Distributed Genetic Tuning of Fuzzy Rule-Based Systems

Ignacio Robles, Rafael Alcalá, José Manuel Benítez and Francisco Herrera

Department of Computer Science and Artificial Intelligence
University of Granada, Spain
Email: ignaciorobles@gmail.com, {alcala, J.M.Benitez, herrera}@decsai.ugr.es

***Abstract*— *The tuning of Fuzzy Rule Base-Systems is necessary to improve their performance after the extraction of rules. This optimization problem can become a hard one when the size of the considered system in terms of the number of variables, rules and data samples is big. To alleviate this growth in complexity, we propose a distributed genetic algorithm which explotes the nowadays available parallel hardware (multicore microprocessors and clusters). The empirical performance in solution quality and computing time is assessed by comparing its results with those from a highly effective sequential tuning algorithm. Both methods are applied for the modeling of four well-known regression problems.**

***Keywords*—** 2-tuples, Distributed Genetic Algorithms, Fuzzy Rule-based Systems Tuning, Lateral Tuning

1 Introduction

Fuzzy rule based-systems (FRBS) have become a wide choice when addressing modeling and system identification problems. An essential component in these systems is the fuzzy rule base, together with the data base compose the knowledge base. The construction of this knowledge base is a key step for obtaining a correct system.

It is very difficult for human beings to obtain appropiate rules when dealing with real-world complex problems with many variables and where the necessary number of rules is high. When an expert determines the rule set for a determined problem, generally it will not be the optimal set in terms of performance. Performance is an important goal design for whatever the system is intended to be used: either with approximation purposes or when searching for an interpretable system. To cope with this problem a refining process that adjusts the system is required. This process is widely known as *tuning*.

Classically, tuning processes involve changing the shape of the Membership Functions (MFs) associated to the labels in the database so that the best cooperation among rules is reached. However as the number of variables and rules increases, tuning methods show poor performance due to the growing complexity of the search space. Moreover, the comptuing time consumed by these approaches grows with the complexity of the search space which would result in a procedures that are not useful in practice.

Nowadays parallel hardware and software has become very affordable. They are broadly available which makes them perfect to deal with complex search spaces in order to improve the poor performance achieved with classical tuning approaches. Clear examples in this line are multicore processors and linux clusters.

In this paper, we address the tuning problem and present a distributed method for lateral FRBS tuning. The paper is structured as follows: in the second section of the paper the lateral tuning of FRBSs problem is stated and an efficient sequential specialized algorithm is reviewed. The third section describe our proposal for the distributed tuning for FRBS. An empirical evaluation of the distributed algorithm is presented in the fourth section. Finally some conclusions and future work is commented in the last section.

2 Lateral Tuning of FRBSs

The tuning of FRBS is a problem long studied by researchers in the community [1, 2]. A quite efficient procedure fuzzy systems tuning was presented recently by Alcalá *et al.* [3]. We choose that as a reference to compare the perfomance of our proposal with. We describe this method briefly in this section.

2.1 Lateral Tuning: The Linguistic 2-Tuples Representation

In [3], a new procedure for FRBSs tuning was proposed. It is based on the linguistic 2-tuples representation scheme introduced in [4], 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 [4]. The symbolic translation of a label is a number in [-0.5, 0.5), expressing this interval the domain of a label when it is moving between its two adjacent lateral labels (see Figure 1.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,

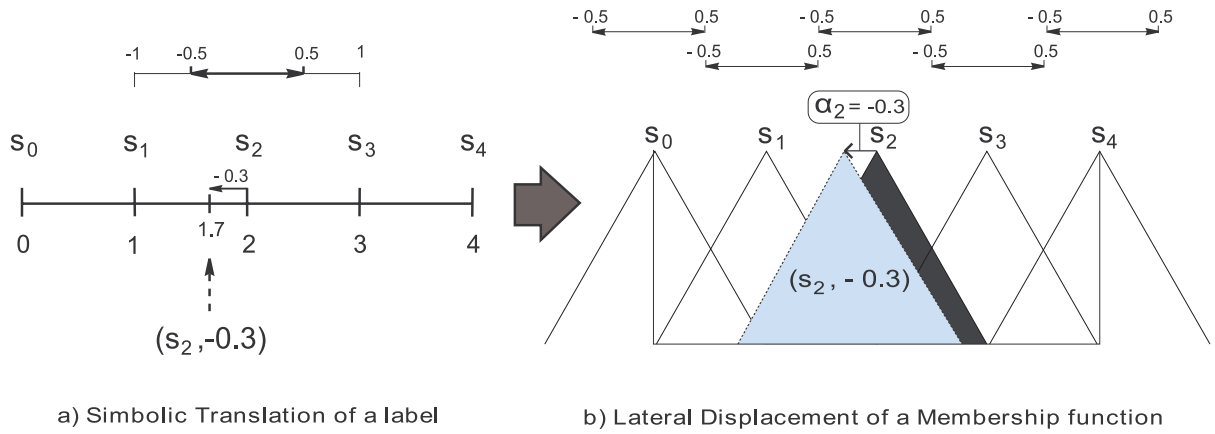
$$(s_i, \alpha_i), \;\; s_i \in S, \; \alpha_i \in [-0.5, 0.5).$$

The symbolic translation of a label involves the lateral variation of its associated MF. Figure 1 shows the symbolic translation of a label represented by the 2-tuple $(s_2, -0.3)$ together with the associated lateral variation.

In the context of FRBSs, the linguistic 2-tuples could be used to represent the MFs comprising 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. 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 [3], two different rule representation approaches have been proposed, a global approach and a local approach. The global approach tries to obtain more interpretable models,

ISBN: 978-989-95079-6-8



a) Simbolic Translation of a label

b) Lateral Displacement of a Membership function

Figure 1: Symbolic Translation of a Label and Lateral Displacement of the associated MF

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 considered, 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 [5]. Refer to [3] for further details on this approach.

2.2 Sequential Algorithm for the Lateral Tuning of FRBSs

In [3], a sequential genetic algorithm was proposed to perform a lateral tuning of previously obtained FRBSs. A short description of this algorithm is given in the next (see [3] for a detailed description).

The used model was the genetic model of CHC [6]. CHC makes use of a "Population-based Selection" approach. N parents and their corresponding offsprings are combined to select the best N individuals to take part of 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 over 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:

$$T = (\#GenesC_T * BITSGENE)/4.0.$$

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.

In the following, the components used to design the evolutionary tuning process are explained. They are: DB codification, chromosome evaluation and genetic operators.

2.3 DB Codification

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),

$$(\alpha_1^1, \ldots, \alpha_1^L, \alpha_2^1, \ldots, \alpha_2^L, \ldots, \alpha_n^1, \ldots, \alpha_n^L)$$

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).

2.4 Chromosome Evaluation

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

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

with E 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.

2.5 Genetic Operators

The genetic operators considered in CHC are crossover and restarting approach (no mutation is used). A short description of these operators comes next:

- Crossover. The crossover operator is based on the the concept of environments. These kinds of operators show a good behavior in real coding. Particularly, the Parent Centric BLX (PCBLX) operator [7] (an operator based on BLX-α) is considered.
- Restarting. To get away from local optima, this algorithm uses a restart approach [6]. 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 [6], performing the restart procedure when threshold L is below zero.

ISBN: 978-989-95079-6-8

3 A distributed genetic algorithm for FRBS tuning

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 to these new approaches is achieving not only gains in time, which is somehow inherent to distributed algorithms, but gains in quality of the solutions found.

Distributed Genetic Algorithms (DGA) are excellent optimization algorithms and have proven to be one of the best options when trying to cope with large scale problems and when the classic approaches take too much time to give a proper solution.

One procedure for the parallelization of GA comes from the consideration of spatial separation of populates. Schematically:

```
1. Generate a random population, P.
2. Divide P into n subpopulations: SPi,
   i = 1, ..., n.
3. Define a topology for SP1, ..., SPn.
4. For i = 1 to n do:
4.1. Apply in parallel during FM
     generations the genetic operators.
4.2. Send in parallel NM chromosomes to
     neighbour subpopulations.
4.3. Receive in parallel chromosomes
     from neigbour subpopulations.
5. If stopping criteria is not meet then
   go back to step 4.
```

In this section the DGA [8] used for Fuzzy Rule-based System tuning is described.

3.1 Gradual Distributed Real-Coded Genetic Algorithm used

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.

The GDRCGA [8] used for FRBS tuning employs 8 subpopulations in a hypercube topology as seen in Figure 2.

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 exploration tuned genetic operators 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.

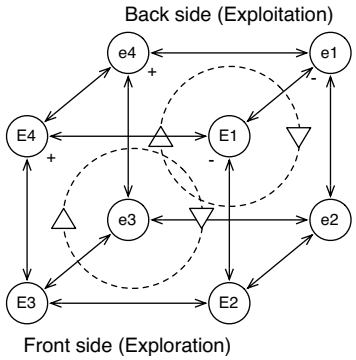


Figure 2: Hypercube topology for GDRCGA

One of the key elements of DGAs is the migration policy of individuals between subpopulations. In this particular model, a *immigration* process [9] is achieved when the best chromosome in every subpopulation abandons it and moves to an immediate neighbour. 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, $i = 1 \dots 4$.

Figure 3 shows the three different migration movements described above.

The genetic operators used in the distributed model are:

- **Selection mechanism**: linear ranking selection (LRS) [10] with stochastic universal sampling [11]. Values of LRS parameter is shown if Table 1.

- **Crossover operator**: BLX-α [12] operator using values for α shown in Table 2.

- **Mutation operator**: non-uniform mutation operator applied with probability $P_{mut} = 0,125$.

Table 1: LRS parameter values for each subpopulation

Exploitation				Exploration			
$+$	$\leftarrow$		$-$	$-$	$\rightarrow$		$+$
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

ISBN: 978-989-95079-6-8

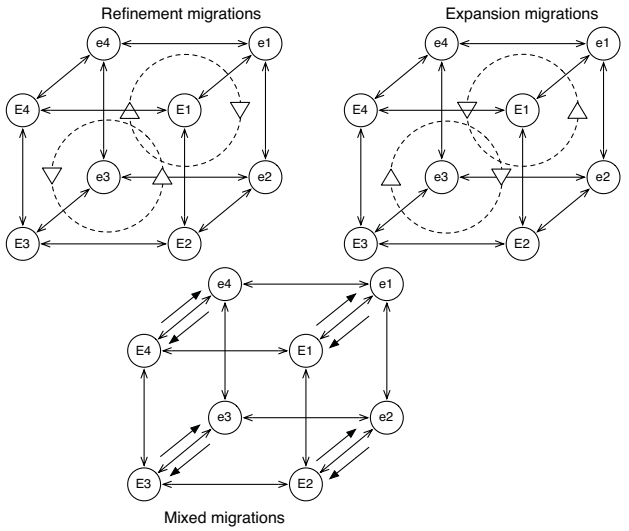


Figure 3: Three different migration movements

Table 2: Values of α for each subpopulation

Exploitation				Exploration			
$+$	$\leftarrow$		$-$	$-$	$\rightarrow$		$+$
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

As stated in [8], 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.

4 Empirical evaluation

In this section we describe the empirical evaluation we performed in order to assess the merits of our proposal.

We consider the tuning of FRBSs constructed for the modeling of some well-known regression problems. We have compared the quality of the tuning performed by the sequential algorithm and our proposal. We have selected four problems, some of their salient features are shown in Table 3 displayed in increasing complexity order.

Table 3: Data sets used to evaluate the algorithm

Data set	Variables	Instances
Electrical Maintenance	5	1056
Trasury	16	1049
Weather-Izmir	10	1461
Abalone	8	4177

A five-fold cross-validation approach has been used. So five runs with different independent test sets have been carried out for each problem. The performance of a fuzzy rule is measured as mean squared error (MSE) over the test set.

Our proposed distributed algorithm is compared with a specialized sequential genetic algorithm (CHC) [3] in terms of quality of the solutions achieved (MSE) as well as in time. As a starting point we have used the results achieved by applying Wang & Mendel fuzzy rule learning method [13]. These initial results are shown in Table 4.

Table 4: Initial results using Wang & Mendel fuzzy rule learning method

Dataset	Training	σ_{tra}	Test	σ_{test}
Electrical M.	57605.83	2840.78	57934.25	4732.66
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

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

$$R = \frac{T_{seq}}{T_{dist}} \quad (1)$$

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 5.

An interesting point to which to pay attention is the *evolution* of the mean squared error 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 6. An important fact to notice is that the mean squared error achieved with the distributed method is lower than the error obtained with the specialized genetic algorithm in all data sets at 50000 evaluations. The distribuited also obtains good results with fewer iterations in some cases (e.g. Electrical Maintenance, 25000 it.), but clearly its real effectiveness will be reached when the computation load is higher. Actually, we are running further experiments with datasets of higher complexity.

Table 5: 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

As shown in Table 5, the time gain ratio, R, increases with the problem complexity. In the less complex data sets the ratio obtained is substantially low because the sequential specialized genetic algorithm is very fast and the time spent in communications of the distributed approach slows it down in comparision. 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 genetic algorithm.

ISBN: 978-989-95079-6-8

Table 6: Mean squared errors in training and test sets. 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**

5 Conclusions and final remarks

We have developed and presented a very promising distributed algorithm for FRBSs lateral tuning that achieves better results than a specialized genetic algorithm. Due to its distributed nature and consequently the spatial separation implied, it needs more evaluations to converge than a classic sequential algorithm. It always presents the same behaviour in comparision to the specialized sequential algorithm: with a small number of evaluations it offers a higher error than the sequential approach but when the evaluations are high it gives better quality solutions.

Empirical results show that when the complexity of the problem grows, our distributed method takes adventage of the large computing times and converges to a better solution in less time. This property makes the distributed tuning algorithm very convenient when dealing with large scale data sets.

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 paralelized and the specialized algorithm is optimized for small size data sets where the search space is not too complex.

Since execution time and quality of the results are two properties always in conflict somehow, our approach is very convenient since it can be graduated in order to achieve faster execution times with a small cost in quality and viceversa.

We expect to achieve great execution time gains when applying the distributed FRBSs tuning algorithm with large scale data sets with a non significant loss of quality.

Acknowledgment

Supported by the Spanish Ministry of Education and Science under grant no. TIN2008-06681-C06-01.

References

[1] R. Alcalá, J. Alcalá-Fdez, M. J. Gacto, and F. Herrera. Rule base reduction and genetic tuning of fuzzy systems based on the linguistic 3-tuples representation. *Soft Computing*, 11(5):401–419, 2007.

[2] F. Herrera, M. Lozano, and J. L. Verdegay. Tuning fuzzy logic controllers by genetic algorithms. *International Journal of Approximate Reasoning*, 12:299–315, 1995.

[3] R. Alcalá, J. Alcalá-Fdez, and F. Herrera. A proposal for the genetic lateral tuning of linguistic fuzzy systems and its interaction with rule selection. *IEEE Trans. Fuzzy Syst.*, 15(4):616–635, 2007.

[4] F. Herrera and L. Martínez. A 2-tuple fuzzy linguistic representation model for computing with words. *IEEE Trans. Fuzzy Syst.*, 8(6):746–752, 2000.

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

[6] L. J. Eshelman. The CHC adaptive search algorithm: How to have safe serach when engaging in nontraditional genetic recombination. In G.J.E. Rawlin, editor, *Foundations of genetic Algorithms*, volume 1, pages 265–283. Morgan Kaufman, 1991.

[7] F. Herrera, M. Lozano, and A. M. Sánchez. A taxonomy for the crossover operator for real-coded genetic algorithms: An experimental study. *International Journal of Intelligent Systems*, 18:309–338, 2003.

[8] F. Herrera and M. Lozano. Gradual distributed real-coded genetic algorithms. *IEEE Transactions on Evolutionary Computation*, 4(1):43–63, April 2000.

[9] B. Kröger, P. Schwenderling, and O. Vornberger. Parallel genetic packing on transputers. *Parallel Genetic Algorithms: Theory and Applications: Theory Applications*, pages 151–186, 1993.

[10] J. E. Baker. Adaptative selection methods for genetic algorithms. In MA Ed. L. Erlbraum Associates: Hillsdale, editor, *Proceedings of the First International Conference on Genetic Algorithms and their Applications*, pages 101–111, 1985.

[11] J. E. Baker. Reducing bias and inefficiency in the selection algorithm. In *Proceedings of the 2nd International Conference on Genetic Algorithms (ICGA'87)*, 1987.

[12] L.J. Eshelman and J.D. Schaffer. Real-coded genetic algorithms and interval-schemata. *Foundations of Genetic Algorithms*, 2:187–202, 1993.

[13] L. X. Wang and J. M. Mendel. Generating fuzzy rules by learning from examples. *IEEE Trans. Syst. Man and Cybernetics*, 22(6), 1992.

ISBN: 978-989-95079-6-8