Journal of Applied Mathematics and Physics, 2020, 8, 1047-1065
https://www.scirp.org/journal/jamp
ISSN Online: 2327-4379
ISSN Print: 2327-4352

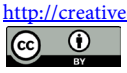
Neuroevolution Strategy for Time Series Prediction

George Naskos[1], Konstantinos Goulianas[1], Athanasios Margaris[2]

[1]Department of Information and Electronic Engineering, International Hellenic University, Thessaloniki, Greece
[2]Department of Digital Systems, University of Thessaly, Larissa, Greece
Email: georgenascos@yahoo.com, gouliana@it.teithe.gr

How to cite this paper: Naskos, G., Goulianas, K. and Margaris, A. (2020) Neuroevolution Strategy for Time Series Prediction. *Journal of Applied Mathematics and Physics*, **8**, 1047-1065.
https://doi.org/10.4236/jamp.2020.86082

Received: April 12, 2020
Accepted: June 2, 2020
Published: June 5, 2020



Abstract

Optimization is a concept, a process, and a method that all people use on a daily basis to solve their problems. The source of many optimization methods for many scientists has been the nature itself and the mechanisms that exist in it. Neural networks, inspired by the neurons of the human brain, have gained a great deal of recognition in recent years and provide solutions to everyday problems. Evolutionary algorithms are known for their efficiency and speed, in problems where the optimal solution is found in a huge number of possible solutions and they are also known for their simplicity, because their implementation does not require the use of complex mathematics. The combination of these two techniques is called neuroevolution. The purpose of the research is to combine and improve existing neuroevolution architectures, to solve time series problems. In this research, we propose a new improved strategy for such a system. As well as comparing the performance of our system with an already existing system, competing with it on five different datasets. Based on the final results and a combination of statistical results, we conclude that our system manages to perform much better than the existing system in all five datasets.

Keywords

Neuroevolution, Neural Networks, Evolutionary Algorithms, Time Series

1. Introduction

The great potential of neural networks has been proven repeatedly in the past years by many studies, as they are applicable to real problems, even in our daily lives [1]. Evolutionary algorithms (EA) are used in optimization problems, with the ability to cope better with problems where the optimal solution is found

among a chaotic set of solutions [2]. Generally speaking, the larger the range of possible solutions to a problem, the more efficient the EA associated with it. The combination of the two approaches results in the neuroevolution (NE) [3] [4]. This approach describes a neural network equipped by an evolutionary optimization algorithm. Bearing in mind the benefits of evolutionary algorithms, they are proven to be useful in deep learning [5], because of the enormous size of such a network that making it ideal for EA use. This allows the achievement of the same or better results in less time than a conventional optimization algorithm. It is also possible to build an EA system to find out which neural network architecture is appropriate for the problem. Then, the proposed network can be used, using a conventional optimization method [6]. In this study we will investigate the use of NE in time series problems [7]. This is a type of problem that has an increased degree of difficulty, having an additional factor, and more specifically, the time. Five datasets will be used, with some of their data to be derived from real problems and some other data, from simulations. Five different experiments will be performed on each of the datasets and then our own system will be compared with the memetic cooperative neuroevolution (MCNE) system [8]. In both systems, the same datasets have been used for the experiments, which are split into training/validation/testing in the same way [9]. Our system is generally based on their own MCNE. At the same time, proposals will be made for different methods and parameters in evolutionary algorithms, where they can be used not only in NE but generally in optimization problems. Finally, the results of our system will be compared to the upgraded version of their own MCNE system.

2. Artificial Neural Network

Neural networks try to replicate the biological function of the human brain. Each neuron receives an excitation and, after processing, sends its own excitation to the other neurons to which it is connected. There are many types of neural networks, one of which is multilayer perceptron (MLP) [10].

It is well known and it can give very good results in a wide range of problems. This is the kind of neural network that will be used in the following experiments. An MLP is described by one or more layers of neurons. Each neuron has inputs and each input is a data characteristic of the situation. Each entry, it associated with a weight value. This value is a number that indicates how important this feature is for that particular neuron. There is an extra weight called bias. The bias moves the activation function to the x-axis so that it fits better depending on the input data. The weights are multiplied by their associated input values and the resulted products are summed each other. The resulted value goes through an activation function whose return value is the output signal of the specific neuron in **Figure 1**. The neuron acquires the ability to learn through an optimization method that appropriately adapts the weights of the neuron.

The main advantages of neural networks are:

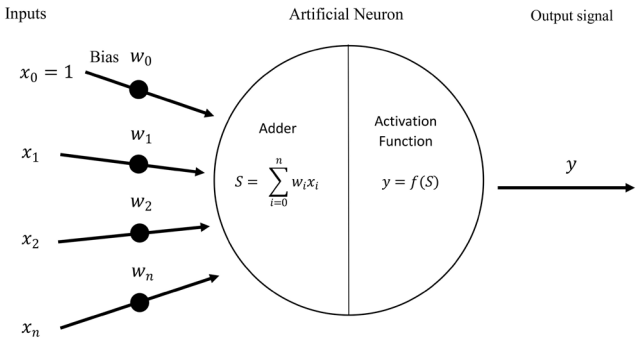


Figure 1. Artificial neuron model.

The ability to learn. More specifically, through their training, they can learn to solve a problem by adjusting to their weights.

Their most powerful feature is generalization. It is defined as their ability to produce results for data they have not seen in their training.

There is a large set of optimization methods based on neural network which help to solve classification and regression problems. In addition, given an entity's past values, they can predict its future values.

Their major disadvantages are:

There is no general rule for the most appropriate use of the activation function, as well as for the optimum number of layers and the number of neurons in each layer. Thus, it takes multiple efforts and continuous testing, to determine the best configuration and specify a network that produces satisfactory results.

Because of its nature, there is no satisfactory control over the network as well as useful information that someone can derive by looking at the weights and connections of the network. In general, the real function of a neural network is that of a black box, for which the only thing we know is the input data and its results.

3. Evolutionary Algorithms

The creation of evolutionary algorithms (EA) is the idea that the biological process of evolution could be the basis for the development of optimization algorithms [1] [2]. The theory associated with these algorithms is based on natural selection and genetic change, with a population of individuals competing with each other to choose the best. Recombining the best among them will produce better offspring. If the process continues repeatedly, the members of the resulting population will be much better with respect to the initial members of this set. In a real problem, the members of the population, are a candidate solution to the problem.

EA are divided into several stages:

1) Initially, a population is created by individuals (**Algorithm 1**-Step 1). Each individual contains chromosomes and the chromosomes contain genes. Genes are the values that have their characteristics, which are the ones that will determine how good the solution is to the problem. This number is the fitness value

of the individual.

2) Once a fitness value has been found for each individual (**Algorithm 1**-Step 2), a new set of individuals is created, selected on the basis of a method based on their fitness (**Algorithm 1**-Step 3). It has to be noted, that in the general case, the best ones are selected, but the worst ones are also given the opportunity to be selected, because they may contain genetic code that is useful for the next step.

3) This stage is the recombination of the total from the previous stage. So, these people will fertilize the next generation (**Algorithm 1**-Step 4). Recombination is the most basic way of population growth. In most cases, from the set produced by the selection method there are two people that are selected. Therefore, by recombining these two individuals, we can produce two even better offspring. The fact that evolutionary algorithms, are targeted search algorithm, and not just a random search algorithms is based to this recombination procedure.

4) The young offspring will go through a method of random mutation of their genes. This happens in order to enable the population to surpass any local optimum as it grows (See **Algorithm 1**-Step 5).

5) Finally, the offspring are integrated into the population, fitness is calculated for each new individual in the population (**Algorithm 1**-Step 6) and the process starts from the beginning.

The algorithm terminates when the specified number of iterations or a computational limit has been reached.

The advantage of EA is that they can solve quickly difficult problems. The greater the number of possible optimal solutions, the more efficient they are in terms of both speed and result. Their main advantages are:

Most conventional methods are stiff or sometimes inappropriate for several problems, due to their need for complex mathematics. The use of such mathematics is indifferent to EA, which makes them suitable for a wide range of problems.

It is one of the few methods that simultaneously explores the search space and takes advantage of the information already processed. Thus, random searches make a good exploration of the space and the search takes advantage of the information.

```
1) BEGIN
2)      Step 1: INITIALISE population with random candidate solutions;
3)      Step 2: EVALUATE each candidate;
4)      REPEAT UNTIL (TERMINATION CONDITION is satisfied) DO
5)          Step 3: SELECT parents;
6)          Step 4: RECOMBINE pears of parents;
7)          Step 5: MUTATE the resulting offspring;
8)          Step 6: EVALUATE new candidates;
9)      DONE
10) END
```

Algorithm 1. Pseudocode of evolutionary algorithm.

Another major advantage is their easy parallelism, a potential that is scarce in other competing methods. Parallelism results, in better results in less time.

Finally, they can be combined with other methods. Although the power of EA is high, there are some cases of problems in which other methods produce very good results, due to their specialization in the problem. This is a result of the great flexibility of EA.

As with any method there are disadvantages:

The first thing anyone should do to solve a problem with EA is to encode the problem in such a way that EA can process it. However, this can sometimes be difficult depending on the nature of the problem. Specifically, this encoding is a process that will take time for one to understand and feel comfortable with.

The next step is to be able to set and implement a goal for EA. This goal is to guide the population of the algorithm to the optimal solution. Therefore, this stage greatly influences finding the optimal solution and in turn requires time and familiarity with it.

4. Proposed Neuroevolution Architecture

Thanks to the great flexibility that EA possess, they are able to combine with other methods. However, neural networks are needed to solve machine learning problems. This results in the combination of EA and neural networks. Neural networks alone cannot be trained, they need to use an optimization algorithm. One of the best known is backpropagation (BP). In NE [3] [4] the EA take the place of the optimization algorithm. The good news is that it is no longer necessary to use sophisticated mathematics to train and optimize the network. There are also many ways to customize and control during training, as are significantly reduced the effort and experimentation needed to find the right network. For example, in very large networks, in deep learning, training times are greatly reduced and consistently good results are produced.

4.1. Existing Study

There are several studies in the NE industry. One of the most recent that deals with time series problems can be found in [7]. An upgraded system to solve such problems is proposed in [8]. In practice, it combines EA with conventional algorithms, in order to achieve better results. Another study that uses the same system, but different strategies is described in [9], with very good results. Having in mind the best results from [9], that were compared to ours and based on an overview of the MCNE strategy described in [8], we were able to produce better results at about the same cost. We have also used the same datasets with [11] [12] [13] [14] [15] from the most recent study [9] to allow a meaningful comparison of the systems.

4.2. Our System

The encoding of the neural network differs in our method from the present study [8], since this is one of the most important pieces of EA and has a great

impact on the good results of the algorithm. Our system puts a subpopulation, that is, a set of neural networks, which implies that the subpopulation individuals are neural networks, while in [8] each neuron is also a subpopulation and its atoms are neurons with different weights. As a result, in the process of evolution, when the worst of the population is replaced by the bests, only the neurons actually change, while in proposed method the whole neural network changes. Meanwhile, the fitness of both systems is calculated in the same way and the overall rationale of the system is the same, so that the results can be compared between the two systems. Among the three methods cited in [9], MCNE [8] is closer to our system and the comparison will be performed between these two systems.

The implementation of the proposed system is not based on a library or on an available neural simulator, but everything has been created from scratch. This decision was made on the grounds that in order to do proper research there must be complete control over the system, a feature that generally is not supported by third-party software. Of course during the implementation, we used the well-known techniques for both EA and neural networks with some customizations. The advantage of this implementation is that it has many parameters that they can easily be controlled, so that the system can adapt and solve the problem in the best possible way. This system was able to give better results across all datasets [11] [12] [13] [14] [15].

In a more detailed description the following methods for EA have been used.

Regarding the selection process, the tournament method has shown far better results than roulette. It seems to help in rapid population growth, while at the same time is able to keep the population in balance in case of a high likelihood of mutation. The tournament has only one parameter that controls how many people will compete against each other. In this way it is possible to control how possible is the selection of a non-good individual. However, there may be some cases in which the use of fitness is not adequate to decide which one to choose, due to the generality of fitness. One solution is to create an extra value for each person. So, we don't have to mess with actual fitness since it is calculated every epoch right after fitness is calculated. This feature is called the "percentage of fitness" because it is a percentage-based fitness. This enables the creation of many methods for its calculation. The selection method can be based on this value instead of fitness, since the logic of fitness calculation remains the same, but at the same time, we can select people with different strategies. In the next paragraph we propose three methods of calculating this value.

1) We called the first method, the rank method, because every person will get its value based on the worst and the best of the current population. In Equation (1) we consider the worst to be 0% and the best 100%.

$$Percent = 100 \cdot \frac{c - min}{max - min} \tag{1}$$

where

c is the fitness of the person;

min is the smallest fitness in the population;

max is the greatest fitness in the population.

2) The current best, calculates the percent based on the best person in the current population. Thus, the range of the percentage is [0, best fitness], and therefore, in order to get a value of 0% the value of fitness should be zero. This one is less stringent with the not-so-good people of the population compared with the rank method.

3) The last method requires the maximum value calculated via the fitness, which is not always available. The range of the percentage is [0, total best fitness] and it is used in problems where the fitness is not available or there is not a good way for its calculation.

The crossover is used for the recombination method with only two parents, producing two offspring. The parameter in this method determines how many genes the crossover will take. The smaller the parameter number, the better the knitting of the genes. The parameter helps to better control the growth.

In the mutation method, it has to be determined how many chances each gene has to mutate. Regarding this method, there are two adjustments that have been implemented. When the mutation is done, it doesn't just get a random value through a range, but a value emerged via the addition or the subtraction of a value from the old one. This makes growth smoother, without killing good people due to a big value change. Also, as the epochs pass, the likelihood of mutation increases. So, the population lets to grow on its own and then when it sticks to local optimum, it is directed to move forward with the high mutation potential. We change the probability based on the epochs with the linear function. At this point, it can be used any function the user wants. However, the experiments show that there was a little change in the results with more complicated functions than linear.

Another important method for growth, as well as for the end results, is the parallel growth with subpopulation. We did this by starting different threads for each population. An interesting feature of the proposed method is the implementation of a method called the "migration". From time to time, a set of the best person from every sub-population is created. It is this group that will replace the worst persons of every subpopulation **Figure 2**. As a consequence, the overall performance and growth are greatly increased, because some subpopulations may not have enough good persons, in such cased the migration method is used to fix this problem. The downside to this method is when it is given to the system to generate many subpopulations, and therefore many threads. Specifically, when the computer processor cannot support so many threads, it requires more time to complete, although it is executed in parallel due to threads creation.

The proposed system as well as the system described in [9] use hybrid EA. When a conventional algorithm is executed at certain times for better development then the EA are called hybrid [16].

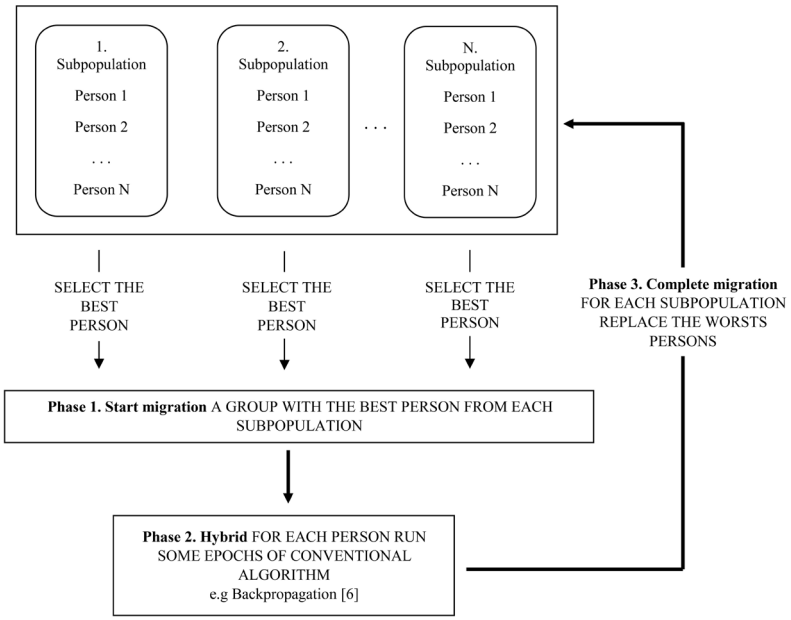


Figure 2. Detailed showing migration and hybrid methods simultaneously.

In the NE, the conventional algorithm is an optimization method for neural networks. In our case it is BP [6]. At a certain number of EA epochs, there are some BP epochs. In our system this is done immediately before migration is complete. Specifically, once the group is destined for migration, these individuals in the group execute some BP epochs **Figure 2**. Without the hybrid, our system would produce good results, but it could not overcome some of the problems, with the results of [9]. With the hybrid it was able to overcome these problems and the best results found in [9].

Specifically, for BP, its simple form was used [6]. The difference is in the behavior of a BP parameter and more specifically, the learning rate. In the proposed method, an adapted learning rate was used. So, the algorithm starts with a relatively large learning rate and during the training process its value progressively decreases. This behavior is due to the fact that for small iteration numbers, namely after the beginning of the training process, the large variation in the values of the neuron weights is not a major issue. On the contrary, it helps in faster growth. However, as the number of epochs increases and the optimum solution is approached, smaller changes in the weight values are required in order for the system to reach the optimum solution as close as possible.

Another feature of our system is the ability to develop EA with a variable neural network, given two parameters. The first of these parameters controls the range of the number of layers, while the other one controls the range of number of neurons per layer. Therefore, the EA in the creation phase of the first population, construct different random neural networks, based on the two previous parameters.

This process creates individuals with a variable set of genes, which implies that a different method is needed for recombination and mutation, simply to

support individuals with a variable set of genes. There are many implementations for EA with variable number of genes. The problems we have encountered are that these implementations require the addition or removal of random N genes in the individual, as well as other actions such as the transfer of a number of genes from one position to another during the population mutation phase [17]. This task is impossible for our system: in our case, a "person" is a neural network and his genes are all the weights of the network; therefore all we can do is to add or subtract neurons only to the hidden layers of the network. So, the above mutation method can be applied to our system but with some limitations. Also, if we consider that a set of N genes is a neuron, then we cannot transfer this neuron to another layer safely, as the neurons of the layer intended to go may not have the same number of inputs as the one we transport, so that a valid network remains after the transport. The recombination we have proposed and implemented in the system consists of one function, which is a map between two one-dimensional arrays A and B, each having a different size. Each array is an individual and each array element is a gene. This map shows, given a position in A, what should be the corresponding position in Array B. Of course, all the positions of the two arrays cannot be fully matched. This function can be easily combined with previous fixed size recombination methods. Given a position in Array A, the corresponding position in Array B is given by Equation (2).

$$P_B = \frac{P_A \cdot L_B}{L_A} \tag{2}$$

where:

P_B is the position of B we are looking for;

P_A is the position of A that we know;

L_B is the size of array B;

L_A is the size of array A.

One of the major problems of conventional neural networks is overtraining. This means that the network loses the ability to generalize and specializes only in some cases. It is a destructive case of training that is usually solved by defining a set of data called a validation set. All of this is used by the system during training, not to train the network, but to be able to monitor whether it loses its generality.

Our system supports this, as it provides the ability to interrupt training or finish normally completing all epochs and selecting the best person from all epochs based on the validation set. Our experiments generally show that the system achieves the best validation error relatively close to the end of the epochs. This indicates that the system does not lose generalization easily and that the use of validation set is only necessary in a few problems (see **Figure 3**).

5. Experiments and Analysis

During the simulation and the experimentation stage, five different datasets were used and more specifically, the following:

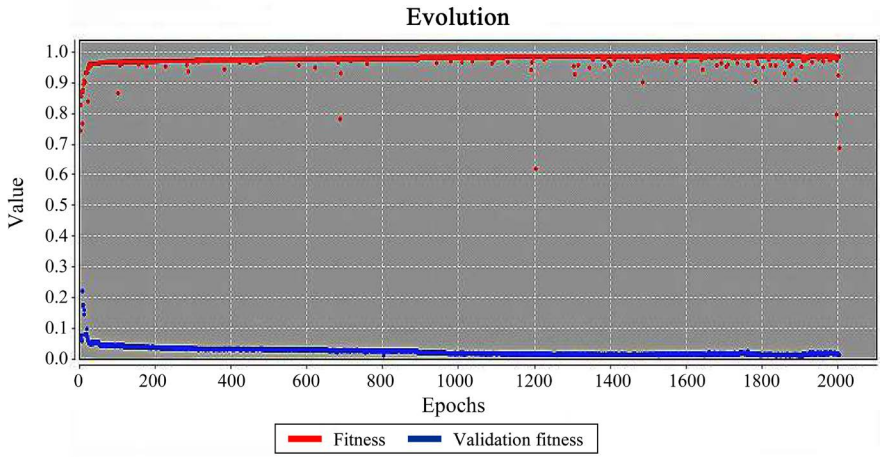


Figure 3. The diagram shows that fitness validation decreases over time. This shows the continuous improvement of the network, without losing its generality.

1) Sunspot [12]. The magnetic field of the Sun makes a periodic motion and each period lasts about 11 years. At the end of each period the poles of the sun have changed their position with each other. It has been observed that during this time period of eleven years, dark spots are formed on the surface of the sun, due to the change in their polarity. This phenomenon greatly affects our solar system and the climate on Earth. The proposed system is configured to predict the number of spots during these periods. It has been observed that this number is difficult to predict, as there are unexpected changes affecting the number of spots.

2) Mackey-Glass [14]. The situation is associated with a set of functions that produce diverse waveforms, which apparently have limitless or "chaotic" solutions. More specifically, these results are related to dynamic breathing and hematopoietic diseases. Essentially, some chronic or even acute diseases have been observed that some of their symptoms exhibit a periodicity and attempt to correlate mathematical functions in this study to predict the periodicity of the symptoms.

3) Lazer [15]. This data set is composed of data recorded using a remote infrared laser, produced for benchmark use, which has been used in many competitions. Its data is low dimensional, non-linear and constant time series.

4) Lorenz [13]. It is made up of data from observations of various hydrodynamic flows that exhibit general periodicity, but over time it appears volatility in its data. It is another natural observation, so it contains the element of the unpredictable.

5) Taiwan Trading Index Exchange (TWI Exchange) [11]. Simple recording of the Taiwanese exchange rate within a given time, this dataset is quite different from the others, as its values range in a much smaller range.

We use the already split format of the datasets, with 60%/20%/20% data, training/validation/testing respectively, where they are taken from their own system (https://github.com/gary-wong-fiji/Meme-Collection-SEQ).

5.1. Experiments Plan

In order to be able to properly compare the two systems, the datasets [11] [12] [13] [14] [15] from the research [9] had to be used for simulation, just as they divided them into their subsets. Therefore, the error is measured using the same root mean square error method (RMSE). The same method is the one that guides the evolution of subpopulations. The best person for each run is selected based on the fitness of the individual in the validation set. The C cost of each experiment is calculated from Equation (3).

$$C = \left(\sum^{n+1} m \cdot t \right) + \left(\sum^{r} z \right) \tag{3}$$

where:

n the epochs of the evolutionary algorithm;

m the crowd of people in the population;

t the multitude of subpopulations;

r the number of times that BP algorithm will be executed (see Equation (5));

z the number of epochs the BP algorithm will execute each time.

$$z = p \cdot n \tag{4}$$

$$r = \frac{n}{z} \tag{5}$$

There is a parameter N that is increased one by one to take into account fitness calculation when initializing the system. Another parameter is the so-called "percentage" p, that defines two things:

1) The number of epochs of the evolutionary algorithm, that the BP algorithm execute.

2) The number of epochs of the BP algorithm.

The motivation for using a variable value for the p parameter, was to keep a balance between how often and for how many epochs BP will executed. For example, if the p parameter increases its value, the BP algorithm will execute for more epochs, but less often.

5.2. Parameters of the Experiments

The experiments were divided into four types. The formulas have some common parameters **Table 1** and differ **Table 2** with respect to some others, whereby the final conclusions will be drawn on the overall picture of the system as well as its efficiency. All four types were applied to all datasets and thirty system executions were performed for each type. This is due to the fact that the evolutionary algorithms by they own nature are characterized by the property of randomness to a large extent. The conclusions are produced via statistical methods applied to the results emerged from all executions.

The parameters described as common, have been selected after many repeated system tests on the datasets [11] [12] [13] [14] [15]. Observing the results of the tests as well as how the system reacts to the changes in the values of the parameters, we came to the following values **Table 1**.

Table 1. Common parameters, for all test types.

Parameter	Value
Epochs	2000
Population size	15
Selection method	Tournament $K = 3$ (Size of tournament group)
Recombination method	Random $K = 5$ (Crossover break size)
Mutation method	Simple random (Add or subtract a random value)
Value range, for variable learning rate	[0.3, 0.00001]
Range of values for the probability of mutation	[200, 80]
Range of values for the mutation value	[−1, 1]
Neural network size	A hidden layer with 9 neurons A neuron in the output layer
Activation function, for hidden layer neurons	Logistic sigmoid [18]
Network Output Function	Ground-relu [19] (except for the Lorenz dataset [13], where it has the linear)

Table 2. Different parameters.

		Parameters	
		percentage (P)	Threads
Experiment type	A	0.1	5
	B	0.3	5
	C	0.1	10

In more detail, the greater the number of epochs, the better, but the epochs are costly in time. So after a point the performance/cost ratio will be very small and unprofitable.

The size of the population helps the system to make great leaps in improvement, from the earliest epochs, due to the larger volume of genetic material in the population. And that, in turn, adds a lot of time cost. Experiments have shown that our system does not need a large population.

Due to the selection of the small population, the total selection method must be small. The break size for the recombination method is small enough to keep the knitting of the genes satisfactory because the neural networks we have chosen are small in size.

The learning rate for the conventional BP [6] method is also small because most progress must be made by the EA [2] and the conventional method has the auxiliary role.

The relatively high probability of mutation was chosen because through the experiments the system showed that it can withstand them. In an even higher probability of mutation, we would see a sharp drop in the course of the population, because it's best individuals are killed, the high probability of mutation destroys their genetic material.

Once our data is normalized, the range we selected for the mutation values is sufficient. In their research [9], they use five neurons in the hidden layer, we chose to have nine. In our test D, where the network has a variable size, in its hidden layer it can have from five to nine neurons. However, the results show that most neurons do not help much in our system for these datasets. Even when we tried it with a hundred neurons in the hidden layer it did not produce better results and made the execution of the program very slow.

The sigmoid function in the hidden layer and the ground-relu at the output of the network, are two simple but efficient methods and very fast in execution time. For the dataset Lorenz [13] we have the linear function at the output because in this dataset there are negative values and the ground-relu method does not produce negative values.

5.3. Extra Experiments

During simulation, another type of experiment, called the D experiment, was performed. With this type of experiment, we can draw conclusions regarding the effectiveness of variable evolution. This experiment has the same parameters as the previous types, except from the parameters descripted at **Table 3**.

To be able to understand the effectiveness of using a percentage method, the test type E was created. It is essentially just like the test D, but without the use of the percentage method. Finally, we have five types of problems A, B, C, D, E.

5.4. Results

After all executions were done, all data were collected from the system output and the following tables were generated (see **Table 4**). In these tables, the results of our system (A, B, C, D, E) are compared with the results of the existing study (Sequential, Concurrent, MCNE) [9]. As mentioned above, EA have the element of randomness to a large extent, and because of this, an additional column, namely the Mean error bias (MEB) has been added in **Table 4**. MEB column values are in the interval [−100, 100], indicating how close (in percentage) the Mean error is to the best error (Best error column). A value of 100% means that the average error is equal to the best error, while a value of −100% means that it is equal to the worst error. This is a very important column, since we can see if the value we got at the best error is a pretty likely value or we just got lucky. However, MEB cannot be compared from one test to another because they have different values in the maximum and minimum error. The type that MEB describes is Equation (6).

$$\mathrm{MEB} = -\left(2 \cdot \frac{\mu_\varepsilon - m_\varepsilon}{M_\varepsilon - m_\varepsilon} - 1\right) \cdot 100 \tag{6}$$

where:

μ_ε is the average error;

m_ε is the minimum error;

M_ε is the maximum error;

Table 3. Different parameters for the extra experiment D.

Parameter	Value
Percentage (P)	0.1
Threads	5
Recombination method	Variable size
Percent of fitness method	Current best ranked
Variable size neural network	Hidden layer: [5, 9] neurons A neuron in the output layer

Table 4. All tests results.

Santa Fe Laser [15]							
Test type	**Best error**	**Mean error**	**Mean error bias**	**Worst error**	**Cost**	**Cost per thread**	**Mean (best validation error epoch)**
A	0.0122094	0.0369875	5.63%	0.0647200	200,075	40,015	1619
B	0.0151571	0.0430133	−17.93%	0.0624001	195,075	39,015	1742
C	0.0138906	0.0323462	11.01%	0.0553674	500,150	50,015	1715
D	0.0186784	0.0606052	38.98%	0.1560946	200,075	40,015	1509
E	0.0203611	0.0656812	54.02%	0.2174987	200,075	40,015	1269
Sequential [9]	0.0571243	0.0695330	−65.25%	0.0721420	269,421	-	-
Concurrent [9]	0.0634781	0.0768557	−68.66%	0.0793412	121,200	-	-
MCNE [8] [9]	0.1471420	0.1949820	−33.44%	0.2188464	100,000	-	-
Lorenz [13]							
Test type	**Best error**	**Mean error**	**Mean error bias**	**Worst error**	**Cost**	**Cost per thread**	**Mean validation error epoch**
A	0.0027403	0.0052443	24.78%	0.0093983	200,075	40,015	1897
B	0.0022897	0.0096347	33.82%	0.0244856	195,075	39,015	1813
C	0.0019629	0.0038730	28.18%	0.0072823	500,150	50,015	1949
D	0.0022847	0.0144763	32.90%	0.0386242	200,075	40,015	1767
E	0.0037872	0.0146563	62.75%	0.0621459	200,075	40,015	1809
Sequential [9]	0.0713540	0.0731450	48.59%	0.0783210	260,668	-	-
Concurrent [9]	0.3214887	0.3445700	22.62%	0.3811421	121,200	-	-
MCNE [8] [9]	0.0747062	0.0753210	73.42%	0.0793321	100,000	-	-
Mackey Glass [14]							
Test type	**Best error**	**Mean error**	**Mean error bias**	**Worst error**	**Cost**	**Cost per thread**	**Mean validation error epoch**
A	0.0027723	0.0037160	45.77%	0.0062529	200,075	40,015	1880
B	0.0027492	0.0037760	14.47%	0.0051503	195,075	39,015	1864
C	0.0023960	0.0033838	11.41%	0.0046261	500,150	50,015	1966
D	0.0010170	0.0040382	6.24%	0.0074617	200,075	40,015	1814
E	0.0020943	0.0043915	36.02%	0.0092757	200,075	40,015	1873
Sequential [9]	0.0019264	0.0045463	−37.10%	0.0057482	271,031	-	-
Concurrent [9]	0.0032004	0.0059527	−56.75%	0.0067121	121,200	-	-
MCNE [8] [9]	0.0123215	0.0252556	−16.57%	0.0345122	100,000	-	-

Continued

Sunspot [12]

Test type	Best error	Mean error	Mean error bias	Worst error	Cost	Cost per thread	Mean validation error epoch
A	0.0077448	0.0100750	13.07%	0.0131057	200,075	40,015	1909
B	0.0083803	0.0109812	13.68%	0.0144062	195,075	39,015	1817
C	0.0063976	0.0077735	30.12%	0.0103354	500,150	50,015	1877
D	0.0078447	0.0128080	27.05%	0.0214516	200,075	40,015	1578
E	0.0077889	0.0134720	38.76%	0.0263481	200,075	40,015	1485
Sequential [9]	0.0107341	0.0127696	53.78%	0.0195412	205,039	-	-
Concurrent [9]	0.0146470	0.0193530	10.14%	0.0251210	121,200	-	-
MCNE [8] [9]	0.0246412	0.0478444	-9.27%	0.0671124	100,000	-	-

TWI Exchange [11]

Test type	Best error	Mean error	Mean error bias	Worst error	Cost	Cost per thread	Mean validation error epoch
A	0.0118479	0.0127522	48.92%	0.0153886	200,075	40,015	1658
B	0.0120645	0.0130102	42.49%	0.0153533	195,075	39,015	1576
C	0.0115597	0.0123720	46.37%	0.0145887	500,150	50,015	1701
D	0.0121878	0.0134124	53.20%	0.0174216	200,075	40,015	1465
E	0.0113766	0.0133122	40.42%	0.0178735	200,075	40,015	1604
Sequential [9]	0.0354120	0.0394227	−38.23%	0.0412148	272,318	-	-
Concurrent [9]	0.0363142	0.0397674	0.59%	0.0432614	121,200	-	-
MCNE [8] [9]	0.0745214	0.0852743	−28.59%	0.0912457	100,000	-	-

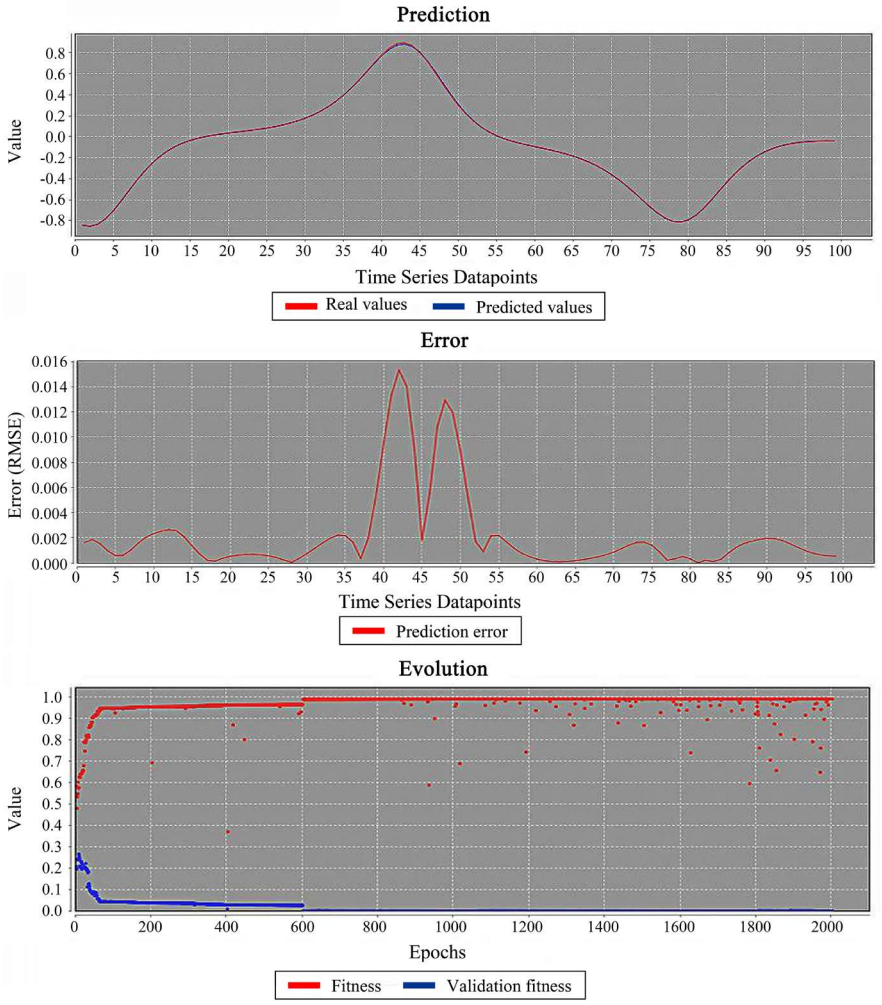
The Cost column is calculated by the type of cost mentioned above. Also, there is an additional "Cost per thread" column, because as the initial cost is calculated, it's like implying that the execution of the program is serial, while in fact it is a parallel task, because of the threads. The Mean (best validation error epoch) (MBVE) column is the average of the epochs, from which the best validation error occurred. It follows that the larger the MBVE, the less the system dependence on the validation error, in order to find the optimal network. All of the above explains, whether the validation dataset is useful for the system and therefore the validation error (**Figure 4**).

5.5. Initial Conclusions

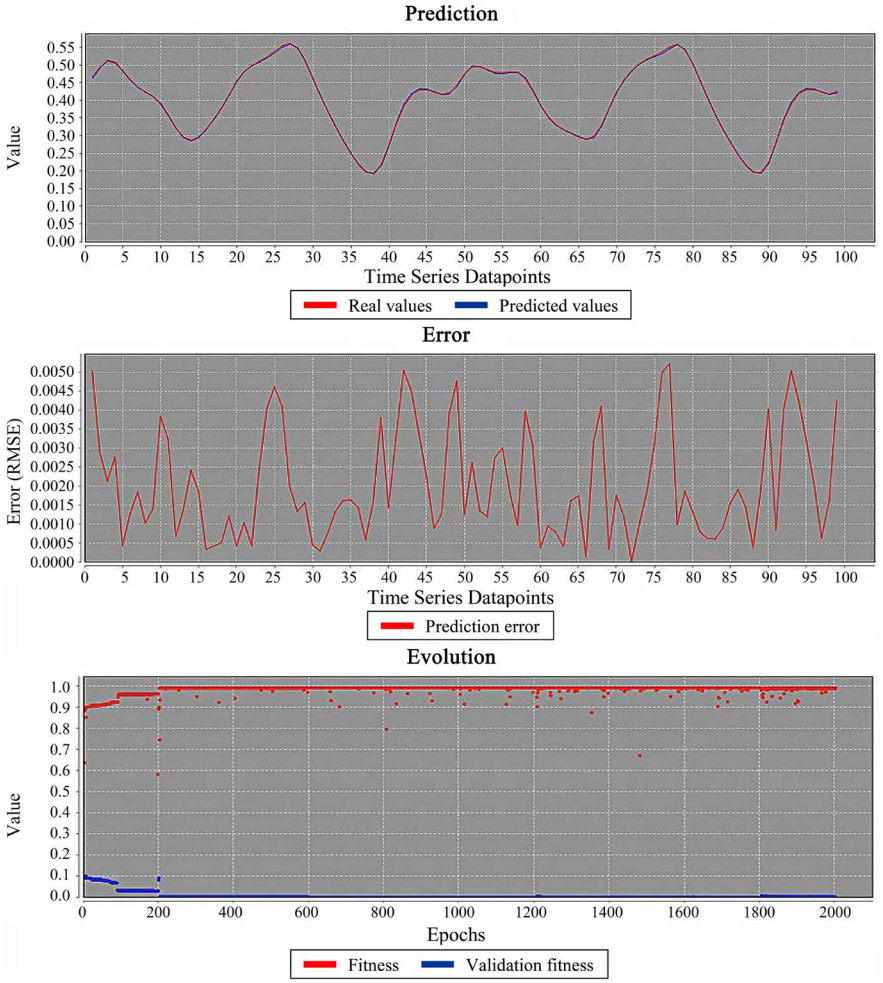
In their most recent study [9] the authors have shown that their system performs better with their Sequential strategy than the other two strategies (Concurrent, MCNE), considering MCNE worse. Although our system is generally based on MCNE, it achieves and produces better results than all of the strategies used in [9], both in the average error and the smallest error, at approximately the same cost. The dominant tests are C and A.

5.6. Additional Analysis

An additional statistical analysis was performed to determine the effectiveness of our own tests with each other. One-way ANOVA [20] was used for statistical analysis, using the PSPP graphic program [21]. In order for the ANOVA method to work, it requires at least two parameters, one factor and one dependent variable. In our problem, the factor is the tests which are a qualitative variable and the dependent variable is the evaluation error, which is a quantitative variable. The first conclusion from the statistical analysis is that the test with the best results in all datasets is C. Having the best average performance, it also has the smallest dispersion, which means that it produces often and consistently the best results. The next best result comes from test A, which is not statistically very different from C, regarding its performance. The remaining tests are ranked in the order B, D and E. From the performance of D relative to E, we can conclude that the method of Percent of fitness helps, but not to a great extent, because statistically the performance of the two tests is not much different. One of the major decision problems, analyzed in [9], is how often and how many times the BP algorithm needs to be executed. By carefully examining the low performance of B, we



(a) Lorenz problem



(b) Mackey problem

Figure 4. Prediction, error (RMSE) and subpopulation development from a run for Lorenz [13] (see (a)) and Mackey problems [14] (see (b)).

can see that many successive BP epochs with low frequency does not help the system, whereas the small number of BP epochs with high frequency of executions increase the system performance.

The general conclusion we get is that the larger the number of subpopulations, the better and more efficient the system is. This implies a larger population sample during migration, which helps the weaker populations, which in turn means more people for evolution than BP. Of course, this comes at a great cost only if multiple threads are not used and at the same time there is no such support from the computer hardware, in particular from its processor. Consequently, if the conditions are met, then the subpopulations will develop in real time, each in its own thread.

6. Future Work

A feature of the problem that deserves further development is that of the variable size of the chromosomes of persons, since it is associated with a proper imple-

mentation of the recombination and mutation methods, based on existing techniques [17]. This feature has to be customized so that it can work harmoniously with the neural networks. Combining EA with different networks such as recurrent neural network (RNN) [22] or long short-term memory (LSTM) [23] will also be very useful to further make the system suitable for dealing with a larger variety of problems. It would also be wise to upgrade BP to a potential learning rate, using a different method from a simple function. For example, one such effective method has been proposed by Adam [24], which has proven its effectiveness, through the multitude of libraries, which generally uses it in machine learning. Another major area for research is deep learning [5] [25] as it appears to be used in a wide range of real problems with great success.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Islam, M.R., Lu, H.H., Hossain, M.J. and Li, L. (2019) A Comparison of Performance of GA, PSO an Differential Evolution Algorithms for Dynamic Phase Reconfiguration Technology of a Smart Grid. *IEEE Congress on Evolutionary Computation*, Wellington, 10-13 June 2019, 858-865. https://doi.org/10.1109/CEC.2019.8790357

[2] Mühlenbein, H., Gorges-Schleuter, M. and Krämer, O. (1988) Evolution Algorithms in Combinatorial Optimization. *Parallel Computing*, **7**, 65-85. https://doi.org/10.1016/0167-8191(88)90098-1

[3] Rodzina, L., Rodzina, O. and Rodzin, S. (2016) Neuroevolution: Problems, Algorithms, and Experiments. *IEEE* 10*th International Conference on Application of Information and Communication Technologies*, Baku, 12-14 October 2016, 1-4. https://doi.org/10.1109/ICAICT.2016.7991745

[4] Stanley, K.O., Clune, J., Lehman, J. and Miikkulainen, R. (2019) Designing Neural Networks through Neuroevolution. *Nature Machine Intelligence*, **1**, 24-35. https://doi.org/10.1038/s42256-018-0006-z

[5] LeCun, Y., Bengio, Y. and Hinton, G. (2015) Deep Learning. *Nature*, **521**, 436-444. https://doi.org/10.1038/nature14539

[6] Robert, H.-N. (1992) Theory of the Backpropagation Neural Network. *Neural Networks for Perception*, **3**, 65-93.

[7] Palmer, A., Montaño, J.J. and Sesé, A. (2006) Designing an Artificial Neural Network for Forecasting Tourism Time Series. *Tourism Management*, **27**, 781-790. https://doi.org/10.1016/j.tourman.2005.05.006

[8] Wong, G., Chandra, R. and Sharma, A. (2016) Memetic Cooperative Neuro-Evolution for Chaotic Time Series Prediction. In: Hirose, A., Ozawa, S., Doya, K., Ikeda, K., Lee, M. and Liu, D., Eds., *Neural Information Processing*, Springer, Cham, 299-308. https://doi.org/10.1007/978-3-319-46675-0_33

[9] Wong, G., Chandra, R. and Sharma, A. (2018) Information Collection Strategies in Memetic Cooperative Neuroevolution for Time Series Prediction. *International Joint Conference on Neural Networks*, Rio, 12-14 October 2016, 1-6.

https://doi.org/10.1109/IJCNN.2018.8489184

[10] Gardner, M. and Dorling, S. (1998) Artificial Neural Networks (the Multilayer Perceptron)—A Review of Applications in the Atmospheric Sciences. *Atmospheric Environment*, **32**, 2627-2636. https://doi.org/10.1016/S1352-2310(97)00447-0

[11] Admin. Exchange Rate (TWI). https://datamarket.com/data/set/22tb

[12] S.W.D. Center. The International Sunspot Number (1843-2001), International Sunspot Number Monthly Bulletin and Online Catalogue. Royal Observatory of Belgium, Avenue Circulaire 3, 1180 Brussels, Belgium [Online]. http://www.sidc.be/silso

[13] Lorenz, E. (1963) Deterministic Non-Periodic Flows. *Journal of Atmospheric Science*, **20**, 130-141.

[14] Mackey, M.C. and Glass, L. (1997) Oscillation and Chaos in Physiological Control Systems. *Science*, **197**, 287-289. https://doi.org/10.1126/science.267326

[15] Weigend, A.S. and Gershenfeld, N.A. (1994) Laser Problem Dataset: The Santa Fe Time Series Competition Data. https://github.com/gary-wong-fiji/Meme-Collection-SEQ/tree/master/Datasets/Laser

[16] Grosan, C. and Abraham, A. (2007) Hybrid Evolutionary Algorithms: Methodologies, Architectures, and Reviews. In: *Hybrid Evolutionary Algorithms*, Springer, Berlin, 1-17. https://doi.org/10.1007/978-3-540-73297-6_1

[17] Hutt, B. and Warwick, K. (2007) Synapsing Variable-Length Crossover: Meaningful Crossover for Variable-Length Genomes. *IEEE Transactions on Evolutionary Computation*, **11**, 118-131. https://doi.org/10.1109/TEVC.2006.878096

[18] Karlik, B. and Olgac, A.V. (2011) Performance Analysis of Various Activation Functions in Generalized MLP Architectures of Neural Networks. *International Journal of Artificial Intelligence and Expert Systems*, **1**, 111-122.

[19] Li, Y. and Yuan, Y. (2017) Convergence Analysis of Two-Layer Neural Networks with ReLU Activation. *Neural Information Processing Systems Conference*, Long Beach CA, 4-9 December 2017, 597-607.

[20] Cuevas, A., Febrero, M. and Fraiman, R. (2004) An Anova Test for Functional Data. *Computational Statistics & Data Analysis*, **47**, 111-122. https://doi.org/10.1016/j.csda.2003.10.021

[21] GNU PSGNU Free Software Foundation. https://www.gnu.org/software/pspp

[22] Mikolov, T., Karafiát, M., Burget, L., Černocký, J. and Khudanpur, S. (2010) Recurrent Neural Network Based Language Model. 11*th Annual Conference of the International Speech Communication Association*, Makuhari Chiba, Japan, 26-30 September 2010, 1045-1048. https://doi.org/10.1109/ICASSP.2011.5947611

[23] Gers, F., Schmidhuber, J. and Cummins, F. (1999) Learning to Forget: Continual Prediction with LSTM. 9*th International Conference on Artificial Neural Networks*, Edinburgh, 7-9 May 2015. https://doi.org/10.1049/cp:19991218

[24] Kingma, D.P. and Jimmy, B. (2015) Adam: A Method for Stochastic Optimization. *International Conference on Learning Representations* (ICLR2015).

[25] Such, F.P., Madhavan, V., Conti, E., Lehman, J., Stanley, K.O. and Clune, J. (2017) Deep Neuroevolution: Genetic Algorithms Are a Competitive Alternative for Training Deep Neural Networks for Reinforcement Learning. https://arxiv.org/abs/1712.06567