Parametric Optimization of a Fuzzy Identification System, Using Distributed Genetic Algorithm with Dynamic Migration Period

Marco Antonio Castro Liera
Instituto Tecnológico de La Paz, Baja California Sur, México
mcastroliera@acm.org

Francisco Herrera Fernández
Universidad Central "Martha Abreu" de Las Villas, Santa Clara, Republic of Cuba
herrera@fie.uclv.edu.cu

Abstract

A type of distributed genetic algorithm (DGA), with dynamic determination of the migration period is proposed. The algorithm is especially well suited for the parametric optimization of fuzzy identification systems and its implementation on heterogeneous clusters. The results of the parametric optimization of a Takagi-Sugeno-Kang (TSK) identification system for a bio-technological (fermentative) process are shown, including the analysis of the solution's quality, and the speedup obtained when nodes are added to the cluster.

1. Introduction

When combining fuzzy systems and genetic algorithms a synergy is created in which, as expressed by Zadeh [1], the main contribution of fuzzy logic is what may be called the calculus of fuzzy if-then rules, while genetic algorithms provide a methodology of systematized random search inspired by evolutionary processes in living species and proposed originally by John H. Holland [2].

The present work centers on the collaborative use of a specially adapted Distributed Genetic Algorithm [3] that runs on a low-cost parallel architecture known as a cluster, and a TSK fuzzy system [4] to solve a highly non-linear identification problem.

One of our main motivations is the use of a low-cost parallel architecture that will allow us to address computationally demanding problems without the need of acquiring parallel computers that are too expensive for most educative institutions at present. The algorithm was designed having in mind the special limitations of cluster computers when compared with parallel computers, particularly, the low inter-processor bandwidth.

Distributed genetic algorithms, also known as coarse-grained genetic algorithms and island model parallel genetic algorithms, are currently studied as one of the more scalable forms of parallel genetic algorithms [3,5-6].

2. The fermentative process

To carry out a fermentative process a quantity of microorganisms (biomass) are suspended in a food rich medium(substratum)

This kind of process presents a higly non-lineal behavior that responds to a dynamic system following the general structure [7]:

$$\frac{dX}{dt} = f(X) + bu \qquad \textbf{(1)}$$

where X is the vector formed by the two state variables [*x,s*], *u* is the input variable (S_{in}), and *b* is the constant dilution rate *D*.

3. The fuzzy model structure

We deal with a model with three input variables x_t, s_t, and s_{in} that represent the biomass, substrate, and input substrate concentration at instant *t* respectively. The model must determine the amount of biomass on the next instant *(t+1)*.

The proposed membership functions for the fuzzy sets of the input variables are generalized bells of the form:

$$\Phi(x,\alpha,\beta,\gamma) = \frac{1}{1+\left|\frac{x-\gamma}{2\alpha}\right|^{2\beta}} \qquad \textbf{(2)}$$

The fuzzy rule-base has the following structure:

IF x_t IS Φ_j AND s_t IS Φ_k THEN

$$x_{i(t+1)} = a_{i0} + a_{i1}x_t + a_{i2}s_t + a_{i3}s_{in} \quad (3)$$

With j varying from 1 to the number of fuzzy sets that divide x_t (in this case 3), k varying from 1 to the number of fuzzy sets that divide s_t (in this case 3) and i varying from 1 to the number of fuzzy rules on the fuzzy rules base (in this case 9).

For those processes like the studied case, where s_{in} remains constant, the terms a_{i0} and $a_{i3}s_{in}$ can be consolidated, giving us a simplified form of the rule base:

IF x_t IS Φ_j AND s_t IS Φ_k THEN

$$x_{i(t+1)} = a_{i0} + a_{i1}x_t + a_{i2}s_t \quad (4)$$

The predicted biomass is calculated as the pondered average of the rules outputs:

$$x_{(t+1)} = \frac{\sum_{i=1}^{9} h_i x_{i(t+1)}}{\sum_{i=1}^{9} h_i} \quad (5)$$

where the degree on which a rule is fulfilled is calculated using the T operator (in this case the product):

$$h_i = T(\Phi_j(x_t), \Phi_k(s_t)) \quad (6)$$

4. The genetic algorithm

The proposed structure of the chromosome for the genetic algorithm is the real vector:

$$[\alpha_j, \beta_j, \gamma_j, \alpha_k, \beta_k, \gamma_k | a_{i0}, a_{i1,} a_{i2}] \quad (7)$$

where α_j, β_j and γ_j represent the parameters for the membership function of the j-th fuzzy set on x_t,

α_k, β_k and γ_k represent the parameters for the membership function of the k-th fuzzy set on s_t.

a_{0i}, a_{1i} and a_{2i} are the i-th rule's output function coefficients.

Therefore we have a total of 18 parameters for the antecedent part of the TSK model and 27 coefficients for the consequent part.

The use of a distributed genetic algorithm was decided to better fit the characteristics of the computational architecture where the application runs, since on a cluster the main limitation is the low inter-processor communication bandwidth. The scalability of this kind of algorithms has been treated on several works [3-4]

A master task is in charge of the critical part of the process, and each sub-population of the algorithm is processed by a slave task.

4.1 Heterogeneous Clusters

We define a heterogeneous cluster as one in which the involved nodes have different architectures and processing speeds.

If the applied algorithm assumes a homogeneous cluster (one in which all the nodes have approximately the same computational power), and assigns the same amount of work load to each task, when the algorithm is executed on a heterogeneous cluster, the nodes with a higher computational power will have to wait until those that are slower have finished their processing before being able to perform migration, resulting on undesirable dead times on the faster nodes.

The chosen alternative to minimize this disadvantage is to dynamically determine the migration period for each sub-population in dependence of its execution time.

Figures 1 and 2 show the proposed modified algorithm.

```
Create P sub-populations
for i = 1 to P
  send parameters to sub-population i
for i = 1 to GMAX/MP
  for j = 1 to MR
    for k = 1 to P
      receive MR individuals from sub-population k
      receive T[k](population's k processing time)
    calculate average execution time TP
    for k = 1 to P
      send MR individuals from sub-population k to
sub-population P-(k+1)
    MP = MP*(TP/T[k])
    if MP > MPMAX
      MP = MPMAX
    Send the new MP to sub-population P-(k+1)
Display the individual with higher aptitude as the
solution of the optimization problem
```

Figure 1. Master task

```
receive parameters from master
generate randomly initial population
number of cycles C = GMAX/MP
for k=1 to C
  for i=1 to MP
     selection
     crossover
     mutation
     g=g+1
  send the MR best fitted individuals to the master
  receive MR individuals from the master
  replace the MR worst fitted individuals with the
received ones
  receive the new MP
```

Figure 2: Slave task

With this new approach those sub-populations with an execution time below the average perform more

iterations of the GA before the next migration, while those with a higher than average execution time perform fewer iterations.

This strategy progressively evens the execution times of all the sub-populations, so the waiting times decrease.

4.2 Two phase strategy

The optimization process was divided into two stages; the first stage aims to find a set of coefficients for the consequent part of the fuzzy rules set that minimize the total error calculated as:

$$\sum_{s=1}^{n} |xc_s - xm_s| \quad \textbf{(8)}$$

where *n* is the total training samples number (on this case 490 samples where available), *xc* is the calculated biomass concentration, and *xm* the measured biomass concentration.

The second stage of the optimization process aims to find a set of membership-function parameters that minimize the maximum error given by:

$$\max(|xc_1 - xm_1|, \ldots, |xc_n - xm_n|) \quad \textbf{(9)}$$

To conduct the first part of the optimization, fixed parameters were chosen for the generalized bell membership functions to create fuzzy sets that evenly divide the input variable spaces.

$$\alpha_j = 2 \quad \alpha_k = 2$$
$$\beta_j = 4 \quad \beta_k = 4$$
$$\gamma_j = x_{\min} + \frac{(x_{\max} - x_{\min})(2j-1)}{6}$$
$$\gamma_k = s_{\min} + \frac{(s_{\max} - s_{\min})(2k-1)}{6} \quad \textbf{(10)}$$

The above mentioned parameters were used to find a set of 27 output function coefficients that minimize the resulting model's total error given by (8).

The chromosome uses real representation on a 3 by 9 matrix that corresponds to the 3 output coefficients that each of the 9 lineal output functions requires.

For the second part of the optimization problem the best set of coefficients is fixed to try to find a set of parameters for the membership functions that minimizes the maximum prediction error given by (9).

4.3 Genetic algorithm parameters:

The chosen parameters for the first phase were:

- Population Size MU=600.
- Maximum generations GMAX=160.
- Tournament selection with tournament size Z=2.
- Crossing probability PC=0.7
- Uniform crossing parameter λ generated on (0,1).
- Uniform mutation, with mutation probability PM=0.2.
- Search ranges: [-7,7] for a_{i0}, [-4,4] for a_{i1} and [-2,2] for $a_{i2.}$
- Migration period MP=20.
- Maximum migration period MPMAX=30.
- Migration rate MR=2.

The chosen parameters for the second phase are:

- Population Size IMU=300.
- Maximum generations IGMAX=60.
- Tournament selection with tournament size IZ=2.
- Crossing probability IPC=0.7
- Uniform crossing parameter λ generated on (0,1).
- Uniform mutation, with mutation probability IPM=0.2.
- Search ranges:

$$\alpha_j = 2 \pm 1 \quad \alpha_k = 2 \pm 1$$
$$\beta_j = 4 \pm 1 \quad \beta_k = 4 \pm 1$$
$$\gamma_j = x_{\min} + \frac{(x_{\max} - x_{\min})(2j-1)}{6} \pm \frac{(x_{\max} - x_{\min})}{6}$$
$$\gamma_k = s_{\min} + \frac{(s_{\max} - s_{\min})(2k-1)}{6} \pm \frac{(s_{\max} - s_{\min})}{6} \quad \textbf{(11)}$$

- Migration period IMP=20.
- Maximum migration period IMPMAX=30.
- Migration rate IMR=2.

5. Results

Several test where conducted to measure the quality of the obtained models, as well as the execution time behavior under different conditions.

The sample size for each set of conditions was 200 runs of the algorithm and where conducted on a heterogeneous parallel virtual machine with a fast-ethernet switch and two types of nodes:

(a) Pentium IV@2.8GHz with 1024 KB cache and 512 MB DDR2 RAM@400 MHz.
(b) Pentium IV@2.0GHz with 512 KB cache and 512 MB DDR2 RAM@266 MHz.

5.1 Quality of the solution

The following table shows the error behavior with various sub-population numbers.

Table 1: Error behavior

SP	ATE	AE	AME
4	17.1492789	0.03499853	0.13239199
8	14.6136543	0.02982378	0.11224275
12	13.7827334	0.02812803	0.10273310
16	13.5702353	0.02769436	0.10423106
18	12.9202716	0.02636790	0.09917504

Where SP is the number of sub-populations, ATE the average total error, AE the average per-sample error, and AME the average maximum error.

The average execution time with the fore-mentioned parameters and a heterogeneous 18 nodes cluster was 220 seconds (using a node for each sub-population).

Figure 3 shows a comparison of the predicted and measured outputs and figure 4 plots the prediction error of a typical model generated with 4 sub-populations.

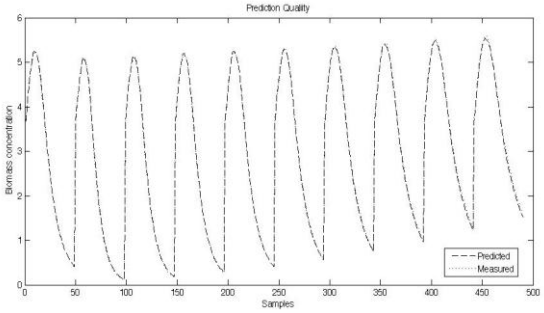

Figure 3: Prediction quality (4 sub-populations)

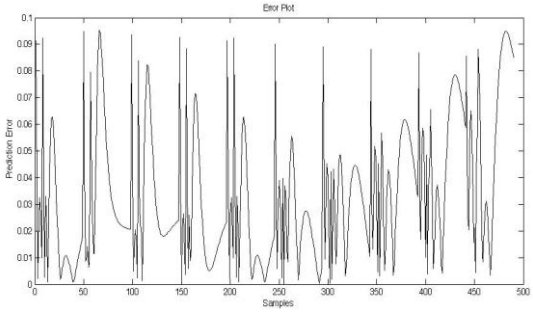

Figure 4: Prediction error (4 sub-populations)

Figure 5 shows a comparison of the predicted and measured outputs and figure 6 plots the prediction error of a typical model generated with 18 sub-populations.

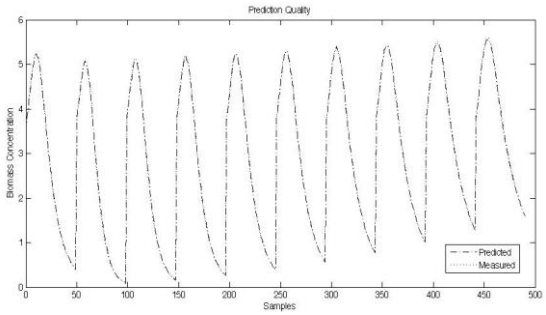

Figure 5: Prediction quality (18 sub-populations)

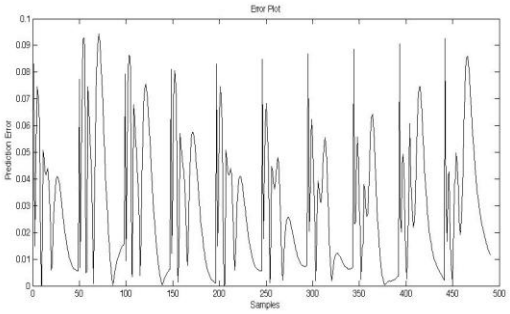

Figure 5: Prediction error (18 sub-populations)

5.2 Execution times

To achieve lower run-times, while maintaining reasonably good solutions the following parameters were modified:

- Population Size MU=400.
- Maximum generations GMAX=120.
- Migration period MP=15.
- Maximum migration period MPMAX=20.
- Population Size IMU=200.
- Maximum generations IGMAX=30.
- Migration period IMP=15.
- Maximum migration period IMPMAX=20.

With 18 sub-populations, an average total error of 15.2724839 that represents an average per-sample error of 0.03116833 was obtained; the average maximum error was 0.132167275.

The execution times that where obtained with a cluster of 9 type (a) nodes are shown on table 2.

Figure 6 shows the comparative speedup against lineal acceleration under the above mentioned conditions.

Table 2: Execution times homogeneous cluster

Nodes	Execution time (seconds)	Speedup
1	879	1.00
2	453	1.94
3	340	2.59
6	177	4.97
9	109	8.70

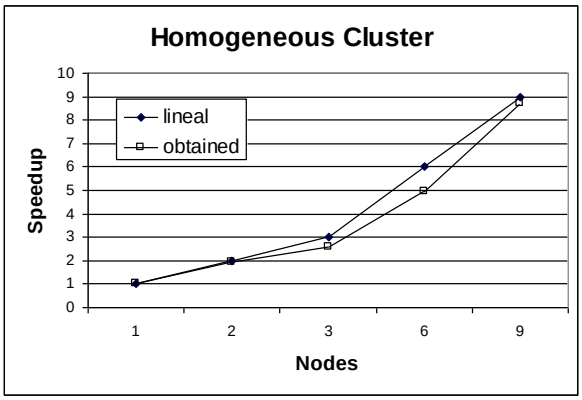

Figure 6: speedup

With a heterogeneous cluster of 10 type (a) and 8 type (b) nodes the following times were obtained using the non-adaptive version of the algorithm:

Table 3: Heterogeneous cluster non-adaptive version

Nodes (a)	Nodes (b)	Execution time (seconds)	Speedup
	1	2448	1
	2	1252	1.96
	3	846	2.91
	6	422	5.80
1	8	279	8.77
10	8	141	17.36

The obtained execution times show that type (a) nodes are much faster than type (b) ones but the algorithm shows a quasi-lineal speedup behavior, because it considers every node as equal to the rest, therefore it is unable to take advantage of the faster nodes added to the cluster (in fact we can observe comparing tables 2 and 3, that 9 nodes of type b ran faster than 18 mixed nodes). On the other hand, the adaptive version of the algorithm produced the following execution times:

Table 4: Heterogeneous cluster adaptive version

Nodes (a)	Nodes (b)	Execution time (seconds)	Speedup
	1	2454	1.00
	2	1253	1.96
	3	852	2.88
	6	404	6.07
1	8	243	10.10
10	8	100	24.54

It is clear that, when adding faster nodes to an existing cluster, this new version of the algorithm is able to exploit their higher computational power.

Figure 7 shows the comparative speedup of both algorithms against a lineal acceleration.

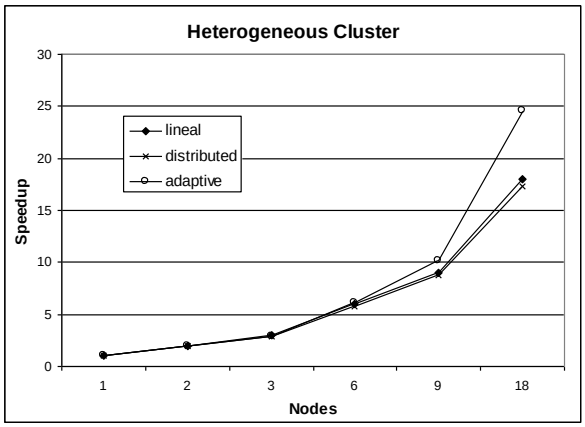

Figure 7: speedup

6. Conclusions

The proposed algorithm is able to generate adequate solutions for the identification problem.

The presented results show that the quality of the solution improves as the number of sub-populations increase; it is also evident that we can maintain low execution times while increasing the amount of sub-populations by adding nodes to the cluster.

The speedup tests for up to 18 nodes clusters show that, at least for this amount of processors, the algorithm can be scaled without considerable loss of performance.

It can be observed that when we add faster nodes to a cluster, hyper-lineal accelerations are obtained with the adaptive version of the algorithm.

The algorithm can be easily adapted for the parametric optimization of different TSK Fuzzy models; the scope of application is vast but limited by the amount of data that needs to be exchanged between

sub-populations for models with a very large number of parameters and large number of sub-populations.

However, these limitations can be reduced with the advent of new networking technologies such as gigabit-Ethernet.

The real limit for the number of sub-populations that the algorithm can exploit on fast-Ethernet networks for this specific problem remains to be empirically found, since for the moment on test runs with 18 nodes and up to 36 sub-populations that limit was not reached.

Source code of the application written in C for GNU/Linux and PVM 3.4 is available at [8].

7. References

[1] Zadeh, Lofti A., Foreword on *Genetic fuzzy systems: evolutionary tuning and learning of fuzzy knowledge bases*. Cordon, O, et all, 2001, Singapore: World Scientific. XXV, 462.

[2] Holland, J.H., *Adaptation in Natural and Artificial Sstems*. (1975) 6 ed. 2001, Michigan: MIT Press. 205

[3] Nowostawski, M. and P. Ricardo, "Parallel Genetic Algorithm Taxonomy". *Third International Conference of Knowledge-Based Intelligent Information Engineering Systems*, 1999.

[4] Cordon, O. et al., *Genetic fuzzy systems: evolutionary tuning and learning of fuzzy knowledge bases*. 2001, Singapore: World Scientific. XXV, 462.

[5] Alba E. and Tomassini, M.: Parallelism and Evolutionary Algorithms, IEEE Transactions on Evolutionary Computation, Vol. 6 No 5, (2002)

[6] Alba, E. and Troya, J.M.: Analyzing Synchronous and Asynchronous Parallel Distributed Genetic Algorithms, Future Generation Computer Systems, 17(4):451-465 (2001)

[7] Herrera Fernández, F., et al., Aplicación de las Técnicas de la Inteligencia Artificial en un Proceso Biotecnológico de Reproducción Celular (Embriogénesis Somática). 2003, Universidad Central "Marta Abreu" de Las Villas: Santa Clara Libre, República de Cuba.

[8] Castro Liera, Marco A. *biomass full model generator source code* http://sistemas.itlp.edu.mx/castroga/biomasa-full.tgz. 2006, Instituto Tecnológico de La Paz.