We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

4,800
Open access books available

122,000
International authors and editors

135M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

CLARIVATE ANALYTICS BOOK CITATION INDEX INDEXED

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

Forecasting Chaotic and Non-Linear Time Series with Artificial Intelligence and Statistical Measures

Aranildo Rodrigues L. J.[1], Paulo S. G. de Mattos Neto[2],
Jones Albuquerque[1], Silvana Bocanegra[1] and Tiago A. E. Ferreira[1]
[1]*Statistics and Informatics Department, Federal Rural University of Pernambuco*
Recife - Pernambuco
[2]*Center for Informatics, Federal University of Pernambuco*
Recife - Pernambuco
Brazil

1. Introduction

The most of the classical time series literature suppose that the series are stationary (or that the time series can be transformed in stationary series through some simple transformation, such as differentiation), and that the series phenomenon is a linear process. In this sense, all time series can be represented by linear models.

However, time series encountered in practice way not always exhibit characteristics of a linear process, then there is not any reason that generalizes the linearity supposition for a real world time series. In fact, the high complexity of the real world phenomena induce to think, more naturally, about non-linear and chaotic structures presents in the data of time series than linear structures.

Loosely speaking, a time series is a set of observations made sequentially in time. Examples of real world time series abound in such fields as economics, business, engineering, natural sciences (commonly in the meteorology, geophysics and biology), social sciences, etc (Lam, 1998). Phenomena like human breath rate, human electrocardiogram, earthquake, stock prizes are some examples of real world time series. A typical intrinsic feature of a time series is that the adjacent observations are dependent, where the nature of this dependence among time series observations is of considerable practical interest. The *time series analysis and forecasting* is concerned in mathematical and statistical (and more recently, computational) modelling for analysis of this dependence.

Mathematical and statistical methods are successfully used for time series analysis and forecasting (Box et al., 1994; Gooijer and Kumar, 1992; Kantz, 2004), but sometimes these approaches are not trivial to apply in practical sense, considering that some times series (mainly real world time series, as the financial or economical series, climatic series, etc) have a chaotic and non-linear behavior and many types of components, such as trends, seasonality, impulses, steps, model exchange and other uncontrolled features.

Alternatively, in the last two decades, the Artificial Neural Network (ANN) model have been widely used in order to solve the time series forecasting problem, presenting less mathematical complexity than the typical non-linear statistical methods. However, the ANN

approach has a critical point, the correct adjust of its parameters, since this adjustment is dependent of the problem. To solve this problem of adjustment, many Intelligent Hybrid Systems have been proposed to model a time series, where an ANN has the parameters automatically adjusted, with a problem dependent procedure. A very promising approach is to combine the ANN with others Artificial Intelligence techniques, as genetic algorithms, evolutionary strategies, simulated annealing, among others, for enhancement the final time series forecast.

In this chapter is presented some intelligent techniques as Artificial Neural Networks (Haykin, 1998), Genetic Algorithms (GA) (Goldberg, 1989), Particle Swarm Optimization (PSO) (Eberhart and Kennedy, 1995; van den Bergh and Engelbrecht, 2004), Greedy Randomized Adaptive Search Procedures (GRASP) (Feo and Resende, 1995; Resende and Ribeiro, 2003) and an Intelligent Hybrid method, composed of an ANN combined with GRASP procedure and Evolutionary Strategies, for chaotic and non-linear time series prediction, called GRASPES.

The GRASPES method is based on a multi-start metaheuristic for combinatorial problems to train, to tune and to adjust the structure and parameters of an ANN. The GRASPES is capable to evolve the parameters configuration and the weights in order to train the ANN, searching, in evolutionary sense, the minimum number of relevant time lags for a correct time series representation. It also looks for an optimal or sub-optimal predictive model. A detail experimental procedure, explained step by step, is shown, where an investigation is conducted with the GRASPES method with four different fitness function and with two time series. The results achieved are discussed and compared, according to five well-known statistical performance measures, like MSE, MAPE, POCID, Statistical of U Theil and ARV. Furthermore, in order to fill some lacks of experimental justified guidelines to help the practitioners to find good predictions using these techniques, an experimental analysis is made according to different types of fitness functions evaluations.

This document is organized as follows. In Section 3, some modelling strategies for non-linear times series analysis are presented. Intelligent methods for computational modeling are described in Section 4 and the problem of time series forecasting is defined in Section 2. The proposed method is developed in Section 5. The statistical performance measures and the fitness functions are presented in Section 6 and 7 respectively. Experimental results and some discussions are presented in Section 8. Finally, Section 9 provides the conclusion and a few remarks.

2. Time Series Forecasting Problem

In the branch of statistics, signal processing, or many other study fields, a time series is a set of data points, measured generally at successive times, spaced at (often uniform) time intervals, defined by,

$$X_t = \{x_t \in \mathbb{R} \mid t = 1, 2, 3 \ldots N\}, \tag{1}$$

where t is the temporal index and N is the number of observations. Therefore, X_t is a sequence of temporal observations orderly sequenced and equally spaced.

The objective of the forecast problem is to apply some prediction techniques for the time series X_t and to identify patterns presents in the historical data, building a model able to identify the next time patterns. This kind of problem is not always easy to solve, considering that a real world time series has a huge complexity.

In this context, a most relevant factor for a good forecasting performance is the correct choice of the time lags considered for a time series representation. For situations where there is a clear linear relationship among historical data of a phenomenon, the functions of auto-correlation

and partial auto-correlation are capable of identifying the important lags. Such procedure is usually applied in linear models, such as Box-Jenkins' models (Box et al., 1994).
However, when working with complex time series, which is the general situation in real world applications, the structures of relationship among historical data are actually non-linear, which makes the analysis procedure of the time lags based on those functions only a crude estimate.
Such relationship structures among historical data constitute a d-dimensional state space, where d is the minimum dimension capable of representing such relationship. Therefore, a d-dimensional state space can be built so that it is possible to unfold a time series in its interior. Takens (Takens, 1980) has proved that if d is sufficiently large, such built state space is homeomorphic to the state space which generated the time series. Thus, Takens Theorem (Takens, 1980) has provided the theoretical justification that it is possible to build a state space using the correct lags, and if this space is correctly rebuilt, guarantees that the dynamics of this space are topologically identical to the dynamics of the original system's state space.
In this way, it verifies that the main problem in the rebuilding of the state space is the correct choice of dimension d, i.e., the correct choice of the relevant time lags necessary for system dynamics characterization. In literature, can be found many methods used for the definition of the lags(Pi and Peterson, 1994; Savit and Green, 1991; Tanaka et al., 2001). Such methods are usually based on measures of conditional probabilities, which consider:

$$x_t = f(x_{t-1}, x_{t-2}, \ldots, x_{t-d}) + R_t \tag{2}$$

where $f(x_{t-1}, x_{t-2}, \ldots, x_{t-d})$ is a possible mapping of the pasts values to the facts of the future and R_t is a noise term. Generally, R_t decreases with the increase of d, and if the system is totally deterministic, R_t tends to zero when d exceeds the minimum embedded dimension necessary for the system description. With this kind of procedure, the objectives are to define the minimum embedded dimension capable of representing the series, to find the sensibility of x_t with respect to the time lags, and to estimate the size of the noise.
The exhibited method in this chapter does not discard any possible correlation that can exist between the series data, even higher order correlations, since it carries out an iterative automatic search for solving the problem of finding the relevant time lags using an evolutionary algorithm.

3. Computing and Mathematical Modelling

The term *model* is normally used for a structure which has been built purposely to exhibit the behavior of some other objects. Generally only some features and characteristics will be retained in the model depending upon the use.
Many models used for time series forecasting have standard forms, where they try to capture the time series features. One of these models is the popular ARIMA model (Box et al., 1994), which is the most common choice among the practitioners for time series prediction, but it is not the best choice for the case of non-linear time series forecasting problems (Rodrigues et al., 2008)
Furthermore, statistical parameters are obtained as a result of modelling uncertainty about problem data by specification of probability distributions over these data. The value of a statistical model stem from the ability to represent solutions that hedge against multiple possible future outcomes. In deterministic and linear model, optimal solutions tend toward extreme point solutions which rely on a limited set of activities (basic variables) and force a solution to meet critical constraints tightly. Thus, to model real problems, statistical parameters have

been used with Mathematical Programming, Differential Equations and Cellular Automata. Particularly, Kantz (2004) has been presented the most common strategies these approaches.
As reported by Derek Holmes (Robert R. Mc Cormick school of Engineering and Applied Science - Northwestern University - `http://users.icms.northwestern.edu`), many decision problems can be modeled using mathematical programs, which seek to maximize or minimize some objective which is a function of decisions. The possible decisions are constrained by limits in resources, minimum requirements. Decisions are represented by variables. Objectives and constraints are functions of the variables, and problem data. Stochastic programs are mathematical programs where some of data incorporated into the objective or constraints is uncertain. Uncertainty is usually characterized by a probability distribution on the parameters. Although the uncertainty is rigorously defined, in practice it can range in detail from a few scenarios (possible outcomes of the data) to specific and precise joint probability distributions. It is possible to formulate a Stochastic Linear Program, as shown on the Argonne National Laboratory (Mathematics and Computer Science Division - `http://www.mcs.anl.gov`), like a task that seek to minimize the cost of the first-stage decision plus the expected cost of the second-stage recourse decision:

$$\textbf{Min} \quad c^T x + E_w Q(x, y),$$

subject to

$$Ax = b \quad \textbf{and} \quad x \geq 0,$$

where

$$Q(x, y) = \quad \textbf{Min} \quad d(w)^T y,$$

subject to

$$T(w)x + W(w)y(w) = h(w).$$

The first linear program(LP) minimizes the first-stage direct costs, $c^T x$ plus the expected recourse cost, $Q(x, y)$, over all the possible scenarios while meeting the first-stage constraints, $Ax = b$.
The cost Q depends both on x, the first-stage decision, and on the random event, w. The second LP describes how to choose $y(w)$ (a different decision for each random scenario w). It minimizes the cost $d^T y$ subject to some function, $Tx + Wy = h$. This constraint can be thought of as requiring some action to correct the system after the random event occurs.
One important issue to notice in stochastic programs is that the first-stage decision, x, is independent of which second-stage scenario actually occurs. This is called the *non-anticipativity property*. The future is uncertain and so today's decision cannot take advantage of knowledge of the future. In this way, we can treat the events as independent ones and approximations on expected values are mathematically possible as to convert the problem in a deterministic equivalent one, which is used in Kantz (2004).
Therefore, intelligent methods for computational modelling have been successful applied to capture the chaotic and non-linear behavior of *forecasting real time series* (Ferreira et al., 2005; Ghiassi et al., 2005; Rodrigues et al., 2008; Zhang et al., 1998).

4. Intelligent Methods for Computational Modelling

In the field of intelligent computing there are several approaches that can be used for computational modeling. The techniques to be used are determined by the problem at hand, for problems such as classification or recognition of patterns can be used RBF networks, Support

Vector Machines (SVMs), KNN (K-Nearest Neighbors Algorithm), Meta-Heuristics, among others. For optimization problems can be utilized Genetic Algorithms, Particle Swarm Optimization, among others. For forecasting problem, Neural Networks , Multi-Layer Perceptron, Jordan type or Elman type, are heavily used and generally are combined with others techniques as Evolutionary Algorithms (Genetic Algorithms, Evolution strategy, Evolutionary programming and Genetic programming) and others approaches as Particle Swarm Optimization or SVMs for example.
In the next sections will describe some techniques used to further the time series forecasting problem.

4.1 Artificial Neural Networks – Multi-Layer Perceptron

An Artificial Neural Networks (ANN) is a mathematical model inspired in a biological neural network which automatically extract useful patterns to represent the desired information. The ability to learn through examples and to generalize the learned information is one of the most attractive characteristics of the ANN.
The key element of this abstract model is the structure of the information processing distributed system. It is composed of a large number of highly interconnected processing units divided in layers, but working in union to solve a determinate problem. Through a learning process, where adjustments to the synaptic connections that exist between the neurones, the ANN seek the best possible solution, being a similar process to biological neurons.
Each ANN is create and configured for a specific application, such as data classification or forecasting problem. The ANN is used with successful in solving time series forecasting problems due to its characteristic of modeling complex non-linear relationships among data (Ghiassi et al., 2005), without any prior supposition of the data nature.

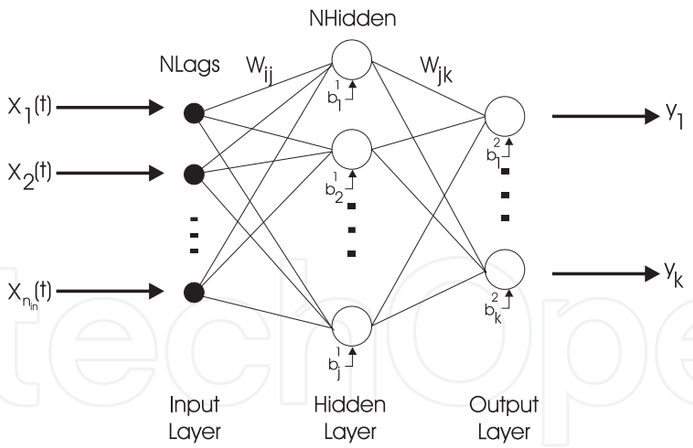


Fig. 1. Multi-Layer perceptron networks with link switches.

The possibility to improve the prediction performance of ANN can be achieved through the correct adjustment of its parameters. The main problem is to determine the optimal or suboptimal values of these parameters. Whereas that all parameters are problem dependent, it is a very complex task to find them into a very wide universe of possibilities. However, a layer ANN just solve a set of linearly separable problems, so to solve non-linearly separable problems, it is necessary to use at least one hidden layer ANN, according to Cybenko(Cybenko,

1989), who also proved that an ANN is an universal approximate function since the ANN has at least one hidden layer.
As the goal of this work is to predict continuous functions, then a MultiLayer Perceptron (MLP) networks with three layers of type *i-j-k* should be used, where i denotes the number of time lags (processing units in input layer), j denotes the number of processing units in hidden layer (sigmoidal units) and k denotes the number of processing units in output layer (the chosen prediction horizon here is of one step ahead, so $k = 1$ should be used).
Three possible distinct forms of modeling the ANN are proposed ($NetMod = 1, 2, 3$), where each one is described below and its parameters are:

- W_{ij}, weight of connections of the input layer for the intermediary layer;
- W_{jk}, weight of connections of the intermediary layer for the output layer;
- b_j^1, bias of the intermediary unit;
- b_k^2, bias of the output unit,

where all these parameters are real values.
The first ANN model ($NetMod = 1$), uses the sigmoidal activation function for all hidden processing units. The output processing unit uses a linear activation function where a sigmoidal function is applied to its bias. The output of ANN is given by,

$$y_k(t) = \sum_{j=1}^{n_h} W_{jk} Sig\left[\sum_{i=1}^{n_{in}} (W_{ij} Z_i(t) - b_j^1)\right] - Sig(b_k^2), \tag{3}$$

where $Z_i(t)$ ($i = 1, 2, \ldots, n_{in}$) are the ANN input values, n_{in} denotes the ANN input number and n_h is the hidden units number. Since the prediction horizon is one step ahead, only one output unit is necessary ($k = 1$). The term Sig is a sigmoidal function,

$$Sig(x) = \frac{1}{1 + exp(-x)}. \tag{4}$$

The utilization of the sigmoidal function to the bias, in this model, is the assumption of the linear correlation between the delay and a possible non-linear behavior of the series.
The Second ANN model ($NetMod = 2$), consists of hidden units activated by a sigmoidal function with its output layer using a linear function, given by:

$$y_k(t) = \sum_{j=1}^{n_h} W_{jk} Sig\left[\sum_{i=1}^{n_{in}} (W_{ij} Z_i(t) - b_j^1)\right] - b_k^2. \tag{5}$$

The Third ANN model ($NetMod = 3$), applies the sigmoidal activation function to all processing units, given by:

$$y_k(t) = Sig\left\{\sum_{j=1}^{n_h} W_{jk} Sig\left[\sum_{i=1}^{n_{in}} (W_{ij} Z_i(t) - b_j^1)\right] - b_k^2\right\}. \tag{6}$$

These three MLP modeling ($NetMod = 1, 2, 3$) can be combined with other systems, and it will search by architecture that better describes the time series phenomenon.

4.2 Genetic Algorithm

Genetic Algorithms (GA) (Goldberg, 1989) are a technique of directed random search widely applied in complex optimization problems. They are particularly interesting for employment in situations where the number of parameters is very large and analytical solutions are very difficult, or impossible, to obtain. The modified GA (MGA) exhibited here was originally proposed by(Leung et al., 2003) where new genetic operations were introduced to improve its performance.

4.2.1 Population

The population is composed of individuals (chromosomes), where each of these individuals represents a possible model for time series prediction: an ANN and its parameters.
Initially, the first set of population, P, is generated randomly, $P = \{\mathbf{ind}_1, \mathbf{ind}_2, \ldots, \mathbf{ind}_{pop\_size}\}$, where $\mathbf{ind}_i$ $(i = 1, 2, \ldots, pop\_size)$ are the individuals that make up the population and pop_size is the population size. Each individual, or chromosome, is composed of genes (the parameters of the solution) given by

$$\mathbf{X} = (x_1, x_2, \ldots, x_p) \tag{7}$$

where x_i $(i = 1, 2, \ldots, p)$ are the solution parameters and p is the maximum parameters number.
Each chromosome in the population is evaluated (Section 7) and the better chromosomes return higher fitness values.

4.2.2 Selection

In each generation, two chromosomes in the population will be selected to undergo a genetic operation (crossover operation) by the fitness proportionate method. A popular selection method is the spinning the roulette wheel(Goldberg, 1989; Leung et al., 2003), where the chromosome having a higher fitness value should therefore have a higher chance of being selected (higher potential parents will produce better offspring).

4.2.3 Genetic Operations

After the selection process, two chromosomes (parents) are combined to generate new chromosomes (offspring) by genetic operations. The genetic operations include the crossover and mutation operations.
The crossover operation is the basic means for exchanging information from the two parents ($\mathbf{p}_1$ and $\mathbf{p}_2$). These parents will produce one offspring composed of four new chromosomes (four sons), according to the following mechanisms:

$$C_1 = [c_1^1 \quad c_1^2 \quad \cdots \quad c_1^{no\_Vars}] = \frac{\mathbf{p}_1 + \mathbf{p}_2}{2} \tag{8}$$

$$C_2 = [c_2^1 \quad c_2^2 \quad \cdots \quad c_2^{no\_Vars}] = \mathbf{p}_{max}(1 - w) + \max(\mathbf{p}_1, \mathbf{p}_2)w \tag{9}$$

$$C_3 = [c_3^1 \quad c_3^2 \quad \cdots \quad c_3^{no\_Vars}] = \mathbf{p}_{min}(1 - w) + \min(\mathbf{p}_1, \mathbf{p}_2)w \tag{10}$$

$$C_4 = [c_4^1 \quad c_4^2 \quad \cdots \quad c_4^{no\_Vars}] = \frac{(\mathbf{p}_{min} + \mathbf{p}_{max})(1 - w) + (\mathbf{p}_1 + \mathbf{p}_2)w}{2} \tag{11}$$

$$\mathbf{P}_{max} = [para_{max}^1 \quad para_{max}^2 \quad \cdots \quad para_{max}^{no\_Vars}] \tag{12}$$

$$\mathbf{P}_{min} = [para^1_{min} \quad para^2_{min} \quad \cdots \quad para^{no\_Vars}_{min}] \tag{13}$$

where $w \in [0 \ldots 1]$ denotes a weight to be determined by user, $\max(\mathbf{p}_1, \mathbf{p}_2)$ and $\min(\mathbf{p}_1, \mathbf{p}_2)$ denotes the vector with each element obtained by taking the maximum and minimum, respectively, among the corresponding element of $\mathbf{p}_1$ and $\mathbf{p}_2$, no_Vars denotes the number of variables to be tuned and $para^j_{min}$ and $para^j_{max}$ are the minimum and maximum values of the parameter $\mathbf{X}^j_i$ for all i, respectively.
According to the Equations 8 – 11, the potential offspring spread over the state space defined by $\mathbf{P}_{max}$ and $\mathbf{P}_{min}$. Equations 8 and 11 result in searching around the center region of the state space (if $w \to 1$ then $C_4 \to C_1$), whereas Equations 9 and 10 move the potential offspring near to the domain boundary (if $w \to 1$ then $C_2 \to \mathbf{P}_{max}$ and $C_3 \to \mathbf{P}_{min}$).
After the potential offspring are generated by the crossover operation, the best offspring is chosen, and if this offspring is better than the worst chromosome from the old population, then this offspring replaces the worst chromosome.
Each one of the four new chromosomes generated by the crossover process is cloned and its clones undergo the mutation operation, where three new chromosomes are generated by,

$$MC_{i,\alpha} = [c^1_i \quad c^2_i \quad \cdots \quad c^{no\_Vars}_i] + [\delta_1 mc^1_i \quad \delta_2 mc^2_i \quad \cdots \quad \delta_{no\_Vars} mc^{no\_Vars}_i] \tag{14}$$

where $\alpha = 1, 2, 3$ is the mutation index, $i = 1, 2, 3, 4$ is the offspring index, δ_u ($u = 1, 2, \ldots, no\_Vars$) can only take values 0 or 1, and mc^u_i ($u = 1, 2, \ldots, no\_Vars$) are randomly generated numbers that satisfy the constraint $para^u_{min} \leq c^u_i + mc^u_i \leq para^u_{max}$. Small mutation are more likely than largest ones(Eiben and Smith, 2003) therefore a gaussian, with normal distribution, is used to perform the mutation operation .
The first mutation operation ($\alpha = 1$) is such that only one δ_u is 1 and all the others are 0 in Equation 14. The second mutation operation ($\alpha = 2$) is obtained by Equation 14, where some δ_u, randomly chosen, are set to 1 and others are set to 0. The third mutation operation ($\alpha = 3$) is obtained with all δ_u equal to 1 in Equation 14. A real number is randomly generated and compared to a user defined number $p_{Mut} \in [0 \ldots 1]$ (accepted mutational probability). If the real number is smaller than p_{Mut} then the mutated chromosome replaces the chromosome with the smallest fitness in the population. However, if the real number is larger than p_{Mut}, then the mutated chromosome replaces the chromosome with the smallest fitness of the population if and only if its fitness is greater than the fitness of the worst chromosome in the population.
The stopping criterion are: training progress, where the GA will stop when occurs a defined number of generations without a percentage increase the average of the population, and the maximum generations number.
The steps necessary for implementing the whole modified GA algorithm are shown below (Algorithm 2).

4.3 Particle Swarm Optimizer Fundamentals

The Particle Swarm Optimization (PSO) is an optimization technique based on a particle population of randomly solutions (i.e. individuals) to the optimization task at hand, where the population is referred to as swarm. At each iteration, each particle moves by the search space in the direction of its own personal best solution found so far, as well as in the direction of the global best position discovered so far by any of the particles in the swarm (van den

begin
  $\tau \rightarrow 0$; // τ: Number of iteration
  initialize **Pop**(τ); // **Pop**(τ): population for iteration τ
  evaluate $f($**Pop**$(\tau))$; // $f($**Pop**$(\tau))$: fitness function
  while *not termination condition* **do**
    $\tau \rightarrow \tau + 1$;
    select two parents $\mathbf{p_1}$ and $\mathbf{p_2}$ from **Pop**$(\boldsymbol{\sigma})$;
    perform *crossover* operation according to Equations 8 to 11;
    perform *mutation* operation according to Equation 14 to generate three new chromosomes $\mathbf{MC_1}$, $\mathbf{MC_2}$ and $\mathbf{MC_3}$;
      // Reproduce a new Population
    The chromosome generated by the crossover operation with the largest fitness value replaces the chromosome with the smallest fitness value in the **Pop**$(\tau - 1)$;
    for *i=1 to 3* **do**
      // p_{Mut}: probability of Mutation acceptance
      if *random number* $< p_{Mut}$ **then**
        $\mathbf{MC_i}$ replaces the chromosome with the smallest fitness value in the **Pop**$(\tau - 1)$
      else
        if $f(\mathbf{MC_i}) >$ *smallest fitness value in the* **Pop**$(\tau - 1)$ **then**
          $\mathbf{MC_i}$ replaces the chromosome with the smallest fitness value in the **Pop**$(\tau - 1)$
    evaluate $f($**Pop**$(\tau))$;
end

Fig. 2. Procedure of the Modified GA

Bergh and Engelbrecht, 2004). In this way, if any particle discovers a promising solution, the swarm is guided to the new solution in order to explore more thoroughly the found region. Assume that the swarm size is given by s. Each individual ($1 \leq i \leq s$) has a current position in search space (x_i), a current velocity (v_i) and a personal best position in the search space (y_i). Assuming that the function f is to be minimized, the swarm consists of n particles, and at each iteration, each swarm particle velocity is updated by

$$v_{i,j}(t+1) = wv_{i,j}(t) + c_1 r_1[y_{i,j}(t) - x_{i,j}(t)] + c_2 r_2[\hat{y}_j(t) - x_{i,j}(t)] \ , \tag{15}$$

where $j \in 1, 2, \ldots, n$, $\hat{y}_j(t)$ denotes the current position in search space (found by any swarm particle), $y_{i,j}(t)$ represents the personal best position in the search space (found by each swarm particle), $v_{i,j}$ is the velocity of the j-th dimension of the i-th particle, c_1 and c_2 represent the acceleration coefficients, which control how far a particle will move in a single iteration, and $r_1 \sim U(0,1)$ and $r_2 \sim U(0,1)$ are elements from two uniform random sequences in the interval [0,1]. The term w is referred to as inertia weight, in which this value is typically setup to vary linearly from 1 to near 0 during the course of the procedure. It is worth to mention that this is reminiscent of the temperature adjustment schedule found in Simulated Annealing algorithms(van den Bergh and Engelbrecht, 2004).

Thus, the new particle position is updated by

$$x_i(t+1) = x_i(t) + v_i(t+1). \tag{16}$$

The personal best particle position and the global best particle found by any particle during all previous iterations are updated by equations (17) and (18), respectively.

$$y_i(t+1) = \begin{cases} y_i(t) & \text{if } f(x_i(t+1)) \geq f(y_i(t)), \\ x_i(t+1) & \text{otherwise.} \end{cases}; \tag{17}$$

$$\hat{y}(t+1) = \text{argmin} f(y_i(t+1)). \tag{18}$$

begin
  initialize the particle population;
  while *stop criterion not satisfied* **do**
    for $i = 1$ *to s of population swarm* **do**
      if $f(x_i(t)) < f(y_i(t))$ **then**
        $y_i(t) = x_i(t)$;
      end
      if $(f(y_i(t)) < f(\hat{y}_i(t)))$ **then**
        $\hat{y}_i(t) = y_i(t)$;
      end
    end
  end
  update the velocity and position of each particle according to equations (13) and (14);
end

Fig. 3. Particle Swarm Optimizer Procedure

The term v_i is normalized in the range $[-v_{max}, v_{max}]$ in order to reduce the likelihood of particles leaving the search space. It is worth to mention that this mechanism doesn't restrict the values of x_i in the range of v_i, it only limits the maximum distance that a particle will move during each iteration(van den Bergh and Engelbrecht, 2004). Figure3 illustrates the PSO procedure.

4.4 The GRASP Method

The GRASP (Resende and Ribeiro, 2003) method is a randomly interactive technique which each iteration consists of two phases: construction and local search.
The construction phase builds a feasible solution, whose neighborhood is investigated until a local minimum is found during the local search phase. The best overall solution is kept as the result. The general expectation is that, given a sub-optimal solution, closed to it there will be, with high probability, other sub-optimal (or optimal) solutions. The search will tend to look around of such solution, stopping when a local optimum model is found.
A problem of combinatorial optimization, is defined by the finite set of data $D = \{1, 2, \ldots, N\}$, the set of possible solutions $G \subseteq 2^D$, and the objective function $f : 2^D \rightarrow \mathbf{R}$. For minimization problems, searches for the excellent solution $S' \in G$ such that $f(S') =< f(S) \forall S \in G$. The ground set D, the cost function f, and the set of feasible solutions G are defined for each specific problem for example as described on Algorithm 4.

```
begin GRASP
    initialize MaxIter, Seed;
    Read Input();
    for i=1 to MaxIter do
        Solution ← Greedy Randomized Construction(Seed);
        Solution ← Local Search(Solution);
        UpdateSolution(Solution, BestSolution);
    end
    return BestSolution;
end
```

Fig. 4. Pseudo-code of the GRASP metaheuristic (Resende and Ribeiro, 2003).

5. The GRASPES Method

The GRASPES (Rodrigues et al., 2008) is basically a combination of Evolutionary Strategies(ES) and the GRASP method (Section 4.4).
For methods based on evolutionary computation (Evolutionary Strategies are a part of evolutionary computation), the process of biological evolution is mimicked. Population is composed by set of trial solutions of the problem, being each solution (individual) coded by a parameter vector (data structure, referred to as chromosome).
Let $\mathbf{X}$ be a chromosome defined by,

$$\mathbf{X} = (x_1, x_2, \ldots, x_p; \sigma_1, \sigma_2, \ldots, \sigma_p) \tag{19}$$

where x_i and σ_i are respectively the solution parameters and the mutation step size of each parameter with $i = 1, 2, \ldots, p$ and p is the maximum parameters number. The model represented by Equation 19, used to describe a three-layer ANN parameters, coded the chromosomes in the population.
The mutation operation is defined by Eiben and Smith (2003) ,

$$\mathbf{X}' = (x_1', x_2', \ldots, x_p'; \sigma_1', \sigma_2', \ldots, \sigma_p'), \tag{20}$$

with

$$\sigma_i' = \sigma_i \cdot exp(\tau' \cdot N(0,1) + \tau \cdot N_i(0,1)), \tag{21}$$

$$x_i' = x_i + \sigma' \cdot N_i(0,1), \tag{22}$$

where $\tau' \propto 1/\sqrt{2f}$, $\tau \propto 1/\sqrt{2\sqrt{f}}$, f is the degree of freedom and $N(0,1)$ is a normal gaussian distribution.
Each individual codifies a three layer Multilayer Perceptron (MLP) ANN, which represents a model for time series forecasting. An ES initialize one individual I, which is a potential solution, generated randomly. The individual will be evaluated by the fitness function, Equation 29 described on Section 7, where betters individuals will return higher fitness values. The ES clones the father's chromosome I_p and will then undergo a operation of mutation which changes the genes of the chromosome. For tuning the ANN structure (Leung et al., 2003), integer random numbers are generated to define the ANN number of time lags (processing units in input layer i), the number of processing units in hidden layer (sigmoidal units j) and the modeling of the ANN. For each weight of the optimal individual I the mutation is applied as

described in the Equations 20, 21 and 22. This new individual is evaluated and will be saved, if and if only, its solution quality (Section 7) is better than the actual father.
This steps will be repeated until the mutated individuals number criterium or the size of the population n is reached. When this fact occur, it will be said that a Parent's Generation (PG) occurs and if this PG has any offspring better than father it will substitute the father. The stopping criterion are: progress of PG evolution, where the method will stop if a PG iteration number occur without better individual generation (a individual is considered "better" when your fitness is greater, a percentage value, than the father), or a maximum PG number. The basic steps of the method are described in the Algorithm, 5.

```
begin GRASPES
    initialize parent;
    evaluate f(parent);  // f(·): fitness function
    while not PG criterium reached do
        clone parent;
        for w=1 to number of iteration per father do
            define the input layer i and hidden layer j;
            perform mutation operation on sons I_τ;
            evaluate f(I_τ);
            if f(I_τ) > parent's fitness value then
                save the offspring;
                if the size of n was reached then
                    break;
                end
            end
        end
        if (f(parent) - f(offspring)) > % of minimal fitness) then
            the individual will be the new parent;
        end
    end
end
```

Fig. 5. Pseudo-code of the GRASPES Method

6. Error Measure

For the forecasting problem, the natural measure of performance is the prediction error. However, there is no universal measure adopted (by the literature of the branch) to evaluate the prediction (Tashman, 2000). Error measures also play an important role in calibrating or refining a model so that it will forecast accurately for a set of time series (Armstrong and Collopy, 1992). The use of only one error for evaluate the model performance (*e.g.*, MSE), not shows the behavior of the predict (Clements and Hendry, 1993) in a clear way, for this reason more performance criteria should be considered for make robust the evaluation of the results and final desirable goals (Tashman, 2000).
Considering T the value to be predicted of the time series (target) and O the model output (prediction), five well known error measure are considered to evaluate the prediction performance:

MSE (Mean Squared Error):the most popular measure used for performance prediction,

$$MSE = \frac{1}{N}\sum_{j=1}^{N}\left(e_j\right)^2, \tag{23}$$

where N is the amount of the target points on the set and $e_j = (T_j - O_j)$.
MAPE (Percentage Average Error):

$$MAPE = \frac{1}{N}\sum_{j=1}^{N}\left|\frac{e_j}{X_j}\right| \tag{24}$$

where X_j is the point of the set in the instant j.
U of Theil Statistics: it is based in the predictor MSE, normalized by a random walk forecast error. A random walk model assumes that optimum value for the time $t+1$ is the value gotten in the time t, plus a noise term. Thus, the U of Theil Statistics can be given by:

$$Theil = \frac{\sum_{j=1}^{N}\left(T_j - O_j\right)^2}{\sum_{j=1}^{N}\left(T_j - T_{j+1}\right)^2}. \tag{25}$$

that associates the model performance with a random walk model. If the U of Theil Statistics is equal to 1, the predictor has the same performance of a random walk model. If the U of Theil Statistics is greater than 1, then the predictor has a worse performance than a Random Walk model, and if the U of Theil Statistics is less than 1, the predictor is better than a random walk model. So, the predictor is usable if its U of Theil Statistics is less than 1, and tends to the perfect model if the U of Theil Statistics tends to zero.
POCID (Prediction of Forecast the Alterations of Direction): measures the percentage of rightness with the trend of the series, if the future value will to go up or to go fall in relation to the current value.

$$POCID = 100\frac{\sum_{j=1}^{N} D_j}{N}, \tag{26}$$

with

$$D_j = \begin{cases} 1 & \text{if } (T_j - T_{j-1})(O_j - O_{j-1}) > 0, \\ 0 & \text{other case.} \end{cases} \tag{27}$$

ARV (Average Relative Variance): measures the relative performance model gain of the prediction of the series average,

$$ARV = \frac{\sum_{j=1}^{N}(O_j - T_j)^2}{\sum_{j=1}^{N}(O_j - \overline{T})^2} \tag{28}$$

where N, T_j, and O_j are the same parameters of the other evaluation measures, and $\overline{T}$ is the time series mean. If the ARV value is equal to 1, the predictor has the same performance as calculating the mean over the series, if the ARV value is greater than 1, the predictor is worse than simply taking the mean, and, if the ARV is less than 1, then the predictor is better than considering the mean as the prediction. So, the predictor is usable if the value of ARV is less than 1, and tends to the perfect model when the ARV tends to zero. In an ideal model, the POCID tends to 100% and all other error measures tends to zero.

7. Fitness Function

The first important feature about fitness computation is that it represents 99% of the total computational cost of evolution in most real-world problems. Second, the fitness function very often is the only information about the problem in the algorithm: any available and usable knowledge about the problem domain should be used (Eiben and Schoenauer, 2002).
Basically, the fitness function (aptitude), assigns a fitness value to each point in the space, where this value can be seen as a measure of how good a solution, represented by that point in the landscape, is to given problem (Hordijk, 1996). The correct choice of the fitness function is fundamental for a good solution of the problem.
For the best ANN model choice (of each individual), it is calculated its fitness function through of error measure, which is:

$$fitness = f_n(I) \tag{29}$$

where the functions are:

$$f_1(I) = \frac{1}{1 + ARV} \tag{30}$$

$$f_2(I) = \frac{1}{1 + MSE} \tag{31}$$

$$f_3(I) = \frac{1}{1 + THEIL} \tag{32}$$

$$f_4(I) = \frac{POCID}{1 + ARV + MSE + THEIL + MAPE} \tag{33}$$

8. Experimental Results

There is no only and universal way adopted for define the cardinality of the data set, but one common example used is divide the series in three sets: training set with 50% of the data, validation set with 25% of the data and test set with the last 25% of data. In order to be able to concentrate on the effects of the methods, it makes sense avoid unnecessary complications due to effects of other much more components (Jansen et al., 2005). For this reason all series investigated were normalized to lie within the interval [0;1] and the MLP networks are not trained by any conventional algorithms like backpropagation (Haykin, 1998) for example, avoiding the possibility that the training method could interfere in the general search.
Conclusions about the accuracy of various forecasting methods typically require comparisons across some time series (Armstrong and Collopy, 1992). Two financial time series were used for evaluation of the GRASPES (Dow Jones Industrial Average Index and S&P500 Stock Index).
For each time series with a different fitness function, ten experiments were repeated and the results with the best individual according with the best value of the validation fitness function (of the test set) is chosen to represent the model. For the predictions (with 1 step ahead of prediction horizon), the methods automatically choose a window time with length between 1 lag and 10 lags for the time series representation and the size of the j-layer (between 1 and 10). In addition, experiments with two hybrid methods to training an ANN (one using a modified genetic algorithm -MGA- and another using particle swarm optimization -PSO) are used for comparison with the proposed method.

The termination conditions for the GRASPES are increase of 1% of the minimal population fitness average value better than the previous (in the case of GRASPES the population is only one individual), after 10000 generations or when the fitness function of the validation set decrease 1% with respect to last round.

8.1 Standard & Poor 500 (S&P500)

The S&P500 Stock Index is a index of market values of the most negotiated actions in the New York Stock Exchange (NYSE), American Stock Exchange (AMEX) and Nasdaq National Market System. The S&P500 series corresponds to the monthly records from January 1970 to August 2003, constituting a database of 369 points. In order to reduce exponential trend of the S&P500 Stock Index, the natural logarithm was applied to the original values of the series.

	MGA	PSO	GRASPES			
Measures	f_3	f_4	f_1	f_2	f_3	f_4
ARV	0.015	0.053	0.009	0.012	0.011	0.044
MAPE	0.012	0.240	0.009	0.010	0.010	0.020
MSE(10^{-4})	1.776	6.988	1.183	1.418	1.416	4.896
POCID	77.419	51.111	67.391	68.817	67.391	72.043
THEIL	1.760	9.704	1.166	1.407	1.395	4.871

Table 1. Results - S&P500 - Best Individuals

The table 1 shows the experiments results, where for the MGA (Modified Genetic Algorithm) methodology the best fitness function was the f_3, for the PSO methodology the best fitness function was the f_4 and for the GRASPES method all fitness functions were shown. The GRASPES has always a superior performance when compared with the hybrid method using PSO. The Table also shows that when the GRASPES uses the fitness function f_1, f_2 and f_3 the MGA beat only the POCID error.
Figures 6, 7, 8 and 9 show the results of the test set according with the target (solid lines) and the predict values (dashed lines).

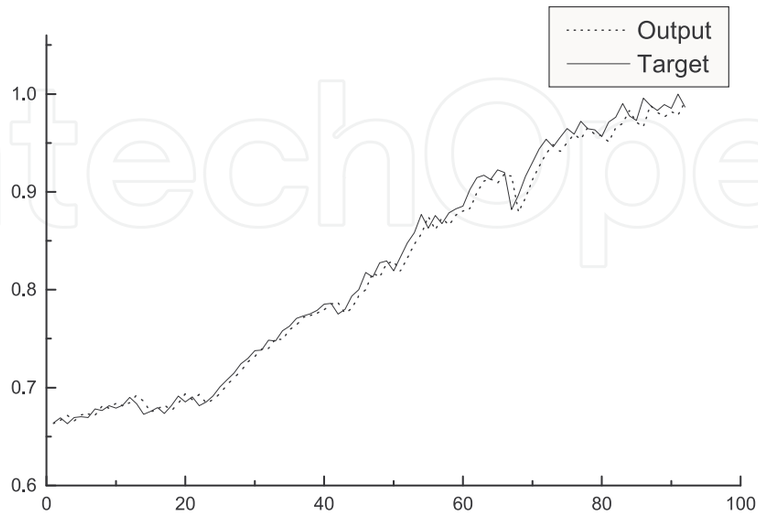


Fig. 6. Prediction os S&P500 series with Fitness Function f_1.

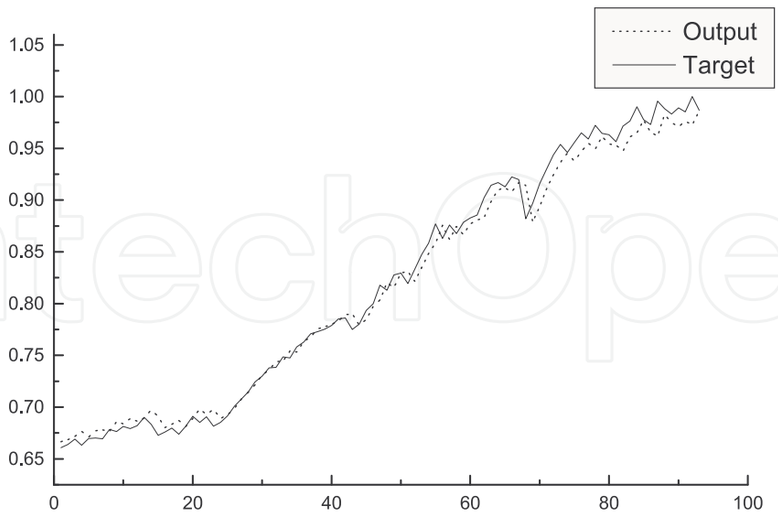


Fig. 7. Prediction os S&P500 series with Fitness Function f_2.

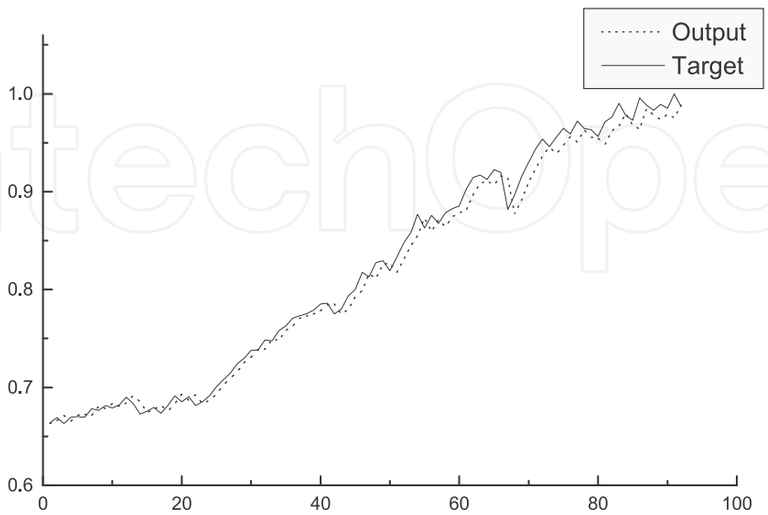


Fig. 8. Prediction os S&P500 series with Fitness Function f_3.

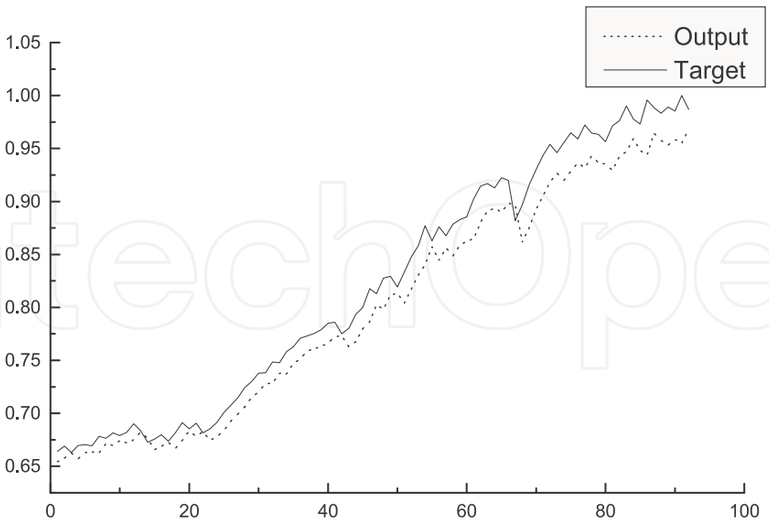


Fig. 9. Prediction os S&P500 series with Fitness Function f_4.

8.2 Dow Jones Industrial Average (DJIA)

The Dow Jones Industrial Average (DJIA) Index series corresponds to daily records from January 1st 1998 to August 26th 2003, constituting a database of 1420 points.

	MGA	PSO	GRASPES			
Measures	f_1	f_4	f_1	f_2	f_3	f_4
ARV	0.034	0.049	0.032	0.034	0.033	0.033
MAPE	0.098	0.143	0.095	0.098	0.096	0.097
MSE(10^{-3})	0.909	1.200	0.831	0.827	0.823	0.842
POCID	52.112	47.025	51.685	52.112	51.267	52.112
THEIL	1.087	1.986	0.998	0.991	0.986	1.009

Table 2. Results - DJIA - Best Individuals

Table 2 shows that the GRASPES has always a superior performance when compared with the hybrid method using PSO. The Table also shows that when the GRASPES uses the fitness function f_2 and f_3 the POCID error is the same as MGA and the others error are better.
Figures 10, 11, 12 and 13 show the results of the test set according with the target (solid lines) and the predict values (dashed lines).

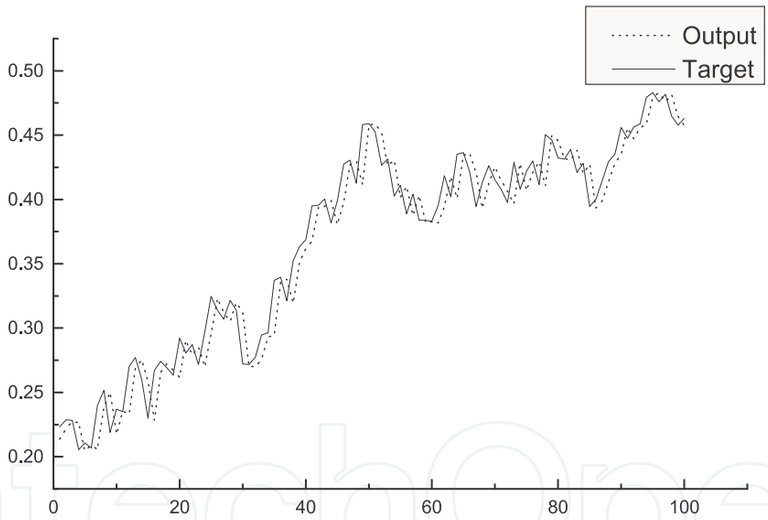


Fig. 10. Prediction os DJIA series with Fitness Function f_1.

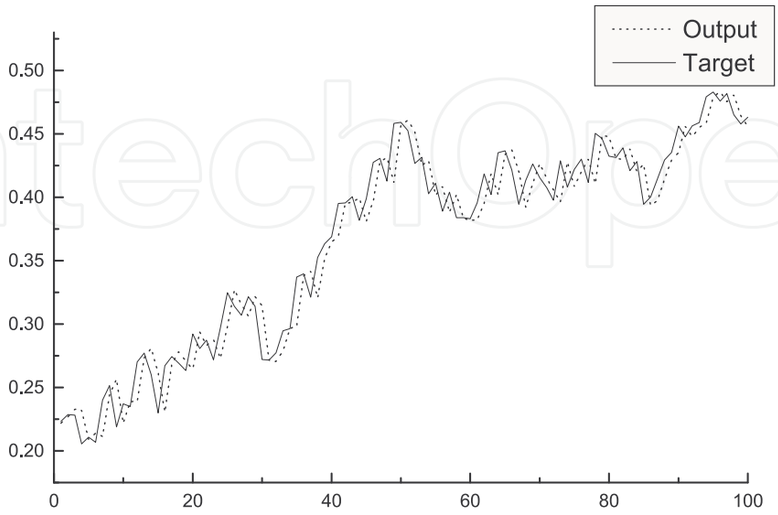


Fig. 11. Prediction os DJIA series with Fitness Function f_2.

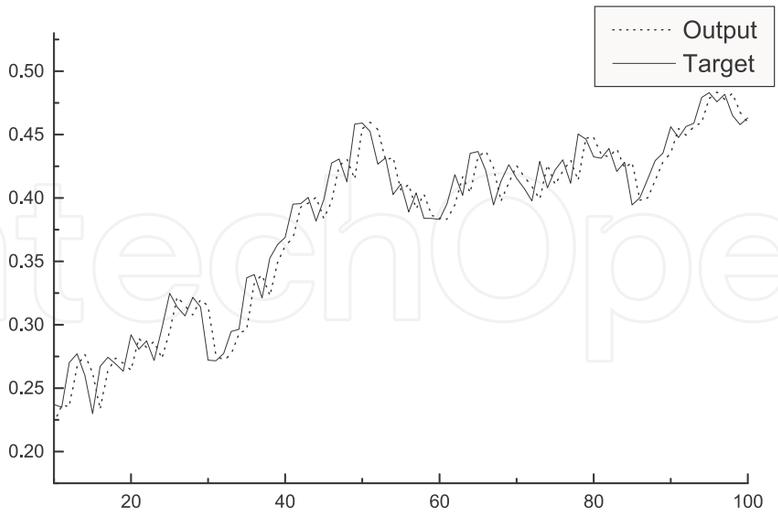


Fig. 12. Prediction os DJIA series with Fitness Function f_3.

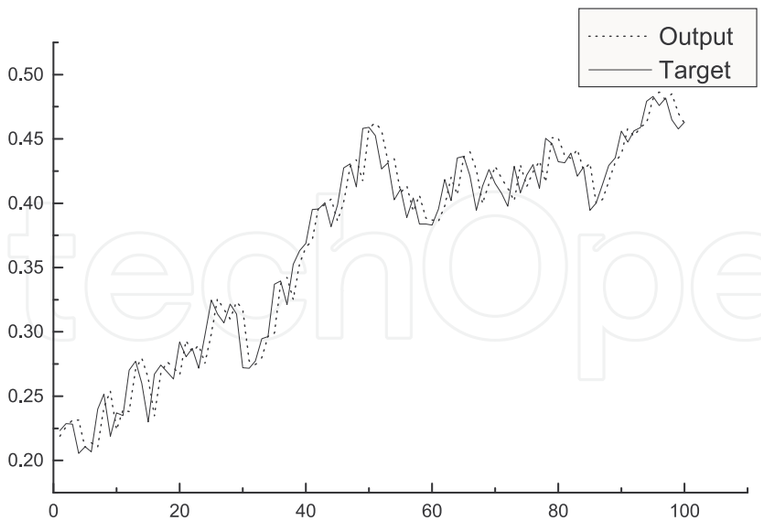


Fig. 13. Prediction os DJIA series with Fitness Function f_4.

9. Conclusions

In this chapter was presented a summary of how use the intelligent computational modelling for time series forecasting and the importance of the correct choice of the fitness function. Three methodology were employed for adjust the parameters of an ANN, a Modified Genetic Algorithm (MGA) (Section 4.2), a Particle Swarm Optimization (PSO) (Section 4.3) and the GRASPES Method (Section 5).

The success of evolution control is highly dependent of the optimization algorithm and the fitness function complexity (Buche et al., 2005). As affirmed in the Section 7, the fitness function assigns a fitness value to each individual in the population of Evolutionary Algorithm, at any time, measuring how good is a solution. This solution is represented by a point in the landscape. When an algorithm has a possibility to be guided by different fitness functions, each one will walk on the space, pointing in a different way. The experiments presented here show that the choice of the fitness function is also very important as the choice of the intelligent method employed for time series modelling.

The results reached with the MGA and PSO methods were developed in independent way and can be found at (de Mattos Neto et al., 2009; Rodrigues et al., 2009), respectively. In the Rodrigues et al. (2009) was employed eight different fitness functions, where the fitness function of Equation 32 achieved the best performance for the S&P500 index series and the fitness function 3 obtained the best result for the Dow Jones Industrial Average index series. However, in the (de Mattos Neto et al., 2009) the main goal was to develop the combination between the intelligent techniques ANN and PSO. For this reason, only one fitness function was applied (fitness function given by Equation 33)

Analyzing the S&P500 index results obtained here is possible to observe that the statistical error measures have a strong accomplished. These statistical measures present many times a competitive behavior. For example, observing the Table 1 for the GRASPES method, the fitness function f_2 (Equation 31) is directly based on MSE error, but f_2 has a inferior performance for the MSE error than the fitness function f_1 (Equation 30) which is based on ARV error. This observation shows that the choice of the fitness function is not a trivial decision.

According to the Table 2, where the experimental results for the Dow Jones Industrial Average are exhibits, it is possible to observe that the variation among the analyzed fitness functions are not significative. This observation shows that the sensibility of the choice of the fitness function for the Dow Jones Industrial Average is very low in comparison with the S&P500 index results, *i.e.*, small changes of the fitness function evaluation could lead to a significantly improved forecast performance.

10. References

Armstrong, J. S. and Collopy, F. (1992). Error measures for generalizing about forecasting methods: Empirical comparisons, *International Journal of Forecasting* **8**(1): 69–80.

Box, G. E. P., Jenkins, G. M. and Reinsel, G. C. (1994). *Time Series Analysis: Forecasting and Control*, third edn, Prentice Hall, New Jersey.

Buche, D., Schraudolph, N. and Koumoutsakos, P. (2005). Accelerating evolutionary algorithms with gaussian process fitness function models, *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews* **35**(2): 183–194.

Clements, M. P. and Hendry, D. F. (1993). On the limitations of comparing mean square forecast errors, *Journal of Forecasting* **12**(8): 617–637.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function, *Mathematical Control Signals Systems* **2**: 303–314.

de Mattos Neto, P. S. G., Petry, G. G., Rodrigues, A. and Ferreira, T. A. E. (2009). Combining artificial neural network and particle swarm system for time series forecasting., *Proceedings of International Joint Conference on Neural Networks*, IEEE, Atlanta - Georgia.

Eberhart, R. C. and Kennedy, J. (1995). A new optimizer using particle swarm theory, *Proceedings of the Int. Symp. Micro Machine and Human Science*, Nagoya, Japan, pp. 39–43.

Eiben, A. E. and Schoenauer, M. (2002). Evolutionary computing, *Information Processing Letters* **82**(1): 1–6.

Eiben, A. E. and Smith, J. E. (2003). *Introduction to Evolutionary Computing*, Natural Computing Series, Springer, Berlin.

Feo, T. and Resende, M. (1995). Greedy randomized adaptive search procedures, *Journal of Global Optimization* **6**(2): 109–133.

Ferreira, T. A. E., Vasconcelos, G. C. and Adeodato, P. J. L. (2005). A new evolutionary method for times series ferecasting, *GECCO '05: Proceedings of the 2005 conference on Genetic and evolutionary computation*, ACM, New York, NY, USA, pp. 2221 – 2222.

Ghiassi, M., Saidane, H. and Zimbra, D. K. (2005). A dynamic artificial neural network model for forecasting time series events, *International Journal of Forecasting* **21**(2): 341–362. http://www.sciencedirect.com/science/article/B6V92-4F011CS-2/1/6ada74e2310edd6e2a37a22516d28e63.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley.

Gooijer, J. G. D. and Kumar, K. (1992). Some recent developments in non-linear time series modelling, testing, and forecasting, *International Journal of Forecasting* **8**: 135–156.

Haykin, S. (1998). *Neural Networks - A Comprehensive Foundation*, second edn, Pearson Education.

Hordijk, W. (1996). A measure of landscapes, *Evolutionary Computation* **4**(4): 335–360. http://www.mitpressjournals.org/doi/abs/10.1162/evco.1996.4.4.335.

Jansen, T., De, K. A. J. and Wegener, I. (2005). On the choice of the offspring population size in evolutionary algorithms, *Evolutionary Computation* **13**(4): 413–440. http://www.mitpressjournals.org/doi/abs/10.1162/106365605774666921.

Kantz, H. (2004). *Nonlinear Time Series Analysis*, second edn, Cambridge University Press.

Lam, L. (1998). *Nonlinear physics for beginners: fractals, chaos, solitons, pattern formation, cellular automata, complex systems.*, World Scientific.

Leung, F. H. F., Lam, H. K., Ling, S. H. and Tam, P. K. S. (2003). Tuning of the structure and parametrs of the neural network using an improved genetic algorithm, *IEEE Transaction on Neural Networks* **14**(1): 79–88.

Pi, H. and Peterson, C. (1994). Finding the embedding dimension and variable dependences in time series, *Neural Computation* **6**: 509–520.

Resende, M. and Ribeiro, C. (2003). Greedy randomized adaptive search procedures, *Handbook of Metaheuristics*, Vol. 57, Springer, pp. 219–250.

Rodrigues, A., de Mattos Neto, P. S. G. and Ferreira, T. A. E. (2009). A prime step in the time series forecasting with hybrid methods: The fitness function choise, *Proceedings of International Joint Conference on Neural Networks*, IEEE, Atlanta - Georgia.

Rodrigues, A., Ferreira, T. A. E. and de A. Araujo, R. (2008). An experimental study with a hybrid method for tuning neural network for time series prediction, *Evolutionary Computation, 2008. CEC 2008. (IEEE World Congress on Computational Intelligence). IEEE Congress on* pp. 3435–3442.

Savit, R. and Green, M. (1991). Time series and dependent variables, *Physica D* **50**: 95–116.

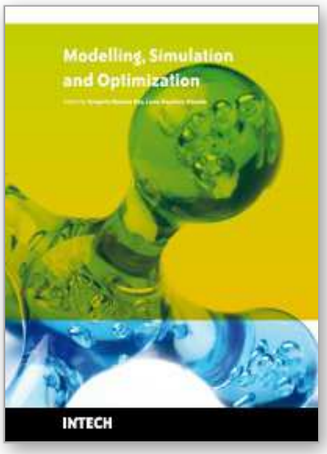
Takens, F. (1980). Detecting strange attractor in turbulence, *in* A. Dold and B. Eckmann (eds), *Dynamical Systems and Turbulence*, Vol. 898 of *Lecture Notes in Mathematics*, Springer-Verlag, New York, pp. 366–381.

Tanaka, N., Okamoto, H. and Naito, M. (2001). Estimating the active dimension of the dynamics in a time series based on a information criterion, *Physica D* **158**: 19–31.

Tashman, L. J. (2000). Out-of-sample tests of forecasting accuracy: An analysis and review, *International Journal of Forecasting* **16**(4): 437–450.

van den Bergh, F. and Engelbrecht, A. P. (2004). A cooperative approach to particle swarm optimization., *IEEE Trans. Evolutionary Computation* **8**(3): 225–239.

Zhang, G., Patuwo, B. E. and Hu, M. Y. (1998). Forecasting with artificial neural networks: The state of the art, *International Journal of Forecasting* **14**: 35–62.



Modelling Simulation and Optimization

Edited by Gregorio Romero Rey and Luisa Martinez Muneta

ISBN 978-953-307-048-3
Hard cover, 708 pages
Publisher InTech
Published online 01, February, 2010
Published in print edition February, 2010

Computer-Aided Design and system analysis aim to find mathematical models that allow emulating the behaviour of components and facilities. The high competitiveness in industry, the little time available for product development and the high cost in terms of time and money of producing the initial prototypes means that the computer-aided design and analysis of products are taking on major importance. On the other hand, in most areas of engineering the components of a system are interconnected and belong to different domains of physics (mechanics, electrics, hydraulics, thermal...). When developing a complete multidisciplinary system, it needs to integrate a design procedure to ensure that it will be successfully achieved. Engineering systems require an analysis of their dynamic behaviour (evolution over time or path of their different variables). The purpose of modelling and simulating dynamic systems is to generate a set of algebraic and differential equations or a mathematical model. In order to perform rapid product optimisation iterations, the models must be formulated and evaluated in the most efficient way. Automated environments contribute to this. One of the pioneers of simulation technology in medicine defines simulation as a technique, not a technology, that replaces real experiences with guided experiences reproducing important aspects of the real world in a fully interactive fashion [iii]. In the following chapters the reader will be introduced to the world of simulation in topics of current interest such as medicine, military purposes and their use in industry for diverse applications that range from the use of networks to combining thermal, chemical or electrical aspects, among others. We hope that after reading the different sections of this book we will have succeeded in bringing across what the scientific community is doing in the field of simulation and that it will be to your interest and liking. Lastly, we would like to thank all the authors for their excellent contributions in the different areas of simulation.

University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen

IntechOpen