section 3.1.2.

3.2.3 Genetic operators. The same selection mechanism mentioned in section 3.1.3 is used here. The genetic operators chosen are:

(1) Crossover operator: The arithmetic crossover operator (Michalewicz 1996) is considered the most adequate for the chromosome representation defined. This operator generates two offspring as a weighted mean of the parent values. A real value, u, in the range [0–1] is selected randomly and used to compute the new offspring by means of equation (10).

$$\begin{aligned} C_t &= u \cdot \text{father} + (1-u) \cdot \text{mother} \\ C'_t &= (1-u) \cdot \text{father} + u \cdot \text{mother} \end{aligned} \tag{10}$$

An advantage of the crossover operator selected is that it assures the validity of the offspring obtained, i.e. the sum of the data proportion for all the classes of each variable is 1.

(1) Mutation operator: Due to the nature of the values stored in the chromosome, the mutation operator proposed in Thrift (1991) is considered. In this case, the data proportion associated to the gene of the selected chromosome is increased or decreased (the decision is made randomly) by a factor in-between the range $[V_{\min} \ldots \text{MAX}]$ set, also, randomly. Where $\text{MAX} = 0.5 - V_{\min} \cdot (N_{\text{label}} - 1)$. The other proportions of the same variable are adjusted in order to maintain the addition to 1.

When the value to be changed plus the factor gets out of the limits of the range $[V_{\min} \ldots V_{\max}]$, the only possible change is done, i.e. increase or decrease by the proportion factor, respectively.

3.2.4 Genetic parameters. The crossover and mutation parameters proposed in Subsection 3.1.4 are also used here, whereas the population size and the stop criteria used are shown also in table 1.

Notice that for the medical application studied, the search space is much bigger than it was in the problem of the determination of the number of classes, and therefore, the stop criteria is now set to 3500 and 7000 chromosome evaluations (section 4).

In the electrical application, we have considered again two stop criteria of the GA: to reach 10,000 and 20,000 chromosome evaluations, due to the fact that in this case the search space is also larger.

The initial population is composed of an individual distributed by the EFP method, and the rest of the chromosomes have their values chosen randomly. No repeated chromosomes are allowed.

4. Central nervous system

The human CNS (figure 4) is composed of five controllers, namely, heart rate (HR), myocardium contractility (MC), peripheral resistance (PR), venous tone (VT) and coronary resistance (CR). All of them are single-input/single-output (SISO) models driven by the same input variable, namely the carotid sinus blood pressure (CSP).

The input and output signals of the CNS controllers were recorded with a sampling rate of 0.12 seconds from simulations of the purely differential equation model obtaining 7279 data points (Nebot *et al.* 1998). The model had been tuned to represent a specific patient suffering a coronary arterial obstruction, by making the four different physiological variables (right auricular pressure, aortic pressure, coronary blood flow and HR) of the simulation model agree with the measurement data taken from the real patient. Each CNS control was

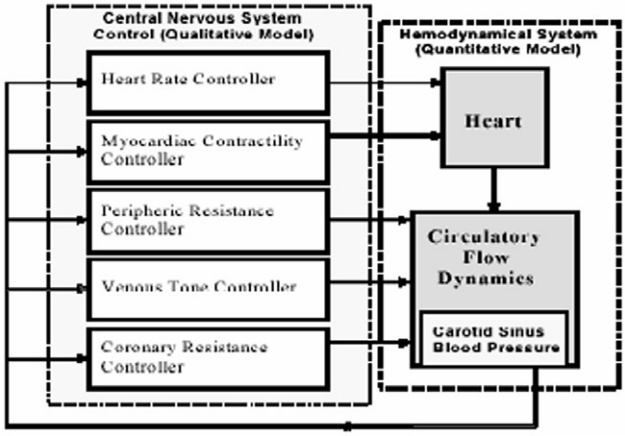

Figure 4. Simplified diagram of the cardiovascular system model, composed of the hemodynamic system and the CNS control.

Table 2. Prediction errors (MSE) of the CNS controller models using NARMAX, TDNN and RNN methodologies (Mean value of the 6 test data sets for each controller).

	HR (%)	PR (%)	MC (%)	VT (%)	CR (%)
NARMAX	9.3	18.5	22.0	22.0	25.5
TDNN	15.3	33.7	34.0	34.0	55.6
RNN	18.3	31.1	35.1	34.7	57.1

validated by using it to forecast six data sets not used in the training process. Each one of these six test data sets, with a size of about 600 data points each, contains signals representing specific morphologies, allowing the validation of the model for different system behaviors.

4.1 *Previous works*

Table 2 contains the predictions achieved when NARMAX (Nebot *et al.* 1998), time delay neural networks and recurrent neural networks (Cueva *et al.* 1997) are used for the same problem. The columns of the table specify the average prediction error of the six test sets for each controller. All methodologies used the same training and test data sets previously described.

4.2 *Learning the optimal number of classes for each system variable (GA1)*

This section evaluates the utility of the GA1 proposed for the optimization of the number of classes. Thirty executions of the GA1 are performed for each objective function and stop criteria. This GA is executed to obtain the optimal number of classes for the input and output variables of each of the five controllers. The EFP is used as the default method to obtain the membership functions of the classes.

As mentioned before, two objective functions were studied in this work. Table 3 shows the results obtained for the CR controller when $1-Q$ was used as objective function and when the objective function is defined as the prediction MSE of a portion of the training data set. In this application, the last 25% of the training signal is used for objective function evaluation and only the first 75% of the signal is used to obtain the FIR models (masks and pattern rule bases).

Table 3 is organized as follows. The first column indicates the number of chromosome evaluations made by the GA1. The second column indicates the partition suggested by the GA when its execution ends up. Note that the partition suggested by the GA is the input parameter to the *fuzzification* process of the FIR methodology. The third and fourth columns present the optimal mask (in position notation) obtained by FIR for this specific partition and the quality associated to this mask, respectively. The fifth column corresponds to the evaluation of the objective function used: $1-Q$ or MSE_{train}. The next column shows the mean value of the prediction errors obtained for the six test data sets. The last column indicates the CPU time (in seconds) used by the GA.

The GA suggests, when it reaches 40 evaluations, 5 and 4 different partitions for the objective functions $1-Q$ and MSE_{train}, respectively. When 64 evaluations are used as the stop criteria, the GA suggests for both objective functions, four different partitions. Looking closer at table 3 it can be seen that the optimal solution for the objective function $1-Q$ corresponds to the partition (9,3) with a mask quality of 0.9787. The partitions found, for both stop criteria, suggest always three classes for the output variable, and five, six, seven, eight or nine classes for the input variable. Notice that the qualities (Q) of all the suggested partitions are very close to the optimal one.

The MSE_{train} objective function corresponds to the prediction error of part of the training data set. As mentioned before, in this case, the mask is obtained using exclusively the first 75% data points of the training signal. Therefore, the data used for the objective function evaluation has not been seen before for the model. This is the reason why the best predictions obtained for the last 25% values of the training set do not correspond, necessarily, to the partitions with associated optimal masks of highest quality. However, the quality of the optimal masks found for the suggested partitions are still high, i.e. 0.9642. The optimal solution is the partition (2,5) with a MSE_{train} of 0.08%, that is really very low. The partition (3,4) is the best suboptimal solution, i.e. with the second lowest MSE_{train}. Therefore, the GA is able to obtain the best partitions. Notice that

Table 3. Number of classes results of the CR controller using: (a) $1-Q$ and, (b) Prediction error of the last 25% of the training data set (MSE_{train}) cost functions. EFP method for the membership functions.

$1-Q$	# Eval	Part.	Opt. mask	Q	$1-Q$	MSE_{test} (%)	Time
	64	(9,3)*	(1,4,6)	0.9787	0.0213	3.92	18
	64	(8,3)	(1,4,6)	0.9776	0.0224	4.33	14
	64	(7,3)	(1,4,6)	0.9776	0.0224	4.85	13
	64	(4,3)	(1,4,6)	0.9748	0.0252	1.36	10
	40	(9,3)*	(1,4,6)	0.9787	0.0213	3.92	12
	40	(8,3)	(1,4,6)	0.9776	0.0224	4.33	9
	40	(7,3)	(1,4,6)	0.9776	0.0224	4.85	9
	40	(6,3)	(1,4,6)	0.9762	0.0238	1.79	7
	40	(5,3)	(1,4,6)	0.9749	0.0251	2.38	9
			Optimal Solution: Partition = (9,3); Q = 0.9787				
MSE_{train}	# Eval	Part.	Opt. mask	Q	MSE_{train} (%)	MSE_{test} (%)	Time
	64	(2,5)*	(1,4,5,6)	0.9642	0.08	0.16	114
	64	(3,4)	(4,5,6)	0.9638	0.13	0.29	111
	64	(2,4)	(3,4,5,6)	0.9630	0.17	0.39	93
	64	(7,4)	(4,5,6)	0.9677	0.19	0.42	84
	40	(2,5)*	(1,4,5,6)	0.9642	0.08	0.16	49
	40	(3,4)	(4,5,6)	0.9638	0.13	0.29	55
	40	(2,4)	(3,4,5,6)	0.9630	0.17	0.39	57
	40	(7,4)	(4,5,6)	0.9677	0.19	0.42	54
			Optimal Solution: Partition = (2,5); MSE_{train} = 0.08%				

*Optimal solution.

the CPU time has increased considerably with respect to the objective function $1-Q$. This is due to the fact that now the *forecasting* process of the FIR methodology needs to be executed for each partition evaluated.

Tables 4 and 5 contain the partition results for the rest of the CNS controllers. The tables are organized as table 3, but the optimal mask, quality and time columns are not included. The GA has been executed 30 times for both objective functions and stop criteria. As can be seen in the tables, the optimal solution is reached for both objective functions and stop criteria in all the controllers.

It is interesting to analyze the MSE_{test} columns of tables 3–5. As expected, the MSE_{train} objective function is able to obtain partitions with higher performance on the prediction of the test data sets than the ones obtained by the $1-Q$ objective function. However, the results obtained for all the controllers are very good if compared to the ones obtained when other inductive methodologies are used (table 2). Moreover, the highest MSE_{test} of 4.85% obtained with the $1-Q$ objective function for the controller CR is five times smaller than the smaller error obtained with these methodologies, 25.5%. Therefore, in this application, both objective functions can be considered good for the task at hand. However, this is not the case for all the controllers when the $1-Q$ objective function is used. Notice that although the GA finds both the best solution and the suboptimal solutions, the prediction error of the test data sets for the HR controller is smaller using the NARMAX approach, i.e. 9.3% vs. 13.76%. In this case, the quality measure used by the FIR methodology is not doing a good job. It can be interesting to study alternative quality measures for the task at hand.

In general, it is observed that in all the controllers, the $1-Q$ objective function needs less time to be evaluated but the performance with respect to the test set prediction is lower. Contrarily, the MSE_{train} objective function is more expensive from the CPU time point of view but the performance is higher. The user should decide which objective function to use taking into account the size of the optimization problem and his/her own needs.

Clearly, the medical application presented in this article is a short optimization problem due to the fact that only two variables are involved and a maximum of nine classes is allowed for each one of them (in fact only eight, because class 1 is not used). Therefore, there exist 64 possible solutions and an exhaustive search can be performed easily.

However, previous works (Schoenauer and Michalewicz 1997, González and Pérez 2001) suggest the scalability of the GAs. Therefore, it can be assumed that this approach will work on large optimization problems as well where an exhaustive search would be impracticable. This is demonstrated by means of the electrical application presented later in this article.

Table 4. Number of classes results of the MC and HR controllers using: (a) $1-Q$ and, (b) Prediction error of the last 25% of the training data set (MSE_{train}) cost functions. EFP method for the membership functions.

MC				
$1-Q$	# Eval	Part	$1-Q$	MSE_{test} (%)
	64	(8,7)*	0.1866	8.37
	64	(7,7)	0.1950	40.77
	40	(8,7)*	0.1866	8.37
	40	(7,7)	0.1950	40.77
MSE_{train}	# Eval	Part	MSE_{train} (%)	MSE_{test} (%)
	64	(4,9)*	0.60	2.53
	64	(2,5)	0.63	2.76
	64	(3,9)	1.10	3.91
	40	(4,9)*	0.60	2.53
	40	(2,5)	0.63	2.76
	40	(3,9)	1.10	3.91
	40	(2,6)	1.13	3.04
HR				
$1-Q$	# Eval	Part	$1-Q$	MSE_{test} (%)
	64	(7,2)*	0.1674	13.76
	64	(8,2)	0.1861	12.94
	64	(6,2)	0.1968	15.99
	64	(7,4)	0.2739	2.68
	64	(9,4)	0.2756	13.28
	64	(8,7)	0.2774	12.39
	40	(7,2)*	0.1674	13.76
	40	(8,2)	0.1861	12.94
	40	(6,2)	0.1968	15.99
	40	(9,2)	0.1973	12.24
	40	(7,4)	0.2739	2.68
	40	(8,4)	0.2763	2.91
	40	(8,7)	0.2774	12.39
	40	(7,8)	0.2804	5.69
	40	(9,7)	0.2811	3.59
MSE_{train}	# Eval	Part	MSE_{train} (%)	MSE_{test} (%)
	64	(3,7)*	0.89	9.37
	64	(5,9)	1.02	2.85
	64	(4,6)	1.03	3.48
	64	(4,7)	1.34	3.82
	40	(3,7)*	0.89	9.37
	40	(5,9)	1.02	2.85
	40	(4,6)	1.03	3.48
	40	(6,7)	1.15	13.73
	40	(4,4)	1.65	2.79

*Optimal solution.

Table 5. Number of classes results of the VT and PR controllers using: (a) $1-Q$ and, (b) Prediction error of the last 25% of the training data set (MSE_{train}) cost functions. EFP method for the membership functions.

VT				
$1-Q$	# Eval	Part	$1-Q$	MSE_{test} (%)
	64	(8,7)*	0.1858	9.49
	64	(7,7)	0.1952	40.20
	40	(8,7)*	0.1858	9.49
	40	(7,7)	0.1952	40.20
MSE_{train}	# Eval	Part	MSE_{train} (%)	MSE_{test} (%)
	64	(2,5)*	0.61	1.69
	64	(2,8)	0.64	1.58
	64	(2,7)	0.77	2.09
	64	(4,9)	0.91	2.06
	40	(2,5)*	0.61	1.69
	40	(2,8)	0.64	1.58
	40	(2,7)	0.77	2.09
	40	(7,7)	1.34	2.08
PR				
$1-Q$	# Eval	Part	$1-Q$	MSE_{test} (%)
	64	(8,7)*	0.1448	6.10
	64	(7,7)	0.1505	4.66
	40	(8,7)*	0.1448	6.10
	40	(7,7)	0.1505	4.66
MSE_{train}	# eval	Part	MSE_{train} (%)	MSE_{test} (%)
	64	(4,9)*	0.93	3.05
	64	(7,7)	1.08	3.40
	40	(4,9)*	0.93	3.05
	40	(7,7)	1.08	3.40

*Optimal solution.

4.3 *Learning the membership functions of the classes (GA2)*

This section evaluates the utility of the GA2 proposed for the optimization of the membership functions. Thirty executions of the GA2 are performed for each objective function. This GA considers the optimal number of classes obtained by the previous algorithm.

The results are presented in two sections, based on the objective function used for the evaluation of the chromosomes.

4.3.1 $1-Q$ objective function. Table 6 presents the results of the GA2 for the five controllers when $1-Q$ objective function is used. The table is organized as follows. The first column is divided in three sections. Section A corresponds to the best result obtained by the GA2 while section B corresponds to its worse result. Section C shows the result when the GA2 is not used, i.e. when the EFP method is used to determine the data proportions for each variable. Therefore, C is taken as a reference. The second column indicates the number of chromosome evaluations made by the GA2. The third column shows the data proportion for the input variable (CSP), and the output variable for each

Table 6. Membership functions results of the MC, HR, VT, PR and CR controllers using $1-Q$ cost function. GA1 results for the number of classes.

	# Eval	Data proportion	Opt. mask	Q	$1-Q$	MSE_{test} (%)
		MC controller; Num. Class.: (8,7)				
A	7000	CSP:(0.17,0.19,0.20,0.08,0.11,0.07,0.07,0.11) MC:(0.06,0.05,0.06,0.06,0.05,0.67,0.05)	(4,5,6)	0.9371	0.0629	3.63
	3500	CSP:(0.18,0.21,0.10,0.08,0.10,0.10,0.15,0.08) MC:(0.07,0.05,0.05,0.05,0.08,0.65,0.05)	(3,4,6)	0.9334	0.0666	82.52
B	7000	CSP:(0.15,0.15,0.13,0.08,0.09,0.14,0.13,0.13) MC:(0.17,0.15,0.12,0.15,0.15,0.16,0.10)	(3,4,6)	0.8347	0.1653	4.03
	3500	CSP:(0.14,0.10,0.14,0.10,0.08,0.16,0.19,0.09) MC:(0.17,0.14,0.17,0.15,0.16,0.11,0.10)	(3,4,6)	0.8345	0.1655	2.53
C		Equal frecuency partition method	(3,4,6)	0.8166	0.1834	8.37
		HR controller; Num. Class.: (7,2)				
A	7000	CSP:(0.05,0.20,0.27,0.08,0.09,0.05,0.26) HR:(0.06,0.94)	(4,5,6)	0.9421	0.0579	11.49
	3500	CSP:(0.05,0.05,0.05,0.36,0.19,0.05,0.25) HR:(0.06,0.94)	(4,5,6)	0.9420	0.0580	11.98
B	7000	CSP:(0.15,0.14,0.11,0.14,0.10,0.14,0.22) HR:(0.07,0.93)	(4,5,6)	0.9212	0.0788	12.48
	3500	CSP:(0.20,0.19,0.13,0.10,0.07,0.09,0.22) HR:(0.06,0.94)	(4,5,6)	0.9192	0.0808	11.49
C		Equal frequency partition method	(4,5,6)	0.8326	0.1674	13.76
		VT controller; Num. Class.: (8,7)				
A	7000	CSP:(0.21,0.17,0.19,0.07,0.13,0.07,0.11,0.05) VT:(0.06,0.08,0.05,0.05,0.06,0.62,0.08)	(3,4,6)	0.9332	0.0668	6.17
	3500	CSP:(0.17,0.26,0.11,0.14,0.11,0.07,0.08,0.06) VT:(0.10,0.07,0.07,0.06,0.60,0.05,0.05)	(3,4,6)	0.9256	0.0744	73.77
B	7000	CSP:(0.14,0.11,0.14,0.11,0.11,0.19,0.10,0.10) VT:(0.17,0.12,0.12,0.14,0.15,0.20,0.10)	(3,4,6)	0.8401	0.1599	4.87
	3500	CSP:(0.13,0.13,0.12,0.11,0.09,0.15,0.11,0.16) VT:(0.13,0.18,0.13,0.16,0.15,0.15,0.10)	(3,4,6)	0.8308	0.1692	9,82
C		Equal frecuency partition method	(3,4,6)	0.8142	0.1858	9.49
		PR controller; Num. Class.: (8,7)				
A	7000	CSP:(0.13,0.13,0.11,0.09,0.15,0.12,0.14,0.13) PR:(0.18,0.11,0.15,0.14,0.15,0.11,0.16)	(4,5,6)	0.8831	0.1169	19.56
	3500	CSP:(0.13,0.14,0.10, 0.10, 0.12,0.13,0.15,0.13) PR:(0.18,0.11,0.15,0.14,0.14,0.12,0.16)	(4,5,6)	0.8807	0.1194	3.61
B	7000	CSP:(0.19,0.15,0.13,0.12,0.09,0.12,0.13,0.07) PR:(0.13,0.16,0.20,0.17,0.18,0.10,0.06)	(3,4,6)	0.8687	0.1313	8.21
	3500	CSP:(0.12,0.13,0.11,0.15,0.10,0.21,0.09,0.09) PR:(0.20,0.09,0.29,0.15,0.11,0.09,0.07)	(4,5,6)	0.8671	0.1329	3.66
C		Equal frecuency partition method	(4,5,6)	0.8552	0.1448	6.10
		CR controller; Num. Class.: (9,3)				
A	7000	CSP:(0.06,0.09,0.18,0.12,0.09,0.22,0.11,0.08,0.05) CR:(0.38,0.43,0.19)	(1,4,6)	0.9845	0.0155	2.03
	3500	CSP:(0.06,0.07,0.10,0.19,0.11,0.23,0.07,0.12,0.05) CR:(0.40,0.41,0.19)	(1,4,6)	0.9844	0.0156	2.03
B	7000	CSP:(0.07,0.14,0.17,0.17,0.18,0.10,0.07,0.05,0.05) CR:(0.38,0.43,0.19)	(1,4,6)	0.9840	0.0160	1.36
	3500	CSP:(0.11,0.20,0.09,0.25,0.12,0.07,0.06,0.05,0.05) CR:(0.38,0.39,0.23)	(1,4,6)	0.9836	0.0164	1.56
C		Equal frecuency partition method	(1,4,6)	0.9787	0.0213	3.92

controller (MC, HR, VT, PR, CR). The number of elements of the data proportion corresponds to the number of classes for that variable (shown in table 6 on top of each controller). The data proportion is the output of the GA2. The fourth column presents the optimal mask, in position notation, encountered by FIR when the data proportion obtained is used to set the landmarks. The fifth column corresponds to the quality associated to the optimal mask. The sixth column is the value of the objective function $1-Q$. The last column shows the mean value of the prediction errors (equation (8)) obtained for the six test data sets.

If we look closer at table 6, it can be seen that for all the controllers, the worse result obtained by the GA2 ($1-Q$ value of row B) is better than the reference result, i.e. when the default EFP method is used to obtain the membership functions of each class. Therefore, although the GA does not assure the optimal solution, all the solutions obtained are better than the one obtained when no GA is used. The prediction errors of the test sets (last column) are also presented to see the accuracy of each model obtained. Notice that although the MSE_{test} is usually smaller when the GA2 is used to obtain the membership functions, this is not always true. This is due to the fact that the test data sets have not been used in the FIR model identification process.

The CPU time needed by the GA to run the 30 executions is, obviously, directly related to the pre-defined number of classes of the variables. For MC, VT and PR controllers with the number of classes set to (8,7), i.e. the partition with a higher number of classes, the CPU time needed by the GA to compute one execution was 1 hour and 45 minutes using a Pentium III (0.6GHz) computer.

4.3.2 MSE_{train} objective function. Table 7 presents the results of the GA2 for the five controllers when the MSE_{train} objective function is used. The table is organized as table 6. The only difference is that the fifth column contains the values of the MSE_{train} instead of the $1-Q$. As can be seen in table 7, the GA2 results are better than the results obtained when the default EFP method is used to obtain the membership functions.

As explained before, the CPU time needed by the GA when the MSE_{train} objective function is used is clearly greater than the time needed when the $1-Q$ is used. In this case, the mean CPU time needed by the GA to compute one execution for each controller was 5 hours, using the same computer as before. If we analyze the last columns of tables 6 and 7, it can easily be seen that the MSE_{train} objective function is able to obtain membership functions with higher performance on the prediction of the test data sets than the ones obtained by the $1-Q$ objective function. Also, the best results obtained by the GA2 (row A) are very good if we compare them with the ones obtained when the EFP method and other inductive methodologies are used (table 2). However, as expected, the MSE_{train} objective function is more expensive from the CPU time point of view. As before, the user should decide which objective function to use taking into account the size of the optimization problem and his/her own needs.

5. Electrical distribution network

The problem of estimating the maintenance cost of the electric network becomes difficult when we deal with medium and low voltage lines. Maintenance cost depends, among other factors, on the total length of the electrical line each company owns, and on its kind, i.e. high, medium or low voltage (Cordón *et al*. 1999). To justify the distribution expenses of the companies, models of the length of the line are used. Although high voltage lines can be easily measured, this is not the case with medium and low voltage lines. These lines are found in cities and villages, and it is very difficult and expensive to measure them, due to the fact that they have been installed incrementally, according to the electrical needs in each moment. Therefore, it is necessary to handle the problem from the modeling perspective.

To deal with the problem under study, we were provided with data of 1059 simulated towns (Cordón *et al*. 1998, Cordón *et al*. 1999).

Four characteristics of each town correspond to the input variables, i.e. the sum of the lengths of all streets in the town (SLS) in kilometers, the total area of the town (TA) in Km^2, the area that is occupied by buildings (AB) in Km^2 and the energy supply to the town (ES) in MWh. The maintenance cost of the medium voltage line (MC) in millions of pesetas is the output variable.

In the previous works (Cordón *et al*. 1998, Cordón *et al*. 1999), the available data were divided into the training set (first 847 towns) and the test set (last 212 towns), corresponding to the 80% and 20% of the whole data set, respectively. The same data distribution is used in the present study in order to compare the results obtained in an accurate way. For the same reason, the medium square error (SE) used in Cordón *et al*. (1998) and described in equation (11) is used for the computation of each model prediction error.

$$SE = \frac{1}{2 * N} \sum_{i=1}^{N} \left(y_i(t) - \hat{y}_i(t) \right)^2 \tag{11}$$

where $\hat{y}(t)$ is the predicted output, $y(t)$ the system output and N the number of samples.

Table 7. Membership functions results of the MC, HR, VT, PR and CR controllers using MSE_{train} cost function. GA1 results for the number of classes.

	# Eval	Data proportion	Opt. mask	Q	MSE_{train} (%)	MSE_{test} (%)
		MC controller; Num. Class.: (4,9)				
A	7000	CSP:(0.20,0.29,0.31,0.20) MC:(0.09,0.08,0.12,0.13,0.12,0.11,0.11,0.11,0.13)	(4,5,6)	0.7452	0.20	0.45
	3500	P:(0.27,0.18,0.32,0.23) MC:(0.09,0.08,0.11,0.11,0.15,0.11,0.15,0.10,0.10)	(4,5,6)	0.7899	0.21	0.48
B	7000	CSP:(0.22,0.16,0.35,0.27) MC:(0.16,0.12,0.11,0.13,0.10,0.10,0.10,0.10,0.08)	(4,5,6)	0.7616	0.33	2.98
	3500	CSP:(0.13,0.18,0.40,0.29) MC:(0.16,0.10,0.13,0.09,0.10,0.09, 0.15,0.10,0.08)	(4,5,6)	0.7507	0.37	1.59
C		Equal frequency partition method	(3,4,6)	0.7346	0.60	2.53
		HR controller; Num. Class.: (3,7)				
A	7000	CSP:(0.32,0.30,0.38) HR:(0.13,0.10,0.12,0.12,0.29,0.12,0.12)	(4,5,6)	0.6837	0.44	1.49
	3500	CSP:(0.30,0.32,0.38) HR:(0.13,0.14,0.17,0.16,0.13,0.14,0.13)	(4,5,6)	0.6517	0.44	1.52
B	7000	CSP:(0.32,0.30,0.38) HR:(0.19,0.12,0.16,0.14,0.12,0.12,0.15)	(4,5,6)	0.6711	0.60	2.30
	3500	CSP:(0.41,0.23,0.36) HR:(0.16,0.09,0.21,0.20,0.14,0.10,0.10)	(4,5,6)	0.6998	0.69	2.73
C		Equal frequency partition method	(4,5,6)	0.6761	0.89	9.37
		VT controller; Num. Class.: (2,5)				
A	7000	CSP:(0.38,0.62) VT:(0.11,0.14,0.31,0.22,0.22)	(1,4,5,6)	0.7475	0.19	0.46
	3500	CSP:(0.39,0.61) VT:(0.11,0.12,0.25,0.30,0.22)	(1,4,5,6)	0.7400	0.20	0.47
B	7000	CSP:(0.41,0.59) VT:(0.14,0.10,0.20,0.22,0.34)	(3,4,5,6)	0.7567	0.29	0.76
	3500	CSP:(0.43,0.57)	(1,4,5,6)	0.7825	0.33	0.81
		VT:(0.13,0.24,0.46,0.09,0.08)	(1,3,4,6)	0.7315	0.61	1.69
C		Equal frequency partition method				
		PR controller; Num. Class.: (4,9)				
A	7000	CSP:(0.23,0.20,0.23,0.34) PR:(0.09,0.10,0.08,0.13,0.14,0.08,0.13,0.10,0.15)	(4,5,6)	0.8064	0.20	0.80
	3500	CSP:(0.24,0.22,0.22,0.32) PR:(0.09,0.09,0.10,0.11,0.14,0.13,0.14,0.10,0.10)	(3,4,6)	0.7995	0.22	0.72
B	7000	CSP:(0.21,0.15,0.50,0.14) PR:(0.14,0.13,0.12,0.13,0.09,0.10,0.07,0.07,0.15)	(4,5,6)	0.7983	0.41	3.88
	3500	CSP:(0.27,0.16,0.28,0.29) PR:(0.14,0.11,0.14,0.09,0.08,0.10,0.10,0.15,0.09)	(4,5,6)	0.7758	0.41	1.06
C		Equal frequency partition method	(3,4,6)	0.7497	0.93	3.05
		CR controller; Num. Class.: (2,5)				
A	7000	CSP:(0.54,0.46) CR:(0.13,0.16,0.55,0.11,0.05)	(1,4,5,6)	0.9488	0.05	0.11
	3500	CSP:(0.49,0.51) CR:(0.12,0.17,0.54,0.09,0.08)	(1,4,5,6)	0.9533	0.05	0.11
B	7000	CSP:(0.53,0.47) CR:(0.20,0.18,0.20,0.29,0.13)	(1,4,5,6)	0.9650	0.07	0.13
	3500	CSP:(0.53,0.47) CR:(0.20,0.23,0.22,0.18,0.17)	(1,4,5,6)	0.9646	0.07	0.13
C		Equal frequency partition method	(1,4,5,6)	0.9642	0.08	0.16

It is interesting to notice that no temporal relation exists between two consecutive samples of the five system variables, due to the fact that each sample represents a specific town. This is the first time that FIR methodology is used to deal with a non-dynamical system. However, this is solved easily by forbidding temporal relations between the system variables. This can be achieved by defining a mask candidate matrix of one row, as shown in equation (12).

$$\begin{array}{c|ccccc|} t\backslash x & \text{SLS} & \text{TA} & \text{AB} & \text{ES} & \text{MC} \\ t & -1 & -1 & -1 & -1 & +1 \end{array} \qquad (12)$$

As explained in Section 2, with the proposed candidate matrix, the qualitative model identification process of FIR methodology computes all possible masks and the one with the maximum quality value is considered the optimal one.

5.1 *Previous works*

Table 8 contains the SE prediction errors achieved when classical methods and hybrid evolutionary techniques were used for the same problem (Cordón *et al*. 1998, Cordón *et al*. 1999). As regards classical methods, Cordón, *et al*. had considered linear models fitted by linear least squares, second-order polynomial models fitted by non-linear least square and three-layer-perceptron neural network (of 4-5-1 neurons). The minimization error algorithm was the conjugate gradient. With respect to other techniques, they studied GFRBS for the optimization of three different fuzzy models, i.e. Wang–Mendel (WM), Mamdami and Takagi-Sugeno-Kang (TSK). Finally, they used two hybrid algorithms, GA-P and Interval GA-P, that combines the traditional GAs with the genetic programming (GP) paradigm (Howeard and D'Angelo 1995). The Interval GA-P is a modified version of the GA-P method that uses interval values instead of punctual ones. All the methodologies used the same training and test data sets explained previously.

The first column of table 8 describes the method evaluated; the second and third columns show the prediction errors using the SE formula [described in equation (11)], of the training and test data sets, respectively. As can be seen in table 8, the GA-P techniques and fuzzy models outperform again classical linear and non-linear regression methods as well as neural networks. The TSK fuzzy model has obtained the best result. A more detailed discussion of the results presented in table 8 can be found in Cordón *et al*. (1999). These values are taken in this article as reference errors to study the performance of the FIR methodology in the same problem.

Table 8. Prediction errors (SE) obtained by classical methods and hybrid evolutionary techniques.

Method	SE_{train}	SE_{test}
Linear	164,662	36,819
Second-order polynomial	103,032	45,332
Three-layer perceptron 4-5-1	86,469	33,105
GA-P	18,168	21,884
Interval GA-P	16,263	18,325
WM fuzzy model	20,318	27,615
Mamdani fuzzy model	19,679	22,591
TSK fuzzy model ($\alpha = 0$)	25,579	26,450
TSK fuzzy model ($\alpha = 0,2$)	11,074	11,836

5.2 *Learning the optimal number of classes for each system variable (GA1)*

In this section, the GA1 proposed for learning the number of classes for each system variable is evaluated for the problem at hand. Thirty executions of the GA1 are performed for each objective function and stop criteria. The GA1 is executed to obtain the optimal number of classes for the input and output variables. The EFP is used as the default method to obtain the membership functions of the classes. Table 9 shows the results obtained for the problem of computing the maintenance costs of medium voltage lines when both cost functions, i.e. 1−Q and prediction MSE of a portion of the training data set are used. In this application, the last 20% of the training signal is used for MSE_{train} objective function evaluation and only the first 80% of the signal is used to obtain the FIR models (masks and pattern rule bases).

Table 9 is organized as follows. The first column indicates the number of chromosome evaluations made by the GA1. The second column indicates the partition suggested by the GA1 when its execution finalizes. Remember that the partition suggested by the GA1 is the first input parameter of the *fuzzification* process of the FIR methodology. The third and fourth columns present the optimal mask (in position notation) obtained by FIR for this specific partition and the quality associated to this mask, respectively. The fifth column corresponds to the evaluation of the objective function used: 1−Q or MSE_{train}. The next column shows the prediction SE error obtained in the test data set. The last column indicates the CPU time (in seconds) used by the GA. The GA1 suggests a single partition for 1−Q objective function and three different partitions for MSE_{train} objective function, when it reaches the 2000 evaluations. When 1000 evaluations are used as the stop

Table 9. Number of classes results of the electrical distribution problem using: (a) $1-Q$ and, (b) prediction error of the last 20% of the training data set (MSE_{train}) cost functions. EFP method for the membership functions. (X means any value between the range [2.9]).

$1-Q$	# Eval	Partition	Opt. mask	Q	$1-Q$	SE_{test}	Time
	2000	(X,X,3,2,2)	(3,4,5)	0.8365	0.1635	5111	85.42
	1000	(X,X,3,2,2)	(3,4,5)	0.8365	0.1635	5111	55.46
	1000	(2,2,2,6,2)	(3,4,5)	0.8308	0.1692	2805	90.09
MSE_{train}	# Eval	Partition	Opt. mask	Q	MSE_{train}	SE_{est}	Time
	2000	(X,X,4,9,7)	(3,4,5)	0.5704	0.1625	3305	150.29
	2000	(X,X,4,9,4)	(3,4,5)	0.6417	0.1667	3379	155.04
	2000	(5,3,7,9,7)	(3,4,5)	0.5742	0.1696	3572	159.33
	1000	(X,X,4,9,7)	(3,4,5)	0.5704	0.1625	3305	75.59
	1000	(X,X,4,9,8)	(3,4,5)	0.5452	0.1648	3375	80.42
	1000	(7,6,7,9,7)	(3,4,5)	0.5742	0.1696	3572	83.55

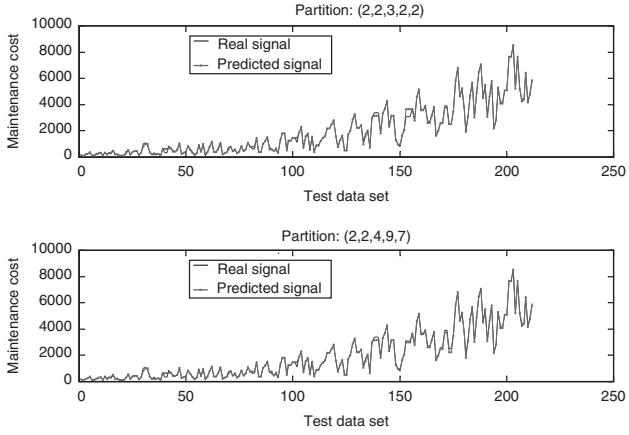

Figure 5. Predictions of the test data set when the best FIR models of the AG1 with $1-Q$ (upper) and MSE_{train} (lower) objective functions are used.

criteria, the GA1 suggests two and three different partitions for the objective functions $1-Q$ and MSE_{train}, respectively. The partitions encountered by the GA1 have very similar cost function values; therefore, they can be considered good solutions, although we do not know the optimal one because an exhaustive search is computationally very expensive. It is interesting to notice that in almost all the partitions suggested by the GA1, the first two input variables, i.e. sum of the lengths of all streets in the town (SLS) and total area of the town (TA), are not considered relevant for the determination of the maintenance cost of the line (MC). FIR encountered that no important causal relation exists between SLS and TA with respect to the output variable MC. Looking closer at table 9, it can be seen that the models obtained by FIR enhanced by the GA1 have a much better performance than the techniques presented in table 8. The lowest SE_{test} obtained in previous works is 11,836, whereas the highest obtained by FIR is 5111. The error reduction is very significant. Therefore, FIR is able to obtain more reliable and precise models for the computation of the maintenance costs of medium voltage lines.

It is also important to analyze the results obtained by the two cost functions used in the GA1, i.e. $1-Q$ and MSE_{train} of the last 20% of the training data set. If we look closer at table 9, it can be observed that the SE_{test} values are in almost every case lower when the MSE_{train} cost function is used. As expected, the CPU time has increased with respect to the $1-Q$ objective function, due to the fact that the *forecast* process of FIR methodology needs to be executed for each partition evaluated. However, notice that the increment of the CPU time is not really significant because in this application the optimal masks encountered for the suggested partitions are very simple (the output only depends of two inputs) and, therefore, the forecasting process is very fast. Figure 5 presents the best predictions obtained for the two cost functions studied. The upper plot shows real and predicted test signals when the partition (2,2,3,2,2) suggested by the GA1 with the $1-Q$ objective function is used, whereas the lower plot shows these signals when using the partition (2,2,4,9,7) suggested by the GA1 with the cost function MSE_{train}. As can be seen in figure 5, the predictions signals obtained by FIR models follow the real maintenance cost signal very accurately.

A new experiment is included (only in this section) to justify the optimization ability of the GAs proposed in this work. Now, we compare the results obtained by GA1 with the solutions generated when a random search of the number of classes for each system variable is performed. In this experiment, 150 solutions are randomly generated for each objective function to evaluate. Repeated solutions are allowed. Tables 10 and 11 show the best and worse random solutions

Table 10. Number of classes results of the electrical distribution problem using *random search*. $1-Q$ cost function.

Partition	Opt. mask	Q	$1-Q$	SE_{test}
(6,3,2,5,2)	(3,4,5)	0.8266	0.1734	2805
(5,8,6,3,2)	(1,5)	0.6666	0.3334	460,432
Mean			0.4000	23430.90
SD			0.0736	74843.21
Median			0.4084	2825.79

Table 11. Number of classes results of the electrical distribution problem using *random search*. MSE_{train} cost function.

Partition	Opt. mask	Q	MSE_{train}	SE_{test}
(5,2,4,9,9)	(3,4,5)	0.5433	0.1650	3342
(8,9,8,3,2)	(2,5)	0.6799	15.235	270791
Mean			1.7254	30874.29
SD			4.3662	75857.24
Median			0.3315	7294.75

Table 12. Statistics of the GA1 for the electrical distribution problem, for both objective functions.

$1-Q$	# Eval	Statistic	$1-Q$	SE_{test}
	2000	Mean	0.1635	5111
		SD	0	0
		Median	0.1635	5111
	1000	Mean	0.1637	5033.93
		SD	0.00104	421.03
		Median	0.1635	5111
MSE_{train}	# Eval	Statistic	MSE_{train}	SE_{test}
	2000	Mean	0.1630	3318.77
		SD	0.00164	51.46
		Median	0.1625	3305
	1000	Mean	0.1649	3621.50
		SD	0.00758	1150.70
		Median	0.1625	3305

obtained for $1-Q$ and MSEtrain objective functions, respectively. The mean, standard deviation and median of the 150 random solutions generated are also presented in each table. Table 12 presents the mean, standard deviation and median of the results obtained by GA1 for the 30 executions performed, for each evaluation number and objective function. As expected, random solutions have larger $1-Q$ and MSEtrain mean values than the results obtained by GA1, i.e. 0.4 vs. 0.16 and 1.72 vs. 0.16, respectively. The SEtest mean values, for both objective functions, when the random search is performed are also much larger than the ones obtained when GA1 is used to find the partition ($1-Q$: 23,420 vs. 5111; MSEtrain: 30,874 vs. 3621). It can also be seen from tables 10 and 11 that the SEtest standard deviations are very high, showing big differences in the errors of the solutions obtained by the random search. This is not the case when GA1 is used. As can be seen from table 12, the standard deviations low values show the consistency of the solutions, in the sense that optimal or suboptimal solutions are always encountered.

5.3 *Learning the membership functions of the classes (GA2)*

This section presents the results obtained when the GA2, proposed for learning the membership functions for each class, is evaluated for the problem at hand. As in the GA1, 30 executions are performed for each objective function and stop criteria. The GA2 considers the optimal number of classes obtained by the previous GA, i.e. GA1.

The results are presented in two sections, based on the objective function used for the evaluation of the chromosomes.

5.3.1 $1-Q$ objective function. Table 13 shows the results obtained when $1-Q$ was used as objective function. The table is organized as follows. The first column is divided in three sections. Section A corresponds to the best result obtained by the GA2 while Section B corresponds to its worse result. Section C shows the results obtained when the GA is not used, i.e. when the EFP method (default value) is used to determine the membership function of each class. Therefore, C is taken as a reference. The second column shows the number of chromosome evaluations of the GA. The third column shows the data proportion for the input variables (SLS, TA, AB, ES) and the output variable (MC). The number of elements of the data proportion corresponds to the number of classes for that variable (shown in the top of the tables). The data proportion is the output of the GA2 and the second input parameter of the *fuzzification* process of the FIR methodology. The fourth column presents the optimal mask, in position notation, encountered by FIR when the data proportion obtained is used to set the landmarks. The fifth column corresponds to the quality associated to the optimal mask. The sixth column is the value of the $1-Q$ objective function. The last column shows the prediction error SE obtained for test data set.

5.3.2 MSE_{train} objective function. Table 14 shows the results obtained when the objective function is defined as the prediction MSE of a portion of the training data set. As before, the last 20% of the training signal is used for objective function evaluation and only the first 80%

Table 13. Membership functions results of the electrical distribution problem using $1 - Q$ cost function. GA1 results for the number of classes.

Partition used: (2,2,3,2,2)						
	# Eval	Data proportion	Opt. mask	Q	$1 - Q$	SE_{test}
A	20000	SLS:(0.85,0.15) TA:(0.49,0.51) AB: (0.79,0.13,0.08) ES:(0.81,0.19) MC:(0.85,0.15)	(1,3,4,5)	0.9475	0.0525	2727
	10000	SLS:(0.92,0.08) TA:(0.58,0.42) AB: (0.82,0.10,0.08) ES:(0.85,0.15) MC:(0.85,0.15)	(1,3,4,5)	0.9505	0.0495	2728
B	20000	SLS:(0.52,0.48) TA:(0.51,0.49) AB: (0.35,0.24,0.41) ES:(0.46,0.54) MC:(0.49,0.51)	(1,3,4,5)	0.8886	0.1114	5094
	10000	SLS:(0.51,0.49) TA:(0.46,0.54) AB: (0.31,0.26,0.43) ES:(0.53,0.47) MC:(0.58,0.42)	(1,3,4,5)	0.8723	0.1277	5063
C		Equal frequency partition method	(3,4,5)	0.8365	0.1635	5111

Table 14. Membership functions results of the electrical distribution problem using MSE_{train} cost function. GA1 results for the number of classes.

Partition used: (2,2,4,9,7)						
	# Eval	Data proportion	Opt. mask	Q	MSE_{train}	SE_{test}
A	20000	SLS:(0.52,0.48) TA:(0.59,0.41) AB (0.19,0.33,0.29,0.19) ES:(0.11,0.12,0.12,0.12,0.12,0.14,0.08,0.08,0.11) MC:(0.17,0.07,0.11,0.19,0.08,0.25,0.13)	(3,4,5)	0.5245	0.1228	2973
	10000	SLS:(0.56,0.44) TA:(0.49,0.51) AB (0.19,0.35,0.28,0.18) ES:(0.11,0.11,0.14,0.12,0.11,0.14,0.09,0.07,0.11) MC:(0.09,0.08,0.09,0.09,0.26,0.31,0.08)	(3,4,5)	0.5343	0.1246	3028
B	20000	SLS:(0.40, 0.60) TA:(0.55,0.45) AB: (0.27,0.27,0.23,0.23) ES:(0.13,0.11,0.11,0.08, 0.10,0.13,0.15,0.14,0.05) MC:(0.12,0.11,0.20,0.12,0.21,0.05,0.19)	(3,4,5)	0.5741	0.1306	3005
	10000	SLS:(0.52,0.48) TA:(0.58,0.42) AB: (0.34,0.26,0.22,0.18) ES:(0.10,0.11,0.12,0.11,0.18,0.14, 0.10,0.05,0.09) MC:(0.19,0.24,0.17,0.12,0.15,0.08,0.05)	(3,4,5)	0.5465	0.1332	2985
C		Equal frecuency partition method	(3,4,5)	0.5704	0.1625	3305

of the signal is used to obtain the FIR models (masks and pattern rule bases). The table is organized as table 13. The only difference is that the sixth column contains the values of the MSE_{train} instead of the $1-Q$ objective cost function.

As explained before, the CPU time needed by the GA2 when the MSE_{train} objective function is used is clearly greater than the time needed when the $1-Q$ is used. The computational time needed to perform 30 executions for 20,000 evaluations when the $1-Q$ and MSE_{train} objective functions are studied is 6:36 and 11:39 hours, respectively, in a Pentium III computer (0.6 GHz).

The worse results obtained using the GA2 (row B in tables 13 and 14) are always better than the reference result (row C of each table). Therefore, although the GA2 does not assure the optimal solution, all the solutions suggested are better than the ones obtained when no GA is used, i.e. when the default EFP method is used to obtain the membership functions.

Once again, the errors obtained by FIR methodology, in this case, are significantly lower than the ones obtained by the methodologies of table 8. The best result of 11,836 SE obtained by the TSK fuzzy model is much bigger than the 2727 SE obtained by FIR methodology enhanced by the GA2. As expected, the results obtained by FIR enhanced by both GAs (GA1 and GA2) (tables 13 and 14) are better than the ones obtained when FIR is enhanced only by the GA1 (table 9). From tables 13 and 14 it can also be seen that the results obtained by both cost functions are equivalent. In this case, the performance of FIR models when the MSE_{train} cost function is used is not superior to the performance of $1-Q$ objective function. Therefore, the $1-Q$ objective function is preferable because of its lower computational cost. The graphical representation of the best results presented in tables 13 and 14 is quite similar to the plots shown in figure 5 and, therefore, it is not presented here.

6. Conclusions

A FIR model is a qualitative, non-parametric, shallow model based on fuzzy logic. Therefore, variations on fuzzy partitions have a direct effect on the performance of the model identification and prediction processes of FIR methodology. In this article, two GA were designed to learn fuzzy partitions (one for the number of classes and another for the membership functions) in the context of the FIR methodology.

In this article, two real applications have been studied, i.e. the human CNS and the estimation of the maintenance cost of medium voltage lines in Spanish towns. For each application studied, two objective functions have been evaluated and compared from the perspective of their performance and computational time.

The results obtained in the CNS application were much better than the ones obtained by other inductive methodologies like NARMAX, time delay neural networks and recurrent neural networks.

On the other hand, the performance when modeling the maintenance cost of medium voltage lines is also superior compared to the performance of other methodologies presented in previous works like linear models, second-order polynomial models, neural networks, hybrid GP and different fuzzy models, for the same problem.

The results show that better models are obtained when FIR methodology is enhanced with the two GAs proposed for learning the number of classes and the membership functions. The GAs suggested provide an efficient, computationally effective and flexible way to optimize fuzzy inductive reasoning models, yet maintaining an acceptable relation between model complexity and design effort.

The next step is the development of a GA that allows the determination of the number of classes and the membership functions at the same time. It is expected that a unique algorithm will achieve better results; however the CPU time will increase considerably, due to the fact that now the search space will be much bigger.

Acknowledgments

This research was supported by Spanish Consejo Interministerial de Ciencia y Tecnología (CICYT), under project DPI2002-03225.

References

G. Alpaydin, G. Dündar and S. Balktr, ''Evolution-based design of neural fuzzy networks using self-adapting genetic parameters'', *IEEE Trans. Fuzzy Syst*., 2, pp. 211–21, 2002.

J.E. Baker, ''Reducing bias and inefficiency in the selection algorithm'', in *Proceedings of the 2nd International Conference on Genetic Algorithms* (*ICGA'87*), Hillsdale, 1987, pp. 14–21.

P.P. Bonissone, ''Soft computing: the convergence of emerging reasoning technologies'', *Soft Comput*., 1, pp. 6–18, 1997.

H.A. Camargo, M.G. Piresand P.A. D. Castro, ''Genetic design of fuzzy knowledge bases – a study of different approaches'', in *Proceedings of IEEE Annual Meeting of the North American Fuzzy Information Processing Society* (*NAFIPS'04*), Alberta, Canada, 2004, pp. 954–9.

P. Carmona, J.L. Castro and J.M. Zurita, ''Strategies to identify fuzzy rules directly from certainty degrees: a comparison and a proposal'', *IEEE Trans. Fuzzy Syst*., 5, pp. 631–40, 2004.

J. Casillas, O. Cordón, M.J. del Jesús and F. Herrera, ''Genetic tuning of fuzzy rule deep structures preserving interpretability and its interaction with fuzzy set reduction'', *IEEE Trans Fuzzy Syst*, 1, pp. 13–29, 2005.

S.M. Chen, I.J. Horng and C.H. Lee, "Document retrieval using fuzzy-valued concept networks", *IEEE Trans. Syst. Man Cybern B*, 1, pp. 111–18, 2001.

Z. Chi, H. Yan and T. Pham, *Fuzzy Algorithms: With Applications to Image Processing and Pattern Recognition*, Singapore: World Scientific, 1996.

I.F. Chung, C.J. Lin and C.T. Lin, "A GA-based fuzzy adaptive learning control network", *Fuzzy Sets Syst*., 1, pp. 65–84, 2000.

O. Cordón, F. Gomide, F. Herrera, F. Hoffmann and L. Magdalena, "Ten years of genetic fuzzy systems: current framework and new trends", *Fuzzy Sets Syst*., 1, pp. 5–31, 2004.

O. Cordón, F. Herrera and L. Sánchez, "Computing the Spanish medium electrical line maintenance costs by means of evolution-based learning processes", in *Proceedings of the 11th International Conference on Industrial and Engineering Applications of AI and ES* (*IEA-98-AIE*), Castellón, Spain, 1998, pp. 478–82.

O. Cordón, F. Herrera and L. Sánchez, "Solving electrical distribution problems using hybrid evolutionary data analysis techniques", *Appl. Intell*., 10, pp. 5–24, 1999.

O. Cordón, F. Herrera, F. Hoffmann and L. Magdalena, *Genetic Fuzzy Systems. Evolutionary Tuning and Learning of Fuzzy Knowledge Bases*, Singapore: World Scientific, 2001.

J. Cueva, R. Alquézar and A. Nebot, "Experimental comparison of fuzzy and neural network techniques in learning models of the central nervous system control", in *Proceedings of the 5th European Congress on Intelligent Techniques and Soft Computing* (*EUFIT'97*), Aachen, Germany, 1997, pp. 1014–18.

M.J. del Jesús, F. Hoffman, I.J. Navascués and L. Sánchez, "Induction of fuzzy-rule-based classifiers with evolutionary boosting algorithms", *IEEE Trans. Fuzzy Syst*., 3, pp. 296–308, 2004.

Y. Dote and S.J. Ovaska, "Industrial applications of soft computing: a review", in *Proc. IEEE*, 9, pp. 1243–65, 2001.

D. Driankov, H. Hellendoorn and M. Reinfrank, *An Introduction to Fuzzy Control*, New York: Springer-Verlag, 1993.

A. González and R. Pérez, "An experimental study about the research mechanism in SLAVE learning algorithm: hill-climbing methods versus genetic algorithm", *Inform. Sci*., 1–4, pp. 159–74, 2001.

J.J. Grefenstette, "Optimization of control parameters for genetic algorithms", in *IEEE Trans. Syst. Man Cybern*, 1, pp. 122–8, 1986.

R. Gudwin, F. Gomide and W. Pedrycz, "Context adaptation in fuzzy processing and genetic algorithms", *Int. J. Intell. Syst*., 10–11, pp. 929–48, 1998.

H.B. Gürocak, "A genetic-algorithm-based method for tuning fuzzy logic controllers", *Fuzzy Sets Syst*., 1, pp. 39–47, 1999.

L.O. Hall, J.C. Bezdek, S. Boggavarpu and A. Bensaid, "Genetic fuzzy clustering", in *Proceedings of NAFIPS'94*, San Antonio, pp. 411–5, 1994.

H. Heider and T. Drabe, "A cascaded genetic algorithm for improving fuzzy-system design", *Int. J. Approx. Reason*., 4, pp. 351–68, 1997.

F. Herrera, M. Lozano and J.L. Verdegay, "Tuning fuzzy controllers by genetic algorithms", *Int. J. Approx. Reason*., 3–4, 299–315.

K. Hirota and M. Sugeno, *Industrial Applications of Fuzzy Technology in The World*, Singapore: World Scientific, 1995.

F. Hoffmann and G. Pfister, "Evolutionary design of a fuzzy knowledge base for a mobile robot", *Int. J. Approx. Reason*, 4, pp. 447–69, 1997.

F. Hoffmann, "Evolutionary algorithms for fuzzy control system design", *Proc IEEE*, 9, pp. 1318–33, 2001.

J. Holland, *Adaptation in Natural and Artificial Systems*, Ann Arbor: The University of Michigan Press, 1975.

L. Howeard and D. D'Angelo, "The GA-P: a genetic algorithm and genetic programming hybrid", *IEEE Expert*., 10, pp. 11–15, 1995.

H. Ishibuchi, T. Nakashima and T. Murata, "Performance evaluation of fuzzy classifier systems for multidimensional pattern classification problems", *IEEE Trans. Syst. Man Cybern B*, 5, pp. 601–18, 1999.

J.S. R. Jang, J.S.R. Sun and E. Mizutani, *Neuro-fuzzy and Soft Computing*, New York: Prentice-Hall, 1997.

A. Jerez and A. Nebot, "Genetic algorithms versus classical search techniques for identification of fuzzy models", in *Proceedings of the 5th European Congress on Intelligent Techniques and Soft Computing* (*EUFIT'97*), Aachen, Germany, 1997, pp. 769–73.

G. Klir, *Architecture of Systems Problem Solving*, New York: Plenum Press, 1985.

L.I. Kuncheva, *Fuzzy Classifier Design*, Heidelberg: Physica-Verlag, 2000.

A. Law and D. Kelton, *Simulation Modeling and Analysis*, New York: McGraw-Hill, 1990.

C.T. Leondes, *Fuzzy Theory Systems: Techniques and Applications*, San Diego: Academic Press, 1999.

L. Magdalena, "Adapting the gain of an FLC with genetic algorithms", *Int. J. Approx. Reason*., 4, pp. 327–49, 1999.

Z. Michalewicz, *Genetic Algorithms+Data Structures=Evolution Programs*, New York: Springer-Verlag, 1996.

S. Miyamoto, "Two approaches for information retrieval through fuzzy associations", *IEEE Trans. Syst. Man. Cybern*., 1, pp. 123–30, 1989.

F. Mugica and F.E. Cellier, "Automated synthesis of a fuzzy controller for cargo ship steering by means of qualitative simulation", in *Proceedings of the European Simulation MultiConference* (*ESM'94*), Barcelona, 1994, pp. 523–8.

A. Nebot, "Qualitative modeling and simulation of biomedical systems using fuzzy inductive reasoning". PhD thesis, Dept. Llenguatges i Sistemes Informàtics, Universitat Politècnica de Catalunya, Barcelona, Spain (1994).

A. Nebot, F. Cellier and D. Linkens, "Synthesis of an anaesthetic agent administration system using fuzzy inductive reasoning", *Artif Intell Med*, 3, pp. 147–66, 1996.

A. Nebot, F. Cellier and M. Vallerdú, "Mixed quantitative/qualitative modeling and simulation of the cardiovascular system", *Comput. Meth. Prog. Biomed*., 55, pp. 127–55, 1998.

A. Nebot, F. Mugica and P. Gómez, "Long term prediction of maximum ozone concentration using fuzzy inductive reasoning", in *Proceedings of the European Symposium on Intelligent Technologies, Hybrid Systems and their implementation on Smart Adaptive Systems (EUNITE'01)*, Tenerife, 2001, pp. 91–101.

W. Pedrycz, *Fuzzy Modelling: Paradigms and Practice*, Norwell, MA: Kluwer Academic Press, 1996.

H. Pomares, I. Rojas, J. González, M. Damas, B. Pino and A. Prieto, "Online global learning in direct fuzzy Controllers", *IEEE Trans. Fuzzy Syst*., 2, pp. 218–29, 2004.

M. Russo, "FuGeNeSys: a fuzzy genetic neural system for fuzzy modeling", *IEEE Trans. Fuzzy Syst*., 3, pp. 373–88, 1999.

F. Sattar and D.B.H. Tay, "Enhancement of document images using multiresolution and fuzzy logic techniques", *IEEE Signal Process Lett*., 10, pp. 249–52, 1999.

M. Schoenauer and Z. Michalewicz, "*Evolutionary Computation, Control and Cybernetics*", 26, pp. 307–338, 1997.

Y. Suzuki, K. Itakura, S. Saga and J. Maeda, "Signal processing and pattern recognition with soft computing", *Proc. IEEE*, 9, pp. 1297–317, 2001.

P.P. Thrift, "Fuzzy logic synthesis with genetic algorithms", in *Proceedings of the 4th International Conference on Genetic Algorithms* (*ICGA'91*), San Diego, 1991, pp. 509–13.

T. Van Le, "Evolutionary fuzzy clustering", in *Proceedings of the 2nd IEEE Conference on Evolutionary Computation (ICEC'95)*, 2, 1995, pp. 753–8.

V. Vapnik, *Statistical Learning Theory*, New York: Wiley & Sons, 1998.

B. Yuan, G.J. Klir and J.F. Swan-Stone, "Evolutionary fuzzy c-means clustering algorithm", in *Proceedings of the 4th IEEE International Conference on Fuzzy Systems* (*FUZZ-IEEE'95*), Yokohama, 1995, pp. 2221–6.

Jesús Acosta was born in Coro-Estado Falcón, Venezuela. He received the Licenciatura degree in information engineering (cum laude) in 1995 from the Universidad Tecnológica del Centro (UNITEC), Guacara, and the M.S. degree in process engineering (cum laude) in 2000 from the Universidad Nacional Experimental Politécnica "Antonio José de Sucre" (UNEXPO), Barquisimeto. At the moment, he is pursuing the Ph.D. degree from Polytechnical University of Catalonia (UPC), Barcelona (Spain), in control engineering. He joined the Departament of *Instrumentación Industrial*, IUTAG, in 1996 as an Assistant Professor and since April 1999 he has been Permanent Professor. He performed research in optimization of fuzzy partitions for inductive reasoning at the Soft Computing Group, UPC. His research interests are in areas including control, fuzzy logic, genetic algorithms, pattern recognition and machine learning.

Mr Acosta received a PhD fellowship in 2001–2003 from the Agencia Española de Cooperación Internacional (Spain) and is currently a research fellow at the UPC. He is a member of IEEE Student since 2004.

Àngela Nebot received her BS and PhD degrees in computer science from the Polytechnical University of Catalonia (UPC), Barcelona, Spain, in 1988 and 1994, respectively. From 1988 to 1992 she was a research student at the *Institut de Cibernètica* (Spanish Consejo Superior de Investigaciones Científicas), Barcelona, holding a pre-doctoral research grant from the Government of Catalonia and finishing the doctoral courses on software and artificial intelligence. She joined the Department of *Llenguatges i Sistemes Informàtics*, UPC, in 1994 as an Assistant Professor, and since March 1998 she has been Associate Professor in the same department. She is currently the head of the Soft Computing group of the UPC. Her current research interests include fuzzy systems, neuro-fuzzy systems, genetic algorithms, simulation and e-learning.

Dr Nebot is an associate editor of the journal "Simulation: Transactions of the Society for Modeling and Simulation International" and collaborates as a reviewer with other international journals like "IEEE Transactions on Fuzzy Systems", "International Journal of General Systems", "Neurocomputing", "Artificial Intelligence in Environmental Engineering" and "Artificial Intelligence Communication".

Pedro Villar was born in Granada, Spain, on August 15, 1968. He received his MS degree in computer science in 1991 from the University of Granada, Spain and his PhD in computer science in 2000 from the University of Vigo, Spain. He joined the Department of *Lenguajes y Sistemas Informáticos* at the University of Vigo in October 1991, as an Assistant Professor, and since May 2002 until October 2004 he was an Associate Professor in the same department. Currently, he is an Assistant Professor in the Department of Software Engineering at the University of Granada. His current main research interests are genetic fuzzy systems, evolutionary algorithms and multi-objective genetic algorithms.

Josep M. Fuertes was born in Barcelona, received his Industrial Engineering in 1976, his PhD degree in 1986, and became Permanent Professor at the Polytechnical University of Catalonia (UPC) in 1987. He was *Researcher* (1975–1986) at the Institut de Cibernètica (Spanish Consejo Superior de Investigaciones Científicas). In 1987, he got a position for a year at the Lawrence Berkeley Laboratory working as *Visiting Scientific Fellow* in the design of the Active Control System of the W.M. Keck 10 meter segmented telescope (Hawaii). From 1992 to 1999, he was the director of the University Research Line in Advanced Control Systems. From 2001 to 2005, he was the director of the Department of *Enginyeria de Sistemes, Automàtica i Informàtica Industrial*, also at UPC.

Dr Fuertes has collaborated as organizer (IEEE-WFCS'97, IEEE-ETFA'99), chairman, session organizer, or member of program committee in several international conferences and has participated with many papers in the areas of Intelligent Control Systems and of Distributed Control Systems and Applications. From 1989, he has coordinated projects at national and international level, related with the above areas of expertise. He acted (1996–2001) as Spanish *Representative* at the Council of the European Union Control Association (EUCA). He is a member of the ADCOM at the IEEE IES (Industrial Electronics Society); He is also a member of CEA-IFAC (Comité Español de Automática, International Federation of Automatic Control) and other scientific and technical societies.