Mining Multivariate Time Series Models with Soft-Computing Techniques: A Coarse-Grained Parallel Computing Approach

Valdés, Julio; Barton, Alan

National Research Council Canada
Conseil national de recherches Canada

Institute for Information Technology
Institut de technologie de l'information

NRC · CNRC

Mining Multivariate Time Series Models with Soft-Computing Techniques: A Coarse-Grained Parallel Computing Approach *

Valdés, J., and Barton, A.
2003

* published in Lecture Notes in Computer Science (Kumar, Gavrilova, Tan, L'Ecuyer eds.) LNCS 2668, pp. 259-268, Springer-Verlag, 2003. NRC 46511.

Canada

Mining Multivariate Time Series Models with Soft-Computing Techniques: A Coarse-Grained Parallel Computing Approach *

Julio J. Valdés and Alan J. Barton

National Research Council of Canada,
Institute for Information Technology,
1200 Montreal Road, Ottawa ON K1A 0R6, Canada.
julio.valdes@nrc-cnrc.gc.ca
alan.barton@nrc-cnrc.gc.ca

Abstract. This paper presents experimental results of a parallel implementation of a soft-computing algorithm for model discovery in multivariate time series, possibly with missing values. It uses a hybrid neural network with two different types of neurons trained with a non-traditional procedure. Models describing the multivariate time dependencies are encoded as binary strings representing neural networks, and evolved using genetic algorithms. The present paper studies its properties from an experimental point of view (using homogeneous and heterogeneous clusters) focussing on: i) the influence of missing values, ii) the factors controlling the parallel computation, and iii) the effectiveness of the time series prediction results. Results confirm that i) the algorithm possesses high tolerance to missing data, ii) Athon-based homogeneous clusters have higher throughput than Xeon-based homogeneous clusters, iii) an increase of the number of slaves reduces the processing time until communication overhead dominates (as expected), and iv) running the algorithm in parallel does not affect the RMS error (as expected). Even though much of this behavior could be qualitatively expected, appropriate tradeoffs between error and time were actually discovered, thereby enabling more effective, systematic, future uses of the system.

1 INTRODUCTION

Multivariate time series modelling and prediction is a very important subject as development in sensor, communication and computer technologies allow the monitoring of complex processes of interest in many domains (industry, environment, medicine, economics, etc.). It is very important to discover patterns of *delayed* cause-effect dependencies relating the different variables and factors, and

* The authors are very grateful to Nicolino Pizzi (Institute for Biodiagnostics, National Research Council of Canada), to Robyn Paul (University of Waterloo) and to Robert Orchard (Institute for Information Technology, National Research Council of Canada).

this situation turns very complex when the multivariate time dependent process is composed by heterogeneous variables (numeric, non-numeric, fuzzy quantities and others), and when missing information affects data quality. Finding models in this situation is a formidable task, given the size of the search space and the computational effort required, thus making a parallel approach immediately appealing. An algorithm oriented to these kinds of problems was developed elsewhere [V02], as well as a parallel implementation [VM02]. This paper studies its properties in more depth from an experimental point of view, focussing on the influence of missing values and the factors controlling the parallel computation, and on the effectiveness of the time series prediction results.

2 THE ALGORITHM

The purpose is to discover *dependency models* in heterogeneous multivariate time varying processes. They expresses the relationship between values of a previously selected time series (the target), and past values of the entire set of series. Heterogeneity means the presence of ratio, interval, ordinal or nominal scales, fuzzy and other magnitudes. Moreover, the series may contain missing values. The class of functional models considered is a generalized non-linear auto-regressive (AR) model (1) (other functional models are also possible),

$$S_T(t) = \mathrm{F}\begin{pmatrix} S_1(t-\tau_{1,1}), \cdots, S_1(t-\tau_{1,p_1}), \\ S_2(t-\tau_{2,1}), \cdots, S_2(t-\tau_{2,p_2}), \\ \cdots \\ S_n(t-\tau_{n,1}), \cdots, S_n(t-\tau_{n,p_n}) \end{pmatrix} \tag{1}$$

where $S_T(t)$ is the target signal at time t, S_i is the i-th time series, n is the total number of signals, p_i is the number of time lag terms from signal i influencing $S_T(t)$, $\tau_{i,k}$ is the k-th lag term corresponding to signal i ($k \in [1, p_i]$), and F is the unknown function describing the process. This approach requires the simultaneous determination of: *i)* the number of required lags for each series, *ii)* the particular lags within each one carrying the dependency information, and *iii)* the prediction function. A requirement on function F is to minimize a suitable prediction error. This is approached with a soft computing precedure based on: *i*) exploration of a subset of the *model space* with a genetic algorithm, and *ii*) use of a similarity-based neuro-fuzzy system representation for the unknown prediction function.

Evolving neuro-fuzzy networks with genetic algorithms has been done for training *single* networks. The situation here involves the construction and evaluation of *thousands* or *millions* of networks, due to the equivalency between the model and the network spaces. Thus, the use of conventional architectures and training procedures becomes prohibitive. Other difficulties with classical approaches include finding the number and composition of hidden layers, using mixed numeric, non-numeric, fuzzy and missing values, etc. The present approach is based on the heterogeneous neuron model [VG97], [B00], [V02a],

which considers a neuron as a general mapping between heterogeneous multidimensional spaces $h : \hat{\mathcal{H}} \times \hat{\mathcal{H}} \rightarrow \mathcal{Y}$, where $\mathcal{Y}$ is an abstract set. If $\overleftarrow{x}, \overleftarrow{w} \in \hat{\mathcal{H}}$ (the input and the neuron weights respectively) and $y \in \mathcal{Y}$, then $y = h(\overleftarrow{x}, \overleftarrow{w})$.

In the *similarity-based* h-neuron model, the aggregation function is given by a *similarity function* $s(x, w)$ between the input and the neuron weights (vectors from a heterogeneous space), whereas the activation is a non-linear function. This neuron maps a n-dimensional heterogeneous space onto the extended [0,1] real interval. The output expresses the degree of similarity between the input pattern and the neuron weights $s : (\hat{\mathcal{H}} \times \hat{\mathcal{H}}) \rightarrow [0, 1] \cup \{\mathtt{X}\}$, where $\mathtt{X}$ is the symbol denoting the missing value (Fig-1 (left)). A hybrid network layout using heterogeneous neurons in the hidden layer and classical neurons in the output layer is suitable for the purpose of model mining. For multivariate heterogeneous time series, where a single time series is targeted for prediction, the network architecture is shown in (Fig-1 (right)).

During network operation each hidden layer neuron computes its similarity with the input vector and the k-best responses are retained (k is a pre-set number of h-neurons to select). They represent the fuzzy memberships of the inputs w.r.t. the classes defined by their weights. Neurons in the output layer compute a normalized linear combination of the expected target values used as neuron weights (W_i), with the k-similarities coming from the hidden layer.

$$output = (1/\Theta) \sum_{i \in \mathcal{K}} h_i W_i, \qquad \Theta = \sum_{i \in \mathcal{K}} h_i \tag{2}$$

where $\mathcal{K}$ is the set of k-best h-neurons of the hidden layer and h_i is the similarity of the i-best h-neuron w.r.t the input vector.The network output is a fuzzy estimate of the predicted value.

Given a similarity function $\mathcal{S}$ and a target series the network is built and trained as follows: Set a similarity threshold $T \in [0, 1]$ and extract the subset $\mathcal{L}$ of input patterns Ω ($\mathcal{L} \subseteq \Omega$) such that for every $x \in \Omega$, there exist a $l \in \mathcal{L}$ such that $\mathcal{S}(x, l) \geq T$. The elements of $\mathcal{L}$ will be the hidden layer h-neurons, while the output layer is built by using the corresponding target outputs as the weights of

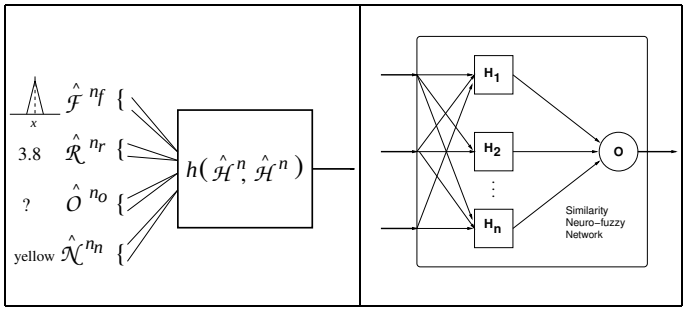

Fig. 1. Left: A heterogeneous neuron. Right: A hybrid neuro-fuzzy network.

the neuron(s). This procedure is *very* fast and allows for the rapid construction and training of many networks.

2.1 PARALLEL IMPLEMENTATION

A parallel implementation following a master-slave approach was made. Parallelization of the algorithm is done at the population level. The master initializes the evolutionary mechanism and distributes the generated chromosomes (model encodings). The master also controls the integration of the corresponding evaluations returned by the slaves. For a received chromosome, a slave constructs the corresponding network, trains it, evaluates it on the test set, and returns the *root mean squared error* (RMS error) to the master. This is thus a coarse grained, embarassingly parallel approach. Medium and fine grained parallel levels also exist, associated with the evaluation of the neural network and the computation of the similarity functions respectively. Further studies should address these latter 2 parallel levels. The coarse-grained system architecture is shown in Fig-2.

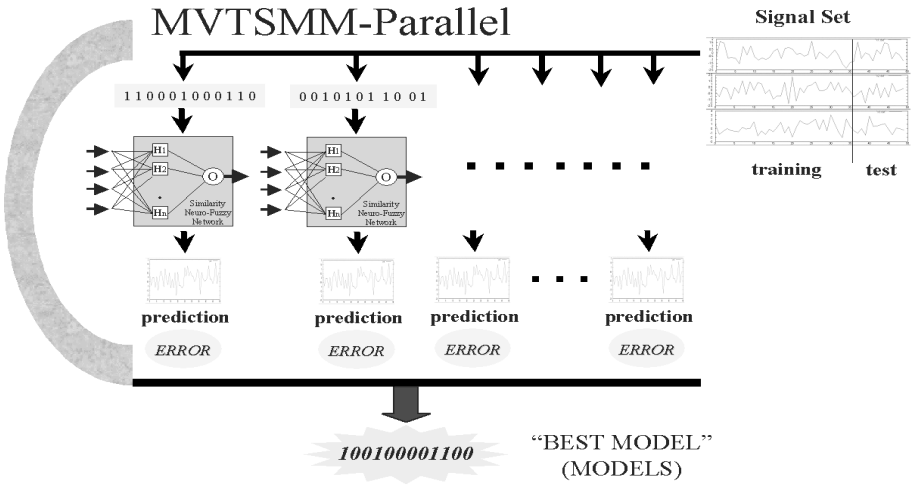

Fig. 2. Multivariate Time Series Model Miner System Architecture. The arc is the parallel genetic algorithm evolving populations of similarity-based networks. They represent different dependency patterns which are generated by the master and evaluated by the slaves during the search process.

The system was implemented in C++ using the GaLib version 2.4.5 [WM96], which was modified to run using LAM-MPI versions 6.5.4/MPI 2 and 6.5.8/MPI 2, C++/ROMIO [MPI].

3 EXPERIMENTAL SETUP

A multivariate time series data set consisting of 10 series with 1140 observations of average monthly temperatures from different sites in the Washington State (USA) was chosen. They were recorded during the period 1895-1989 [M95], and compiled by the National Oceanic and Atmospheric Administration (USA). Originally this data had no missing values and is shown in Fig-3. The West Olympic Coastal drainage region (the top series) was chosen as the target. No preprocessing was applied to the time series in order to test the approximation capacity and robustness of the algorithm in the worst conditions.

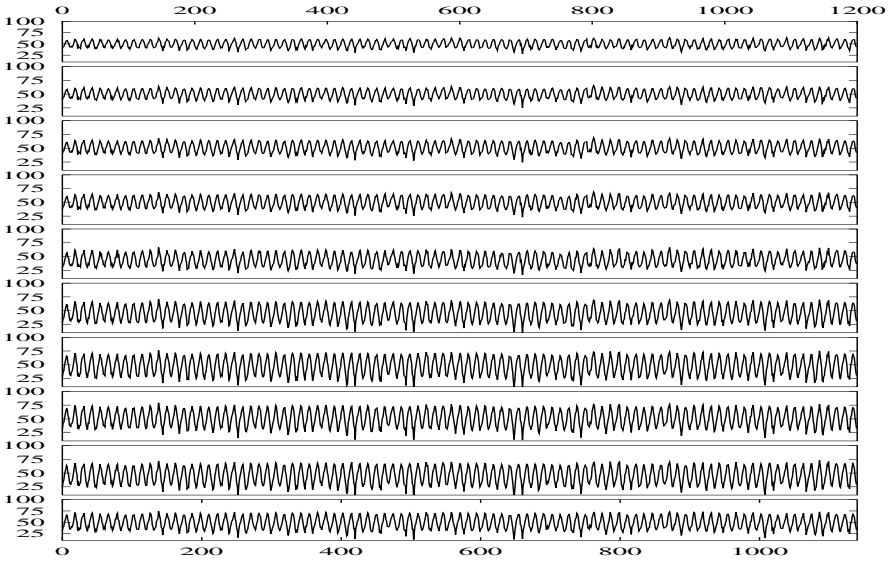

Fig. 3. Temperature data from 10 Washington State sites (in degrees Farenheit).

Three new sets of time series were constructed by introducing 25%, 50% and 75% of uniformly distributed missing values into all 10 original series. Missing values were introduced in a *signal-wise* manner. Each series was divided evenly into a training set and a test set. The training set for each signal contains the same percentage of introduced missing values, while the test sets were left intact. In this way, all signals contain exactly the same amount of missing values, as defined by the corresponding preset percentage.

The similarity function used is the non linear transformation ($s = 1/(1+d)$, where s is a similarity and d a distance) of a modified Euclidean distance, accepting missing values. Given two vectors $\overleftarrow{x} =< x_1, \cdots, x_n >, \overleftarrow{y} =< y_1, \cdots, y_n > \in \mathbb{R}^n$, defined by a set of variables (i.e. attributes) $A = \{A_1, \cdots, A_n\}$, let $A_c \subseteq A$ be the subset of attributes s.t. $x_i \neq \texttt{X}$ and $y_i \neq \texttt{X}$. The modified distance function is

$d_e = (1/card(A_c)) \sum_{A_c} (x_i - y_i)^2$, which is a normalized distance and therefore, independent of the number of attributes. Consequently, *no imputation* of missing values to the data set is performed. The number of responsive neurons in the hidden layer was fixed at 7. The similarity threshold was set to 1, the maximum lag depth to 20, and the relative percentage of training/test to 50%. For the genetic algorithm, the number of generations was fixed at 10 and the population size to 50. Binary chromosomes encoding model components as given by (1) were used with a double point crossover operator and standard bit-reversal mutation. Selection was kept constant (roulette wheel method) and complete population replacement with elitism were used. Crossover and mutation probabilities were 0.6 and 0.01 respectively.

Experiments were conceived to observe how timing and RMS error were affected by several factors in the parallel implementation: the number of slaves, the number of units, and the use of homogeneous and heterogeneous clusters. A total of 640 experiments were evenly distributed among 8 different cluster configurations with 80 experiments per configuration (see Table-1). Within each configuration (a fixed number of units), the percentage of missing spans $[0\%, 25\%, 50\%, 75\%]$ and the number of slave processes ranges over $[6, 8, 10, ..., 44]$.

Cluster Identifier	Cluster Type	Number of Units	Composition
1	Homogeneous (IIT)	3	3 dual Xeons
2	Homogeneous (IBD)	3	3 dual Athlons
3	Homogeneous (IIT)	2	2 dual Xeons
4	Homogeneous (IBD)	2	2 dual Athlons
5	Homogeneous (IIT)	1	1 dual Xeon
6	Homogeneous (IBD)	1	1 dual Athlon
7	Heterogeneous (IBD)	3	1 Xeon, 2 PIII
8	Heterogeneous (IBD)	17	all active units

Table 1. Selected Cluster Configurations from the National Research Council Canada. IIT: Institute for Information Technology, IBD: Institute for Biodiagnostics.

The homogeneous Beowulf cluster configurations located at IIT, consist of subsets from 3 dual Xeon processor units operating at 2 Ghz frequency, each with 1Gb RAM using Red Hat Linux 7.2 with LAM/MPI 6.5.4.

The cluster at IBD is composed by 4 dual Athlon processor units operating at 1.666Ghz frequency, each with 2Gb RAM, 5 Xeon processor units operating at 1.6Ghz with 1Gb RAM, 2 Pentium III units at 1Ghz with 256Mb RAM, 1 Pentium III unit at 1Ghz with 512Mb RAM, 1 Pentium III unit at 930Mhz with 256Mb RAM, and 4 Pentium III units at 860Mhz with 512Mb RAM. This cluster uses Red Hat Linux 7.3, LAM/MPI 6.5.8 and has a shared network drive.

4 RESULTS

Results are organized by the type of cluster used for the parallel experiments.

4.1 EXPERIMENTS WITH HOMOGENEOUS CLUSTERS

The behavior of time and RMS error with the number of slaves and the % of missing values for cluster 1 is shown in Fig-4(a-b). Time is inversely proportionate to the % of missing values (due to data dilution), and almost independently of the number of slaves (as expected). Time is almost unaffected by the increase of the number of slaves within the range $[6 - 28]$, for all missing value variants. However, for more than 28 slaves, time slowly increases (almost linearly) for the whole missing value range. RMS error is independent of the number of slaves (as expected) and % missing except for the extreme case (75%), which is an indication of the algorithm's robustness.

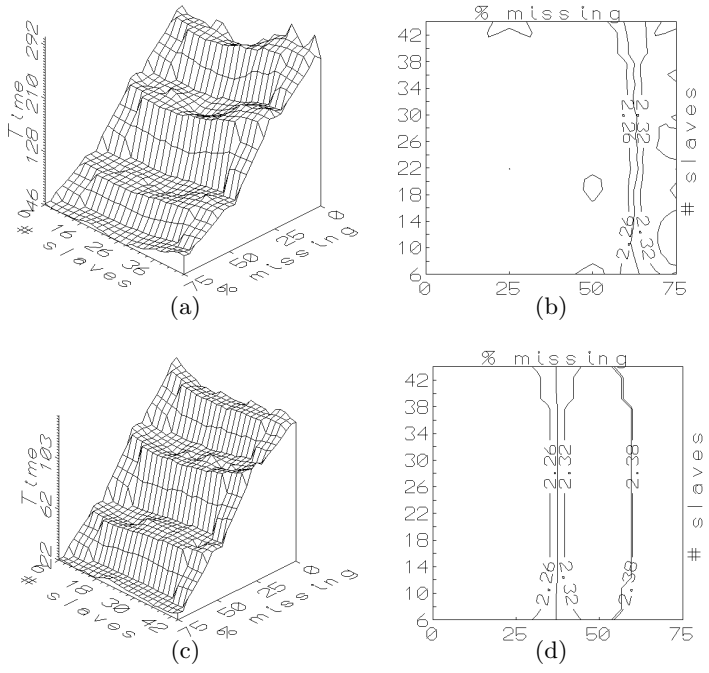

Fig. 4. Behavior of overall time and RMS error for Cluster 1 (a,b), and Cluster 2 (c,d).

The behavior of time and RMS error with the number of slaves and the % of missing values for cluster 2 (Fig-4(c-d)) is similar to that of cluster 1. However, there is a significant time difference, of approximately one half, between the two.

This shows the higher throughput of the Athlon processor-based cluster (with the same number of units but smaller frequency). The time is much less affected by the number of slaves than for cluster 1.

Clusters $3-4$ exhibit similar characteristics (figures deleted for brevity). However, overall execution time increases w.r.t. clusters $1-2$ by approximately a 1.5 factor, and the influence of the number of slaves (on time) is close to none. Again, the Athlon-based cluster outperforms the Xeon-based one substantially. RMS errors behave almost the same. There are changes due to the increased influence of the missing values, but of small magnitude (again, evidence of the algorithm's robustness).

Clusters 5-6 (Fig-5), exhibits slightly different characteristics. Overall execution time increases w.r.t. clusters 3-4 by approximately a factor of 2. It is interesting to note that cluster 6 has similar time characteristics to cluster 3, except for the different trend w.r.t. the number of slaves. That is, as the number of slaves increases within cluster 3, the number of slaves within cluster 6 decreases slightly, and then flattens. The RMS error is affected by the number of slaves and the % of missing values as in clusters 3-4 (no markedly different differences).

In terms of time, cluster 6 is approximately 4 times faster than cluster 5. Again, this shows the Athlon-based cluster outperforms the Xeon-based one substantially (Fig-5(a),(c)). RMS errors behave almost the same for cluster 5 and 6 (Fig-5(b),(d)) when compared against to each other and to cluster 3 and 4, indicating the number of processing units does not affect the RMS error (as expected).

4.2 EXPERIMENTS WITH HETEROGENEOUS CLUSTERS

The performance of cluster 7 is shown in Fig-6(a)(b). When compared against clusters 1 and 2 (see Fig-4), time is approximately 1.5 times slower, and approximately 3 times slower than cluster 2. This is due to the slower processors used to compose cluster 7. Its timing seems to most closely match cluster 3 and cluster 6. It is interesting to observe that this 3 unit heterogeneous cluster is outperformed by a one unit cluster. Further, time does not appear to be influenced by the number of slaves.

Cluster 8's timing (Fig-6(c)) is significantly smaller than all previous clusters. This is due to the large increase in the number of units (17 in this case vs. a maximum of 3 in all previous cases). The % of missing values influences time in the same way as all previous cases. However, time is dramatically reduced as the number of slaves increase from 6 to approximately 28. From 6 to 17, this phenomenon can be explained because the cluster is using more an more units (at least 1 slave per unit). Possibly, the reduction in processing time when the number of slaves is increased from 18 to 28 is due to slaves not having to wait for data to be transferred. That is, one slave can be computing while a second slave is blocked. Further studies are required to understand more clearly this process. Notice that time is not affected when the number of slaves increases past 28.

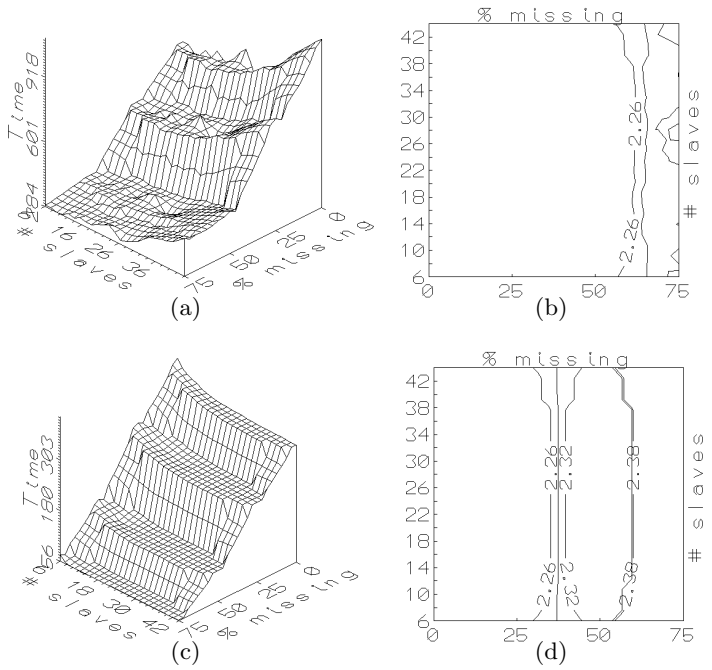

Fig. 5. Behavior of overall time and RMS error for Cluster 5 (a,b), and Cluster 6 (c,d).

The RMS error (Fig-6(d)) is not significantly different from all previous clusters (again, as theoretically expected).

5 CONCLUSIONS

The results are conditioned to the properties of the data set and cluster configurations used. The discovered dependency model errors are not affected by parallelizing the algorithm, but are affected by the increased presence of missing information. The choice of cluster configuration dramatically affects time performance. The results also show which parallel settings may lead to close to optimal use of the clusters for this algorithm implementation. The speed with which models can be generated and explored, makes it a suitable data mining technique for rapid prototyping. Further experiments are necessary with larger more complex data sets using different levels of granularity of parallelization.

References

[B00] Belanche, Ll. : Heterogeneous neural networks: Theory and applications. PhD Thesis, Department of Languages and Informatic Systems, Polytechnic University of Catalonia, Barcelona, Spain, July,(2000).

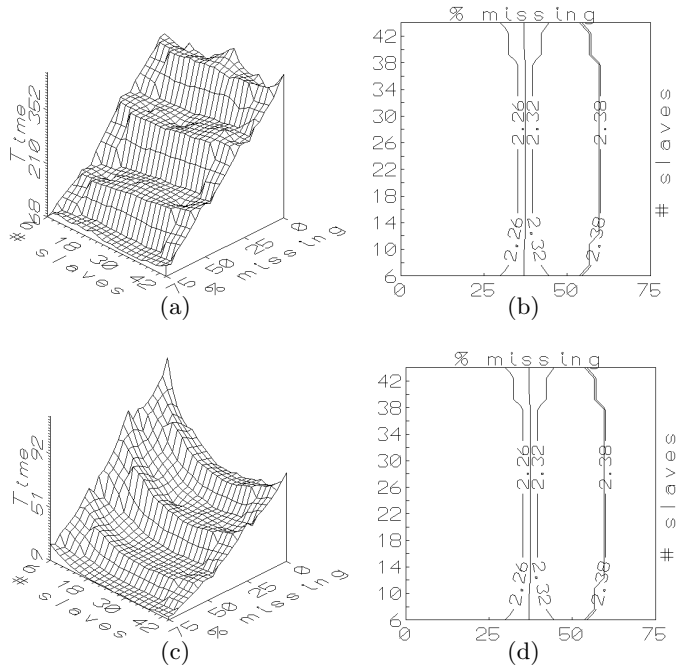

Fig. 6. Behavior of overall time and RMS error for Cluster 7 (a,b), and Cluster 8 (c,d).

[M95] Masters,T. : Neural, Novel & Hybrid Algorithms for Time Series Prediction. John Wiley & Sons, (1995).

[MPI] Pacheco,P. : Parallel Programming with MPI. Morgan Kaufmann, (1997).

[VG97] Valdés,J.J. García, R. : A model for heterogeneous neurons and its use in configuring neural networks for classification problems. Proc. IWANN'97, Int. Conf. On Artificial and Natural Neural Networks. Lecture Notes in Computer Science **1240**, Springer Verlag, (1997), pp.237-246..

[V02] Valdés,J.J. : Similarity-based Neuro-Fuzzy Networks and Genetic Algorithms in Time Series Models Discovery. NRC/ERB-1093, 9 pp. NRC 44919.(2002).

[VM02] Valdés,J.J., Mateescu, G. : Time Series Model Mining with Similarity-Based Neuro-Fuzzy Networks and Genetic Algorithms: A Parallel Implementation. Third. Int. Conf. on Rough Sets and Current Trends in Computing RSCTC 2002. Malvern, PA, USA, Oct 14-17. Alpigini, Peters, Skowron, Zhong (Eds.) Lecture Notes in Computer Science (Lecture Notes in Artificial Intelligence Series) LNCS 2475, pp. 279-288. Springer-Verlag , 2002.

[V02a] Valdés, J.J. : Similarity-based heterogeneous neurons in the context of general observational models. *Neural Network World*, **12** (5), (2002), 499–508.

[WM96] Wall,T. : GaLib: A C++ Library of Genetic Algorithm Components. Mechanical Engineering Dept. MIT (http://lancet.mit.edu/ga/), (1996).