Hybrid Approach for Time Series Prediction using Neural Networks and Simulated Annealing

Adrian L. Arnaud
UFPE - Recife, Brasil
ala2@cin.ufpe.br

Paulo J. L. Adeodato
UFPE - Recife,Brasil
pjla@cin.ufpe.br

Germano C. Vasconcelos
UFPE - Recife,Brasil
gvc@cin.ufpe.br

Rosalvo F. O. Neto
UFPE -Recife,Brasil
rsv@cin.ufpe.br

Abstract

This paper proposes a new hybrid approach which combines simulated annealing and standard backpropagation for optimizing Multi Layer Perceptron Neural Networks (MLP) for time series prediction. Experimental tests were carried out on four simulated series with known features and on the Sunspot series. The results have shown that this approach selects the appropriate time series lags and builds an MLP with the minimum number of hidden neurons required for achieving good performance on the task. The performance attained was better than some results recently reported for hybrid systems combining Genetic Algorithms (GA) and MLPs for the same purpose presented here.

1. Introdução

In the last decade, there has been intense research on the application of neural networks for the problem of time series prediction. Part of this interest is due to some features of the Multi Layer Perceptrons not found altogether in the techniques traditionally used for this purpose. Neural networks are universal function approximators [1], non-parametric systems capable of mapping complex non-linear relations among input and output data, and achieve excellent generalization capacity.

Nevertheless, despite its ability to learn without making any assumption on the data distributions, the performance of a Multi Layer Perceptron depends, among other factors, on the initial weights setting, on the training algorithm and on the network topology. Particularly in time series prediction problems, the selection of the relevant lags which define the network input space and the number of neurons in the hidden layers are fundamental matters in determining the generalizing power of the MLP.

Berardi and Zhang [2] have experimentally shown that the selection of the input lags and of the number of hidden neurons affect differently the bias and the variance of the MLP models applied to time series data. In one hand, exceeding the appropriate number of lags in the input selection worsens the variance of the model while exceeding the appropriate number of hidden neurons worsens the model's bias. On the other hand, selecting less lags and/or hidden neurons affects even more severely the model's bias, thus strongly degrading the network performance.

The above constraints imposed on the selection of the network topology produce a neural network architecture optimization problem which, usually, can not be efficiently tackled simply by the trial and error approach. For this type of problem, global search approaches such as simulated annealing (SA) [3] are more suitable. Simulated annealing is capable of systematically finding optimal or suboptimal solutions in complex search spaces through the generation of candidate solutions, a set of operators for searching the solution space and an adequate cost function for evaluating each candidate solution produced.

In this paper, we propose a new hybrid approach for global optimization of neural networks' topology and weight adjustment using alternately SA for the former and standard backpropagation for the latter, until convergence of the training process. This method comprises a series of cycles where SA produces candidate network topologies by activating or deactivating neurons with all their associated connections and the standard backpropagation algorithm gradually adjusts, during a short predefined number of epochs, the connections' weights. These cycles continue until the optimal topology for the architecture and the optimal weight adjustment for the connections is found. This optimization is particularly suited for the application of MLP networks to the time series prediction problem since the lag detection becomes an automatic search inherent to the training process.

The remainder of this paper is divided in three sections. Section-2 describes the proposed hybrid method. Section-3 presents the experimental results on four simulated series and on the sunspot series with comments on the performance. Finally, Section-4 contains the final remarks stating the main advances achieved and the limitations of the approach to be tackled by further research.

2. Proposed method

The simulated annealing algorithm (SA) has been inspired by the physical annealing process and by Boltzmann [4] equation. It has been proposed by

Kirkpatrick *et al.* [3] as an algorithm for solving combinatorial optimization problems. Besides ease of implementation, this algorithm presents the attractive feature of finding global optimal solutions due to its flexibility to investigate suboptimal solutions along the method execution. Opposed to SA, the backpropagation algorithm based on the Delta Rule [5] is a local search algorithm, thus not capable of finding global optimal solutions. The heuristics of gradient descent used in the backpropagation algorithm partly compensates in the target function optimization, particularly, for neural networks' connection weights adjustment.

The new optimization method proposed in this paper combines the ability of SA for finding global solutions with the ability of backpropagation for finely adjusting the MLP connection weights. The method, coined here SANNO (*Simulated Annealing to Neural Networks Optimization*), alternates the use of SA for topology optimization and standard backpropagation for weight adjustment, until the training process converges. This method comprises a series of cycles where SA produces candidate network topologies by activating or deactivating neurons with all their associated connections and the standard backpropagation algorithm gradually adjusts, during a short predefined number of epochs, the connections' weights. The following subsections present the method details.

2.1. Solutions' representation

According to SANNO method, each point in the solution space represents an MLP network with a single hidden layer and a single output neuron. All MLPs in the solution space are feedforward networks only with connections between adjacent layers. All hidden and output neurons have a bias parameter.

Both the input and the hidden neurons have an associated Boolean variable indicating the neuron's current activation state: TRUE for actively participating in the network topology and FALSE for inactive or out of the topology. When a neuron is inactive, all its input and output connections are disconnected. For instance, if a hidden layer neuron is inactive all its connections from the input layer and to the output layer are disconnected.

The network topologies are only valid if they comply with all the following criteria: (a) have at least 1 active connection between the output layer and a neuron of the hidden layer N_h and (b) have at least 1 active connection between the same neuron of the hidden layer N_h and a neuron of the input layer.

The maximum neural network topology (initial) is always an MLP with a single hidden layer and a single output neuron. In this paper, the experiments were carried out with MLPs with up to 10 input neurons and 10 hidden neurons.

2.2. Generation of candidate solutions

The specification of the operators for candidate solution generation is essential for the methods' success. Differently from Yamazaki and Ludemir's approach [6] and from some known pruning algorithms [7][8][9], the proposed method does not operate at the connection level for MLP topology optimization; it works at the neuron level, instead. This apparently simple change drastically reduces the size of the solution search space. For instance, an MLP with 10 input and 10 hidden neurons has 121 connections, including the bias. Topology optimization at the connection level, in one hand, represents a search space of 2^{121} possible solutions (all the states of 121 binary variables: active or inactive). Optimization at the neuron level, on the other hand, represents a search space of only 2^{20} possible solutions, as there are only 10 input and 10 hidden neurons being tackled in the optimization problem. This search space reduction allows for the exploration of bigger network topologies which may be necessary for modeling more complex time series.

Furthermore, the optimization at the neuron level is sufficient for capturing the fundamental aspects of neural networks topology for time series prediction: the time lags in the input layer and the function approximation complexity in the hidden layer.

Executing the SANNO algorithm, at each iteration of the SA algorithm, a new candidate solution is produced by changing the neurons' activation states of a previous solution. The generation follows the steps below: (1) a neuron of either the input or the hidden layer form the previous network is randomly selected; (2) that neuron's activation state is changed according to a previously defined state reversal probability p (p<1). The execution of these 2 steps generates a neighbor topology with probability p or preserves the same topology with probability 1-p. In the experiments carried out here p=0.5 was the value used. After a new network solution has been generated, its validity is checked according to the criteria defined above. If the generated solution is not valid, the generation step is repeated until a valid solution is produced.

The valid network is then submitted to a pre-specified number λ of training epochs with backpropagation. After training, with all its connection weights adjusted, the network is evaluated according to the adopted cost function. If the network cost decreases compared to the previous, the solution is promptly accepted. Otherwise, it can still be accepted with a probability $P = exp(-\Delta E/T)$, where ΔE is the increase in cost and T is the current temperature of the SA algorithm. This is the Metropolis criterion [4]. It states that the probability of acceptance of a higher cost solution decrease with its cost increase and with the temperature decrease. In the experiments carried out here λ = 2 backpropagation training epochs for the incremental weight adjustment before a new SA iteration.

2.3. Update of the deactivated connection weights

During the experiments, we observed that neurons which had been deactivated in the early stages of the training process had difficulties in being reactivated in later iterations. We detected that the weight vector norm of these neurons' input connections was frequently one order of magnitude smaller than the norms of the input weight vectors of the active neurons. This is a direct consequence of the use of backpropagation for weight training: the norm of the active connections, in general, increases at each training epoch while the inactive neurons connections remain the same.

For minimizing the drawbacks of this difference between the norms of the weights of active and inactive neurons, the weights of the inactive connections started being updated according to the following rule: w_{ij} (t+1) = w_{ij} (t) + $\eta\delta_{kj}o_{kj}$, where w_{ij} (t) represents the connection weight between neuron i and neuron j in time t, η is the learning rate of the backpropagation algorithm, δ_{pj} is the error term for the pattern k in neuron j and o_{kj} is the output of neuron j for pattern k. This is the same as the original weight adjustment rule for the standard backpropagation [5]. However for deactivated neurons, the output o_{kj} is constant and equal to 0.5 corresponding to the response given by a sigmoid to the input of x = 0 (or $\Sigma w_{ij}a_k = 0$, since all the weights are deactivated). Thus, the connection weights of the deactivated neurons are updated by the following equation: w_{ij} (t + 1) = w_{ij} (t) + $0.5\eta\delta_{kj}$

This approach has efficiently solved the problem of the difference in magnitude between the norms of the weights of active and deactivated neurons. Experiments have shown that this procedure helped reactivating inactive neurons even at late stages of the training.

2.4. Cost function

At the end of each cycle, the candidate solution is evaluated based on a cost function proportional to the Mean Squared Error (MSE), calculated on the training set and proportional to the number of active neurons in the network topology. The cost function is given by the following equation:

$$\cos t = \frac{\frac{MSE_{current}}{MSE_{initial}} W_{MSE} + \frac{\beta_{current}}{\beta_{initial}} W_{\beta}}{W_{MSE} + W_{\beta}} \qquad 1$$

where $MSE_{current}$ is the current MSE error for the training set patterns, $MSE_{initial}$ is the initial MSE calculated at the first iteration, $\beta_{current}$ is the number of neurons currently active, $\beta_{initial}$ is the number of neurons active at the first interaction, W_{MSE} is the weight previously defined for the relation between MSE errors and W_β is the weight also previously defined for the relation between the number of active neurons.

This cost function drives the search for solutions towards neural networks with optimal topology and optimal performance. It is possible, however, to control the network topology and performance through the parameter setting of the weights W_β and $W_{MSE.}$

2.5. Temperature control

The SA algorithm performance is directly related to the temperature reduction scheme. In this paper, we have used the geometric cooling rate. According to this rule, the temperature is modified by the equation below: T(t + 1) = αT (t), where T(t) is the temperature value at time t and α is temperature factor, a constant less than 1 and usually close to 1 [10]. In the experiments carried out here, we have used α = 0,9. The temperature was reduced at each 30 cycles and always initially equal to 1.

2.6. Stopping criteria

The algorithm execution is interrupted either if the maximum number of cycles is reached or if any substantial generalization loss is detected over an independent validation set. This loss is measured by the generalization (GL_5) loss criterion described in *Proben1* [11]. More specifically:

$$GL_5 = 100.\left(\frac{MSE_{current}}{MSE_{\min}} - 1\right) \qquad 2$$

where $MSE_{current}$ is the current MSE error for the validation set and MSE_{min} is the minimum MSE error achieved so far during training. Training stops when $GL_5 \geq 5\%$. The GL_5 value is updated at every 30 cycles.

3. Experimental results

Five time series were used for testing the proposed method focusing on its power to optimize the MLP topology and to adjust its connection weights and on its performance for time series prediction. Four of these series are artificially produced by known generating functions: two of them reproduce AR(p) models and the others, ARMA(p,q) models. The fifth series is the annual sunspot series.

The latter series has been chosen because of its unique features which suggest a non-linear, non-Gaussian and non-stationary process. Due to these particular features, the annual sunspot time series has been widely used as reference for comparison and evaluation of several works in the field [12].

In all of the experiments, the values of the series were normalized in the interval [0,1], before being used for training the MLP networks. All the series have also been divided in three sub series: 80% for optimization and training, 10% for validation and 10% for testing the models. The parameters for SANNO algorithm were defined for each series also after a previous test section.

3.1. Simulated series

The following auto-regressive models were considered in the study of simulated time series: AR(3), AR(4), ARMA(3,1) and ARMA(4,4). These models are given by:

$$\begin{array}{ll}
\text{AR(3)} & y_t = 0.3y_{t-1} + 0.6y_{t-3} + \varepsilon_t \\
\text{AR(4)} & y_t = 0.3y_{t-1} + 0.6y_{t-3} - 0.5y_{t-4} + \varepsilon_t \\
\text{ARMA(3,1)} & y_t = 0.3y_{t-1} + 0.6y_{t-3} + 0.5\varepsilon_{t-1} + \varepsilon_t \\
\text{ARMA(4,4)} & y_t = 1 - 1.4(y_{t-2} - \varepsilon_{t-2})^2 + 0.3(y_{t-4} - \varepsilon_{t-4}) + \varepsilon_t
\end{array} \quad 3$$

where y_t is the current value for the series and y_{t-k} are the past observations with lag-k. ε_t represents the uniformly distributed noise in the interval [-0.01, 0.01] and ε_{t-k} are temporal dependences on the current noise with a k time lag.

For all the simulated series, 400 values were generated from which the first 200 were discarded for preventing any initial effect caused by the artificial generation. The remaining 200 values were used for training, validation and testing of the forecasting systems.

The ARMA (4,4) model, defined above, is known in the literature as Hénon series. As well as the sunspot series, Hénon series is very popular and has been used in several works for investigating predictive models due to its complexity and chaotic dynamics.

For the AR(3), AR(4) and ARMA(3,1) models, SANNO algorithm started with a topology of 5 input and 5 hidden neurons. The connection weights were initially set at random values uniformly distributed in the interval $[-10^{-4}, 10^{-4}]$.

The probability of neuron activation reversal was set to $p = 0.5$. At each cycle, the topology currently selected by the SANNO system was trained with backpropagation for 2 epochs with a learning rate of $\eta = 0.01$ and a momentum of 0.8. The SA algorithm initial temperature was set equal to 1 and was gradually reduced at each 30 cycles by a temperature factor of $\alpha = 0.9$.

For the Hénon series, the system used the same initial topology and same parameters except for the weights of the cost functions. While the other auto-regressive models used $W_{MSE} = W_\beta = 1$, Hénon series used $W_{MSE} = 5$ and $W_\beta = 1$.

For all the simulated series, the topology optimization and the weight training were interrupted when $GL_5 \geq 5\%$ or when 1000 training cycles was reached.

The training process for all the four simulated series was repeated 30 times. The best 10 and the worst 10 networks in terms of MSE on the validation set were discarded. The remaining 10 results were used for the final results presented here. Table-1 presents the average final topologies for all auto-regressive models after optimization with the proposed method.

Table 1. MLP topologies selected by the SANNO method on the four artificial auto-regressive models.

Model	Lags	Hidden Neurons		MSE (test)	
		Mean	S	Mean	S
AR(3)	1, 3	1	0	0,0243	1,7E-05
AR(4)	1, 3, 4	1	0	0,0193	8,3E-04
ARMA(3,1)	1, 3	1	0	0,0092	3,6E-05
ARMA(4,4)	2, 4	2,3	0,48	0,0028	1,4E-04

The results presented on Table-1 show that the proposed method is able to efficiently select all relevant time lags for each time series, independently of its complexity. As expected, none of the fundamental lags belonging to the series generating process was out of the selection. Furthermore, no extra lags were selected either. This capability of only selecting the relevant lags complies with the topology constraints reported by Berardi and Zhang [2].

The amount of hidden neurons selected by the proposed method is coherent to the complexity of the specified tasks. The topologies with a single hidden neuron represent the minimum complexity possible because of the imposed constraint of having at least 1 neuron in the hidden layer for the MLP to exist. In fact, apart from the effects of noise, the AR(3), AR(4) and ARMA(3,1) present only linear dependencies on past observations and could possibly be tackled by a simple perceptron.

After the SANNO method found the optimized topologies for each task, MLP networks with such topologies were trained by the standard backpropagation algorithm from the start. Their performance was then compared to that achieved with SANNO method.

While the SANNO method had to optimize network topology and connection weights, the MLP with backpropagation had only to optimize the connection weights for the optimal topology.

For having balanced conditions for both approaches in the comparison, the MLP networks trained solely by the backpropagation algorithm were subjected to the same weight initialization methodology, were trained with the same learning rate and same momentum term as they had been trained with the SANNO method. The only difference was in the maximum allowed number of training epochs which was doubled to 2000 for the SANNO method runs 2 epochs for each of the 1000 training cycles.

For three of the tasks listed SANNO approach found a unique network topology for the problem. For the ARMA(4,4) task there was a small variation on the topology found by SANNO. Therefore the corresponding optimal MLP network was that with only two hidden neurons (the closest to the average 2.3).

Table 2. MSE on the test set for the SANNO method and for MLP networks with optimal topology trained with standard backpropagation (BP).

Model	Using SANNO		Only using BP	
	Mean	S	Mean	S
AR(3)	0,0243	1,7E-05	0,0247	3,3E-08
AR(4)	0,0193	8,3E-04	0,0196	4,9E-08
ARMA(3,1)	0,0092 *	3,6E-05	0,0094	1,4E-08
ARMA(4,4)	0,0028 *	1,4E-04	0,0569	2,2E-06

The t-test was applied to each of the ten pairs of measurements in all four tasks for detecting statistical significance at 95% confidence level in the comparison of both approaches. Table-2 shows the MSE for both approaches applied to the independent test sets (SANNO vs. using only backpropagation on optimal topology MLP). The values with statistically significant difference are marked with an asterisk (*).

From table-2, there is no difference between the approaches for the models AR(3) and AR(4) (linear models). For the ARMA(3,1) and ARMA(4,4) models, the SANNO algorithm performed significantly better than the MLP with backpropagation, even considering that the latter had started training from an already optimal topology.

In the case of the ARMA(4,4) model, the results of SANNO are one (1) order of magnitude better than that of optimal topology MLP trained with backpropagation. This more complex task brings global optimization techniques to the fore which might more easily find local minima deeper than those found by local optimization techniques.

These results show that the proposed method is capable of optimizing both the neural network topology and its connection weights yet preserving or even improving its performance measured by the MSE.

3.2. Sunspot series

In 1849, Wolf [12] introduced the sunspot number as a yearly index for measuring the sun's activity. Every year new readings of data are gathered by the Swiss Federal Observatory and published on the Journal of Geophysical Research. In this paper, the data set consisted of the annual measurements observed between 1700 and 1988 (289 observations).

For this data series, the initial MLP topology had 10 input and 10 hidden neurons, an amount slightly greater than typically needed for this task. The training rate was $\eta = 0.005$ and the ratio of costs was $W_\beta / W_{MSE} = 1/100$. All the other training parameters had the same values as in the simulated series above.

The proposed training process has been executed 30 times resulting thus, in 30 networks with their respective performance indicators. The 10 best and 10 worst performance results have been discarded. In all cases, training stopped with $GL_5 \geq 5\%$, which happened in average after 2000 cycles of training totaling around 4000 backpropagation epochs (2000x2).

Despite the excess of neurons in the initial network topology for this task, the SANNO approach converged to a topology which detected the same input lags for all the 10 trials namely lags 1, 2, 3, 9 and 10. In the hidden layer, the network topologies ended up with an average of 7.7 and a standard deviation of 1.16 neurons.

Both results agree with those reported on the literature. Weigend *et al.*[7], after network pruning, found strong connections corresponding to lags 1, 2 and 9. Later, Pi and Peterson's [13] δ-test, based on conditional probabilities, have selected the lags 1, 2, 3, 4, 9 and 10 as the most relevant for this series prediction. Pi and Peterson have also shown that an MLP network with only those input lags and 8 hidden neurons is sufficient for high level prediction performance on this time series.

In term of prediction performance, the SANNO method has also produced good results. The MSE (average error) obtained for the independent test set was 0.0087 (with standard deviation = 4.25E–08). This result is substantially superior to other results recently reported on the use of GAs for time series prediction. Leung *et al.* [14] used a modified GA for training MLP networks and their solution attained an MSE error equivalent to 0.061 for the test set. Terui and Dijk [15] employed a method which combined the AR, TAR and ExpAR models and obtained an MSE error equivalent to 0.039 for the test set. More recently, Ferreira *et al.* [16] used a hybrid system based on F. Takens theorem and obtained an MSE error of 0.016 for the test set.

4. Final remarks

This paper has presented a new hybrid method (SANNO) which uses simulated annealing for topology optimization and backpropagation for connection weight training in neural networks for time series prediction. The approach has been tested on 4 simulated time series and on the sunspot series

The experimental results on the simulated series (AR(*p*) and ARMA(*p*,*q*) models) and on the annual sunspot series have shown that this approach generates near optimal MLP network topologies that both identify the time lags for the input layer and provide the required function approximation complexity in the hidden layer. For the simulated series, in all cases, the network model has selected exactly the time lags specified on the series generating functions. For the sunspot series, the model has selected time lags that agree, to a large extent, with those reported on the literature, particularly with Pi and Peterson's [13] results with the non-parametric δ-test.

In terms of performance, the proposed approach is also outstanding. For the simulated time series, SANNO has been compared to the MLP network topologically optimized for each of the four tasks and trained with standard backpropagation. For the two AR series there was no significant difference in performance. For the two ARMA series, the SANNO approach has achieved significantly better (t-test at 95% confidence). For the sunspot series, SANNO has been compared to several recent hybrid MLP approaches and has performed far better than what they have reported [14] [15] [16].

Despite all these promising results, the SANNO approach has just been proposed and there is a lot yet to be investigated for consolidating this method. As seen in this paper, performance does not seem to be a problem but the system's performance sensibility to the training parameters and the training algorithm need to be evaluated. Further research also involves the extension of SANNO method to recurrent neural networks such as Elman and Jordan networks aiming at improving the system's performance for times series prediction.

References

[1] K. Hornik, M. Stinchcombe, and H. White. "Multilayer Feedforward Networks are Universal Approximators", *Neural Networks*, vol. 2, 1989, pp. 359-366..

[2] V.L. Berardi, and G.P. Zhang. "An Empirical Investigation of Bias and Variance in Time Series Forecasting: Modeling Considerations and Error Evaluation", *IEEE Transactions on Neural Networks*, vol. 14, no. 3, 2003, pp. 668-679.

[3] S. Kirkpatrick, C.D. Gellat Jr., and M.P. Vecchi. "Optimization by Simulated Annealing", *Science*, vol. 220, no. 4598, 1983, pp. 671-680.

[4] N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, and E. Teller, "Equation of state calculations by fast computing machines", *Journal of Chemical Physics*, vol. 21, no. 6, 1953, pp. 1087-1092.

[5] D.E. Rumelhart, G.E. Hinton, and R.J. Williams, "Learning internal representations by error propagation", *Parallel Distributed Processing*, Cambridge, MIT Press, vol. 1, 1986, pp. 318-362.

[6] A. Yamazaki and T. B. Ludermir, "Neural Network Training with Global Optimization Techniques", *International Journal of Neural Systems*, vol. 13, no. 2, 2003, pp. 77-86.

[7] A.S. Weigend, B.A. Huberman, and D.E. Rumerlhart, "Predicting the future: A connectionist approach". *International Journal On Neural Systems*, vol. 1, no.3, 1990, pp. 193-209.

[8] K.L. Ho, Y.Y. Hsu, and C.C. Yang, "Short-term load forecasting using a multilayer neural network with an adaptative learning algoritm", *IEEE Transactions Power Systems*, vol. 7, no. 1, 1992, pp. 141-149.

[9] M. Cottrel, B. Girard, Y. Girard, M. Mangeas, and C. Muller, "Neural Modeling for Time Series: A Statistical Stepwise Method for Weight Elimination", *IEEE Transactions on Neural Networks*, vol. 6, no.6, 1995, pp. 1355-1364.

[10] Pham, D.T., and D. Karaboga, *Intelligent Optimization Techniques*, Springer Verlag, 2000.

[11] L. Prechelt. "Proben 1- a set of neural network benchmark problems and benchmarking rules", *Technical Report 21/94*, Kakultat fur Informatik, universitat Karlsruhe, Germany, 1994.

[12] A.J. Izenman, "J.R. Wolf and the Zürich Sunspot Relative Numbers", *The Mathematical Intelligencer*, no.1, 1985, pp. 27-33.

[13] H. Pi, and C. Peterson, "Finding the Embedding Dimension and Variable Dependences in Time Series", *Neural Computation*, vol. 6, 1994, pp. 509-520.

[14] F.H.F. Leung, H.K. Lam, S.H. Ling, and P.K.S. Tam, "Tuning of the Structure and Parameters of a Neural Network using an Improved Genetic Algorithm". *IEEE Transactions On Neural Networks*, vol. 14, no. 1, 2003, pp. 79-87.

[15] N. Terui, and H.K. Van Dijk, "Combined forecasts form Linear and Nonlinear Time Series Models", *International Journal of Forecasting*, vol. 18, 2002, pp. 421-438.

[16] T. Ferreira, G. Vasconcelos, and P. Adeodato, "A Hybrid Intelligent System Approach for Improving the Prediction of Real World Time Series", *Congress on Evolutionary Computation*, Portland, Oregon, 2004.