Shuffle Design to Improve Time Series Forecasting Accuracy

Juan Peralta, German Gutierrez, Araceli Sanchis

***Abstract*—In this work new improvements from a previous approach of an Automatic Design of Artificial Neural Networks applied to forecast time series is tackled. The automatic process to design Artificial Neural Networks is carried out by a Genetic Algorithm. These improvements, in order to get an accurate forecasting, are related with: to shuffle train and test patterns obtained from time series values and improving the fitness function during the global learning process (i.e. Genetic Algorithm) using a new patterns set called validation apart of the two used till the moment (i.e. train and test). The object of this study is to try to improve the final forecasting getting an accurate system. Results of the Artificial Neural Networks got by our system to forecast a set of famous time series are shown.**

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. For real applications, i.e. time series, this is equivalent to forecast unknown numerical values in time *t, t+1,…, t+n*, from the known past values in time *t-1, t-2,..., t-k*. 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].

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. On the other hand, one new improvement will be used to beat the forecasting.

The research presented in this paper was motivated by NN3 (2007) and NN5 (2008) competition [5].

Two different steps, as it was explained in an earlier work [6], 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 this approach it has been chosen Full Connected Multilayer Perceptron (MLP) with one hidden layer, as computational model and Backpropagation (BP) as learning algorithm (developed with Stuttgart Neural Network Simulator (SNNS) tool [7]).

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 set carrying out a search process performed by a Genetic Algorithm (GA) to set the architecture (the topology and connection weights set) of the net.

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 (ADANN) designs ANN with GA to forecast time series and two new improvements of the system will be also explained. 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 reveals the full consideration of ANN (as a data driven learning machine) into forecasting theory [8].

Before using an ANN to forecast, it has to be designed, i.e. establishing the suitable value for each freedom degree of the ANN [9] (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 [8] Zhang, Patuwo and Hu present a "state of the art" of ANN into forecasting task, in [10] is proposed an "extensive modeling approach" to review several designs of ANN.

In order to obtain a single ANN to forecast time series values, an initial step has to be done with the original values from 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.

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

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).

978-1-4244-2959-2/09/$25.00 ©

series (t-1, t-2,.., t-k) to obtain a function as it is shown in (1):

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

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 test each ANN generated during the GA execution. So patterns set will be split into two subsets, train and test. Initially, the complete patterns set are ordered into the same way the time series is. But, as it will be explained in section III, there will be two options to generate the train and test patterns subsets from the total patterns set.

First way is explained as follows: the first x% from the total patterns set will generate the train patterns subset, and the test subset will be obtained from the rest of the complete patterns set. The second way will be called "*shuffle*" and it will consist on obtaining the train patterns subset from the x% of the total patterns set but in a random way, the test subset will be also randomly generated from the rest of the total patterns set left after having obtained the train subset. The validation subset will be the future (and unknown) time series values that want to be forecasted.

If hand design of ANN is carried out, several topologies (i.e. different number of inputs nodes and number of hidden neurons in only one hidden layer), with different learning rates are trained. For each of them, train and test error are obtained, and one with better generalization capability (i.e. less test error and a good train error) is selected to generate forecasted values (i.e. validation subset).

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 [11,12,13,14,15].

Some of them use Direct Encoding Schemata (DES) [11,12], others use Indirect Encoding Schemata (IES) [13,14,15]. 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). Ajith Abraham [16] shows an automatic framework for optimization ANN in an adaptive way, and Xin Yao et. al. [17] try to spell out the future trends of the field.

III. ANN DESIGN WITH GA (ADANN)

A. ADANN

The problem of designing ANN could be seen as a search problem into the space of all possible ANN. And that search can be done by a GA [18] 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 looking for, translated into the fitness function.

In this approach it has been chosen Multilayer Perceptron (MLP) as computational model due to its approximation capability, according to [19], and inside this group, Full Connected MLP with only a hidden layer and Backpropagation (BP) as learning algorithm. This is because ANN with only one hidden layer are faster to be trained and easier to work with them.

As it was mentioned before the design of the ANN will be done 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, 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 [6] to design ANN to forecast Time Series, a Direct Encoding Scheme 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, are used 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 value of the seed of initialization if the connection weights (s) (seed in SNNS [7] is of *"long int"* type, that is why it has been used 10 genes (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):

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

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 test patterns subsets are obtained for this individual, depending on the number of inputs nodes of each net, as it was said above.
   2.3 The net is trained with BP (using SNNS binary tools [7]). The architecture (topology and connections weights set) of the net when the test error (i.e. error for test patterns subset) is minimum during the training process is saved (i.e. early stopping). So this architecture is the final phenotype of the individual.
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 1 and 2 are iteratively executed till a maximum number of generations are reached.

A schema of the whole search process can be seen at fig. 1.

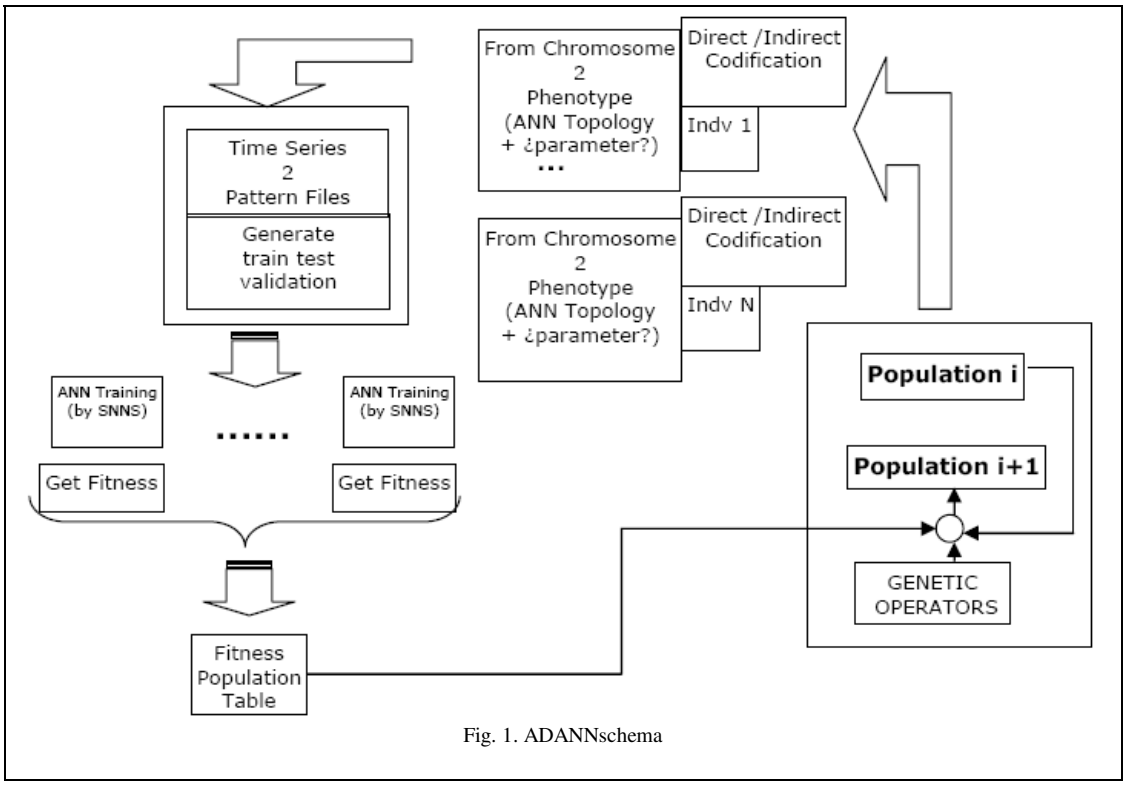

Fig. 1. ADANNschema

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

$$\text{fitness function} = \text{minimum test error} \quad (3)$$

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 method will be a random place to cut the parents chromosomes and join the first part of the father with the second part of the mother and on the other hand the second part of the father joined with the first part of the mother so two new offspring will be obtained ; mutation probability will be one divided between the length of the chromosome (1/length_chrom = 0.7), and it will be carried out for each gen of the chromosome.

Once that GA reaches the last generation, one ANN obtained from the best individuals of the last generation is used to forecast the future (and unknown) time series values (i.e. validation set), using the architecture saved when test error is minimum for that individual.

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}). So, to forecast several consecutive values (a_{t+1}, $a_{t+2,\ldots}$) every time a new value is forecasted, it will be included in order into the known previous values set and used to forecast the next one.

A. *Shuffle improvement*

"Shuffle" refers to an improvement of the system. Up till now, and as it was explained before, to train the ANN obtained from the chromosome, its total patterns set is generated and split into two different subsets, train and test patterns subset. Train patterns subset will be used to train the ANN and the test patterns subset will be used to estimate the generalization capability and obtaining its fitness value (stopping the training process before overtraining is reached). Till now, train and test sets are obtained in a sequentially manner, i.e. first part (initial 70%) of the time series was used to generate train patterns set and last part (last 30%) of time series was used to generate test patterns set (Fig 2).

Nevertheless, in this improvement the process of splitting the patterns set will consist of obtaining train and test sets in a random way. So it will let different parts of the time series to train the ANN and also different parts of the time series to test the ANN (Fig 3), in order to obtain better generalization ability.

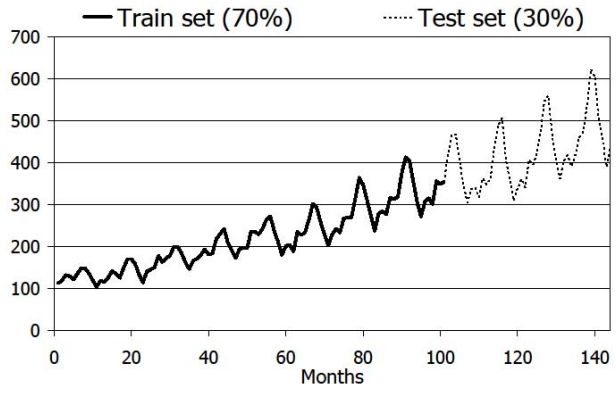

Fig. 2. Passengers: train and test patterns sequentially obtained

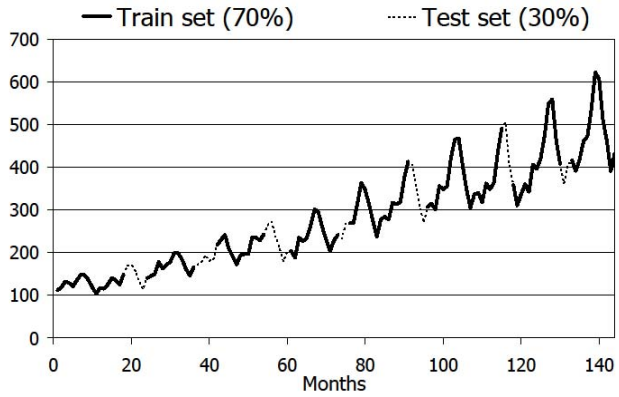

Fig. 3. Passengers: train and test patterns randomly obtained

But considering that now we will have three different patterns subsets (i.e. train, test and a new one called validation) due to the another new improvement called *"fitness improvement"* that will be seen below, the whole patterns set will have to be split into these three different patterns subsets.

So depending on if the new improvement (shuffle) is used or not two different options could appear. First option will be taking first x% from the whole patterns set to generate the train patterns subset, next y% to generate the test patterns subset and the validation subset will be obtained from the rest of the complete patterns set (Fig 4).

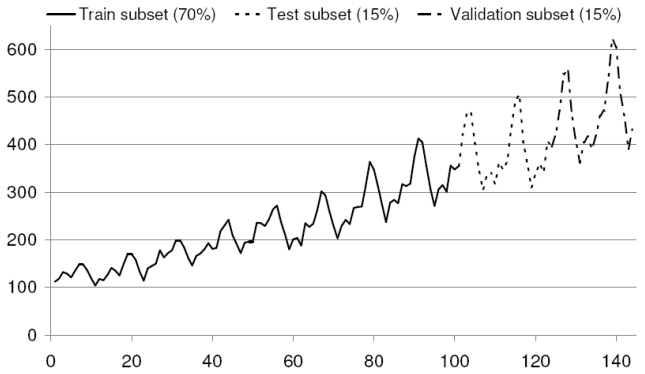

Fig. 4. Passengers: train, test and validation patterns sequentially obtained

The second manner, i.e. applying *"shuffle"*, will consist on obtaining the train patterns subset from the x% of the total patterns set but in a random way, the y% of the total patterns set also in a random way for the test patterns subset and the validation subset will be the rest of the total patterns set left (Fig 5).

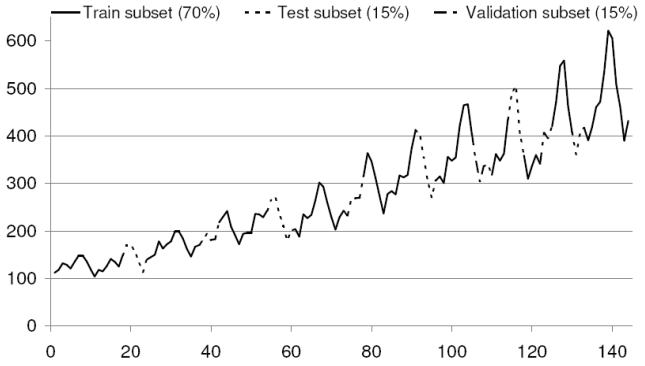

Fig. 5. Passengers: train, test and validation patterns randomly obtained

B. *FITNESS improvements*

As it was commented above, two different subsets have been used during the learning process, train subset (used to modify the connection weights along learning process) and test subset (to avoid the overfitting problem during the learning process and to get the architecture used to obtain the fitness value for each individual).

In previous works [6], the fitness function was just the minimum test error got by the architecture (i.e. topology + connection weights) along the learning process.

But a new improvement will be carried out in this approach. It will be used an extra subset, neither for learning process nor to establish the phenotype, but just to obtain

fitness value as a linear combination of test error and the error for this extra patterns subset. That extra subset will be known as validation subset, and now the future and unknown values to be forecasted will be called forecasting subset. So, it will be tried to demonstrate if using this new validation subset into fitness function will help to get an accurate result, i.e. better forecasting.

To carry out this task and as it was said before now three subsets (train, test and validation) will be used. Train and test subsets will have the same functionality than before and the new validation subset will be used to complement the fitness function.

This new validation set will be obtained as it was done in section II from the total patterns set. The total percent that will be used of the total patterns set to generate these three subsets (train, test a validation) will now depend on if there is any previous information of the time series. For NN3 times series there was no information about the time series data, just the values. So the train test and validation patterns subsets were, respectively, 70%, 15% and 15% of the whole patterns set. However for NN5 time series the information that represents each value is known (daily amount of money took from different cash machines in England measured during two years). Then the measure has an inherent periodicity of days, weeks, months or years, etc (related with social/human events or actions). If there is any periodicity in data, the input-output mapping will be learned better if that periodicity is not disrupted, i.e. train data will correspond to an integer of weeks, months or years. Moreover, the same could be said about generalization capability estimation with the test subset. That estimation will be appropriately representative if test and validation data also corresponds to an integer of weeks, months or years.

As NN5 time series represent the daily amount of money took from different cash machines in England measured during two years then the patterns set will be split into one year (i.e. 50 %) for train, 25% for test and 25 % for validation.

Besides, as it was explained above, there will be two options to generate the train, test and validation patterns subsets from the total patterns set.

First option will be taking first x% from the total patterns set to generate the train patterns subset, another y% to generate the test patterns subset and the validation subset will be obtained from the rest of the complete patterns set. As it can be seen in figure 3.

Second option, *"shuffle"* will consist on obtaining the train patterns subset from the x% of the total patterns set but in a random way, the y% of the total patterns set also in a random way for the test patterns subset and the validation subset will be the rest of the total patterns set left. As it can be seen in figure 4.

Once that these three subsets have been generated, the learning process explained above will be slightly altered. Again the net will be trained with BP and the architecture of the net when the test error is minimum along the training process will be saved. But now error in validation subset is obtained after the training process is finished. So, validation set will be totally unknown for the ANN evaluation. Also now, not only the test error will be used as the fitness value, the new fitness function will consist in the linear combination of the test and validation error. It will be decided how much weight want to be assigned to the test (α) and validation (β) errors. In this work, three different coefficients pairs have been chosen for the test and validation error respectively for the experiments, 1.0 and 0, 0.4 and 0.6, and finally, 0 and 1.0. eq (4):

$$\text{fitness function} = (\alpha \cdot \text{test error}) + (\beta \cdot \text{valid error}) \quad (4)$$

IV. Experimental results and setup

A. Time Series

As it was commented before, the research presented in this paper was motivated by NN3 (2007) and NN5 (2008) time series competition [4]. NN3 and NN5 forecasting competition consisted of two sets of time series: reduced dataset (11 time series) and full data set (reduced dataset plus 100 more time series). The objective of the competition was to develop a unique computational intelligence method to use it to design an ANN for each time series considered (from reduced or full data set). So, for each time series a specific ANN is obtained applying always the same automatic method described above.

For NN3 time series there is no previous knowledge about what each value of the time series means, as well as each time series had from 126 to 150 known values and the following 18 values had to be forecasted. On the other hand, for NN5 the competitors knew that the data of the time series represented the daily amount of money took from different cash machines in England measured during two years (i.e. 730 elements) and the following 56 values had to be forecasted.

However, other known seven time series (three similar to NN3 time series and other four similar to NN5 time series) will be used to evaluate our method. These time series are named Passengers, Temperature, Dow-Jones, Quebec, Mackey-Glass, IBM and Jokulsa. 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. On the other hand Jokulsa represents the mean daily flow at Jokulsa Eystri River from 1st January of 1972 till 31st December of 1974 and IBM shows the stock level data that IBM Company had from 1st of January of 1980 till the 8th of October of 1992.

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 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. Shuffle improvement results

Both ways to generate train and test subsets are evaluated into the system, sequentially and randomly (*shuffle*), so results (i.e. forecasted values) obtained for time series are checked. To evaluate *shuffle* improvement, *fitness* improvement is not integrated into the method yet. 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 is used at NN3 and NN5 forecasting competitions. Results are shown in Tables I and II.

In Table I is shown the results obtained for Passengers, Temperature and Dow-Jones. These time series (as it was commented before) are similar to those presented in NN3 competition, they are small (only from 126 to 150 elements) monthly measured time series. For these three time series the system has been run eight times for each time series, four without applying shuffle (represented as *"without shuffle x"* into the table) and four applying it (i.e. *"with shuffle y"*).

In these tables, the columns will show: MSE and SMAPE error in forecasting (i.e. validation set) for each time series. These errors are relative to the best individual from the last generation of the GA.

TABLE I
SMAPE AND MSE PASSENGERS, TEMPERATURE AND DOW-JONES WITH AND WITHOUT SHUFFLE

		Without Shuffle 1	Without Shuffle 2	Without Shuffle 3	Without Shuffle 4	With Shuffle 1	With Shuffle 2	With Shuffle 3	With Shuffle 4
Passengers	SMAPE (%)	4.094	2.609	2.505	3.053	13.870	8.753	9.556	3.817
	MSE	0.000075	0.00046	0.00046	0.00059	0.00708	0.00290	0.00376	0.0071
Temperature	SMAPE (%)	3.328	3.798	4.845	6.038	3.991	4.884	3.997	4.893
	MSE	0.00213	0.00307	0.00406	0.00675	0.00305	0.00428	0.00364	0.00440
Dow-Jones	SMAPE (%)	4.760	5.512	4.721	3.132	8.179	7.805	5.838	4.936
	MSE	0.0106	0.0143	0.0097	0.0043	0.0295	0.0262	0.0150	0.0119

As it can be observed applying shuffle improvement to these time series doesn't achieve better forecasting (MSE/SMAPE) in any of the time series.

It could be explained as follows: having so less elements in the time series (from 126 to 150) the less number of patterns it can be obtained. Also, if train and test (and validation if it is necessary) pattern subsets obtained are split in a random way, then all the patterns used to adjust the connection's weights does not correspond to consecutives time series values. So the relationship between inputs and output could be harder to learn if there are few patterns for learning and they are not consecutive (i.e. mixing up the training and test patterns).

On the other hand, the same experiment (applying shuffle or not) was also carried out with Quebec, Mackey-Glass, IBM and Jokulsa (time series similar to NN5 competition time series), larger than previous ones (now about 730 elements) and daily measured time series during two years. Results for these time series are shown in Table II. Again the columns represent the results (SMAPE/MSE) got *"without shuffle"* and *"with shuffle"* respectively for each time series.

It can be observed in Table II that now applying *shuffle* to these time series gets a better result. It stands out specially Mackey-Glass that gets an error of SMAPE (*1.818 %*) *with shuffle*, which means that the forecasted values are almost the same than the real values. Only IBM

time series doesn't improve the forecast but it doesn't get worse (16.0% *without shuffle*, 16.2% *with shuffle*).

TABLE II
SMAPE AND MSE QUEBEC, MACKEY-GLASS, IBM AND JOKULSA WITH AND WITHOUT SHUFFLE

		Without Shuffle	With Shuffle
Quebec	SMAPE (%)	12.121	9.218
	MSE	0.02149	0.01312
Mackey	SMAPE (%)	8.672	1.818
	MSE	0.00363	0.00016
IBM	SMAPE (%)	16.0	16.200
	MSE	0.12059	0.12298
Jokulsa	SMAPE (%)	18.903	14.589
	MSE	0.00176	0.00099

A. *Fitness improvement results*

When fitness improvement is applied, as it was commented above, not only the test error will be used as the fitness value. The new fitness function will consist of a linear combination of the test and validation errors, so the coefficients of test and validation error are new parameters.

In this work, as it can be observed in Table III and Table IV, three different weights are established to the test and validation errors to obtain the fitness value: first 1.0 for test and 0 for validation (i.e. what has been done in experiments till the moment), second 0.4 for test and 0.6 for validation and last one, 0 for test and 1.0 for validation (fitness value is just the validation error). Again experiments for Passengers, Temperature and Dow-Jones can be seen in Table III and Quebec, Mackey-Glass, IBM and Jokulsa in Table IV.

TABLE III
SMAPE AND MSE PASSENGERS, TEMPERATURE AND DOW-JONES WITH DIFFERENT FITNESS FUNCTIONS

Test coef – Valid coef		1.0 - 0	0.4 - 0.6	0 - 1.0
Passengers	SMAPE(%)	3.053	3.858	5.376
	MSE	0.000595	0.000779	0.001364
Temperature	SMAPE(%)	3.798	3.901	4.053
	MSE	0.003077	0.003014	0.00336
Dow-Jones	SMAPE(%)	4.76	4.628	5.477
	MSE	0.01064	0.00988	0.01331

TABLE IV
SMAPE AND MSE QUEBEC, MACKEY-GLASS, IBM AND JOKULSA WITH DIFFERENT FITNESS FUNCTIONS

Test coef – Valid coef		1.0 - 0	0.4 – 0.6	0 – 1.0
Quebec	SMAPE(%)	12.121	11.272	11.847
	MSE	0.02149	0.01938	0.02069
Mackey	SMAPE(%)	8.672	7.977	1.394
	MSE	0.00363	0.00296	0.00008
IBM	SMAPE(%)	16.00	22.429	21.643
	MSE	0.12059	0.21822	0.20613
Jokulsa	SMAPE(%)	18.903	18.445	25.001
	MSE	0.00176	0.00167	0.00323

It can be seen that for small time series like Passengers in Table III, there is no profit using the new fitness functions, even more, the results are worse.

For larger time series (Table IV), only one of them (Mackey-Glass) there has been an improvement in the results. For coefficients 0 and 1 for test and validation errors respectively, the SMAPE decrease till 1.384%. For Quebec there is no change in results obtained. For IBM and Jokulsa, results are even worse.

B. *Shuffle+ Fitness improvement results*

After having checked that for large time series (i.e. Quebec, Mackey-Glass, IBM and Jokulsa) shuffle improvement get a better results in general and new fitness functions do the same in just one case, a mix of both methods will be carried out to see if the results of the forecasts of the large time series can be improved. To do this, it will be shown in Table V the SMAPE and MSE error for Quebec, Mackey-Glass, IBM and Jokulsa using the three different fitness functions (the old one and the new two ones) but applying also shuffle (i.e. mix of new fitness functions and shuffle at the same time).

TABLE V
SMAPE AND MSE QUEBEC, MACKEY-GLASS, IBM AND JOKULSA WITH DIFFERENT FITNESS FUNCTIONS AND SHUFFLE

Test coef – Valid coef		1.0 - 0	0.4 – 0.6	0 – 1.0
Quebec	SMAPE(%)	9.218	15.879	9.531
	MSE	0.01312	0.03541	0.01478
Mackey	SMAPE(%)	1.818	13.124	6.166
	MSE	0.000016	0.00999	0.00178
IBM	SMAPE(%)	16.200	21.967	22.129
	MSE	0.12298	0.21048	0.21365
Jokulsa	SMAPE(%)	14.589	29.685	17.741
	MSE	0.00099	0.00567	0.00154

As it can be observed in Table V, applying both methods (shuffle and new fitness functions) does not improve the results of the forecasts, even more; it gets worse results in general for all time series. For example, applying shuffle in Quebec with the "old" fitness function (coefficient 1.0 for test and 0 for validation) and shuffle gets a better forecast (1.818%) than using the two new ones (13.124% and 6.166% respectively) with shuffle.

V. CONCLUSIONS AND FUTURE WORKS

The results of the experiments disclose that using shuffle for small time series (i.e. time series with a few number of elements, as Passengers, Temperature and Dow-Jones) does not improve the forecasting of the ANN obtained. But for larger time series (i.e. time series with about 700 elements, as Quebec, Mackey-Glass, IBM and Jokulsa) in which shuffle improvement has been applied forecasting was enhanced.

An issue arises at this point: how the positive/negative effect of shuffle depends on the number of time series elements (i.e. size of training/test subsets). Different experiments with a time series like Quebec or Mackey-Glass but using different number of elements could clarify this question. Then it could be define what a small time

series is and what is not so using shuffle could be an advantage.

Although two new fitness functions have been used to be compared on the forecasting results, it has been demonstrated that the best fitness function that could be used till the moment is still the old/previous one, i.e. fitness value is just test error.

Anyway, new fitness functions or new ways to evaluate each ANN obtained during GA execution will be designed to try to improve the forecasting results in future works.

On the other hand it has been also demonstrated that mixing these two new improvements (i.e. shuffle and new fitness functions) doesn't get a better result (forecasting result in our case).

This approach without shuffle improvement and error test as fitness value, 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 and IBM (as stationary, chaotic and financial time series respectively) will allow us to obtain more accurate conclusions about the effect of shuffle improvement, and new fitness function. 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; and exchanging the test and validation pattern subset in each generation so any possible overfitting would be avoided.

ACKNOWLEDGMENT

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

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] Hyndman, R.J. (n.d.) Time Series Data Library, http://www.robhyndman.info/TSDL. Accessed on October 2008.

[6] J. Peralta, G. Gutierrez, A. Sanchis, "ADANN: Automatic Design of Artificial Neural Networks". ARC-FEC 2008 (GECCO 2008). ISBN 978-1-60558-131-6.

[7] Prof. Dr. Andreas Zell,, WSI Computer Science Department, Computer Arquitecture, Software, Artificial Neural Networks http://www-ra.informatik.uni-tuebingen.de/SNNS/

[8] 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.

[9] Haykin, S. Simon & Schuster (ed.) Neural Networks. A Comprehensive Foundation Prentice Hall, 1999.

[10] 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.

[11] 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.

[12] D.B. Fogel, Fogel L.J. and Porto V.W. Evolving Neural Network, Biological Cybernetics, 63, 487-493, 1990.

[13] 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.

[14] Yao, X. and Lin, Y. A new evolutionary system for evolving artificial neural networks, Transactions on Neural Networks, 8(3): 694-713, 1997.

[15] Kitano, H.: Designing Neural Networks using Genetic Algorithms with Graph Generation System, Complex Systems, 4, 461-476, 1990.

[16] Ajith Abraham, Meta-Learning Evolutionary Artificial Neural Networks, Neurocomputing Journal, Elsevier Science, Netherlands, Vol. 56c, pp. 1-38, 2004.

[17] X. Yao (1993), "A review of evolutionary artificial neural networks", International Journal of Intelligent Systems, 8(4):539-567.

[18] Fogel, D. Evolutionary Computation: Toward a New Philosophy of Machine Intelligence. Wiley-IEEE Press, 1998.

[19] G. Cybenko. Approximation by superposition of a sigmoidal function. Mathematics of Control, Signals and Systems, 2, 303-314, 1989.

[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.