Multivariate Time Series Model Discovery with Similarity -Based Neuro-Fuzzy Networks and Genetic Algorithms

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

*Multivariate Time Series Model Discovery with Similarity -Based Neuro-Fuzzy Networks and Genetic Algorithms **

Valdés, J., and Barton, A.
April 2003

* published in Proceedings of the IEEE, INNS, IJCNN 2003 International Joint Conference on Neural Networks. IEEE Catalog Number: 03CH37464C, ISBN: 0-7803-7899-7. Oregon, Portand, USA..

Canada

Multivariate Time Series Model Discovery with Similarity-Based Neuro-Fuzzy Networks and Genetic Algorithms

Julio J. Valdés
National Research Council Canada
Institute for Information Technology
1200 Montreal Road, Ottawa ON K1A 0R6
Canada

Email: julio.valdes@nrc.ca

Alan J. Barton
National Research Council Canada
Institute for Information Technology
1200 Montreal Road, Ottawa ON K1A 0R6
Canada

Email: alan.barton@nrc.ca

***Abstract*— This paper studies the properties of a hybrid technique for model discovery in multivariate time series, using similarity based hybrid neuro-fuzzy neural networks and genetic algorithms. This method discovers *dependency patterns* relating future values of a target series with past values of all examined series, and also constructs a prediction function. It accepts a mixture of numeric and non-numeric variables, fuzzy information, and missing values. Experiments were made with a real multivariate time series for studying the model discovery ability and the influence of missing values. Results show that the method is very robust, discovers relevant interdependencies, gives accurate predictions and is tolerant to considerable proportions of missing information.**

I. Introduction

Multivariate time-varying processes occur in many domains and their importance is increasing with the developments in sensor technologies and advanced monitoring systems. Processes of this kind involve many variables changing simultaneously with time. In general, these processes are heterogeneous in nature, consisting of numeric and non-numeric quantities, typically with missing values (i.e. gaps in the observations occur due to sensor saturation, malfunctioning, etc.), which do not necessarily distribute in the same way in the different observed variables. Also, measurements and observations are obtained with different degrees of precision and indetermination (e.g. data may be fuzzy). One of the most important data mining and knowledge discovery tasks in the study of time dependent information is finding *interesting dependencies* between past and future values of the observed variables (i.e. *dependency patterns* or *models*). Another goal is to find suitable prediction estimators for forecasting purposes. The use of classical methods is limited by different factors. Some factors are related to the underlying assumptions about the data concerning type, volume, homogeneity, complexity, precision, the curse of dimensionality, etc. In many cases these methods are based on assumptions which don't hold or are unpractical to verify. From a soft-computing approach to solving this problem, neural networks have been applied extensively for time series and signal analysis, however, the multivariate case is less frequently studied. A technique for model discovery and prediction in multivariate time series was introduced recently in [6]. That method accepts heterogeneous, large series with different degrees of imprecision (possibly with missing data) and uses hybrid networks mixing different neuron models (similarity-based and classical). These networks operate in a neuro-fuzzy mode. Preliminary applications showed interesting behavior with respect to speed, performance and sensitivity to detect internal dependencies. This paper studies the behavior of this kind of network in a strongly multivariate time series modeling and forecasting problem. The paper also shows the network's robustness w.r.t. increased presence of missing values in the time series, and choice of algorithm parameters.

II. METHOD OUTLINE

The objective is to extract plausible *dependency models* in heterogeneous multivariate time varying processes, expressing the relationship between future values of a previously selected time series (the target), and the entire set of series. Heterogeneity means the presence of ratio, interval, ordinal or nominal scales, and fuzzy magnitudes. Moreover, the series may contain missing values. The first step is to set a conceptual class of functional models and in this case a generalized non-linear auto-regressive (AR) model was used (1) (other classes of functional models are also possible),

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

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 $\mathbf{F}$ is the unknown function describing the process. The second step in the proposed method, is 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 natural requirement on function F is the property of minimizing a suitable prediction error. This is approached with a soft computing precedure based on: (a) exploration of a subset of the entire *model space* with a genetic algorithm, and (b) 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 a long time, but only for training purposes and in the context of a *single* network. The situation here is very different: it involves the construction and evaluation of *thousands* or even *millions* of networks, since the search in the space of dependency models is equivalent to the search in the space of networks. Thus, the use of conventional architectures and training procedures becomes prohibitive. Other difficulties with classical approaches include finding the number of hidden layers and their composition, using mixed numeric, non-numeric, fuzzy and missing values, etc. The present approach is based on the heterogeneous neuron model [5], [1], [7], 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. For the h-neuron used in the experiments, the chosen aggregation is a similarity function constructed by non-linearly transforming a distance function (allowing missing values), and the chosen activation function is the identity. Several distance functions were used in the experiments (see section III). This neuron maps a n-dimensional heterogeneous space onto the extended [0,1] real interval in such a way that the output expresses the degree of similarity between the input pattern and neuron weights $s : (\hat{\mathcal{H}} \times \hat{\mathcal{H}}) \rightarrow [0, 1] \cup \{\mathtt{X}\}$, where `X` is the symbol denoting the missing value. 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. In the particular case of multivariate heterogeneous time series, where a single time series is targeted for prediction based on the entire signal set, a suitable network architecture is shown in (Fig-1).

Network operation is as follows: Each neuron in the hidden layer 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 the hidden layer neurons. 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, representing a fuzzy estimate for the predicted value.

Assuming that a similarity function $\mathcal{S}$ has been chosen and that the target is a single time series, this *case-based* neuro-fuzzy network is built and trained as follows: Define a similarity threshold $T \in [0, 1]$ and extract the subset

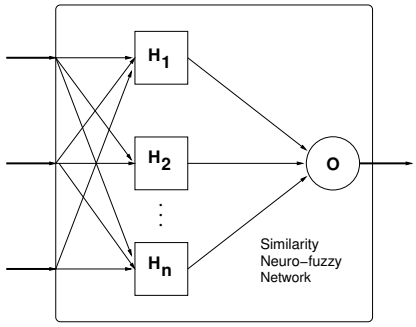

Fig. 1. Neuro-fuzzy network composed by h-neurons in the hidden layer and one classical neuron in the output layer.

$\mathcal{L}$ of the set of input patterns Ω ($\mathcal{L} \subseteq \Omega$) such that for every input pattern $x \in \Omega$, there exist a $l \in \mathcal{L}$ such that $\mathcal{S}(x, l) \geq T$. The hidden layer is constructed by using the elements of $\mathcal{L}$ as h-neurons, while the output layer is built by using the corresponding target outputs as the weights of the neuron(s). This training procedure is *very* fast and allows for the rapid construction and testing of many networks, as training complexity is $O(n^2)$ (In the case of $T = 1$, training is exactly $\Omega(n)$=$O(n)$).

A parallel implementation following a master-slave approach was made using LAM/MPI [3] and the GaLib [8]. The slaves construct and evaluate individual neuro-fuzzy networks based on models received from the master, which controls the genetic algorithm process at the population level. The system architecture is shown in Fig-2.

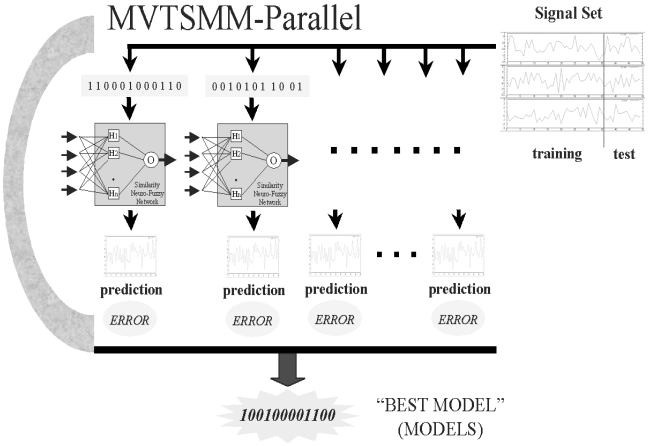

Fig. 2. Multivariate Time Series Model Miner System Architecture. The arc is the parallel genetic algorithm evolving populations of similarity-based networks. They represent different dependency patterns which are generated and evaluated during the search in the space of multivariate time series models.

The system's behavior is controlled by three classes of factors related to: *i)* the neuro-fuzzy network, *ii)* the genetic algorithm, and *iii)* the parallel implementation. Related to *(i)* are the specific similarity function modeling the neuron's computation (Sf), the number of responsive neurons in the hidden layer (Rn) representing the number of terms used

to compute (2), the similarity threshold (St) determining the hidden layer composition, the maximun lag depth (Ld), and the relative percentage of the training set vs test set (Rp) when learning the prediction function for a given time series dependency model. In all experiments St was kept fixed and equal to 1.

The process of model search is performed by the genetic algorithm. Binary chromosomes coding model components as given by (1) were used with single and double point crossover operators and standard bit-reversal mutation. Selection was kept constant (roulette wheel method) and complete population replacement with elitism were used. In *(ii)* the influence of the number of generations (Ng), and the crossover operator were investigated. Population size P_s controls the richness of the "genetic pool" used in the evolutionary process. In these experiments it was fixed at 50. As for *(iii)* the number of physical nodes was fixed at three (dual CPUs), and the number of slaves fixed at 15 in all runs.

III. 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 [4], 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 for a model mining study. No preprocessing was applied to the time series. This is not the usual way to analyze time series data, but by eliminating additional effects, the properties of the proposed procedure, in terms of approximation capacity and robustness, are better exposed.

With the purpose of investigating the behavior of the hybrid heterogeneous network, three new sets of time series were constructed by introducing 25%, 50% and 75% of uniformly distributed missing values into all 10 original series. The introduction of the missing values was done 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 precentage 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. These training set variants were used by the evolutionary algorithm to explore the model space. The reported error measure is, in all cases, the *root mean squared error* (RMS error) computed by applying the trained networks to the test set.

The similarity functions were constructed from versions of the Euclidean, Clark and Canberra distance functions [2], and account for 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 $y_i \neq \texttt{X}$. Then the corresponding distance functions are given in Table-I. Note that they are normalized distances and therefore, are independent of the number of attributes, and that *no imputation* of missing values to the data set is performed.

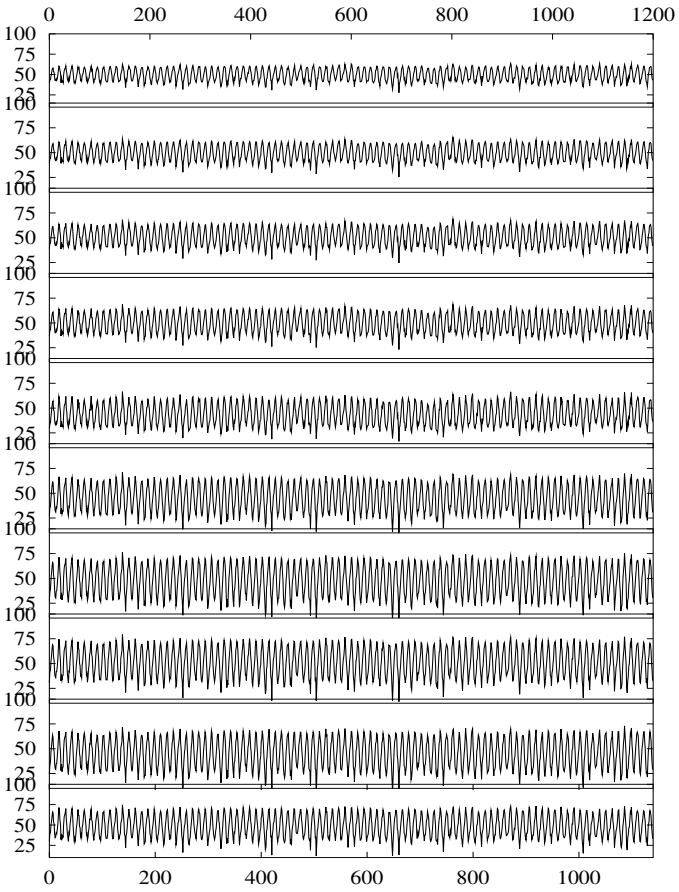

Fig. 3. Temperature data from 10 Washington State sites (Farenheit). See text for details.

Name	Distance
Euclidean	$\frac{\sum_{A_c}(x_i-y_i)^2}{card(A_c)}$
Clark	$\frac{\sum_{A_c}\frac{(x_i-y_i)^2}{(x_i+y_i)^2}}{card(A_c)}$
Canberra	$\frac{\sum_{A_c}\frac{\lvert x_i-y_i\rvert}{(x_i+y_i)}}{card(A_c)}$

TABLE I
EUCLIDEAN, CLARK, AND CANBERRA MODIFIED DISTANCES

A total of 288 experiments were made varying the two classes of controlling factors and their corresponding parameters. In fact, 72 experiments were performed for each of the 4 missing value data set variants ($[0\% - 75\%]$). See Table-II.

The experiments were conducted on a Beowulf cluster consisting of three dual Xeon processor units operating at 2 Ghz frequency, each with 1Gb RAM. The cluster operates with 100 Mbit Ethernet connections. The operating system is Red Hat Linux 7.2 running LAM-MPI version 6.5.4/MPI 2, C++/ROMIO.

Factors	Parameter	Values
(i)	Sf	$(1/(1+d))$
	Rn	$< 1, 3, 5, 7, 13, 20 >$
	Ld	$< 5, 10, 20, 30, 50 >$
	Rp	$< 50\% >$
(ii)	Ng	$< 2, 10, 100 >$
	Ps	$< 50 >$
	Cp	$< 0.6 >$
	Mp	$< 0.01 >$
	Ct	$< single, double >$

TABLE II

EXPERIMENTAL PARAMETERS. IN *(i)* d STANDS FOR EUCLIDEAN, CLARK, OR CANBERRA NORMALIZED DISTANCES.

IV. RESULTS

The distribution of the RMS error for all data sets ($[0\% - 75\%]$ of missing data) is shown in Fig-4. All distributions are highly skewed towards the lower end of the RMS error measure. For the present analysis the range corresponding to the best error was considered to be the interval $[2.167 - 2.3]$. Due to the skewness of the RMS error distribution, the selected range comprises 75% of all the models found with 0% missing values. It is interesting that the series with 75% missing values still have more than 25% of their models with error in this lower end. The Q1-Q3 interquartile range (between the first and third quartiles) gets broader as would be expected since the information content in the series decreases. However it does it very slowly up until 75%, when it abruptly increases. Remarkably, the absolute minimum errors are almost constant, even considering the extreme 75% case. Thus, the algorithm exhibits a very robust behavior and a capacity to retrieve good models in this data set, even though it contains scarce information.

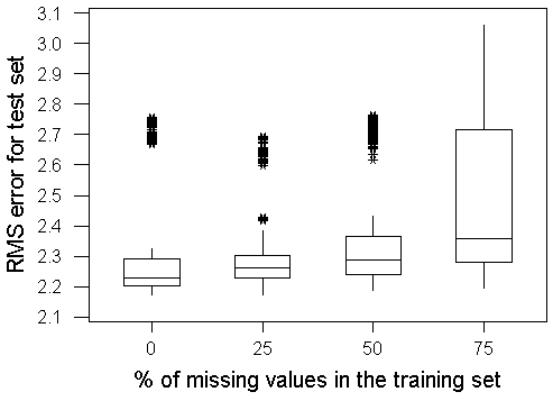

Fig. 4. Boxplots showing the main distribution parameters of the RMS Error for the different percentages of missing values for all experiments. Stars represent outlying elements in the tail of the distribution.

A. Influence of neuro-fuzzy network parameters

The neuro-fuzzy network relies on the responses of the heterogeneous neurons in the hidden layer which happen to be similarity-based units. Similarity functions are known to be sensitive to data structure, which in turn, is influenced by data dilution. The relations between the combination of percentage of missing values and the different similarity functions are shown in Fig-5 as boxplots of their corresponding error distributions.

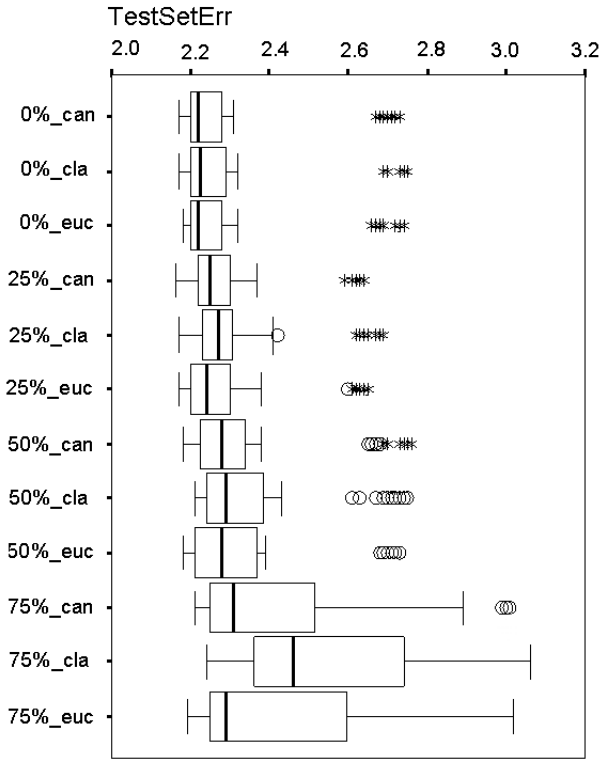

Fig. 5. Boxplots showing the main distribution parameters of the RMS Error for the different percentages of missing values and similarity measures for all experiments. Stars represent outlying elements in the tail of the distribution.

The different similarity measures don't appear to exert a large influence on the RMS error for most of the missing value variants, with the exception of 75%. Overall, errors for all cases except this last one, are within a relatively narrow band at the lower end. In the range $[0\% - 50\%]$ there are no within variant pairwise differences between similarities. In the 75% case the similarity function based on Clark 's distance clearly under performs and actually is responsible for the increase of the interquartile range in the overall (see Fig-4). It is also interesting to observe that with the exception of this last case, models with good prediction errors can be found with any choice of distance measure for all missing data set variants. Within the set of selected similarities, none of them performs significantly better than the others. In the context of this kind of data, the similarity function was not influential w.r.t the quality of the models discovered.

The number of responsive neurons (R_n) is an important parameter controlling the neuro-fuzzy network output. It determines the number of terms used in computing the fuzzy interpolation (2). The dependency of the RMS error with the

number of responsive neurons (R_n) and the percentage of missing values is shown in Fig-6. In the specific case when $R_n = 1$ (representing the "winner take all" strategy), errors are systematically several times higher than those obtained from other larger choices of R_n, for all missing value variants. In general, in order to achieve a good fuzzy estimate for the predicted output, comparatively few terms (small R_n) are required in (2).

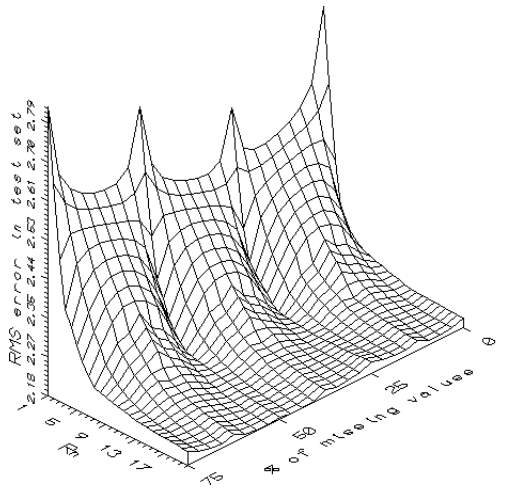

Fig. 6. Dependency of the mean RMS errors with the percentage of missing values and the number of responsive neurons (R_n).

The maximum lag (L_d) parameter defines the size of the search space of models by imposing an upper bound in the number of lag terms available for model construction. A value too small may preclude the discovery of good models if the memory of the process is large, whereas a value too large will increase unnecessarily the search space (it does it exponentially), thus decreasing the chances of discovery by "diluting" the good models. Moreover, it introduces noise in the search process. The behavior of L_d for the cases of $[0\% - 75\%]$ missing values is shown in Fig-7

Clearly, when information is complete (0%), maximum lag depth has no influence on the error level. Therefore, small L_d values lead to compact, short-memory models. When data is severely affected by missing information larger L_d values (medium-memory models) are necessary in order to obtain comparable error levels. As missing values increases, the required lag depth increases and it does it linearly with a highly significant correlation coefficient (0.977). Moreover, the model mining procedure proved that looking deeper into the past of the process, does not necessarily improve the explanation of the target series, as shown by the large plateau in the investigated ranges of missing values and lag depths.

B. Influence of genetic algorithm parameters

Most of the parameters controlling the behavior of the genetic algorithm responsible for the evolutionary process of model discovery were kept fixed. However, a few experiments

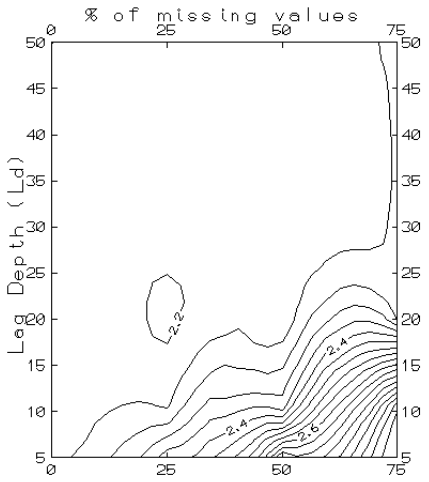

Fig. 7. Dependency of the RMS Error with the percent of missing values and the Lag Depth (L_d).

varying the number of generations (N_g) and crossover type (C_t) were performed. The behavior of the number of generations of the genetic algorithm (N_g) is shown in Table-III.

N_g	0% miss	25% miss	50% miss	75% miss
2	2.3013	2.2987	2.3020	2.3870
10	2.2842	2.2832	2.2849	2.2875
100	2.2374	2.2146	2.2296	2.2510

TABLE III
DISTRIBUTION OF $\overline{RMSerror}$ W.R.T. THE NUMBER OF GENERATIONS.

Clearly, the experiments show that it is enough to let the system evolve a medium-to-small number of generations in order to discover accurate models. Since a small number of generations are required, appropriate models can be found quickly. For a fixed N_g, doesn't affect the average error substantially, another indication of the robustness of the neuro-fuzzy network.

Single and double point crossover operators were used and their relation with the percentage of missing values and the $RMSerror$ is shown in the boxplots of Fig-8.

The interquartile ranges up until 50% missing value are comparable. However, there seems to be a slight advantage for choosing double point over single point. This is demonstrated most clearly in the 75% case. Again, the overall minimum for each of the combinations is comparable. That is, model quality is not affected by the particular crossover operator.

C. Prediction Example

The performance of the overall best model found for the test set is shown in Fig-9. In the model, all 10 signals are contributing with different lag terms to the prediction of the target series. This best selected model is not significantly different from the top 20 models in terms of RMS error and further 65% of these models contained complete information. The other 35% contain 25% missing values.

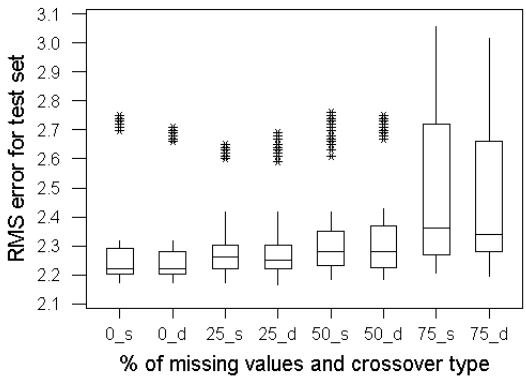

Fig. 8. Dependency of the RMS Error with the percent of missing values and the Crossover Type (C_t).

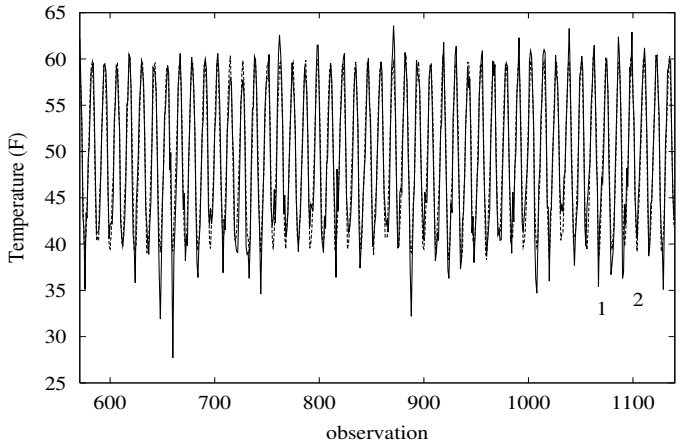

Fig. 9. Behavior of the overall best model found for the target series (West Olympic Coastal) in the test set. RMS error = 2.167 (1: observed, 2: predicted).

V. CONCLUSIONS

The results obtained show that the method studied here is robust, effective, and able to discover accurate models in a consistent way. Taking into account the astronomical size of the search space for the experiments performed, it is remarkable that the exploration of the extremely small fraction covered by the chosen parameters lead to extremely simple and very accurate models, even with 75% missing data. These features make this method appropriate for the study of poorly known or unknown processes for data of this kind. The experiments proved that there is an optimal combination of neuro-fuzzy and genetic algorithm parameters which maximizes the chances of discovering good models. Moreover, it is clear that the construction of a fuzzy estimate of the predicted signal based on more than one responsive neuron in the heterogeneous layer is decisive in obtaining good predictions. Clearly, these results are conditioned to multivariate data of the kind used in this paper and no claims are made outside of this context. Further studies must be carried out with multivariate series data coming from different processes in order to study the properties of the model mining technique proposed and determine the conditions of its optimal use.

ACKNOWLEDGMENT

The authors would like to thank Robyn Paul from the University of Waterloo for her assistance during the final stages of this research.

REFERENCES

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

[2] Chandon, J.L., Pinson, S. : *Analyse Typologique. Thorie et Applications*. Masson, Paris, (1981)

[3] : *MPI Primer/ Developing with LAM*. Ohio Supercomputer Center. The Ohio State University, (1996)

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

[5] 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), 237–246

[6] Valdés, J.J. : Time Series Models Discovery with Similarity-Based Neuro-Fuzzy Networks and Evolutionary Algorithms. *IEEE World Conference on Computational Intelligence* WCCI'2002, Hawaii, USA, 2002.

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

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