Time series forecasting by evolving artificial neural networks using genetic algorithms and estimation of distribution algorithms

Juan Peralta, German Gutierrez, Araceli Sanchis

Abstract— **Accurate time series forecasting are important for displaying the manner in which the past continues to affect the future and for planning our day to-day activities. In recent years, a large literature has evolved on the use of evolving artificial neural networks (EANNs) in many forecasting applications. Evolving neural networks are particularly appealing because of their ability to model an unspecified non-linear relationship between time series variables. This paper evaluates two methods to evolve neural networks architectures, one carried out with genetic algorithm and a second one carry out with estimation of distribution algorithms. A comparative study between these two methods, with a set of referenced time series will be shown. The object of this study is to try to improve the final forecasting getting an accurate system.**

I. Introduction

IN order to acquire knowledge, it is interesting to know what the future will look like, i.e. forecast the future from the observed past. Time series forecasting is an essential research field due to its effectiveness in human life. It is a discipline that finds each day more applications in areas like planning, management, production, maintenance and control of industrial processes, economy, and weather forecasting.

The forecasting task can be performed by several techniques as Statistical methods [1], and others based on Computational Intelligence like Immune Systems [2] and Artificial Neural Networks (ANN) [3].

ANNs provide a methodology for solving many types of nonlinear problems that are difficult to solve by traditional techniques. Most time series processes often exhibit temporal and spatial variability, and are suffered by issues of nonlinearity of physical processes, conflicting spatial and temporal scale and uncertainty in parameter estimates. The ANNs have capability to extract the relationship between the inputs and outputs of a process, without the physics being explicitly provided. Thus, these properties of ANNs are well suited to the problem of time series forecasting.

This contribution reports the methodology to carry out the automatic design of ANN that tackles the forecasting of a referenced set of time series [4]. The task will consist of forecasting several time series, not all of them with the same ANN, but an automatic method will be used to obtain a different ANN to forecast each time series.

Two different steps, as it was explained in an earlier work [5], will be done to get an ANN to forecast each time series. The first step will consist of setting the kind of ANN that will solve the forecasting task, and the learning algorithm used.

In the second step the design of the ANN will be done setting the parameter values of the ANN, i.e. number of input nodes, number of hidden nodes, learning rate for BP and finally all connections weights. These parameters are given by carrying out a search process performed by two different evolving algorithms, a Genetic Algorithm (GA), and Estimation Distribution Algorithm (EDA).

The paper is organized as follows. Sec II reviews the related work about how to tackle forecast task with ANN, and design of ANN with Evolutionary Computation. Sec III will explain how our system designs ANN with GA and EDA to forecast time series. In Sec IV experimental setup and results are shown. And finally, conclusions and future works are described in Sec V.

II. Related work

A. Time series and ANN

Several works have tackled the forecasting time series task with ANN, not only computer science researchers, but statistics as well [1]. This shows the full consideration of ANN (as a data driven learning machine) into forecasting theory [6].

Before using an ANN to forecast, it has to be designed, i.e. establishing the suitable value for each freedom degree of the ANN [7] (kind of net, number of input nodes, number of outputs neurons, number of hidden layer, number of hidden neurons, the connections from one node to another , connection weights, etc). The design process is more an "art" based on test and error and the experience of human designer, than an algorithm. In [6] Zhang, Patuwo and Hu present a "state of the art" of ANN into forecasting task, in [8] is proposed an "extensive modeling approach" to review several designs of ANN.

The research reported here has been supported by the Spanish Ministry of Science and Innovation under project TRA2007-67374-C02-02.

J. Peralta. Computer Science Department, University Carlos III of Madrid,, SPAIN (phone: +34 916249424; fax: +34 916249129; e-mail:jperalta@inf.uc3m.es)

German Gutierrez. Computer Science Department, University Carlos III of Madrid,, SPAIN (phone: +34 916249135; fax: +34 916249129; e-mail: ggutierr@inf.uc3m.es)

Araceli Sanchis. Computer Science Department, University Carlos III of Madrid,, SPAIN (phone: +34 916249423; fax: +34 916249129; e-mail: araceli.sanchis@uc3m.es).

978-1-4244-8126-2/10/$26.00 ©2010 IEEE

The problem of forecasting time series with ANN is considered as modeling the relationship of the value of the element in time *"t"* (due to the net will only have one output neuron) and the values of previous elements of the time series (*t-1, t-2,.., t-k*) to obtain a function as it is shown in (1):

$$a_t = f(a_{t-1}, a_{t-2}, \ldots, a_{t-k}) \quad (1)$$

B. ANN and Evolutionary Computation

Several works show methods to obtain ANN design by an automatic way; among them, those that use Evolutionary Computation (EC) reveal that the search process carried out by evolutionary techniques, obtain good results [9,10,11,12,13].

Some of them use Direct Encoding Schemata (DES) [9,10], others use Indirect Encoding Schemata (IES) [11,12,13]. For DES the chromosome contains information about parameters of the topology, architecture, learning parameters, etc. of the Artificial Neural Network. In IES the chromosome contains the necessary information so that a constructive method gives rise to an Artificial Neural Network topology (or architecture). Abraham [14] shows an automatic framework for optimization ANN in an adaptive way, and Xin Yao et. al. [15] try to spell out the future trends of the field.

III. ANN DESIGN WITH GA AND EDA

A. Learning pattern set

In order to obtain a single ANN to forecast time series values, an initial step has to be done with the original values of the time series, i.e. normalize the data. And once the ANN gives the resulting values, the inverse process is carried out. This step is important as the ANN will learn just the normalized values.

Therefore, the time series known values will be transformed into a patterns set, depending on the k inputs nodes of a particular ANN. Each pattern consists in:

- "k" inputs values, that correspond to "k" normalized previous values of period t: $a_{t-1}, a_{t-2}, \ldots, a_{t-k}$.
- One output value, that corresponds to normalized time series value of period t.

This patterns set will be used to train and validate each ANN generated during the GA execution. So patterns set will be split into two subsets, train and validation. The first x% from the total patterns set will generate the train patterns subset, and the validation subset will be obtained from the rest of the complete patterns set.

B. ANN design carried out with GA

The problem of designing ANN could be seen as a search problem into the space of all possible ANN. Moreover, that search can be done by a GA [16] using exploitation and exploration. Therefore there are three crucial issues: i) the solution's space, what information of the net is previously set and what is included into the chromosome; ii) how each solution is codified into a chromosome, i.e. encoding schema; iii) and what is being looked for, translated into the fitness function.

In this approach it has been chosen Multilayer Perceptron (MLP) as computational model due to its approximation capability and inside this group, Full Connected MLP with only a hidden layer and Backpropagation (BP) as learning algorithm, according to [17]. This is because ANN with only one hidden layer are faster to be trained and easier to work than two or more hidden layer MLP.

As it was mentioned before the design of the ANN will be done by setting the parameter values of the ANN. In the case of MLP with only one hidden layer and BP, the parameters are: number of inputs nodes, number of hidden neurons, number of output neurons, (only one, it is set by the forecasting problem), which is the connection patterns (how the nodes are connected), and the whole set of connection weights (implemented by the seed used to initialize the connection weights as it will be explained later).

For our approach [5] to design ANN to forecast time series, a Direct Encoding Schema for Full Connected MLP has been considered. For this Direct Encoding Scheme the information placed into the chromosome will be: two decimal digits, i.e. two genes, to codify the number of inputs nodes (i); other two for the number of hidden nodes (h); two more for the learning factor (α); and the last ten genes for the initialization seed value of the connection weights (s) (seed in SNNS [18] is *"long int"* type, that is why it has been used 10 genes (decimal digits) to encode "s"). This way, the values of "i", "h", "α" and "s" are obtained from the chromosome as it can be seen in eq (2):

$$\begin{aligned} &\text{Chrom}: \\ &g_{i1}\, g_{i2}\, g_{h1}\, g_{h2}\, g_{\alpha 1}\, g_{\alpha 2}\, g_{s1}\, g_{s2}\, g_{s3}\, g_{s4}\, g_{s5}\, g_{s6}\, g_{s7}\, g_{s8}\, g_{s9}\, g_{s10} \\ &0 \le g_{xy} \le 9,\ x = i, h, \alpha,\ y = 1..10 \\ &i = \max\_inputs \cdot ((g_{i1} \cdot 10 + g_{i2})/100) \\ &h = \max\_hiddens \cdot ((g_{h1} \cdot 10 + g_{h2})/100) \\ &\alpha = ((g_{\alpha 1} \cdot 10 + g_{\alpha 2}))/100 \\ &s = g_{s1}\, g_{s2}\, g_{s3}\, g_{s4}\, g_{s5}\, g_{s6}\, g_{s7}\, g_{s8}\, g_{s9}\, g_{s10} \end{aligned} \quad (2)$$

The search process (GA) will consist of the following steps:

1. A randomly generated population, i.e a set of randomly generated chromosomes, is obtained.
2. The phenotypes (i.e. ANN architectures) and fitness value of each individual of the actual generation is

obtained. To obtain the phenotype and fitness value associated to a chromosome:

2.1 The phenotype (i.e. ANN) of an individual of the actual generation is first obtained (using SNNS).

2.2 The train and validation patterns subsets are obtained for this individual, depending on the number of inputs nodes of each net, as it was commented above.

2.3 The net is trained with BP (using SNNS binary tools [18] again). The architecture (topology and connections weights set) of the net when the validation error (i.e. error for validation patterns subset) is minimum during the training process is saved (i.e. early stopping). So this architecture is the final phenotype of the individual.Once that fitness value for whole population is already done, the GA operators as Elitism, Selection, Crossover and Mutation are applied in order to generate the population of the next generation, i.e. set of chromosomes.

3. Once that fitness value for whole population is already done, the GA operators as Elitism, Selection, Crossover and Mutation are applied in order to generate the population of the next generation, i.e. set of chromosomes.
4. The steps 2 and 3 are iteratively executed till a maximum number of generations are reached.

The fitness value for each individual will be then the minimum validation error during the learning process (training of ANN topology), as it can be seen in eq (3):

$$fitness\ function = minimum\ validation\ error \quad (3)$$

The parameters for the GA are: population size, 50; maximum number of generations, 100; percentage of the best individual that stay unchangeable to the next generation (percentage of elitism), 10%; crossover: parents are split in one point randomly selected, offspring are the mixed of each part from parents; mutation probability will be one divided between the length of the chromosome (*1/length_chrom = 1/16 ≈ 0.7*), and it will be carried out for each gen of the chromosome.

Once that GA reaches the last generation, the best individual (i.e. ANN) from the last generation is used to forecast the future (and unknown) time series values.

The future unknown values (a_{t+1}) will be forecasted one by one using the k previous known values (a_t, a_{t-1}, ..., a_{t-k}). To forecast several consecutive values (a_{t+1}, $a_{t+2,...}$), every time a new value is forecasted, it will be included in order into the previous known values set of the time series and used to forecast the next one.

C. Estimation Distribution Algorithm (EDA)

Estimation of Distribution Algorithms (EDA), sometimes called Probabilistic Model-Building Genetic Algorithms (PMBGA), are an outgrowth of genetic algorithms. In a genetic algorithm, a population of candidate solutions to a problem is maintained as part of the search for an optimum solution. This population is typically represented explicitly as an array of objects. Depending on the specifics of the GA, the objects might be bit strings, vectors of real numbers, LISP style S expressions or some custom representation. In an EDA, this explicit representation of the population is replaced with a probability distribution over the choices available at each position in the vector that represents a population member.

For example, if the population is represented by bit strings of length 4, the EDA for the populations would be a single vector of four probabilities (p1, p2, p3, p4) where each p is the probability of that position being a 1 or any other possible value. Using this probability vector it is possible to create an arbitrary number of candidate solutions.

In evolutionary computation new candidate solutions are often generated by combining and modifying existing solutions in a stochastic way. The underlying probability distribution of new solutions over the space of possible solutions is usually not explicitly specified. In EDAs a population may be approximated with a probability distribution and new candidate solutions can be obtained by sampling this distribution. This may have several advantages, including avoiding premature convergence and being a more compact representation. Better-known EDA include:

- UMDA: Univariate Marginal Distribution Algorithm, no dependencies between variables, univariate distributions.
- MIMIC: Mutual Information Maxminization for Input Clustering, variables with order one dependencies, probability distribution.
- EBNA: Estimation of Bayesian Networks Algorithm, no restriction on the numbers of dependencies.

So far it has been observed, maybe because the estimation distribution algorithm is a recent technique (came into use just a few years ago); it has only been carried out few hybrids studies (i.e. ANN+EDA) applied to classification domains [19]. The bibliography does not show any article or study of hybrid systems using ANN and estimation of distribution algorithms to perform time series forecasting.

Due to all this, here it is proposed a new hybrid method using advantages of EDA and ANN to forecast all kind of time series. Besides, as it is a totally automatic method, it is no necessary the user to be an expert at all.

D. ANN design carried out with EDA

As it was commented above, there are different kinds

of EDA, but for our approach it has been chosen UMDA, with no dependencies between variables, according to [19], due to it is faster and easier to work with them. Here we have the process for a general EDA:

1. D_0 <= Generate and evaluate an initial population of solutions
2. Repeat for k = 0, 1, 2, …, until a stopping criterion is met
   a. D_k^{Sel} <= Select a subset of solutions from D_l
   b. $P_k(X)$ <= Estimate the empirical probability distribution of D_k^{Sel}
   c. D_k^{New} <= Sample solutions from $p_k(X)$
   d. Evaluate solutions in D_k^{New}
   e. D_{k+1} <= Create the new population with solutions from D_k^{New} and D_k

Considering the previous schema of an EDA process, here there is an example of how a binary UMDA would work, applying this simplest version (i.e. with no dependencies between variables) to a trivial example of optimization. Suppose we are trying to maximize the function OneMax defined in a search space of 6 dimensions. So we are trying to maximize the function $h(X) = \sum_{i=1}^{6} Xi$ with X_i=0,1.

First step will be to obtain a random first population and all they are evaluated as it can be seen at figure 1.

	X_1	X_2	X_3	X_4	X_5	X_6	$h(x)$
1	1	0	1	0	1	0	3
2	0	1	0	0	1	0	2
3	0	0	0	1	0	0	1
4	1	1	1	0	0	1	4
5	0	0	0	0	0	1	1
6	1	1	0	0	1	1	4
7	0	1	1	1	1	1	5
8	0	0	0	1	0	0	1
9	1	1	0	1	0	0	3
10	1	0	1	0	0	0	2
11	1	0	0	1	1	1	4
12	1	1	0	0	0	1	3
13	1	0	1	0	0	0	2
14	0	0	0	0	1	1	2
15	0	1	1	1	1	1	5
16	0	0	0	1	0	0	1
17	1	1	1	1	1	0	5
18	0	1	0	1	1	0	3
19	1	0	1	1	1	1	5
20	1	0	1	1	0	0	3

Fig. 1. Initial population D_0

In a second step, best x individuals (i.e. those with higher h(X) value) from first population (i.e. D_0) are selected. In this case we have selected best 10 individuals (i.e. D_0^{Se}) as it can be seen at fig 2.

	X_1	X_2	X_3	X_4	X_5	X_6
1	1	0	1	0	1	0
4	1	1	1	0	0	1
6	1	1	0	0	1	1
7	0	1	1	1	1	1
11	1	0	0	1	1	1
12	1	1	0	0	0	1
15	0	1	1	1	1	1
17	1	1	1	1	1	0
18	0	1	0	1	1	0
19	1	0	1	1	1	1

Fig. 2. D_0^{Se}, selected individuals from initial population

After this, it has to be expressed in an explicit way, using the joint probability distribution, the characteristics of those selected individuals. Using mathematical notation, as it can be seen in eq (4):

$$p_l(X) = p(X_1,...,X_6) = \prod_{i=1}^{6} p(X_i \mid D_0^{Se}) \quad (4)$$

So, it is only necessary six parameters to specify the model. Each parameter, $p(X_i \mid D_0^{Se})$ with i = 1,…,6, will be estimated from the set D_0^{Se} through their corresponding relative frequencies, $p(X_i = 1 \mid D_0^{Se})$. Next values are obtained from the parameters:

$$p(X_1 = 1 \mid D_0^{Se}) = 0.7 \qquad p(X_2 = 1 \mid D_0^{Se}) = 0.7$$
$$p(X_3 = 1 \mid D_0^{Se}) = 0.6 \qquad p(X_4 = 1 \mid D_0^{Se}) = 0.6$$
$$p(X_5 = 1 \mid D_0^{Se}) = 0.8 \qquad p(X_6 = 1 \mid D_0^{Se}) = 0.7$$

Sampling this joint probability distribution, $p_l(X)$, it is obtained a new population of individuals, D_k. In fig. 3 it can be observed the new 20 individuals (i.e. D_0^{Se} plus the ten new ones).

	X_1	X_2	X_3	X_4	X_5	X_6	$h(x)$
1	1	1	1	1	1	1	6
2	1	0	1	0	1	1	4
3	1	1	1	1	1	0	5
4	0	1	0	1	1	1	4
5	1	1	1	1	0	1	5
6	1	0	0	1	1	1	4
7	0	1	0	1	1	0	3
8	1	1	1	0	1	0	4
9	1	1	1	0	0	1	4
10	1	0	0	1	1	1	4
11	1	1	0	0	1	1	4
12	1	0	1	1	1	0	4
13	0	1	1	0	1	1	4
14	0	1	1	1	1	0	4
15	0	1	1	1	1	1	5
16	0	1	1	0	1	1	4
17	1	1	1	1	1	0	5
18	0	1	0	0	1	0	2
19	0	0	1	1	0	1	3
20	1	1	0	1	1	1	5

Fig. 3. Next population with new individuals, D_1

If we also have a look to the process for a general GA, it would be:

1. D_0 <= Generate and evaluate an initial population of solutions
2. Repeat for k = 0, 1, 2, …, until a stopping criterion is met
   a. $D_k^{Elitism}$ <= Select a subset of solutions from D_k
   b. $D_k^{Crossover}$ <= Apply crossover to solutions from D_k
   c. $D_k^{Mutation}$ <= Apply mutation to solutions from $D_k^{Crossover}$
   d. D_{k+1} <= Create the new population with solutions from $D_k^{Mutation}$ and $D_k^{Elitism}$
   e. Evaluate solutions in D_{k+1}

To apply EDA to our approach it was necessary to replace the GA, who is responsible of carrying out the global search into the hybrid system, for EDA.

It can be observed that the principal differences between GA and EDA consist of steps *"b"* and *"c"*, where in stead of carrying out crossover and mutation, it is estimated the empirical probability of each individual and sampling the solutions.

IV. Experimental Setup and Results

A. Time Series

Five time series will be used to evaluate our methods. These time series are named Passengers, Temperature, Dow-Jones, Quebec, and Mackey-Glass. Passengers time series has the information about the number of passengers of an international airline in thousands, measured monthly from January of 1949 till December of 1960, the source is Box & Jenkins (1976). Temperature time series shows the mean monthly of air temperature measured at Nottingham Castle from 1920 till 1939; in this case the source is O.D. Anderson (1976). Dow-Jones is about the monthly closings of the Dow-Jones industrial index from August of 1968 till August of 1981, the source is Hipel and Mcleod (1994). Quebec represents the number of births daily measured in Quebec from 1st of January of 1977 till 31 of December of 1978. And the last one called Mackey-Glass is based on the Mackey-Glass differential equation and is widely regarded as a benchmark for comparing the generalization ability of different methods. This series is a chaotic time series generated from a time-delay ordinary differential equation.

B. Experimental setup

The time series values have to be rescale, into the numerical range value [0,1], considering not only the known values, but the future values (those to be forecasted) [20].

So, the maximum and minimum limits for normalizing (*max4norm, min4norm* respectively) cannot be just the maximum (*max*) and minimum (*min*) known time series values. A margin from max and min has to be set if future values were higher or lower than known values already are. This margin will depend on another parameter (*Prct_inc*). In those cases in which the time series is stationary a *Prct_inc* of 10% will be enough, but when the time series is increasing or decreasing *Prct_inc* should be at least of 50%. As it could be forecasted new values for a time series that will rise of a fall, it is needed a enough big margin so the new values, obtained as output of ANN, can be into the numerical range [0,1]. This Equation (5) shows how are obtained max4norm and min4norm.

$$\begin{aligned} \text{max4norm} &= \text{max} + (\text{Prct\_inc} \cdot (\text{max} - \text{min})) \\ \text{min4norm} &= \text{min} - (\text{Prct\_inc} \cdot (\text{max} - \text{min})) \end{aligned} \quad (5)$$

C. GA versus EDA

Both ways to forecast time series, hybrid system with GA and with EDA, have been executed five times for each time series a total of 200 generations each one and the average result obtained for each time series has been calculated.

To evaluate the error for each method, forecasted values are compared with real values and two error formulas are used: MSE (mean squared error) and SMAPE (symmetric mean absolute percent error [4]); SMAPE has been used at NN3 and NN5 forecasting competitions. Results are shown in Tables I and II.

In Table I, it is shown the results obtained for Passengers, Temperature, Dow-Jones, Quebec and Mackey-Glass time series in generation number 100. In

this table, the columns will show: MSE and SMAPE error in forecasting (i.e. test set) for each time series.

In Table II, it is shown the results obtained for Passengers, Temperature, Dow-Jones, Quebec and Mackey-Glass time series in generation number 200. In this table, the columns will show: MSE and SMAPE error in forecasting (i.e. test set) for each time series. These errors (as it was commented before) are relative to the average of the five times experiments have been run, choosing each execution the best individual from the last generation of the GA or the EDA.

TABLE I
SMAPE AND MSE PASSENGERS, TEMPERATURE, DOW-JONES, QUEBEC AND MACKEY-GLASS WITH GA AND EDA FOR 100 GENERATIONS

100 Generations		GA	EDA
Passengers	SMAPE (%)	3.180	3.572
	MSE	0.00061	0.00069
Temperature	SMAPE (%)	4.308	3.978
	MSE	0.00358	0.00314
Dow-Jones	SMAPE (%)	6.662	6.812
	MSE	0.02150	0.02429
Quebec	SMAPE (%)	12.643	13.270
	MSE	0.02540	0.02550
Mackey-Glass	SMAPE (%)	8.672	6.266
	MSE	0.00363	0.00186

TABLE II
SMAPE AND MSE PASSENGERS, TEMPERATURE, DOW-JONES, QUEBEC AND MACKEY-GLASS WITH GA AND EDA FOR 200 GENERATIONS

200 Generations		GA	EDA
Passengers	SMAPE (%)	3.148	3.218
	MSE	0.00058	0.00061
Temperature	SMAPE (%)	4.239	3.936
	MSE	0.00347	0.00324
Dow-Jones	SMAPE (%)	6.307	5.263
	MSE	0.01993	0.01459
Quebec	SMAPE (%)	12.121	10.964
	MSE	0.02149	0.01726
Mackey-Glass	SMAPE (%)	8.042	1.798
	MSE	0.00309	0.00016

As it can be observed in Table I, applying EDA in stead of GA to these time series doesn't achieve better forecasting (MSE/SMAPE) in many of the time series when the experiment has been run only 100 generations. Just Mackey-Glass and Temperature obtain a better SMAPE result with EDA and in Mackey-Glass case; the improvement is about 2.4%.

But if the experiment is run over 200 generations, it can be seen in Table II an important improvement in almost all the time series, where EDA obtain a better forecast than GA in four of the five time series. Only in Passengers time series GA is still better than EDA although both results are really close. A special consideration has to be taken on Mackey-Glass time series where the SMAPE error result is 1.798%, being the values forecasted by our method almost identical to the real time series values.

The better results obtained by EDA compared with GA after having run the experiments 200 generations could be explained because EDA, unlike GA, avoid premature convergence. So when GA gets stock in a local minimum value, EDA don't, and they keep on looking for a better result if they are given more time, in this case, more generations to look for.

To have a better idea about the forecast of each time series and how close to the real values each forecasting method was, a graph for each time series showing all the forecasts done by each method will be carried out. Figure 4 shows Passengers forecast for each method. Figure 5 shows Temperature, Figure 6 shows Dow-Jones, Figure 7 shows Quebec and Figure 8 Mackey-Glass. A zoom of Quebec will be done in Figure 9. A zoom of Mackey-Glass will be done in Figure 10. TS(tr+val) represents the known values of the time series with which we have worked and TS(test) shows the future unknown real values that have to be forecasted (i.e. test subset).

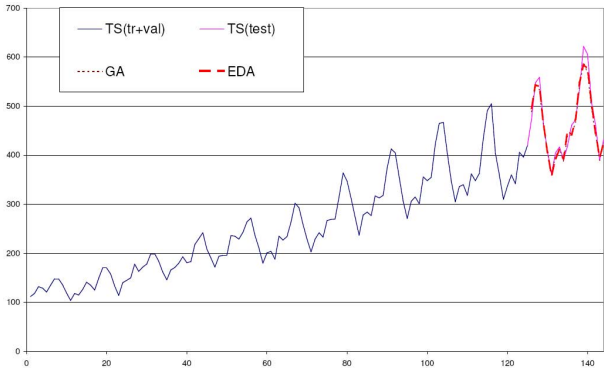

Fig. 4. Passengers forecast with GA and EDA

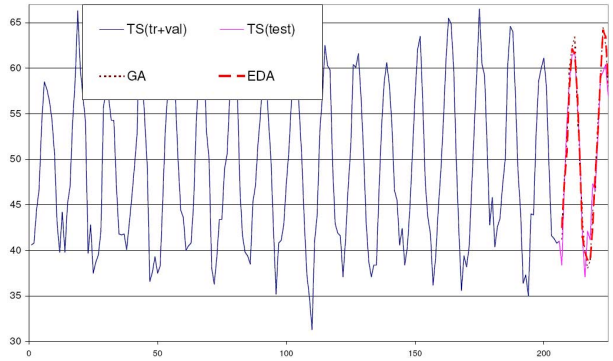

Fig. 5. Temperature forecast with GA and EDA

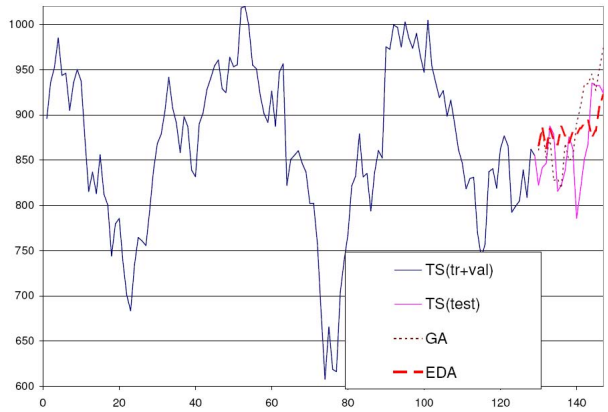

Fig. 6. Dow-Jones forecast with GA and EDA

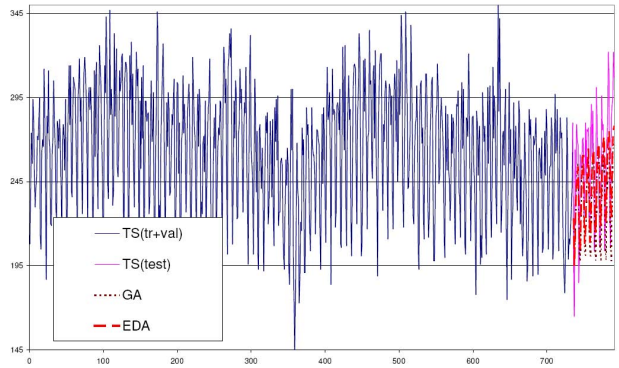

Fig. 7. Quebec forecast with GA and EDA

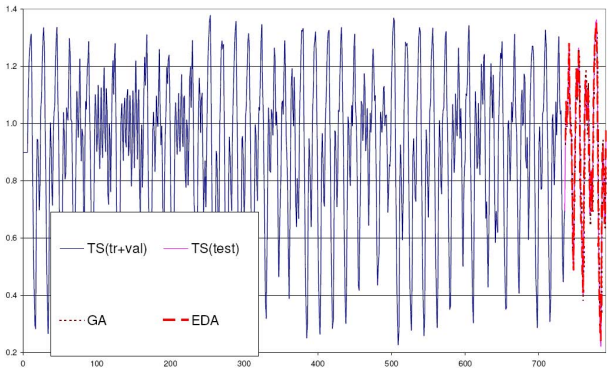

Fig. 8. Mackey-Glass forecast with GA and EDA

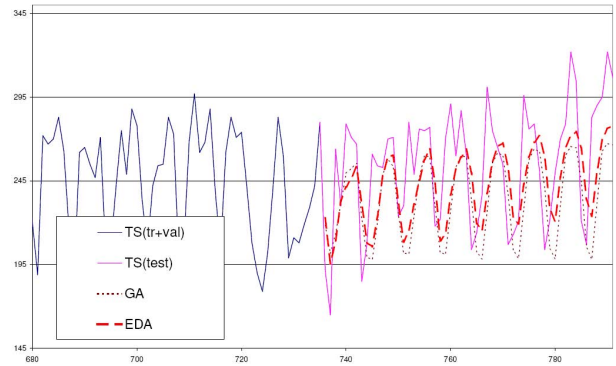

Fig. 9. Zoom of Quebec forecast with GA and EDA

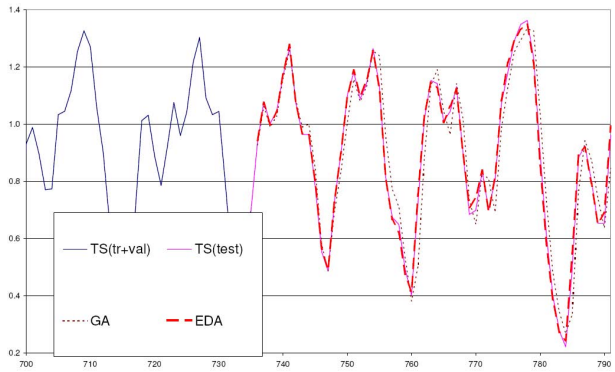

Fig. 10. Zoom of Mackey-Glass forecast with GA and EDA

V. CONCLUSIONS AND FUTURE WORKS

The results of the experiments disclose that using EDA in stead of GA obtain different results, depending on the number of generations they are executed. With only 100 generations, EDA results don't improve too much compared to GA. But if 200 generations are reached, it can be observed a significant improvement, some times with a gain of 1.2% in the results, as it happens in Dow-Jones time series. Special attention should be paid to Mackey-Glass results, where a 6.2% improvement is obtained.

As it was commented before, obtaining better results by EDA than with GA after having run the experiments 200 generations could be explained because EDA, unlike GA, avoid premature convergence. So when GA gets stock in a local minimum value, EDA don't, and they keep on looking for a better result if they are given more time, in this case, more generations to look for.

As it is a totally automatic method, it will not be necessary any previous knowledge from the user so the user will not have to be an expert in time series, statistics, mathematics or computational intelligence. The user just have to give the time series he wants to forecast and the number of future elements he wants to be forecasted to the system; and this method will give these forecasted values as result to the user.

This approach was presented as an automatic method to design ANN in NN5 competition, getting the 6th position with SMAPE error of 21.9% in Neural Nets and Computational Intelligence methods (NNCI) ranking, for the reduced dataset (i.e. 11 time series). Best result on NNCI ranking and reduced data was a SMAPE error of 19.0%. Autobox tool [21] based on Box-Jenkins forecasting methodology got an error of 23.9%.

Future works with additional time series, with similar characteristics to Quebec, Mackey-Glass will allow us to obtain more accurate conclusions about the effect of using EDA in stead of GA. On the other hand, it would be really interesting to use EDA with dependencies between its variables like MIMIC (i.e. variables with order one dependencies) or even "tree" EDA, with no restriction on the numbers of dependencies.

Other interesting future works are: to use *"cross validation"* into the GA for a better evaluation of each individual; using sparsely connected ANN to try to improve the forecast and getting an accurate system.

ACKNOWLEDGMENT

The research reported here has been supported by the Spanish Ministry of Science and Innovation under project TRA2007-67374-C02-02.
The author wants to thank specially Ramon Sagarna for introducing him into the subject.

REFERENCES

[1] Spyros G. Makridakis, Steven C. Wheelwright, Rob J Hyndman. Forecasting: Methods and Applications.
[2] Ian Nunn, Tony White."The application of antigenic search techniques to time series forecasting". Genetic and Evolutionary Computation Conference 2005. ISBN:1-59593-010-8.
[3] Paulo Cortez, José Machado, José Neves. "An evolutionary artificial neural network time series forecasting system". IASTED 1996.
[4] Time Series Forecastig Competition for Neural Networks and Computational Intelligence. http://www.neural-forecasting-competition.com. Accessed on October 2008.
[5] J. Peralta, G. Gutierrez, A. Sanchis, "ADANN: Automatic Design of Artificial Neural Networks". ARC-FEC 2008 (GECCO 2008). ISBN 978-1-60558-131-6.
[6] Zhang, G.; Patuwo, B.E. & Hu, M.Y. Forecasting with artificial neural networks: The state of the art International Journal of Forecasting, 1998, 14, 35-62.
[7] Haykin, S. Simon & Schuster (ed.) Neural Networks. A Comprehensive Foundation Prentice Hall, 1999.
[8] Crone, S. F. Stepwise Selection of Artificial Neural Networks Models for Time Series Prediction Journal of Intelligent Systems, Department of Management Science Lancaster University Management School Lancaster, United Kingdom, 2005.
[9] T. Ash. Dynamic Node Creation in Backpropagation Networks ICS Report 8901, The Institute for Cognitive Science, University of California, San Diego (Saiensu-sh, 1988), 1988.
[10] D.B. Fogel, Fogel L.J. and Porto V.W. Evolving Neural Network, Biological Cybernetics, 63, 487-493, 1990.
[11] Gruau F. "Genetic Synthesis of Boolean Neural Networks with a Cell Rewriting Developmental Process". Proceedings of COGANN-92 International Workshop on Combinations of Genetic Algorithms and Neural Networks, pp. 55-74, IEEE Computer Society Press, 1992.
[12] Yao, X. and Lin, Y. A new evolutionary system for evolving artificial neural networks, Transactions on Neural Networks, 8(3): 694-713, 1997.
[13] Kitano, H.: Designing Neural Networks using Genetic Algorithms with Graph Generation System, Complex Systems, 4, 461-476, 1990.
[14] Ajith Abraham, Meta-Learning Evolutionary Artificial Neural Networks, Neurocomputing Journal, Elsevier Science, Netherlands, Vol. 56c, pp. 1-38, 2004.
[15] X. Yao (1993), "A review of evolutionary artificial neural networks", International Journal of Intelligent Systems, 8(4):539-567.
[16] Fogel, D. Evolutionary Computation: Toward a New Philosophy of Machine Intelligence. Wiley-IEEE Press, 1998.
[17] G. Cybenko. Approximation by superposition of a sigmoidal function. Mathematics of Control, Signals and Systems, 2, 303-314, 1989.
[18] Prof. Dr. Andreas Zell,, WSI Computer Science Department, Computer Arquitecture, Software, Artificial Neural Networks http://www-ra.informatik.uni-tuebingen.de/SNNS/
[19] C. Cotta, E. Alba, R. Sagarna, P. Larrañaga. 2002. Adjusting Weights in Artificial Neural Networks using Evolutionary Algorithms. In Estimation of Distribution Algorithms. A new tool for Evolutionary Computation, P. Larrañaga and J.A. Lozano (eds.). Kluwer Academic Publishers, pp. 357-373.
[20] S. Crone (2005) Stepwise Selection of Artificial Neural Network Models for Time Series Prediction, Journal of Intelligent Systems, Vol. 14, No. 2-3, 2005, pp. 99-122.
[21] Automatic Forecasting Systems. http://www.autobox.com. Accessed on October 2008.