On the Use of Distributed Genetic Algorithms for the Tuning of Fuzzy Rule Based-Systems

Ignacio Robles, Rafael Alcalá, José M. Benítez, and Francisco Herrera

Abstract. The tuning of Fuzzy Rule-Based Systems is often applied to improve their performance as a post-processing stage once an appropriate set of fuzzy rules has been extracted. This optimization problem can become a hard one when the size of the considered system in terms of the number of variables, rules and, particularly, data samples is big. Distributed Genetic Algorithms are excellent optimization algorithms which exploit the nowadays available parallel hardware (multicore microprocessors and clusters) and could help to alleviate this growth in complexity.

In this work, we present a study on the use of the Distributed Genetic Algorithms for the tuning of Fuzzy Rule-Based Systems. To this end, we analyze the application of a specific Gradual Distributed Real-Coded Genetic Algorithm which employs eight subpopulations in a hypercube topology.

The empirical performance in solution quality and computing time is assessed by comparing its results with those from a highly effective sequential tuning algorithm. We applied both, the highly effective sequential algorithm and the distributed method, for the modeling of four well-known regression problems. The results show that the distributed approach achieves better results in terms of quality and execution time as the complexity of the problem grows.

Keywords: Genetic Fuzzy Systems, Fuzzy Rule Based-Systems, Distributed Genetic Algorithms, Genetic Tuning.

1 Introduction

Fuzzy rule based-systems (FRBS) have become a wide choice when addressing modeling and system identification problems [1, 2, 3, 4]. One of the most popular

Ignacio Robles · Rafael Alcalá · José M. Benítez · Francisco Herrera
Dept. of Computer Sciences and Artificial Intelligence,
University of Granada, 18071-Granada, Spain
e-mail: ignaciorobles@gmail.com,
{alcala,J.M.Benitez,herrera}@decsai.ugr.es

F. Fernández de Vega, E. Cantú-Paz (Eds.): Paral. and Distrib. Comp. Intel., SCI 269, pp. 235–261.
springerlink.com

approaches for the design of FRBSs is the hybridization between fuzzy logic [5, 6] and Genetic Algorithms (GAs) [7, 8] leading to the well-known Genetic Fuzzy Systems (GFSs) [9, 10, 11]. A GFS is basically a fuzzy system augmented by a learning process based on evolutionary computation, which includes GAs, genetic programming, and evolutionary strategies, among other evolutionary algorithms [12].

The predominant type of GFS is that focused on FRBSs, since the automatic definition of FRBSs can be seen as an optimization or search problem, and GAs are a well known and widely used global search technique with the ability to explore a large search space for suitable solutions only requiring a performance measure. In addition to their ability to find near optimal solutions in complex search spaces, the generic code structure and independent performance features of GAs make them suitable candidates to incorporate a priori knowledge. In the case of FRBSs, this a priori knowledge may be in the form of linguistic variables [13], fuzzy membership function (MF) parameters, fuzzy rules, number of rules, etc. These capabilities extended the use of GAs in the development of a wide range of approaches for designing FRBSs over the last few years.

In this framework, a widely-used technique to enhance the performance of FRBSs is the genetic tuning of MFs [14, 15, 16, 17, 18, 19]. It consists of improving a previous definition of the Data Base (DB) once the Rule Base (RB) has been obtained. The classic approaches to perform genetic tuning [14, 15] consist of using a GA in order to refine the definition parameters that identify the MFs associated to the linguistic terms comprising the initial DB.

Since the real aim of the genetic tuning process is to find the best global configuration of the MFs and not only to find independently specific ones, this optimization problem can become a hard one when the size of the considered system in terms of the number of variables, rules and, particularly, data samples (typically used to guide the search) is big. Moreover, the computing time consumed by these approaches grows with the complexity of the search space.

In order to deal with this complexity, Distributed Genetic Algorithms (DGAs) [20, 21, 22] are found to be excellent optimization algorithms for high dimensional problems. They are able to take advantage of the parallel hardware and software that has become very affordable and broadly available nowadays. Clear examples in this line are multicore processors and linux clusters [23, 24, 25]. This situation makes them perfect to deal with complex search spaces.

In this work, we present a study on the use of the Distributed Genetic Algorithms for the tuning of FRBS from two points of view: solution quality and computing time improvements. To this end, we analyze the application of a specific Gradual Distributed Real-Coded Genetic Algorithm (GDRCGA) to perform an effective genetic tuning of FRBSs [26]. This algorithm employs eight subpopulations in a hypercube topology [27] and makes use of a particular linguistic rule representation model that was proposed in [17] to perform a genetic *lateral tuning of MFs*. This approach is based on the linguistic 2-tuples representation [28] which simplifies the search space by considering only one parameter per MF and, therefore, eases the derivation of optimal models, particularly in complex or high-dimensional problems.

The empirical performance in solution quality and computing time has been assessed by comparing the results of the distributed approach with those obtained from the specialized sequential algorithm, proposed in [17], to perform a lateral tuning of the MFs. Both methods are applied for the modeling of four well-known regression problems. The results show that the distributed approach achieves better results in terms of quality and execution time as the complexity of the problem grows.

The contribution is structured as follows: in the second section DGAs are presented and briefly discussed. In the third section, we present a brief introduction to FRBSs. Next, lateral tuning of FRBSs problem is stated and an efficient sequential specialized algorithm is reviewed. The fifth section describes the DGA used for FRBS tuning. An empirical evaluation of the distributed algorithm is presented in the sixth section. Finally, we close this chapter with some conclusions and final remarks.

2 Distributed Genetic Algorithms

The availability of extremely fast and low cost parallel hardware in the last few years benefits the investigation on new approaches to existing optimization algorithms. The key of these new approaches is achieving gains not only in time, which is somehow inherent to parallel computation, but also gains in quality of the solutions found.

Generally, there are two ways to parallelize GAs. The first way is by means of local parallelization: fitness evaluation of the individuals and, sometimes, the application of the genetic operators are carried out in a parallel way [29, 30]. The second way is by means of global parallelization: complete subpopulations evolve in parallel [31, 32, 33, 34, 35, 36, 27, 37] (distributed approach or DGAs). While the first one is only achieving gains in time, the second one is also able to improve the global performance of the underlying algorithm, subsequently achieving additional gains in the quality of the final solutions. In fact, DGAs [20, 22] are excellent optimization algorithms and have proven to be an interesting approach when trying to cope with large scale problems and when the classic approaches take too much time to give a proper solution.

In this section, our goal is to present an introductory vision of the distributed models. Firstly, we present a taxonomy of the state of the art of DGAs. Finally, the key elements to obtain a well-designed DGA are presented.

2.1 Taxonomy of Distributed Genetic Algorithms

Several categorizations of DGAs can be found in literature [20, 21, 22] according to a wide range of criteria. Some of the most used categories when referring to DGAs are:

- **According to the migration policy:**
  - **Isolated:** no migrations between subpopulations. These DGAs are also known as *Partitioned Genetic Algorithms* [31].
  - **Synchronous:** migrations between subpopulations are synchronized, for example, they are carried out at the same time [31, 32].

  - **Asynchronous:** migrations are carried out when some events occur, generally related to the activity of subpopulations [33].

- **According to the connection schema:**
  - **Static schema:** connections between subpopulations are stablished at the start of the execution and they are not modified.
  - **Dynamic schema:** connection topology changes dynamically along the execution of the algorithm. Connection reconfigurations may occur depending on the degree of evolution of the subpopulations.

- **According to the homogeneity:**
  - **Homogeneous:** genetic operators are the same for all subpopulations as well as parameters, fitness function, coding scheme, etc. The vast majority of DGAs proposed in the literature are homogeneous.
  - **Heterogeneous:** subpopulations are all alike [34, 35, 36]. They can differ from the parameters used, genetic operators, coding scheme, etc. One example of these heterogeneous GAs are the *Gradually Distributed Genetic Algorithms* where genetic operators are applied with different intensities [27].

- **According to the granularity:**
  - **Coarse-grained parallelization:** The population is split into small subpopulations that are assigned to different processors. Each subpopulation evolves independently and simultaneously according to a GA. Periodically, a *migration operator* exchanges individuals among subpopulations, which gives them some additional diversity.
  - **Fine-grained parallelization:** The population is split into a big number of small subpopulations. Generally only one subpopulation is assigned to each processor. The selection and crossover operators are applied considering adjacent individuals. For example, each individual chooses its best neighbor for crossover (see Figure 1) and the resulting individual replaces the original one. When a single individual is assigned to each processor, this type of algorithms are known as **Cellular Genetic Algorithms** [37].

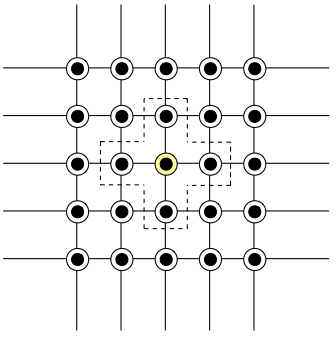

Fig. 1. Cellular Genetic Algorithm: a extreme case of fine-grained parallelization

2.2 Design of Distributed Genetic Algorithms

There are two classic problems [27] in DGAs. A main drawback in DGAs is that the insertion of a new individual coming from a different subpopulation may not be effective. The new individual could be highly incompatible with the receiving subpopulation and therefore it might be ignored or conquer the subpopulation. This probably happens when subpopulations involved are at different stages of evolution.

The arrival of a highly evolved individual comming from a *strong* subpopulation will result in a higher selection ratio than for local individuals which are less evolved. In this way, the subpopulation that sends the highly evolved individual is imposing it to the receiving subpopulation. This problem is known as the *Conquest Problem*.

Symmetrically, when a less evolved individual migrates to a highly evolved subpopulation it will not be selected for reproduction and therefore it will be abandoned. This means a waste of computational and communication efforts. This problem is known as the *Non-effect Problem*.

Both problems could appear in DGAs since subpopulations tend to converge at different speeds. For example, if parameters used for the genetic operators are different, convergence speed will be very different in subpopulations. These problems can directly affect the global convergence leading to non-optimal solutions and losing the efectiveness of the distributed approach.

Subsequently, proposing a well-designed DGA is not a trivial task due to the existence of several factors that can have an influence over the exploration/exploitation balance of the algorithm. There are several elements to consider when designing DGAs:

1. **Topology:** structure of the distributed algorithm which defines relationships between subpopulations and individuals [31, 32, 38, 39]
2. **Migration rate** (MRATE): amount of individuals to be exchanged between subpopulations.
3. **Migration frequency** (MFREQ): number of generations between two consecutive migrations.
4. **Selection strategy:** generally there are two ways of selecting the genetic material to be copied. The first way is randomly selecting an individual from the current subpopulation. The second way consists on selecting the individual with the best fitness in every subpopulation to be copied to another. This last would lead into a more direct evolution because individuals would not have traces of less adapted individuals. The main disadvantage of selecting the best individual is that it could lead into premature convergency [40].
5. **Replacement strategy:** different replacement strategies can be considered, as replacing the worst individuals with the ones received due to migrations, as replacing an individual randomly choosen, etc.
6. **Replication of emigrants:** should individuals be moved, or copied among subpopulations? Exchanging copies of individuals could lead to a highly evolved individuals dominating several less evolved subpopulations [40].

All these parameters have a deep interaction among them and should be carefully determined since a poor choice in one of them can have a strong impact on the global performance of the algorithm. For instance, choosing a extremely high MFREQ can lead to an excessive communication load of the network and the effect of the migrated individuals could be almost imperceptible. Besides, these parameters should be fixed having in mind the hardware that will be used to execute the algorithm: depending on the network it might be better migrating more individuals less frequently than the other way around.

Finally, one procedure for designing DGAs comes from the consideration of spatial separation of subpopulations. Schematically:

1. Generate a random population, P.
2. Divide P into m subpopulations: SP_i, $i = 1, \ldots, m$.
3. Define a topology for $SP_1, \ldots, SP_m$.
4. For i = 1 to m do:
    4.1. Apply in parallel during *MFREQ* generations the genetic operators.
    4.2. Send in parallel *MRATE* chromosomes to neighbor subpopulations.
    4.3. Receive in parallel chromosomes from neighbor subpopulations.
5. If stopping criteria is not meet then go back to step 4.

3 Fuzzy Rule-Based Systems

FRBSs constitute one of the main contributions of fuzzy logic. The basic concepts which underlie these fuzzy systems are those of linguistic variable and fuzzy IF-THEN rule. A linguistic variable, as its name suggests, is a variable whose values are words rather than numbers, e.g., "small", "young", "very hot" and "quite slow". Fuzzy IF-THEN rules are of the general form: if antecedent(s) then consequent(s), where antecedent and consequent are fuzzy propositions that contain linguistic variables. A fuzzy IF-THEN rule is exemplified by "if the temperature is high then the fan-speed should be high". With the objective of modeling complex and dynamic systems, FRBSs handle fuzzy rules by mimicking human reasoning (much of which is approximate rather than exact), reaching a high level of robustness with respect to variations in the system's parameters, disturbances, etc. The set of fuzzy rules of an FRBS can be derived from subject matter experts or extracted from data through a rule induction process.

In this section, we present a brief overview of the foundations of FRBSs, with the aim of illustrating the way they behave. In particular, in Section 3.1, we introduce the important concepts of fuzzy set and linguistic variable. Finally, in section 3.2, we deal with the basic elements of FRBSs.

3.1 Fuzzy Set and Linguistic Variable

A *fuzzy set* is distinct from a crisp set in that its elements belong to it to a certain degree. The core of a fuzzy set is its MF: a surface or line that defines the relationship between a value in the set's domain and its degree of membership. In particular, according to the original ideal of Zadeh [5], membership of an element x to a fuzzy set A, denoted as $\mu_A(x)$ or simply $A(x)$, can vary from 0 (full non-membership) to 1 (full membership), i.e., it can assume all values in the interval $[0,1]$. Clearly, a fuzzy set is a generalization of the concept of a crisp set whose MF takes values in $\{0,1\}$.

The value of $A(x)$ is the degree of membership of x in A. For example, consider the concept of *high temperature* in an environmental context with temperatures distributed in the interval [0, 50] defined in centigrade degrees. Clearly 0°C is not understood as a "high temperature" value, and we may assign a null value to express its degree of compatibility with the high temperature concept. In other words, the membership degree of 0°C in the class of high temperatures is zero. Likewise, 30°C and over are certainly high temperatures, and we may assign a value of 1 to express a full degree of compatibility with the concept. Therefore, temperature values in the range [30, 50] have a membership value of 1 in the class of high temperatures. From 0°C to 30°C, the degree of membership in the fuzzy set high temperature gradually increases, as exemplified in Figure 2, which actually is a MF $A : T \rightarrow [0,1]$ characterizing the fuzzy set of high temperatures in the universe $T = [0,50]$. In this case, as temperature values increase they become more and more compatible with the idea of high temperature.

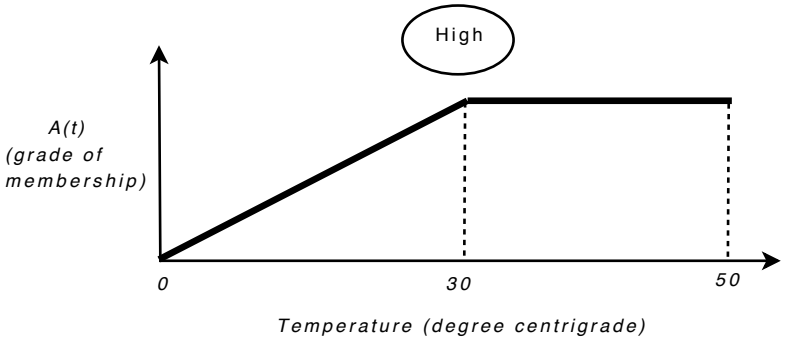

Fig. 2. Membership function

Linguistic variables are variables whose values are not numbers but words or sentences in a natural or artificial language. This concept has clearly been developed as a counterpart to the concept of a numerical variable. In concrete, a linguistic variable L is defined as a quintuple [41]: $L = (x, A, X, g, m)$, where x is the base

variable, $A = \{A_1, A_2, \ldots, A_N\}$ is the set of *linguistic terms* of L (called *term-set*), X is the domain (universe of discourse) of the base variable, g is a syntactic rule for generating linguistic terms and m is a semantic rule that assigns to each linguistic term its *meaning* (a fuzzy set in X). Figure 3 shows an example of a linguistic variable *Temperature* with three linguistic terms "Low, Medium, and High". The base variable is the temperature given in appropriate physical units.

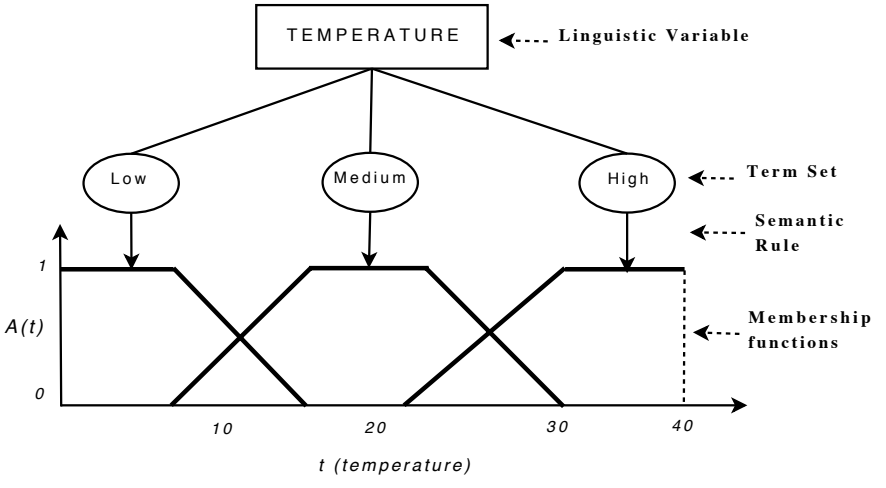

Fig. 3. Example of linguistic variable *Temperature* with three linguistic terms

Each underlying fuzzy set defines a portion of the variable's domain. But this portion is not uniquely defined. Fuzzy sets overlap as a natural consequence of their elastic boundaries. Such an overlap not only implements a realistic and functional semantic mechanism for defining the nature of a variable when it assumes various data values but provides a smooth and coherent transition from one state to another.

3.2 Basic Elements of FRBSs

The essential part of FRBSs is a set of IF-THEN linguistic rules, whose antecedents and consequents are composed of fuzzy statements, related by the dual concepts of fuzzy implication and the compositional rule of inference.

An FRBS is composed of a *knowledge base* (KB), that includes the knowledge in the form of IF-THEN fuzzy rules;

IF *a set of conditions are satisfied*
THEN *a set of consequents can be inferred*

and an inference engine module that includes:

- A *fuzzification interface*, which has the effect of transforming crisp data into fuzzy sets.
- An *inference system*, that uses them together with the KB to make inference by means of a reasoning method.
- A *defuzzification interface*, that translates the fuzzy rule action thus obtained to a real action using a defuzzification method.

FRBSs can be categorized into different families:

- The first includes linguistic models based on collections of IF-THEN rules, whose antecedents are linguistic values, and the system behaviour can be described in natural terms. The consequent is an output action or class to be applied. For example, we can denote them as:
  R_i : If X_{i1} is A_{i1} and $\cdots$ and X_{in} is A_{in} then Y is B_i
  or
  R_i : If X_{i1} is A_{i1} and $\cdots$ and X_{in} is A_{in} then C_k with w_{ik}
  with $i = 1$ to M, and with X_{i1} to X_{in} and Y being the input and output variables for regression respectively, and C_k the output class associated to the rule for classification, with A_{i1} to A_{in} and B_i being the involved antecedents and consequent labels, respectively, and w_{ik} the certain factor associated to the class. They are usually called *linguistic* FRBSs or *Mamdani* FRBSs [42].
- The second category based on a rule structure that has fuzzy antecedent and functional consequent parts. This can be viewed as the expansion of piece-wise linear partition represented as
  R_i : If X_{i1} is A_{i1} and $\cdots$ and X_{in} is A_{in} then $Y = p(X_{i1}, \cdots, X_{in})$,
  with $p(\cdot)$ being a polynomial function, usually a linear expression, $Y = p_0 + p_1 \cdot X_{i1} + \cdots + p_n \cdot X_{in}$. The approach approximates a nonlinear system with a combination of several linear systems. They are called *Takagi and Sugeno's* type fuzzy systems [43] (TS-type fuzzy systems).
- Other kind of fuzzy models are the approximate or scatter partition FRBSs, which differ from the linguistic ones in the direct use of fuzzy variables [44]. Each fuzzy rule thus presents its own semantic, i.e., the variables take different fuzzy sets as values (and not linguistic terms from a global term set). The fuzzy rule structure is then as follow:
  R_i : If X_{i1} is $\hat{A}_{i1}$ and $\cdots$ and X_{in} is $\hat{A}_{in}$ then Y is $\hat{G}_i$
  with $\hat{A}_{ij}$ to $\hat{A}_{in}$ and $\hat{G}_i$ being fuzzy sets. The major difference with respect to the rule structure considered in linguistic FRBSs is that rules of approximate nature are semantics free whereas descriptive rules operate in the context formulated by means of the linguistic semantics.

In linguistic FRBSs, the KB is comprised by two components, a *data base* (DB) and a *rule base* (RB).

- A DB, containing the linguistic term sets used in the linguistic rules and the MFs defining the semantics of the linguistic labels.

Each linguistic variable involved in the problem will have associated a fuzzy partition of its domain represented by the fuzzy sets associated to its linguistic terms. Figure 4 shows an example of fuzzy partition with five labels. In this case, the linguistic term set for each variable (denoted in a common way by y) is {Negative Medium (NM), Negative Small (NS), Zero (ZR), Positive Small (PS), Positive Medium (PM)}, which has associated the fuzzy partition of their corresponding domains shown in the Figure. This can be considered as a discretization approach for continuous domains where we establish a membership degree to the items (labels), we have an overlapping between them, and the inference engine manages the matching between the patterns and the rules providing an output according to the rule consequents with a positive matching. The determination of the fuzzy partitions is crucial in fuzzy modelling [45], and the granularity of the fuzzy partition plays an important role in the FRBS behaviour [46].

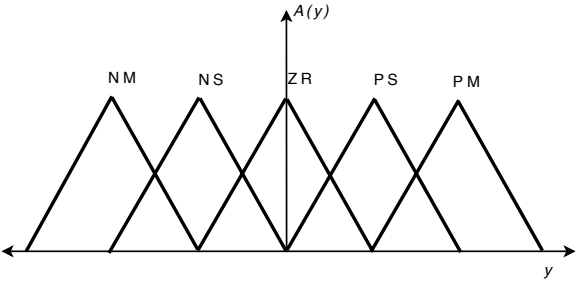

Fig. 4. Membership functions of the linguistic variables (where y stands for each variable involved in the system)

If we manage approximate FRBSs, then we do not have a DB due to the fact that rules use fuzzy values rather than linguistic terms.

- A RB comprises of a collection of linguistic rules that are joined by a rule connective ("also" operator). In other words, multiple rules can fire simultaneously for a given input.

The inference engine of an FRBS acts in a different way depending on the kind of problem (classification or regression) and the kind of fuzzy rules (linguistic ones, TS-ones, etc). It usually includes a fuzzification interface that serves as the input to the fuzzy reasoning process; an inference system that infers from the input to several resulting output (fuzzy set, class, etc); and the defuzzification interface or output interface that converts the fuzzy sets obtained from the inference process into a crisp action that constitutes the global output of the FRBS. This last component appears in the case of regression problems, or provide the final class associated to the input pattern according to the inference model.

The generic structure of an FRBS is shown in Figure 5.

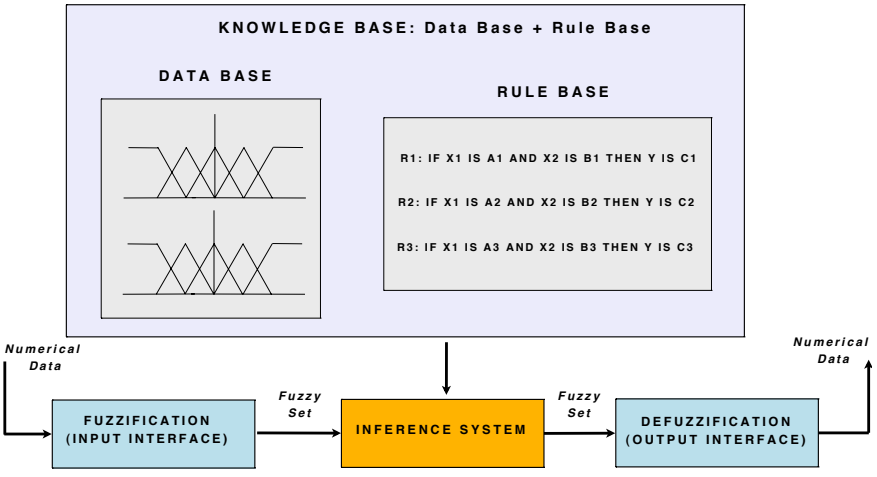

Fig. 5. Structure of an FRBS

For more information about fuzzy systems the following books may be consulted [1, 2, 3, 4, 9, 47, 48]. For different issues associated to the trade-off between interpretability and accuracy of FRBSs, the two following edited books present a collection of contributions in the topic [18, 49].

Finally, we must point out that we can find a lot of applications of FRBSs in all areas of engineering, sciences, medicine, etc. At the present it is very easy to find these applications by using the publisher web search tools and by focusing the search on journals of different application areas.

4 Genetic Tuning of FRBSs

With the aim of making a FRBS performs better, some approaches try to improve the preliminary DB definition or the inference engine parameters once the RB has been derived [9, 10, 11]. In order to do so, a tuning process considering the whole KB obtained (the preliminary DB and the derived RB) is used a posteriori to adjust the MFs or the inference engine parameters. A graphical representation of the tuning process is shown in Figure 6.

Among the different possibilities to perform tuning, one of the most widely-used approaches to enhance the performance of FRBSs is the one focused on the DB definition, usually named *tuning of MFs*, or *DB tuning* [17, 14, 15, 19, 50]. In [14], we can find a first and classic proposal on the tuning of MFs. In this case, the tuning methods refine the parameters that identify the MFs associated to the labels

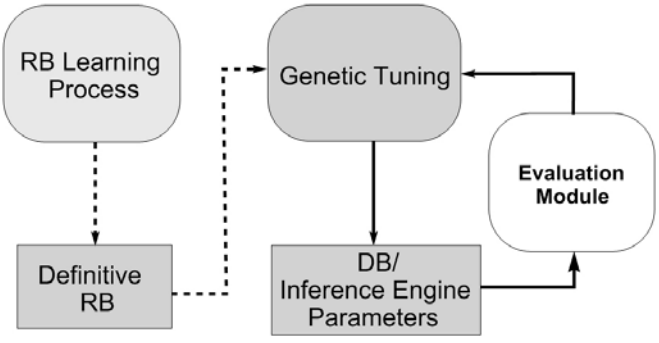

Fig. 6. Genetic tuning process

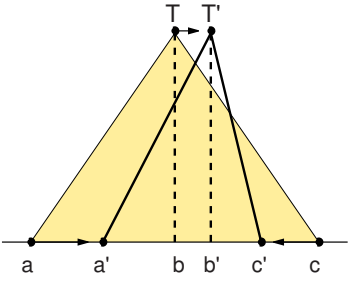

Fig. 7. Tuning by changing the basic MF parameters

comprising the DB. Classically, due the wide use of the triangular-shaped MFs, the tuning methods [19, 9, 15, 14] refine the three definition parameters that identify these kinds of MFs (see Figure 7).

Since the parameters of the MF are interdependent among themselves, in the case of large scale problems, the tuning process becomes an optimization problem on a very complex search space. This, of course, affects the good performance of the optimization methods. A good alternative to solve this problem is the lateral tuning of MFs [17]. This approach makes use of the linguistic 2-tuples representation [28] which simplifies the search space and, therefore, eases the derivation of optimal models, particularly in complex or high-dimensional problems. In order to better handle the complex search space that the tuning of MFs represents, in this work, we analyze the use of the DGAs when performing a lateral tuning of the MFs.

In the next subsection, we describe the efficient lateral tuning of FRBSs. Then, the sequential evolutionary algorithm proposed in [17] to perform the lateral tuning of FRBS is briefly described.

4.1 Lateral Tuning of FRBSs: The Linguistic 2-Tuples Representation

In [17], a new procedure for FRBSs tuning was proposed. It is based on the linguistic 2-tuples representation scheme introduced in [28], which allows the lateral displacement of the support of a label and maintains the interpretability at a good level. This proposal introduces a new model for rule representation based on the concept of symbolic translation [28]. The symbolic translation of a label is a number in $[-0.5, 0.5)$ which expresses its displacement between two adjacent lateral labels (see Figure 8.a). Let us consider a generic linguistic fuzzy partition $S = \{s_0, \ldots, s_{L-1}\}$ (with L representing the number of labels). Formally, we represent the symbolic translation of a label s_i in S by means of the 2-tuple notation,

$$(s_i, \alpha_i), \;\; s_i \in S, \; \alpha_i \in [-0.5, 0.5). \tag{1}$$

The symbolic translation of a label involves the lateral variation of its associated MF. Figure 8 shows the symbolic translation of a label represented by the 2-tuple $(s_2, -0.3)$ together with the associated lateral variation.

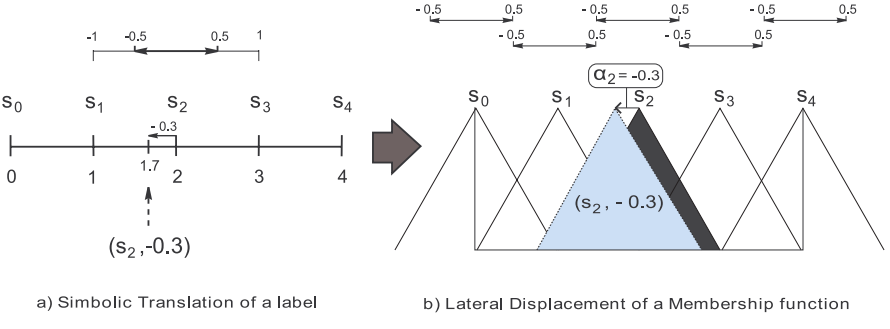

Fig. 8. Symbolic Translation of a Label and Lateral Displacement of the associated MF

In the context of FRBSs, the linguistic 2-tuples could be used to represent the MFs used in the linguistic rules. This way to work, introduces a new model for rule representation that allows the tuning of the MFs by learning their respective lateral displacements. Next, we present this approach by considering a simple control problem.

Let us consider a control problem with two input variables (X_1, X_2), one output variable (Y) and an initial DB defined by experts to determine the MFs for the following labels:

- X_1: $Error \rightarrow \{Negative, Zero, Positive\}$
- X_2: $\nabla Error \rightarrow \{Negative, Zero, Positive\}$
- Y: $Power \rightarrow \{Low, Medium, High\}$

Based on this DB definition, examples of classic and linguistic 2-tuples represented rules are:

- *Classic Rule*:
  R_i: If the **Error** is Zero and the ∇Error is Positive Then the **Power** is High.
- *Rule with 2-Tuples Representation*:
  R_i: If the **Error** is (Zero,0.3) and the ∇Error is (Positive, -0.2) Then the **Power** is (High, -0.1).

With respect to the classic tuning, usually considering three parameters in the case of triangular MFs, this way to work involves a reduction of the search space that eases a fast derivation of optimal models, improving the convergence speed and avoiding the necessity of a large number of evaluations.

In [17], two different rule representation approaches have been proposed, a global approach and a local approach. The global approach tries to obtain more interpretable models, while the local approach tries to obtain more accurate ones. In our case, tuning is applied at the level of linguistic partitions (global approach). By considering this approach, the label s_i^v of a variable v is translated with the same α_i^v value in all the rules where it is used, i.e., a global collection of 2-tuples is used in all the fuzzy rules.

Notice that from the parameters α_i^v applied to each label we could obtain the equivalent triangular MFs. Thus, an FRBS based on linguistic 2-tuples can be represented as a classic Mamdani FRBS [51]. Refer to [17] for further details on this approach.

4.2 Sequential Algorithm for the Lateral Tuning of FRBSs

In [17], an effective sequential GA was proposed to perform a lateral tuning of previously obtained FRBSs. A short description of this algorithm is given below (see [17] for a detailed description).

As the basis optimization procedure the genetic model of CHC [52] was used. Evolutionary model of CHC makes use of a "Population-based Selection" approach. N parents and their corresponding offsprings are combined to select the best N individuals to compose the next population. The CHC approach makes use of an incest prevention mechanism and a restarting process to provoke diversity in the population, instead of the well known mutation operator.

This incest prevention mechanism is considered in order to apply the crossover operator, i.e., two parents are crossed if their hamming distance divided by 2 is higher than a predetermined threshold, T. Since a real coding scheme is considered, each gene is transformed by considering a Gray Code with a fixed number of bits per gene ($BITSGENE$) determined by the system expert. In our case, the threshold value is initialized as:

$$T = (\#GenesC_T * BITSGENE)/4.0. \tag{2}$$

Following the original CHC scheme, T is decreased by one when the population does not change in one generation. In order to avoid very slow convergence, T is also decreased by one when no improvement is achieved with respect to the best chromosome of the previous generation. The algorithm restarts when T is below zero. A scheme of the evolutionary model of CHC is shown in Figure 9.

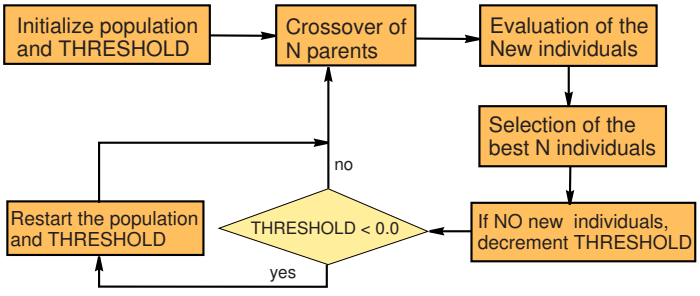

Fig. 9. Scheme of CHC

In the following, the components used to design the evolutionary tuning process are explained. They are: DB codification and initial gene pool, fitness function, crossover operator and restarting process.

4.2.1 Data Base Codification and Initial Population

A real coding scheme is considered, i.e., the real parameters are the GA representation units (genes). Let us consider n system variables and a fixed number of labels per variable L. Then, a chromosome has the following form (where each gene is associated to the tuning value of the corresponding label),

$$(\alpha_1^1, \ldots, \alpha_1^L, \alpha_2^1, \ldots, \alpha_2^L, \ldots, \alpha_n^1, \ldots, \alpha_n^L) \tag{3}$$

To make use of the available information, the initial FRBS obtained from an automatic fuzzy rule learning method is included in the population as an initial solution. To do so, the initial pool is obtained with the first individual having all genes with value '0.0', and the remaining individuals generated at random in [-0.5, 0.5).

4.2.2 Fitness Function

To evaluate a given chromosome the well-known Mean Square Error (MSE) is used:

$$\text{MSE} = \frac{1}{2 \cdot N} \sum_{l=1}^{N} (F(x^l) - y^l)^2, \tag{4}$$

with N being the data set size, $F(x^l)$ being the output obtained from the FRBS decoded from the said chromosome when the l-th example is considered and y^l being the known desired output.

4.2.3 Crossover Operator

The crossover operator is based on the the concept of environments. These kinds of operators show a good behavior in real coding. Particularly, the BLX-α operator [53] is considered.

This operator allows tuning the degree of exploration and exploitation of the crossover in an easy way. BLX-α crossover works as follows: let us assume that $X = (x_1, \ldots, x_g)$ and $Y = (y_1, \ldots, y_g)$ with $x_i, y_i \in [a_i, b_i) = [-0.5, 0.5) \subset \mathbb{R} (i = 1, \ldots, g)$ are the two real-coded chromosomes that are going to be crossed. Using the BLX-α crossover, one descendant $Z = (z_1, \ldots, z_g)$ is obtained, where z_i is randomly (uniformly) generated within the interval $[l_i, u_i]$, with $l_i = max\{a_i, c_{min} - A\}$, $u_i = min\{b_i, c_{max} + A\}$, $c_{min} = min\{x_i, y_i\}$, $c_{max} = max\{x_i, y_i\}$ and $A = (c_{max} - c_{min}) \cdot \alpha$.

Figure 10 shows how the BLX-α operator works at different stages of the evolution process.

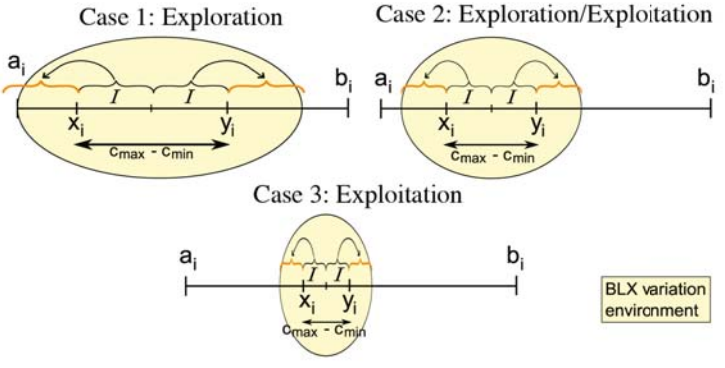

Fig. 10. Different cases/stages of the application of the BLX-α crossover operator, where α = 0.5

4.2.4 Restarting Process

To get away from local optima, this algorithm uses a restart approach [52]. In this case, the best chromosome is maintained and the remaining are generated at random within the corresponding variation intervals [-0.5, 0.5). It follows the principles of CHC [52], performing the restart procedure when the threshold T is below zero.

5 A Distributed Genetic Algorithm for the Lateral Tuning of FRBSs

One of the problems when performing tuning with complex data sets is the complexity of the search space. Sometimes even an advanced GA can not deal with the complex search space in terms of time and quality of the results.

Gradual Distributed Real-Coded Genetic Algorithms (GDRCGAs) are a kind of heterogeneous DGAs based on real coding where subpopulations apply genetic operators in different levels of exploitation/exploration. This heterogeneous application of genetic operators produce a *parallel multiresolution* which allows a wide exploration of the search space and effective local precision. Due to appropiate connections between subpopulations in order to gradually exploit multiresolution, these algorithms achieve refinement or expansion of the best emerging zones of the search space.

In order to analyze how DGAs can help the tuning problem, we have selected an efficient GDRCGA [27], that keeps a good balance between exploration and exploitation of the search space. As we said before, we apply this algorithm to perform a lateral tuning of previously obtained FRBSs. In this section, we describe the different characteristics of the DGA used: topology, migrations scheme and the main components of the different subpopulations.

5.1 Main Components of the DGA

The GDRCGA [27] used for FRBS tuning employs 8 subpopulations in a hypercube topology as seen in Figure 11.

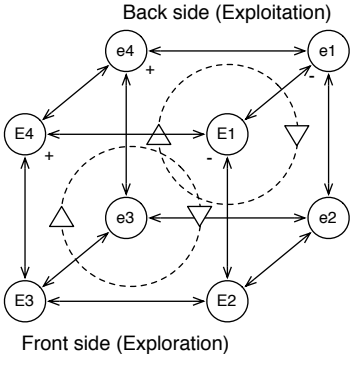

Fig. 11. Hypercube topology for GDRCGA

In this topology two important groups of subpopulations can be clearly identified:

1. **Front side:** this side of the hypercube is oriented to explore the search space. In this side, four subpopulations, $E_1, ..., E_4$, apply genetic operators adapted for exploration in a clockwise increasing degree.
2. **Back side:** subpopulations in the back side of the hypercube, $e_1, ..., e_4$, apply exploitation oriented genetic operators in a clockwise increasing degree.

One of the key elements of DGAs is the migration policy of individuals between subpopulations. In this particular model, an *immigration* process [54] is achieved since the best chromosome in every subpopulation abandons it and moves to an immediate neighbor. Due to this immigration policy, three different immigration movements can be identified depending on the subpopulations involved:

1. **Refinement migrations:** individuals in the back side move clockwise to the immediate neighbor, i.e. from e_2 to e_3. Chromosomes in the front side move counterclock from a more exploratory subpopulation to a less exploratory oriented one.
2. **Expansion migrations:** individuals in the back side move counterclock to the immediate neighbor and chromosomes in the front side move clockwise from a less exploratory subpopulation to a more exploratory oriented one, i.e. from E_4 to E_1.
3. **Mixed migrations:** subpopulations from one side of the hypercube exchange their best individual with the counterpart subpopulation in the other side: interchange between E_i and e_i, $i = 1 \ldots 4$.

Figure 12 shows the three different migration movements described above.

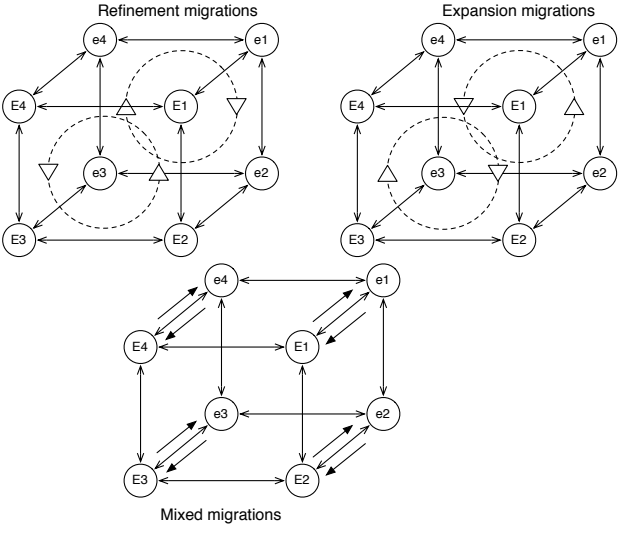

Fig. 12. Three different migration movements for the GDRCGA

As stated in [27], the frequency in which migration movements occur is crucial to avoid the classic withdraws of DGAs: the conquest and noneffect problems. In order to reduce the negative effect of these problems, immigrants stay in the receiving subpopulations for a brief number of generations. Besides, a restart operator is used to avoid stagnation of the search process. This restart operator randomly reinitializes all subpopulations if non-significant improvement of the best element is achieved for

a number of generations. Also an elitism strategy is used in order to keep the best adapted individual of every subpopulation.

5.2 Common Components of Individual Subpopulations

The main component used in the different subpopulations of the distributed model are:

- **DB codification and initial subpopulations:** the coding scheme used to represent the displacement parameters is the same one described in section 4.2.1 for the specialized sequential algorithm. Each subpopulation is also initialized in the same way explained in section 4.2.1, i.e., by including the initial FRBS as the first individual in each subpopulation and the remaining individuals generated at random.
- **Selection mechanism:** linear ranking selection (LRS) [55] with stochastic universal sampling [56]. Using LRS the selective pressure can be easily adjusted. In LRS, the individuals are sorted in order of decreasing raw fitness, and then the selection probability, p_s, of each individual I_i is computed according to its rank $rank(I_i)$, with $rank(I_{best}) = 1$, by using the following non-increasing assignment function:

$$p_s(I_i) = \frac{1}{N} \cdot (\eta_{max} - (\eta_{max} - \eta_{min}) \cdot \frac{rank(I_i) - 1}{N - 1}) \tag{5}$$

  where N is the population size, and $\eta_{min} \in [0, 1]$ specifies the expected number of copies for the worst individual. The selection pressure is determined by η_{min}. If η_{min} is low, high pressure is achieved. The values of η_{min} used for each subpopulation are shown in Table 1.

Table 1. Values of η_{min} for each subpopulation

Exploitation				Exploration			
+	←		−	−		→	+
e_4	e_3	e_2	e_1	E_1	E_2	E_3	E_4
0,9	0,7	0,5	0,1	0,9	0,7	0,5	0,1

- **Crossover operator:** the crossover operator used, BLX-α, is the same that was used in the specialized sequential algorithm and it is described in section 4.2.3. As stated before, distinct parameter values are used between subpopulations in order to achieve different degrees of exploitation/exploration. The values used for each subpopulation are shown in Table 2.

  In the absence of selection pressure, values of α, which are $\alpha < 0.5$ make the subpopulations converge towards values in the center of their ranges, producing low diversity levels in the population and inducing a possible premature convergence towards non-optimal solutions. Only when $\alpha = 0.5$, there is a balanced

relationship reached between convergence (exploitation) and divergence (exploration). In this case, the probability that a gene will lie in the exploration interval is equal to the probability that it will lie in an exploration interval [53].

Table 2. Values of α for each subpopulation

Exploitation				Exploration			
+	←		−	−	→		+
e_4	e_3	e_2	e_1	E_1	E_2	E_3	E_4
0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8

- **Mutation operator:** non-uniform mutation operator [57] applied with probability $P_{mut} = 0.125$. This operator is one of the most used mutation operators in real-coded GAs. The mutation operator works as follows. Let $X^t = (x_1^t, \ldots, x_n^t)$ the chromosome selected for mutation. This operator generates a mutated chromosome $X^{t+1} = (x_1^{t+1}, \ldots, x_n^{t+1})$ where:

$$x_i^{t+1} = \begin{cases} x_i^t + \Delta(t, x_i^u - x_i^t) & \text{if } r \leq 0.5 \\ x_i^t - \Delta(t, x_i^t - x_i^l) & \text{otherwise} \end{cases} \tag{6}$$

where t is the current generation number and r is an uniformly distributed random number between 0 and 1. x_i^l and x_i^u are lower and upper bounds of the i-th gene of the chromosome. The function $\Delta(t, y)$ is defined as follows:

$$\Delta(t, y) = y(1 - u^{(1-\frac{t}{T})^b}) \tag{7}$$

where u is an uniformly distributed random number between 0 and 1, T is the maximum number of generations and b is a parameter determining the strength of the mutation.

6 Empirical Evaluation

In this section, we analize the empirical results we obtained in order to assess the merits of the distributed approach when applied to lateral tuning of MFs. To evaluate the usefulness of the studied approach, we have used four real-world problems. Table 3 summarizes the main characteristics of the four datasets and shows the link to the KEEL software tool webpage [58] (http://www.keel.es/) from which they can be downloaded.

The studied distributed algorithm described in Section 5 (GDRCGA) is compared with the specialized sequential GA (CHC) [17] in terms of quality of the solutions achieved (MSE) as well as in running time. In both cases, the well-known ad-hoc data-driven learning algorithm of Wang and Mendel [59] is applied to obtain an initial set of candidate linguistic rules. The initial linguistic partitions are comprised of five linguistic terms in the case of datasets with less than 9 variables

Table 3. Data sets used to evaluate the algorithm

Data set	Variables	Instances
Electrical Maintenance	5	1056
Abalone	9	4177
Weather-Izmir	10	1461
Treasury	16	1049

and three linguistic terms in the remaining ones. We consider strong fuzzy partitions of triangular-shaped MFs. Once the initial RB is generated, the different post-processing algorithms can be applied.

A five-fold cross-validation approach has been used, i.e., we randomly split the data set into 5 folds, each containing the 20% of the patterns of the data set, and used four folds for training and the other for testing. So, a total of five runs have been carried out with different independent test sets. For each dataset, we therefore consider the average results of the five runs. The average results of the initial FRBSs obtained by the Wang and Mendel algorithm are shown in Table 4 (initial reference results).

Table 4. Initial mean squared errors and deviations in both, training and test sets, obtained by Wang & Mendel

Dataset	Training	σ_{tra}	Test	σ_{test}
Electrical M.	57606	2841	57934	4733
Treasury	1.636	0.121	1.632	0.182
Weather-Izmir	6.944	0.720	7.368	0.909
Abalone	3.341	0.130	3.474	0.247

An interesting point to be taken into account is the *evolution* of the MSE as the number of evaluations increases. So, three different numbers of evaluations have been choosen: 10000, 25000 and 50000 evaluations per run. The results in terms of quality of the solutions attained are shown in Table 5. An important fact to notice is that the MSE achieved with the distributed method is lower than the error obtained with the specialized GA in all data sets at 50000 evaluations. The distribuited approach also obtains good results with fewer iterations in some cases (e.g. Electrical Maintenance with 25000 iterations), but clearly its real effectiveness will be reached when the computation load is higher.

Generally, when comparing a distributed or parallel approach with any other sequential algorithm, an interesting measure is the execution time gain ratio. This ratio could be defined as follows:

$$R = \frac{T_{seq}}{T_{dist}} \tag{8}$$

Table 5. Mean squared errors in training and test sets obtained by CHC and GDRCGA. The winner for each pair of training is *in italics*. The winner for each pair of test is **boldfaced**

		CHC		GDRCGA	
Data set	Evaluations	Training	Test	Training	Test
Electrical Maintenance	10000	*2.59363671E+04*	2.92591821E+04	2.65539710E+04	**2.89024830E+04**
	25000	2.48690100E+04	2.80510895E+04	*2.39248797E+04*	**2.67720415E+04**
	50000	2.46214328E+04	2.78282761E+04	*2.26682075E+04*	**2.54097540E+04**
Abalone	10000	*2.61355003E+00*	2.79981355E+00	2.65916770E+00	**2.79026550E+00**
	25000	2.60333453E+00	2.79298130E+00	*2.59992700E+00*	**2.76143590E+00**
	50000	2.60303744E+00	2.79117626E+00	*2.57035010E+00*	**2.75904570E+00**
Weather-Izmir	10000	*1.68875432E+00*	**1.89318352E+00**	1.89195950E+00	1.95458830E+00
	25000	*1.64117336E+00*	**1.86996710E+00**	1.66238700E+00	1.87669540E+00
	50000	1.64010963E+00	1.86891124E+00	*1.57019250E+00*	**1.86195430E+00**
Treasury	10000	*1.71238672E-01*	**1.86722425E-01**	2.12486800E-01	2.16882700E-01
	25000	*1.33618274E-01*	**1.50895419E-01**	1.42194200E-01	1.67407400E-01
	50000	1.20604483E-01	1.37784224E-01	*1.15845500E-01*	**1.31803000E-01**

where T_{seq} is the time spent by the sequential algorithm and T_{dist} is the execution time of the distributed approach. The higher the value of R, the better. Time gain ratio values obtained in the empirical experimentation are shown in Table 6.

Table 6. Time gain ratio with 50000 evaluations

Data set	T_{seq}	T_{dist}	R
Electrical Maintenance	187,3	391,6	0,479
Trasury	525,3	739,7	0,710
Weather-Izmir	849,8	867,1	0,980
Abalone	1980,9	942,5	2,101

In Table 6, the time gain ratio, R, increases with the problem complexity. In the less complex data sets the ratio obtained is substantially lower because the sequential specialized GA is very fast and the time spent in communications of the distributed approach slows it down in comparison. As the complexity of the data set increases the time gain ratio also increases, showing that the distributed approach in the most complex data set is more than two times faster than the sequential specialized GA. The distributed algorithm takes longer than the sequential algorithm when dealing with small size data sets mainly due to two reasons: interprocess communication in the distributed approach implies additional execution time which can not be parallelized and the specialized algorithm is optimized for small size data sets where the search space is not too complex.

On the other hand, observation of the evolution of the MSE is also an interesting factor to take into account. Two different data sets have been choosen in order to study the evolution of the MSE: Electrical Maintenance and Treasury. These two data sets were choosen because of their different complexity: Treasury data set is far more complex than Electrical Maintenance.

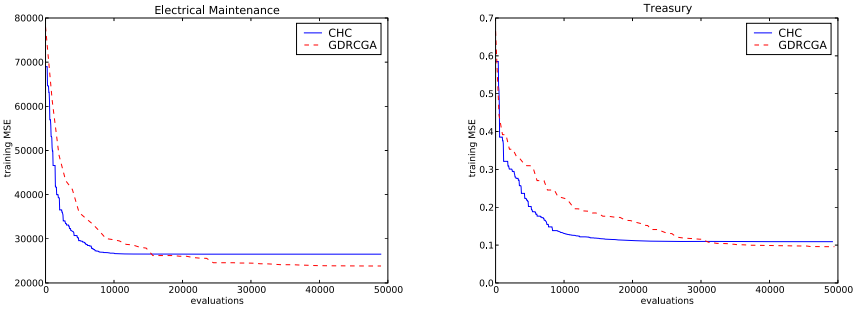

Fig. 13. Evolution of the MSE in training (convergence): Electrical Maintenance and Treasury datasets

Figure 13 shows the convergence of both algorithms in both problems. Due to the distributed nature of the algorithm and consequently the spatial separation implied, it needs more evaluations to converge than the sequential algorithm. It always presents the same behaviour in comparision to the sequential approach: with a small number of evaluations it yields a higher error than the sequential one, but when the number of evaluations is high it gives solutions with a better quality.

As it has been stated, the distributed approach needs more iterations to achieve convergency for complex data sets. This situation can be observed in Figure 13 (right side): GDRCGA achieves better MSE values when the search process has consumed two thirds of the number of evaluations. On the other hand, when dealing with less complex data sets like Electrical Maintenance (Figure 13 left side), the distributed approach quickly achieves better MSE values from almost the begining of the search process and keeps gaining distance from the sequential CHC algorithm. In fact, GDRCGA begins achieving better MSE values shortly after the search process has consumed one third of the number of evaluations available. These two situations can be also verified in Table 5.

Besides studying the evolution of the MSE in training (convergence), it is also interesting to analyze the effects that it produces on the MSE in test regarding the same data sets. Figure 14 shows the MSE in test of Treasury and Electrical Maintenance datasets. Again, we can observe that the distributed approach needs more evaluations to outperform the sequential algorithm in the more complex problem (see Figure 14 right side) while better results are obtained practically from the beginning in the simpler one (see Figure 14 left side). However, two interesting characteristics can be highlighted. Firstly, the evolution in the test error shown by GDRCGA seems quite more stable in both problems. Secondly, GDRCGA shows practically the same trend in training and test in both datasets, while the sequential approach worsen the test error once the half of the evaluations are consumed in the more complex dataset (overfitting). These characteristic are quite recommendable in the fuzzy modeling framework.

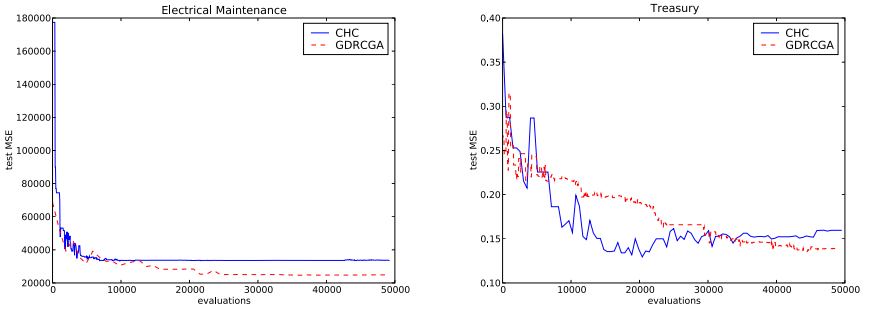

Fig. 14. Effects on the MSE in test: Electrical Maintenance and Treasury data sets

7 Conclusions and Final Remarks

In this chapter, we have presented an study on the use of the DGAs for the lateral tuning of FRBSs. To this end, we have analyzed the performance of a specific GDRCGA employing 8 subpopulations in a hypercube topology [26]. This algorithm has been compared with the specialized GA presented in [17] to perform the lateral tuning of FRBSs.

From the empirical results obtained, we can conclude that as the complexity of the problem grows, the distributed approach outperforms the specialized sequential algorithm. Moreover, the distributed procedure makes effective use of the wall time in relation to the computing times required by the sequential algorithm. Also, when dealing with complex search spaces, the distributed approach is able to converge to better quality solutions than the sequential algorithm. This behaviour makes the distributed tuning algorithm very useful when dealing with large scale problems where the complexity of the search space is high.

Since execution time and quality of the results are two properties always in conflict somehow, the distributed approach could be graduated in order to achieve faster execution times with a small cost in quality and viceversa.

References

1. Driankow, D., Hellendoorn, H., Reinfrank, M.: An introduction to fuzzy control. Springer, Berlin (1993)
2. Pedrycz, W.: Fuzzy Modelling: Paradigms and practice. Kluwer Academic Publishers, Dordrecht (1996)
3. Palm, R., Driankov, D., Hellendoorn: Model based fuzzy control. Springer, Heidelberg (1997)
4. Ishibuchi, H., Nakashima, T., Nii, M.: Classification and modeling with linguistic information granules: Advances approaches to linguistic data mining. Springer, Heidelberg (2004)
5. Zadeh, L.A.: Fuzzy sets. Information and Control 8, 338–353 (1965)

6. Zadeh, L.A.: Outline of a new approach to the analysis of complex systems and decision processes. IEEE Trans. Syst., Man, Cybern. 3, 28–44 (1973)
7. Goldberg, D.E.: Genetic algorithms in search, optimization, and machine learning. Addison-Wesley, New York (1989)
8. Holland, J.H.: Adaptation in natural and artificial systems. The University of Michigan Press, Michigan (1975); The MIT Press, London (1992)
9. Cordón, O., Herrera, F., Hoffmann, F., Magdalena, L.: Genetic Fuzzy Systems: evolutionary tuning and learning of fuzzy knowledge bases. World Scientific, Singapore (2001)
10. Herrera, F.: Genetic fuzzy systems: Taxonomy, current research trends and prospects. Evolutionary Intelligence 1, 27–46 (2008)
11. Cordón, O., Gomide, F., Herrera, F., Hoffmann, F., Magdalena, L.: Ten years of genetic fuzzy systems: current work and new trends. Fuzzy Sets and Systems 141(1), 5–31 (2004)
12. Eiben, A.E., Smith, J.E.: Introduction to evolutionary computation. Springer, Berlin (2003)
13. Zadeh, L.A.: The concept of a linguistic variable and its applications to approximate reasoning, parts i, ii and iii. Information Science 8, 8, 9, 199–249, 301–357, 43–80 (1975)
14. Karr, C.: Genetic algorithms for fuzzy controllers. AI Expert 6(2), 26–33 (1991)
15. Herrera, F., Lozano, M., Verdegay, J.L.: Tuning fuzzy logic controllers by genetic algorithms. International Journal of Approximate Reasoning 12, 299–315 (1995)
16. Alcalá, R., Alcalá-Fdez, J., Casillas, J., Cordón, O., Herrera, F.: Hybrid learning models to get the interpretability-accuracy trade-off in fuzzy modeling. Soft Computing 10(9), 717–734 (2006)
17. Alcalá, R., Alcalá-Fdez, J., Herrera, F.: A proposal for the genetic lateral tuning of linguistic fuzzy systems and its interaction with rule selection. IEEE Transactions on Fuzzy Systems 15(4), 616–635 (2007)
18. Casillas, J., Cordón, O., del Jesus, M.J., Herrera, F.: Accuracy improvements in linguistic fuzzy modeling. Springer, Heidelberg (2003)
19. Casillas, J., Cordón, O., del Jesus, M.J., Herrera, F.: Genetic tuning of fuzzy rule deep structures preserving interpretability and its interaction with fuzzy rule set reduction. IEEE Trans. Fuzzy Syst. 13(1), 13–29 (2005)
20. Cantu-Paz, E.: Efficient and Accurate Parallel Genetic Algorithms. Kluwer Academic Publishers, Norwell (2000)
21. Fernández de Vega, F., Cantu-Paz, E.: Special issue on distributed bioinspired algorithms. Soft Computing 12(12), 1143–1144 (2008)
22. Alba, E.: Parallel Metaheuristics: A New Class of Algorithms. Wiley, Chichester (2005)
23. Sterling, T., Becker, D.J., Savarese, D.F.: How to build a beowulf: A guide to the implementation and application of PC clusters. The MIT Press, Cambridge (1999)
24. Spector, D.H.M.: Building Linux Clusters. O'Reilly, Sebastopol (2000)
25. Dowd, K., Severance, C.: High Performance Computing. O'Reilly, Sebastopol (1998)
26. Robles, I., Alcalá, R., Benítez, J.M., Herrera, F.: Distributed genetic tuning of fuzzy rule-based systems. In: Proceedings of the International Fuzzy Systems Association - European Society for Fuzzy Logic and Technology (IFSA-EUSFLAT) Congress (in press, 2009)
27. Herrera, F., Lozano, M.: Gradual distributed real-coded genetic algorithms. IEEE Transactions on Evolutionary Computation 4(1), 43–63 (2000)
28. Herrera, F., Martínez, L.: A 2-tuple fuzzy linguistic representation model for computing with words. IEEE Trans. Fuzzy Syst. 8(6), 746–752 (2000)

29. Bäck, T., Beielstein, T.: User's group meeting. In: Proceedings of the EuroPVM 1995: Second European PVM, pp. 277–282 (1995)
30. Punch, W., Goodman, E., Pei, M., Chai-shun, L., Hovland, P., Enbody, R.: Further research on feature selection and classification using genetic algorithms. In: Forrest, S. (ed.) Proceedings of the Fifth International Conference on Genetic Algorithms, pp. 557–564 (1993)
31. Tanase, R.: Distributed genetic algorithms. In: Proceedings of the Third International Conference on Genetic Algorithms, pp. 434–439 (1989)
32. Mülhlenbein, H., Schomisch, M., Born, J.: The parallel genetic algorithm as function optimizer. Parallel Computing 17(6), 619–632 (1991)
33. Lin, S.C., Punch III, W.F., Goodman, E.D.: Coarse-grain parallel genetic algorithms: Categorization and new approach. In: Proceedings of the Sixth IEEE Parallel and Distributed Processing, pp. 28–37 (1994)
34. Alba, E., Luna, F., Nebro, A., Troya, J.M.: Parallel heterogeneous genetic algorithms for continuous optimization. Parallel Computing 30(5), 699–719 (2004)
35. Schlierkamp-Voosen, D., Mülhlenbein, H.: Strategy adaptation by competing subpopulations. In: Davidor, Y., Männer, R., Schwefel, H.-P. (eds.) PPSN 1994. LNCS, vol. 866, pp. 199–208. Springer, Heidelberg (1994)
36. Schnecke, V., Vornberger, O.: An adaptative parallel algorithm for vlsi-layout optimization. In: Ebeling, W., Rechenberg, I., Voigt, H.-M., Schwefel, H.-P. (eds.) PPSN 1996. LNCS, vol. 1141, pp. 22–27. Springer, Heidelberg (1996)
37. Alba, E., Dorronsoro, B.: Cellular Genetic Algorithms. Springer, Heidelberg (2008)
38. Tanase, R.: Parallel genetic algorithm for a hypercube. In: Proceedings of the 2nd International Conference on Genetic Algorithms and their Applications, pp. 177–183 (1987)
39. Cohoon, J.P., Hedge, S., Martin, W.: Punctuated equilibria: A parallel genetic algorithm. In: Proceedings of the 2nd International Conference on Genetic Algorithms and their Applications, pp. 148–154 (1987)
40. Ryan, C.: Niche and species formation in genetic algorithms. In: Chambers, L. (ed.) Practical Handbook of Genetic Algorithms: Applications, pp. 57–74. CRC Press, Boca Raton (1995)
41. Klir, G., Yuan, B.: Fuzzy sets and fuzzy logic; theory and applications. Prentice-Hall, Englewood Cliffs (1995)
42. Mamdani, E.H.: Application of fuzzy algorithms for control of simple dynamic plant. Proc. Inst. Elect. Eng. 121(12), 1585–1588 (1974)
43. Takagi, T., Sugeno, M.: Fuzzy identification of systems and its application to modelling and control. IEEE Trans. Syst. Man and Cybernetics 15(1), 116–132 (1985)
44. Alcalá, R., Casillas, J., Cordón, O., Herrera, F.: Building fuzzy graphs: features and taxonomy of learning non-grid-oriented fuzzy rule-based systems. International Journal of Intelligent Fuzzy Systems 11, 99–119 (2001)
45. Au, W.-H., Chan, K., Wong, A.K.C.: A fuzzy approach to partitioning continous attributes for classification. IEEE Transactions on Knowledge and Data Engineering 18(5), 715–719 (2006)
46. Cordón, O., Herrera, F., Villar, P.: Analysis and guidelines to obtain a good fuzzy partition granularity for fuzzy rule-based systems using simulated annealing. International Journal of Approximate Reasoning 25(3), 187–215 (2000)
47. Yager, R., Filev, D.: Essentials of fuzzy modeling and control. John Wiley and Sons, Chichester (1994)
48. Kuncheva, L.: Fuzzy classifier design. Springer, Heidelberg (2000)
49. Casillas, J., Cordón, O., Herrera, F., Magdalena, L.: Interpretability issues in fuzzy modeling. Springer, Heidelberg (2003)

50. Gürocak, H.B.: A genetic-algorithm-based method for tuning fuzzy logic controllers. Fuzzy Sets and Systems 108(1), 39–47 (1999)
51. Mamdani, E.H., Assilian, S.: An experiment in linguistic synthesis with a fuzzy logic controller. International Journal of Man-Machine Studies 7, 1–13 (1975)
52. Eshelman, L.J.: The CHC adaptive search algorithm: How to have safe serach when engaging in nontraditional genetic recombination. In: Rawlin, G.J.E. (ed.) Foundations of genetic Algorithms, vol. 1, pp. 265–283. Morgan Kaufman, San Francisco (1991)
53. Eshelman, L.J., Schaffer, J.D.: Real-coded genetic algorithms and interval-schemata. Foundations of Genetic Algorithms 2, 187–202 (1993)
54. Kröger, B., Schwenderling, P., Vornberger, O.: Parallel genetic packing on transputers. In: Parallel Genetic Algorithms: Theory and Applications: Theory Applications, pp. 151–186 (1993)
55. Baker, J.E.: Adaptive selection methods for genetic algorithms. In: Proceedings of the First International Conference on Genetic Algorithms and their Applications, pp. 101–111. Erlbraum Associates, Hillsdale (1985)
56. Baker, J.E.: Reducing bias and inefficiency in the selection algorithm. In: Proceedings of the 2nd International Conference on Genetic Algorithms, ICGA 1987 (1987)
57. Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs. Springer, Heidelberg (1992)
58. Alcalá-Fdez, J., Sánchez, L., García, S., del Jesus, M.J., Ventura, S., Garrell, J.M., Otero, J., Romero, C., Bacardit, J., Rivas, V.M., Fernández, J.C., Herrera, F.: KEEL: A software tool to assess evolutionary algorithms to data mining problems. Soft Computing 13(3), 307–318 (2009)
59. Wang, L.X., Mendel, J.M.: Generating fuzzy rules by learning from examples. IEEE Trans. Syst. Man and Cybernetics 22(6) (1992)