Iranian Journal of Electrical and Electronic Engineering

Journal Homepage: ijeee.iust.ac.ir

Research Paper

Chaotic Time-Series Prediction using Intelligent Methods

M. Nezhadshahbodaghi*, K. Bahmani*, M. R. Mosavi*(C.A.),and D. Martín**

Abstract: Today, it can be said that in every field in which timely information is needed, we can use the applications of time-series prediction. In this paper, among so many chaotic systems, the Mackey-Glass and Loranz are chosen. To predict them, Multi-Layer Perceptron Neural Network (MLP NN) trained by a variety of heuristic methods are utilized such as genetic, particle swarm, ant colony, evolutionary strategy algorithms, and population-based incremental learning. Also, in addition to expressed methods, we propose two algorithms of Bio-geography-Based Optimization (BBO) and fuzzy system to predict these chaotic systems. Simulation results show that if the MLP NN is trained based on the proposed meta-heuristic algorithm of BBO, training and testing accuracy will be improved by 28.5% and 51%, respectively. Also, if the presented fuzzy system is utilized to predict the chaotic systems, it outperforms approximately by 98.5% and 91.3% in training and testing accuracy, respectively.

Keywords: Time Series, Neural Networks, Heuristic Methods, Fuzzy Systems.

1 Introduction

NOWADAYS, chaotic time-series predictions are used in various fields such as marketing policies, supply chain management, signal processing, traffic jam condition, weather forecasting, sunspot forecasting, and many other fields. Predicting chaotic time series has been one of the significant challenges during the last few decades, which can be considered as a subset of non-linear processes that produce complex and irregular results [1]. Chaos theory has been an appropriate method to express the characteristics of a dynamic and complex system that shows complex trends. This kind of chaotic system has two important features: (1) Unpredictable behaviors, and (2) regular patterns that are embedded in them. Therefore, the chaos system is not only a non-random system, but also regulars and orders exist within appearance irregularities and disorders [2, 3].

Threshold method, exponential model, local linear model, nearest neighbor, regression, and Neural Network (NN) are some of the methods which are utilized for modeling and predicting non-linear time series [4, 5]. Artificial Neural Network (ANN) methods have better performance than other methods, which can be attributed to low computation and complexity, no need for a system model, etc. [6].

ANNs are kinds of new computational methods that are used to forecast the output responses of complex systems. The main idea of these networks is inspired by the performance of the biological nervous system, which is able to learn and create knowledge by engaged to process information. Making a new structure for the information processing is a key point of this idea. This system is composed of an intensive number of interconnected processing elements called neurons that work in unison to solve a problem.

Iranian Journal of Electrical and Electronic Engineering, 2023.
Paper first received 24 Oct 2022, revised 18 Apr 2023, and accepted 26 Apr 2023.
*The authors are with School of Electrical Engineering, Iran University of Science and Technology, Tehran, Iran.
E-mails: m_nezhadshahbodaghi@elec.iust.ac.ir, k_bahmani@elec.iust.ac.ir, and m_mosavi@iust.ac.ir.
**The author is with ETSI de Telecomunicación, Universidad Politécnica de Madrid, Madrid, Spain.
E-mail: diego.martin.de.andres@upm.es.
Corresponding Author: M. R. Mosavi.
https://doi.org/10.22068/IJEEE.19.2.2692.

The first attempt to simulate the NN was made using a logical model, which today this kind of model is utilized as a main structural block of most ANNs [7, 8]. In reference [9], the perceptron NN is introduced, which has three layers and the second layer is known as the bond layer. By training this type of system, it assigns the corresponding output for each input. The error post-propagation algorithm is utilized to train this network that has considerable disadvantages. Therefore, NNs are used to predict short-term and long-term time series such as Mackey-Glass, Lorenz, etc. [10-12].

Reference [13] uses the limited learning machine, which is a kind of pioneer NN with a hidden layer. In these networks, the weight and threshold of the input layer are selected randomly and these parameters are not trained and learned by a special algorithm. In this algorithm, based on the least mean square algorithm, only the weight of the hidden layer to the output layer is calculated. Another approach to predict time series is the combination of NNs and meta-heuristic algorithms. In order to predict the short-term and long-term chaotic time series such as Mackey-Glass, the combination of NN and particle swarm optimization algorithm is used in reference [14].

As mentioned, due to dataset properties and high data size, the use of conventional and traditional methods based on derivatives and gradient decent algorithms does not lead to achieve high prediction accuracy. Therefore, the meta-heuristic or evolutionary algorithms such as particle algorithm, genetics, ant colony and evolutionary strategy can be used to train NNs [14-18].

In this paper, we propose an improved meta-heuristic algorithm based on bio-geography, and a fuzzy system to increase the accuracy of chaotic time-series prediction. The simulation results confirm that using the improved meta-heuristic algorithm based on bio-geography and fuzzy system outperform the other evolutionary methods in terms of the training and testing accuracy by 28.5%-51% and 98.5%-91.3%, respectively.

The remainder of this article is organized as follows. Section 2 demonstrates the Mackey-Glass and Lorenz time-series prediction. In Section 3, the Multi-Layer Perceptron (MLP) NN will be introduced. The two proposed methods are described in Section 4. In Section 5, the proposed and traditional methods are evaluated and compared in terms of training and testing accuracy. Finally, in the last section, the conclusions will be given.

2 Chaotic Time Series

The Mackey-Glass time series is a time series with chaotic behavior that can be expressed as Eq. (1):

$$dx(t)/dt = \beta x(t-\tau)/1+x^n(t-\tau)-\gamma x(t)$$
$$\gamma,\ \beta,\ \tau, \text{ and } n>0 \quad (1)$$

where by adjusting system parameters ($\gamma, \beta, \tau,$ and n), different dynamic behaviors can be made. In Equation (1), the parameter τ determines the properties of the equation. By adjusting $\tau < 4.43$ a fixed-point absorber, $4.43 < \tau < 13.3$ a fixed cycle absorber, $13.3 < \tau < 16.8$ a double-cycle absorber, and $16.8 < \tau$ a chaos will be produced. Figures 1 and 2 illustrate Mackey-Glass time-series fluctuations and dynamic behaviors, respectively which the parameters are adjusted according to the reference [19]. The values are as Eq. (2):

$$\gamma = 0.15,\ \beta = 0.3,\ \tau = 35, \text{ and } n = 11 \quad (2)$$

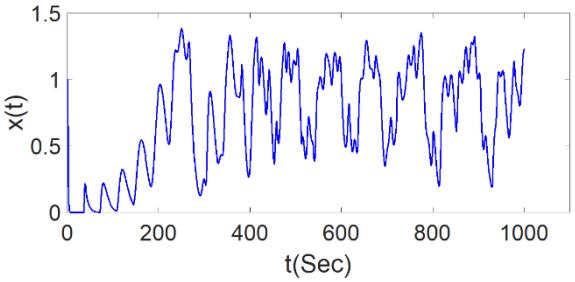

Fig. 1 Mackey-Glass time-series fluctuations in terms of time.

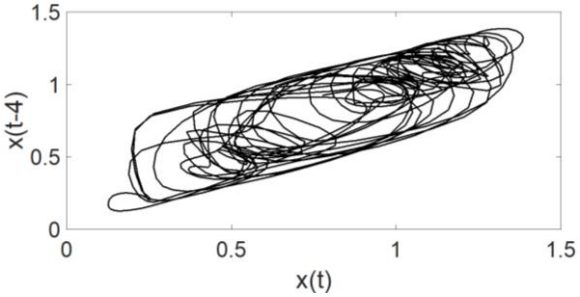

Fig. 2 Mackey-Glass time-series dynamic behavior.

According to Fig. 2, the dynamics of this time series are constantly fluctuating and never converge to a single spot.

The Lorenz system has the following chaotic system, which is widely utilized to establish chaotic theory:

$$\frac{dx(t)}{dt} = \sigma[y(t)-x(t)]$$
$$\frac{dy(t)}{dt} = x(t)[r-z(t)]-y(t) \quad (3)$$
$$\frac{dz(t)}{dt} = x(t)y(t)-bz(t)$$

where σ, r, and b are parameters whose values are adjusted based on the reference [20] as Eq. (4):

$$b = 8/3,\ r = 28,\ \text{and}\ \sigma = 10 \tag{4}$$

Figures 3 and 4 illustrate Lorenz time-series fluctuations and dynamic behaviors, respectively.

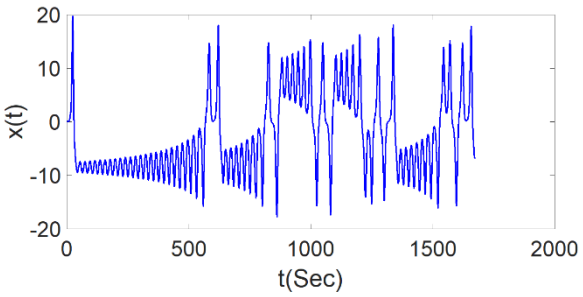

Fig. 3 Lorenz time-series fluctuations in terms of time

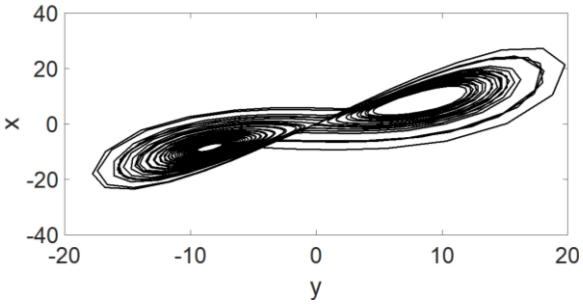

Fig. 4 Lorenz time-series dynamic behavior.

3 Multi-Layer Perceptron Neural Network

Feeder multi-layer networks are one of the most important structures of ANNs. Typically, these networks consist of a set of sensory units (basic neurons) that comprise the input layer, one or more hidden layers, and an output layer. The input signal is propagated in the forward path throughout the network. Figure 5 shows the structure of this type of NN, which includes *p* input nodes, *q* neurons in the hidden layer, and *r* output nodes.

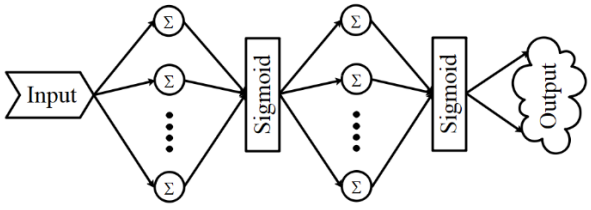

Fig. 5 The MLP NN structure.

The stimulation of the j-th hidden layer is obtained from the Eqs. (5) and (6):

$$v_j = \sum_{i=1}^{p} x_i \times w_{ji} + \theta_j, \qquad j = 1,2,...,q \tag{5}$$

$$s_j = sigmoid(v_j) = \frac{1}{1+\exp(v_j)} \tag{6}$$

where x_i denotes the i-th node in the input layer, w_{ji} represents the weight of the connection line between the i-th node in the input layer and j-th neuron in the hidden layer, and θ_j is the threshold value of j-th neuron in the hidden layer. The Eqs. (7) and (8) are used to calculate network output:

$$z_k = \sum_{j=1}^{q} s_j \times w_{kj} + \varphi_k, \qquad k = 1,2,..,r \tag{7}$$

$$y_k = sigmoid(z_k) = \frac{1}{1+\exp(z_k)} \tag{8}$$

where w_{kj} is the weight of the connection line between j-th neuron in the hidden layer and k-th neuron in the output layer, and φ_k represents the threshold value of k-th neuron in the output layer. The weight of connection lines and threshold values are the most significant parts of MLP NNs which determine the final output.

In this paper, in order to compare the traditional methods with the proposed methods, the same NN structure is utilized to teach the NN. Based on this structure, the number of NN inputs is four, a hidden layer consists of nine sigmoid function neurons, and a linear stimulus function is used in the output layer. The experimental results show that this network structure can give us the desired results.

Neural Network Training

MLP NN training means finding the best value for the weight of connection lines and thresholds in order to achieve the best output result for specific inputs. In traditional methods such as the post-propagation algorithm, the output of the objective function is known and learning is done in a supervisory manner. This algorithm can find the optimal weights for a MLP NN by using the downgrade and error minimization. The calculation of the output sensitivity to the weights starts from the end of the network and all weights are updated concurrently. Due to low convergence speed, ability to get stuck in local minima, and low accuracy, this algorithm is not suitable for high-dimensional problems, large critical points, and practical problems. Therefore, the tendency to use meta-heuristic algorithms to train NNs and engage fuzzy systems has increased dramatically [21]. In the following, we will introduce the first proposed method, which is based on a boosted bio-geographic algorithm.

4.1 The First Proposed Method

The Bio-geography-Based Optimization (BBO) algorithm was first proposed in 2008. The main idea

of this algorithm is inspired by a field in biology that discusses the distribution of animals and plants (in time and space). In this field of study, different ecosystems (habitats) are examined to find the relationship between different species (inhabitants) in terms of immigration, emigration, and mutation [22]. In other words, the evolution of ecosystems to achieve stable conditions is the basis of the BBO algorithm. This algorithm uses a number of search factors called habitats, which are similar to the chromosomes in the Genetic Algorithm (GA). In this algorithm, each variable indicates a location that is considered as a vector of species (similar to the genes in the GA). In addition, the Habitat Suitability Index (HSI) is defined for each habitat area, and a high HSI index means better conditions. Habitat areas are always determined according to the three main rules as follows:

(1) Species that live in high-HSI are more likely to migrate to lower-HSI.
(2) Species that live in low-HSI are more likely to absorb immigrants from high-HSI [23].
(3) Habitat areas must change their inhabitants randomly, regardless of their HSI.

This phenomenon makes a balance between different ecosystems in nature. In other words, nature tends to boost the sustainability of different biological areas. The BBO algorithm uses these concepts to improve the HSI of all habitat areas. In the BBO algorithm, there are three main operators of migration, mutation, and elitism, which we will discuss below.

The BBO algorithm starts with the migration operator and the random selection of habitat areas. Each habitat area has *k* different species that are determined based on the variables of a particular problem. In addition, each habitat area has specific immigration, emigration, and mutation rates, which is modeled based on distinct biological area in nature. Immigration rate λ_i and emigration rate μ_i are functions of the number of inhabitants (species) in the habitat, which are expressed as Eqs. (9) and (10):

$$\lambda_k = I \times (1 - \frac{k}{N}) \tag{9}$$

$$\mu_k = E \times (\frac{k}{N}) \tag{10}$$

where *k* represents the number of current species and *N* indicates the maximum number of species that is increased based on the HSI. In other words, more HSI means more species number. *I* is the maximum immigration number, and E expresses the maximum emigration number. In this paper, *I* and *E* are normalized and the maximum rate is 1. Another operator of the BBO algorithm is the mutation, which improves the exploration power of the BBO algorithm and preserves the diversity of species. This operator is defined as Eq. (11):

$$m_k = m_{\max} \times (1 - \frac{p_k}{p_{\max}}) \tag{11}$$

where m is the initial mutation user-defined value, p_k indicates the probability of mutation in the n-th habitat and $p_{\max}$ is the maximum value of p_k. In this algorithm, after calculating the HSI for each habitat, the rates of immigration, emigration, and mutation are also updated (calculated). According to these updated rates, non-elite species migrate or mutate. After that, some of the best pre-defined habitats are considered for the production of later generations. Finally, the BBO algorithm finishes with the achievement of the final approaches. Figure 6 shows how the migration and mutation operators work. In this figure, habitat 1 has the lowest HSI, which indicates that the mean squared error is minimal for all test specimens, so it is more suitable than habitats 2, 3, and 4. It should be noted that elitism prevents the destruction of the best habitats in the face of immigration. As a result, some of the best habitats are maintained in each iteration. Figure 7 illustrates the general operation steps of a BBO algorithm.

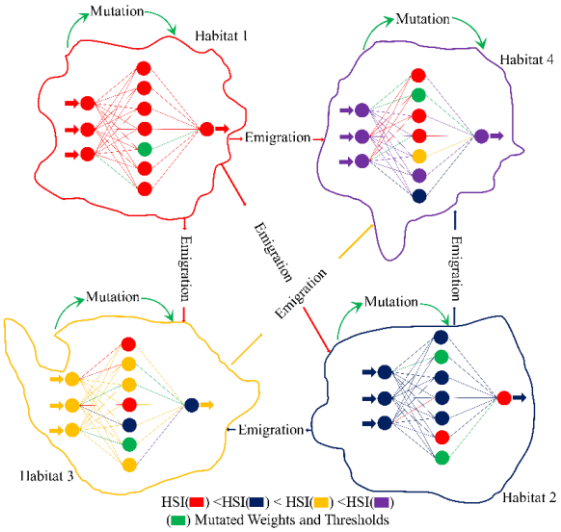

Fig. 6 Behavior of migration and mutation operators based on habitat HIS.

In the following, we will train the NN with the help of the proposed method. Figure 8 shows the structure of the MLP NN that we have utilized in the first proposed method. Also, the final NN vector is given in Eq. (12):

$$\text{Habitat} = [w_{11},..,w_{19},w_{21},..,w_{49},\theta_1,....,\theta_9,\theta_{out}] \tag{12}$$

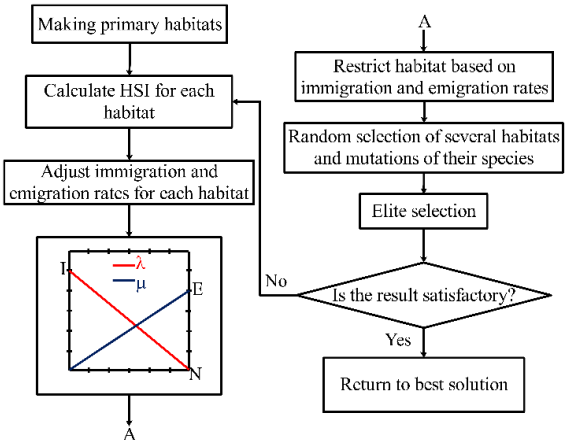

Fig. 7 The general operation steps of a BBO algorithm.

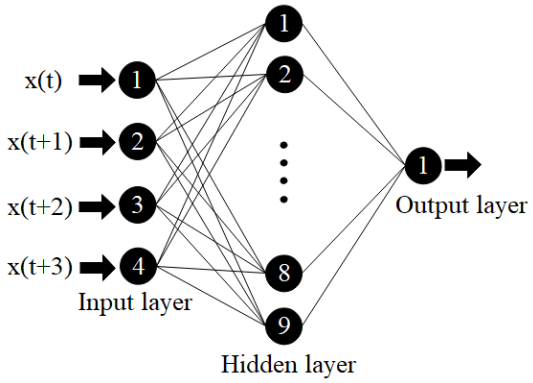

Figure 8: The MLP NN used in the proposed methods.

As mentioned, the training of ANNs is the purpose of learning methods. In this paper, the HSI function for all training samples is calculated by the mean squared error method as Eq. (13):

$$\sum_{k=1}^{q}\frac{\sum_{i=1}^{m}(o_i^k-d_i^k)^2}{q} \tag{13}$$

where q indicates the number of training samples, m represents the number of outputs, d_i^k is the optimal output value for the i-th input when the k-th training sample is applied to the input, and o_i^k is the real output value for the i-th input when the k-th training sample is applied to the input.

As shown in Fig. 7, the first proposed method starts with generating primary habitats that are randomly selected based on the number of defined habitats and MLP NNs. Each MLP NN corresponds to a habitat area, and each weight or threshold corresponds to the species. After the initial step, the mean squared error of each MLP NN is calculated by Eq. (13). In the next step, the rates of immigration, emigration, and mutation are updated by Eqs. (9) and (10). Then, MLP NNs are combined based on immigration and emigration rates. In the next step, according to the mutation rate, each MLP NN is changed. The elite selection is the last step of the first proposed method. At this step, the best MLP NNs are maintained to prevent damage by mutation and evolutionary operators in the next generation. These steps continue until a satisfactory result is achieved.

4.2 The Second Proposed Method

In the second proposed method, first, we design the fuzzy system utilizing input and output datasets. Then, future time-series values are predicted by the designed fuzzy system. In other words, the *x(t)* value is estimated by the *x(t-n+1)*, *x(t-n+2)*,…, and *x(t-1)* values. This is expressed in Eq. (14):

$$\{x(t-n+1),....,x(t-n+1)\}\in R^n\rightarrow\{x(t)\}\in R \tag{14}$$

Figure 9 shows the structure of a fuzzy system. In this design, we have considered four inputs for the fuzzy system. In other words, the values of the Mackey-Glass and Lorenz time series at t depend on their values at *t-4*, *t-3*, *t-2*, and *t-1*.

Fig. 9 The fuzzy system structure.

Triangular, trapezoidal, and Gaussian membership functions can be used to design a fuzzy system. Since we want to design fuzzy systems using the gradient decent algorithm, it is not possible to use triangular and trapezoidal functions due to rupture during derivation. Therefore, we have used Gaussian functions to compare the output results of this designed fuzzy system with the results of gradient descent and clustering algorithms. Table 1 represents a summary of the specifications of the designed fuzzy system. The designed fuzzy system is as Eq. (15):

$$f(x)=\frac{\sum_{l=1}^{M_1}\bar{y}_1^{\,l}\left[\prod_{i=1}^{4}\exp(-(\frac{x_i-\bar{x}_{i,1}^{l}}{\sigma_{i,1}^{l}})^2)\right]}{\sum_{l=1}^{M_1}\left[\prod_{i=1}^{4}\exp(-(\frac{x_i-\bar{x}_{i,1}^{l}}{\sigma_{i,1}^{l}})^2)\right]} \tag{15}$$

Table 1 Summary of the specifications of the designed fuzzy system.

Mamdani multiplication	Inference engine
Singleton	Fuzzification
Center of Area	Defuzzification
Gaussian	Membership functions
4	Number of input
1	Number of output

Since all system inputs and outputs are the same type, we divide each input and output interval into 4

parts. For example, the membership functions of the first input *x(t-4)* are shown in Fig. 10. Also, the membership functions of other inputs and outputs are similar to Fig. 10, but the membership functions center will be different. In this case, the maximum number of rules is calculated by Eq. (16):

$$\text{Rules} = m_{x1} \times m_{x2} \times m_{x3} \times m_{x4} \times m_{x5} = 4^5 = 1024 \quad (16)$$

where m represents the number of membership functions in each of the inputs and outputs. The final rules are selected by (1) the membership functions with higher membership values for the input-output data, (2) the ranking of rules, and (3) selection of higher-ranking rules (delete rules with the same "if" section and different "then" section). Figure 11 shows a three-dimensional view of the output changes in terms of the changes of the first two inputs.

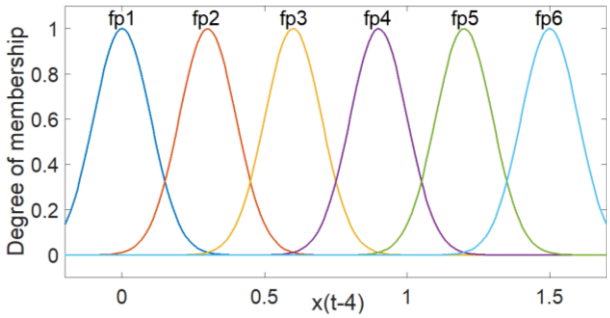

Fig. 10 Membership functions related to *x (t-4)*.

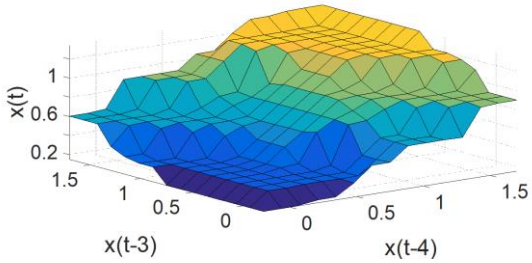

Fig. 11 The output changes in terms of the changes of the first two inputs.

The design of the fuzzy system ends with the look-up table method, but we utilize the gradient descent algorithm to improve the proposed fuzzy system. In other words, to achieve an optimal structure, the parameters of the membership functions are changed to reduce the prediction error. In Eq. (15), the parameters $\bar{x}$, $\bar{y}$ and σ can be considered as variable parameters, which by changing them, the criterion function of Eq. (17) is minimized and the desired results are achieved.

$$e = \frac{1}{2}[y - f(x)] \quad (17)$$

where *f(x)* is the estimated value by the fuzzy system and y is the desired value that we want the fuzzy system to achieve. In other words, our goal is to minimize time prediction error by changing the specifications of Gaussian membership functions. For this purpose, we define the following equations:

$$z = \prod_{i=1}^{n} \exp(-(\frac{x_i - \bar{x}_i^l}{\sigma_i^l})^2) \quad (18)$$

$$a = \sum_{l=1}^{M} \bar{y}^l [\bar{z}^l] \quad (19)$$

$$b = \sum_{l=1}^{M} [\bar{z}^l] \quad (20)$$

$$f = a / b \quad (21)$$

where *n* represents the number of inputs, which here is *n = 4*, and *M* indicates the number of winning rules, where obtained from the look-up table method. Then, according to the above relations, the values of the variable parameters are updated by Eq. (22):

$$\begin{aligned}
&\bar{y}^L(k+1) = \bar{y}^L(k) - \alpha \frac{\partial e(k)}{\partial \bar{y}^L(k)} = \\
&\bar{y}^L(k) - \alpha \frac{f - y(k)}{b} \bar{z}^L(k) \\
&\bar{x}_i^L(k+1) = \bar{x}_i^L(k) - \beta \frac{\partial e(k)}{\partial \bar{x}_i^L(k)} = \\
&\bar{x}_i^L(k) - \beta \frac{f - y(k)}{b} * (\bar{y}^L(k) - f) z^L * 2(\frac{x_i(k) - \bar{x}_i^L(k)}{\sigma_i^L(k)}) \\
&\sigma_i^L(k+1) = \sigma_i^L(k) - \beta \frac{\partial e(k)}{\partial \sigma_i^L(k)} = \\
&\sigma_i^L(k) - \gamma \frac{f - y(k)}{b} (\bar{y}^L(k) - f) z^L * 2(\frac{x_i(k) - \bar{x}_i^L(k)}{\sigma_i^L(k)^3})^2
\end{aligned} \quad (22)$$

where α, β, and γ are called training coefficient parameters that have a value between zero and one. Also, the value of L = 1, 2, …, M. In this method, in each iteration, the value of the parameters is updated in a specific period according to Eq. (22).

We utilized the look-up table and gradient descent algorithms to design the initial fuzzy system and optimize the membership function parameters, respectively. Now, we use the clustering algorithm to optimize the number of fuzzy rules (*M*). As a result, processing speed and system accuracy increase. This method consists of three main steps:

(1) In the first step, by using the look-up table the first input and output dataset are selected and then the following equation parameters are updated:

$$\underline{X}_c^1 = \underline{x}_0^1 \quad (23)$$

$$A^1(1) = y_0^1 \quad (24)$$

$$B^1(1) = 1 \quad (25)$$

where $\underline{X}_c^1$ is the first cluster center, $A^1(1)$ indicates the first rule center, and $B^1(1)$ represents the number of data in the first cluster.

(2) In the second step, we assumed that, the kth of the input and output data is being processed and before that the number of clusters examined is equal to M with centers $X_c^1, X_c^2 \ldots, X_c^{M-1}, X_c^M$. Then, the distance between the kth data to all these centers is calculated. We use the Euclidean distance, which is calculated from Eq. (26):

$$r_L = \sqrt{\{(X_c^L(1) - x_0^k(1))^2 + (X_c^L(2) - x_0^k(2))^2 + (X_c^L(3) - x_0^k(3))^2 + (X_c^L(4) - x_0^k(4))^2\}} \quad (26)$$

Then, the r_L value is compared to a predefined radius. If the r_L value is greater, a new cluster is formed and Eqs. (23) to (25) are expressed for the new cluster. Otherwise, the cluster with the closest value to k is selected and Eqs. (24) and (25) are updated.

(3) In the third step, the value of k is increased by one unit and the second step is repeated. These steps are repeated until all input and output data are processed. Suppose that after processing all the data, the values of each of the formed clusters are as Eq. (27):

$$\begin{aligned} &X_c^L \quad L = 1,\ldots,Q \\ &A^L \quad L = 1,\ldots,Q \\ &B^L \quad L = 1,\ldots,Q \end{aligned} \quad (27)$$

where Q is the total number of clusters formed. Finally, the desired fuzzy system is obtained by Eq. (28):

$$f(x) = \frac{\sum_{l=1}^{Q} A^l \left[\prod_{i=1}^{4} \exp(-(\frac{x_i - x_c^l}{\sigma})^2)\right]}{\sum_{l=1}^{Q} B^l \left[\prod_{i=1}^{4} \exp(-(\frac{x_i - x_c^l}{\sigma})^2)\right]} \quad (28)$$

Simulation Results

In this section, we compare the proposed methods (BBO and fuzzy system) with genetic, bird or particle swarms, ant colonies, evolutionary strategies, and population-based incremental learning algorithms, and discuss the effectiveness of the proposed methods.

To examine the MLP NN trained by the proposed methods, 2000 data have been extracted from the Mackey-Glass and Lorenz chaotic time series. Also, the free parameters of these systems are based on Eqs. (2) and (4). In this simulation, 80% of this data was used as training data and 20% as testing data. It is worth noting that, explaining the method of training the MLP NN is out of the scope of this article, because these methods have been widely published. So, the readers to obtain more information can refer to these references [24-29].

In evolutionary algorithms and the first proposed method, the initial population is 150, which in the proposed method is equivalent to habitat, the number of generations is 100, and the optimization variable length is 55. In the designed fuzzy system, the number of membership functions is 6, which is based on the look-up table and the gradient descent algorithms. Furthermore, the value of training coefficients in the gradient descent algorithm is considered to be 0.01. The value of the radius for the clustering algorithm is 0.5, which was used in the design of the fuzzy system. It should be noted that these parameters have been obtained with many trials and errors.

In the proposed methods and traditional methods, the accuracy of training and testing of algorithms will be compared with each other. Figure 12 shows the accuracy of NN training by the proposed BBO method for the Mackey-Glass system. Figure 13 depicts the training error rate in terms of the number of samples. In this case, the mean square error is approximately 0.2581. Figures 14 and 15 show the accuracy of NN training and the training error rate by the proposed BBO method for the Lorenz system, respectively. In this case, the mean square error is almost 1.871.

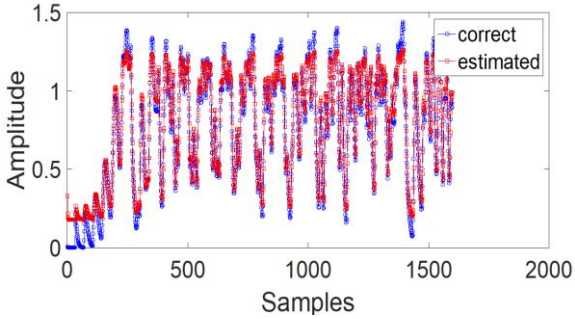

Fig. 12 Training simulation results by the proposed BBO method for the Mackey-Glass system.

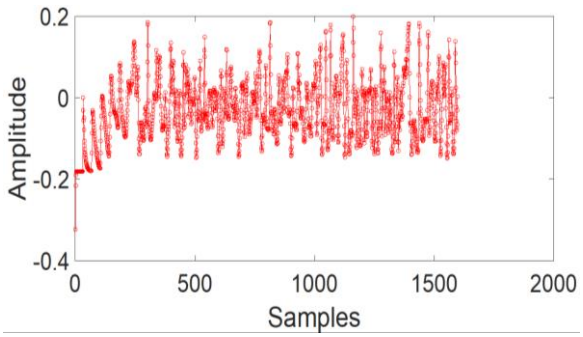

Fig.13 The training error rate in terms of the number of samples by the proposed BBO method for the Mackey-Glass system.

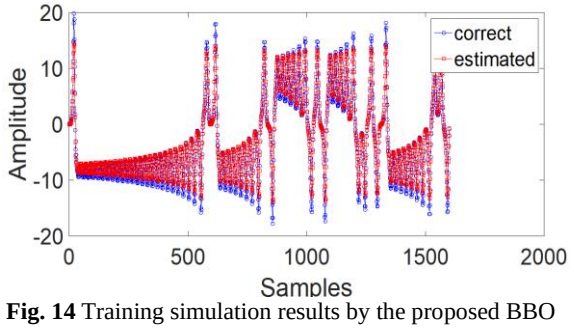

Fig. 14 Training simulation results by the proposed BBO method for the Lorenz system.

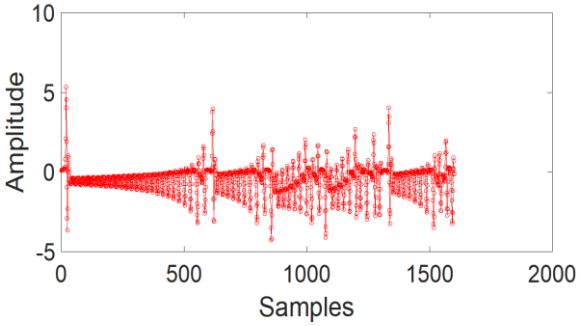

Fig. 15 The training error rate in terms of the number of samples by the proposed BBO method for the Lorenz system.

Figure 16 illustrates the accuracy of fuzzy system training by the clustering algorithm for the Mackey-Glass system. In addition, the training error rate for this system is shown in Fig. 17. Furthermore, Figs. 18 and 19 show the accuracy of fuzzy system training and the training error rate by the clustering algorithm for the Lorenz system, respectively.

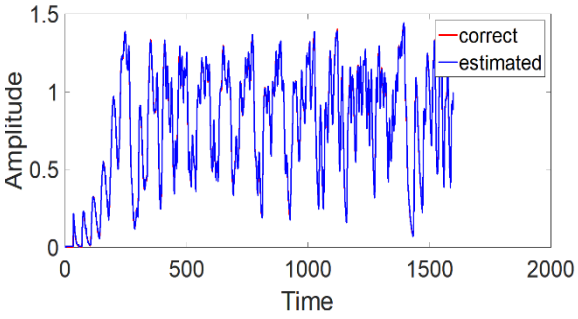

Fig. 16 The accuracy of fuzzy system training by the clustering algorithm for the Mackey-Glass system.

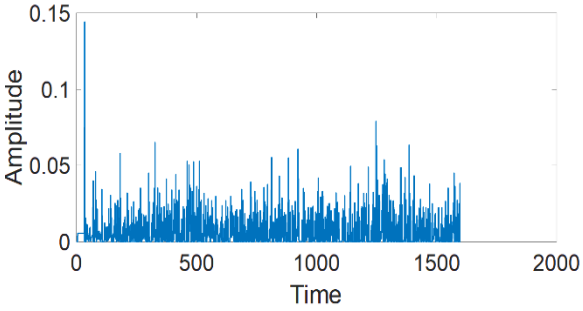

Fig. 17 The training error rate in terms of the number of samples by the second proposed method for the Mackey-Glass system.

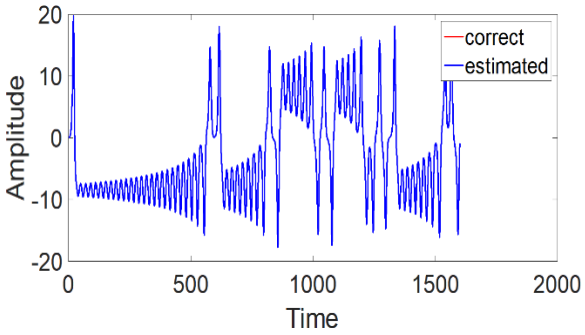

Fig. 18 The accuracy of fuzzy system training by the clustering algorithm for Lorenz system.

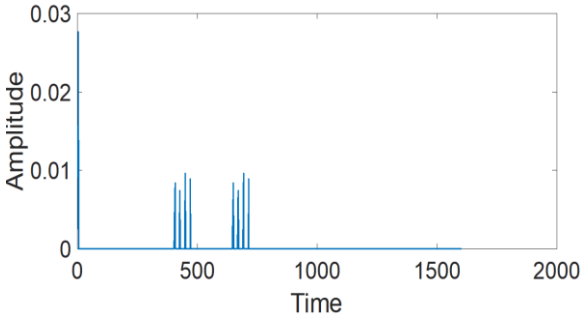

Fig. 19 The training error rate in terms of the number of samples by the second proposed method for the Lorenz system.

As mentioned, about 20% of the data is allocated to the test. Figs. 20 to 27 depict the simulation results of the test data in the two proposed methods for the Mackey-Glass and Lorenz chaotic systems, respectively.

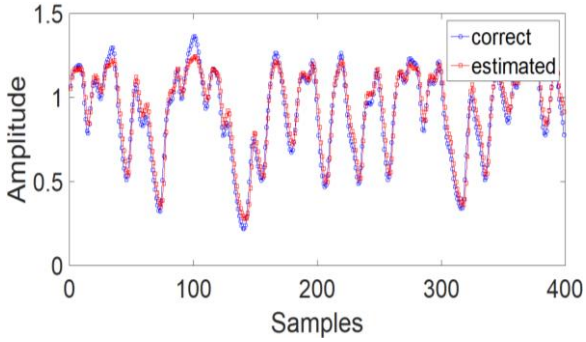

Fig. 20 Simulation results of test data in the first proposed method for the Mackey-Glass system.

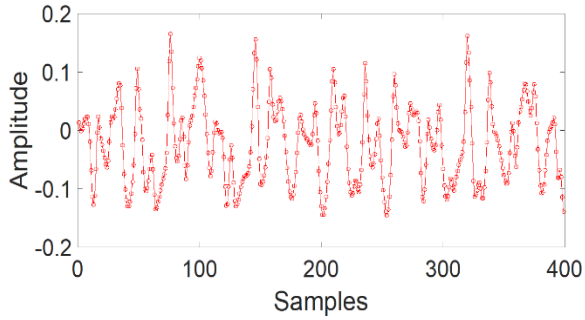

Fig. 21 Simulation results of test data error in the first proposed method for the Mackey-Glass system.

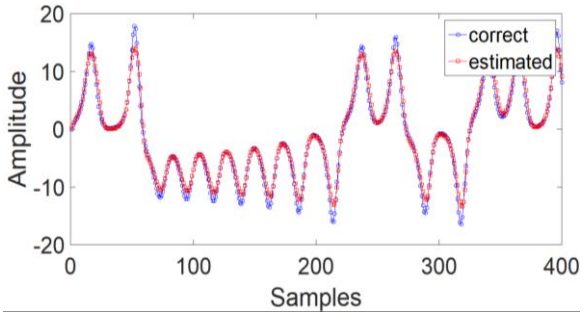

Fig. 22 Simulation results of test data in the first proposed method for the Lorenz system.

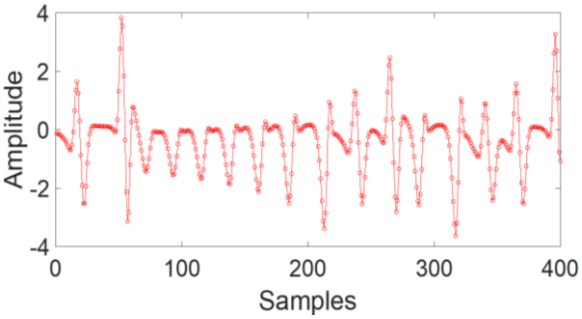

Fig. 23 Simulation results of test data error in the first proposed method for the Lorenz system.

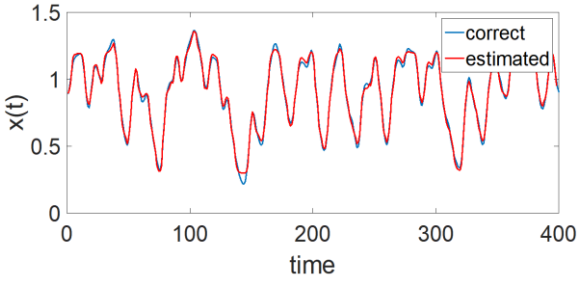

Fig. 24 Simulation results of test data in the second proposed method for the Mackey-Glass system.

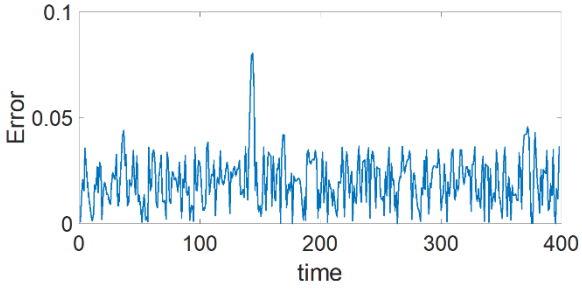

Fig. 25 Simulation results of test data error in the second proposed method for the Mackey-Glass system.

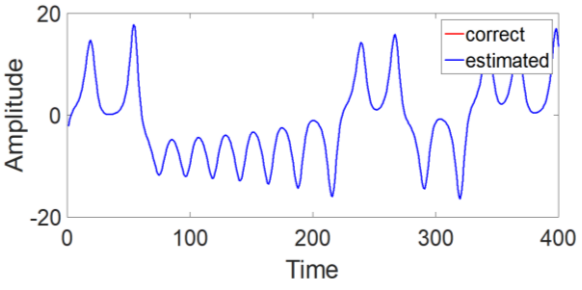

Fig. 26 Simulation results of test data in the second proposed method for the Lorenz system.

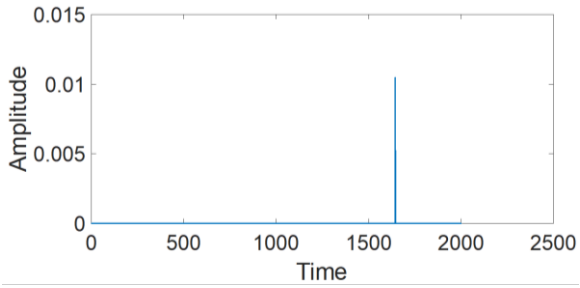

Fig. 27 Simulation results of test data error in the second proposed method for the Lorenz system.

In order to compare the first proposed method with other meta-heuristic algorithms, the convergence speed curves over the generations for Mackey-Glass and Lorenz chaotic systems are shown in Figs. 28 and 29, respectively. Table 2 shows a comparison between the proposed method and other algorithms in terms of the accuracy of the training data (TRD) and the testing data (TED) for the Mackey-Glass and Lorenz systems.

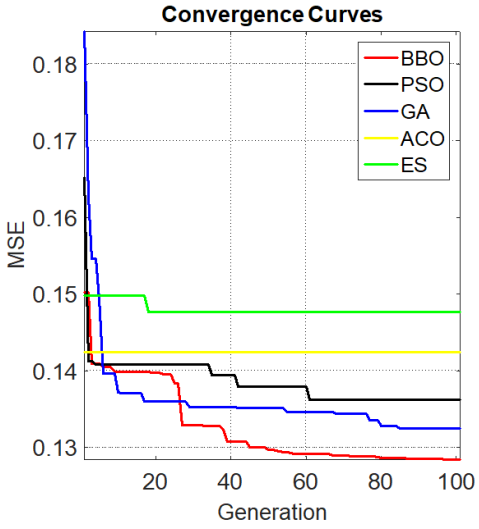

Fig. 28 Comparison of convergence speed curves over the generations for Mackey-Glass system in different algorithms.

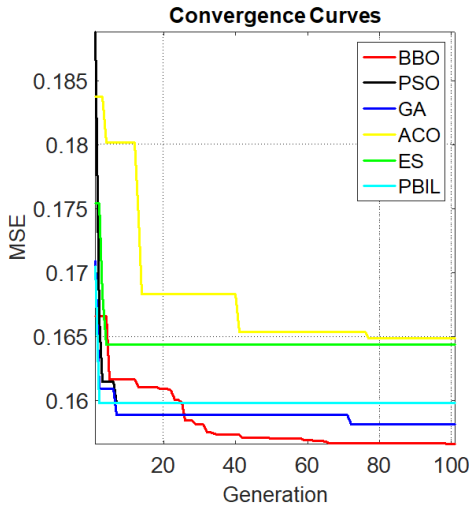

Fig. 29 Comparison of convergence speed curves over the generations for Lorenz system in different algorithms.

Table 2 Comparison between the proposed and other algorithms in terms of the accuracy of the training data and the test data for the Mackey-Glass and Lorenz systems.

Method	Mackey-Glass		Lorenz	
	MSD for TRD	MSD for TED	MSD for TRD	MSD for TED
GA	0.4178	0.4564	2.3178	2.4464
ACO	0.9940	0.9844	3.3242	3.6221
PSO	0.7550	0.7842	2.8150	2.9142
ES	1.1733	1.1601	3.1833	3.4214
PBIL	-	-	2.8241	2.9333
BBO	0.2581	0.1165	1.871	1.8912
FS based on LUT	0.0193	0.0321	1.1145	1.9012
FS based on GD	0.0187	0.0255	1.0147	1.8914
FS based on CA	0.0073	0.0201	0.0005	0.00025

6 Conclusion

In this paper, in order to predict chaotic time series, the MLP NN was trained by various meta-heuristic methods and fuzzy systems. The performance of training is based on three principles: (1) speed, (2) quality, and (3) stability. Higher speed in training allows the system to be used for high-speed applications. A system with high-quality training has fewer final errors so the system is more reliable. Training stability means more samples in the network. As a result, the network learns more and is not trapped in a local minimum. In the first proposed method, we utilized the BBO algorithm, which increased the speed, quality, and stability of training. In the second proposed method, a fuzzy system was designed by the clustering algorithm, which had a better training and test accuracy than the first proposed method. The simulation results confirmed that to increase accuracy, this method requires more membership functions. Therefore, the second proposed method is more complex than the first proposed method. In addition, the first and second proposed methods outperform the other mentioned meta-heuristic methods in terms of training and testing accuracy by approximately 28.5%-51%, and 98.5%-91.3%, respectively.

Intellectual Property

The authors confirm that they have given due consideration to the protection of intellectual property associated with this work and that there are no impediments to publication, including the timing to publication, with respect to intellectual property.

Funding

No funding was received for this work.

Credit Authorship Contribution Statement

M. Nezhadshahbodaghi: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Data curation, Writing-original draft, Writing - review & Editing, Visualization. **K. Bahmani:** Writing-original draft, Writing - Review & editing, Visualization. **M. R. Mosavi:** Conceptualization, Methodology, Validation, Investigation, Resources, Data curation, Writing -original draft, Writing - Review & editing, Supervision, Project administration. **Diego Martín:** Writing - review & Editing, Supervision.

Declaration of Competing Interest

The authors hereby confirm that the submitted manuscript is an original work and has not been published so far, is not under consideration for publication by any other journal and will not be submitted to any other journal until the decision will be made by this journal. All authors have approved the manuscript and agree with its submission to “Iranian Journal of Electrical and Electronic Engineering”.

References

[1] M. Ardalani-Farsa and S. Zolfaghari, "Chaotic Time Series Prediction with Residual Analysis Methodusing Hybrid Elman-NARX Neural Network", *Neurocomputing*, Vol. 73, No. 13, pp. 2540-2553, 2013.

[2] M. Han and S. Wang, "Analysis and Modeling of Multivariate Chaotic Time Series based on Neural Network", *Expert Systems with Applications*, Vol. 36, No. 2, pp. 1280-1290, 2009.

[3] M. Frank, R. Gencay,and T. Stengos, "International Chaos", *European Economic*

*Review***,** Vol. 32, No. 8, pp. 1569-1584, 1998.

[4] A. Weingessel and K. Hornik, "Local PCA Algorithms", *IEEE Transactions on Neural Networks*, Vol. 11, No. 6, pp. 1242-1250, 2000.

[5] H. Hassanpour, M. M. AlyanNezhadi, and M. Mohammadi, "A Signal Processing Method for Text Language Identification", *Transactions C: Aspects*, Vol. 34, No. 6, pp. 1413-1418, 2021.

[6] B. Samanta, "Prediction of Chaotic Time Series using Computational Intelligence", *Expert Systems with Applications*, Vol. 38, No. 9, pp. 11406-11411, 2011.

[7] S. Haykin, "Kalman Filtering and Neural Networks", *John Wiley & Sons*, 2004.

[8] J. S. R. Jang, C. T. Sun, and E. Mizutani, "Neuro-Fuzzy and Soft Computing -A Computational Approach to Learning and Machine Intelligence", *IEEE Transactions on automatic control*, Vol. 42, No. 10, pp. 1482-1484, 1997.

[9] O. Nelles, "Nonlinear System Identification: from Classical Approaches to Neural Networks and Fuzzy Models", *Springer Science and Business Media*, 2013.

[10] Y. Chen, B. Yang, and J. Dong, "Time-Series Prediction using a Local Linear Wavelet Neural Network", *Neurocomputing***,** Vol. 69, No. 4, pp. 449-465, 2006.

[11] D. T. Mirikitani and N. Nikolaev, "Recursive Bayesian Recurrent Neural Networks for Time-Series Modeling", *IEEE Transactions on Neural Networks*, Vol. 21, No. 2, pp. 262-274, 2010.

[12] N. I. Sapankevych and R. Sankar, "Time Series Prediction using Support Vector Machines: a Survey", *IEEE Computational Intelligence Magazine*, Vol. 4, No. 2, pp. 24-38, 2009.

[13] Y. Zheng, S. Wang, J. Feng, and K. T. Chi, "A Modified Quantized Kernel Least Mean Square Algorithm", *Digital Signal Processing*, Vol. 48, pp. 130-136, 2016.

[14] C. H. López-Caraballo, I. Salfate, J. A. Rojas, M. Rivera, and L. Palma-Chilla, "Mackey-Glass Noisy Chaotic Time Series Prediction by a Swarm-Optimized Neural Network", *Journal of Physics***,** Vol. 720, No. 1, p. 012002, 2016.

[15] J. Ghasemi and J. Esmaily, "A Novel Intrusion Detection Systems based on Genetic Algorithms-suggested Features by the Means of Different Permutations of Labels' Orders",*Transactions A: Basics*, Vol. 30, No. 10, pp. 1494-1502, 2017.

[16] S. Jam, S. Shahbahrami, and S. H. S. Ziyabari, "Parallel Implementation of Particle Swarm Optimization Variants Using Graphics Processing Unit Platform", *Transactions A: Basics*, Vol. 30, No. 1, pp. 48-56, 2017.

[17] Z. Dorrani, H. Farsi, and S. Mohamadzadeh, "Image Edge Detection with Fuzzy Ant Colony Optimization Algorithm", *International Journal of Engineering*, Vol. 33, No. 12, pp. 2464-2470, 2020.

[18] Q. He and L. Wang, "An Effective Co-Evolutionary Particle Swarm Optimization for Constrained Engineering Design Problems", *Engineering Applications of Artificial Intelligence*, Vol. 20, No. 1, pp. 89-99, 2007.

[19] J. Zhang and K. F. Man, "Time Series Prediction using RNN in Multi-Dimension Embedding Phase Space", *IEEE Conference on Systems, Man, and Cybernetics*, pp. 1868-1873, 1998.

[20] E. N. Lorenz, "Deterministic non-Periodic Flows", *Journal of Atmospheric Science***,** Vol. 20, No. 2, pp. 130-141, 1963.

[21] G. Li, X. Stoten, and X. Ren, " Neural Network Feed forward Control for Dynamically Sub structured Systems", *IEEE Transactions on Control Systems Technology*, Vol. 22, No. 3, pp. 944-954, 2014.

[22] X. Yao, Y. Liu, and G. Lin, "Evolutionary Programming Made Faster", *IEEE Transactions on Evolutionary Computation*, Vol. 3, No. 2, pp. 82-102, 1999.

[23] W. Gong, Z. Cai, C. Ling, and H. Li, "A Real-Coded Biogeography-based Optimization with Mutation", *Applied Mathematics and Computation*, Vol. 216, No. 9, pp. 2749-2758, 2010.

[24] H. Chiroma, S. Abdulkareem, A. Abubakar, and T. Herawan, "Neural Networks Optimization through Genetic Algorithm Searches: A Review", *Applied Mathematics and Information Sciences*, Vol. 11, No. 6, pp. 1543-1564, 2017.

[25] K. M. Salama, and A. M. Abdelbar, "Learning Neural Network Structures With Ant Colony Algorithms", *Swarm Intelligence*, Vol. 9, pp. 229-265, 2015.

[26] K. Mehrdad, and M. S. Mesgari, "Application of Meta-Heuristic Algorithms for Training Neural Networks and Deep Learning Architectures: A Comprehensive Review", *Neural Processing Letters*, pp. 1-104, 2022.

[27] R. R. Chandan, S. Soni, A. Raj, V. Veeraiah, D. Dhabliya, S. Pramanik, and A. Gupta, "Genetic Algorithm and Machine Learning", *Advanced Bioinspiration Methods for Healthcare Standards*, pp. 167-182, 2023.

[28] M. Agarwal, K. G. Suneet, and K. K. Biswas, "Genetic Algorithm Based Approach to Compress and Accelerate the Trained Convolution Neural Network Model", *International Journal of Machine Learning and Cybernetics*, pp. 1-17, 2023.

[29] E. Bas, E. Egrioglu, and E. Kolemen, "Training Simple Recurrent Deep Artificial Neural Network For Forecasting Using Particle Swarm Optimization", *Granular Computing*, Vol. 7, No.2, pp. 411-420, 2022.

M. Nezhadshahbodaghi received his B.S. and M.S. degrees in Electronic Engineering from respectively Shahid Bahonar University of Kerman in 2016 and Iran University of Science and Technology (IUST) in 2018, Tehran, Iran. He is currently a Ph.D. student of IUST Department of Electrical Engineering. His research interests include signal processing, artificial intelligence, INS, visual odometry, and GPS applications.

K. Bahmani received his B.S. degree in Electronic Engineering from University of Kurdistan in 2017. He is currently a M.S. student of Iran University of Science and Technology (IUST) Department of Electrical Engineering. His research interests include signal processing, GPS applications, 3-D FPGAs, and artificial intelligence.

M. R. Mosavi received his B.S., M.S., and Ph.D. degrees in Electronic Engineering from Iran University of Science and Technology (IUST), Tehran, Iran in 1997, 1998, and 2004, respectively. He is currently a faculty member (full professor) of the Department of Electrical Engineering of IUST. He is the author of more than 500 scientific publications in journals and international conferences in addition to 12 academic books. His research interests include circuits and systems design. He is also editor-in-chief of "Iranian Journal of Marine Technology" and editorial board member of "Iranian Journal of Electrical and Electronic Engineering" and "GPS Solutions".

D. M. De Andrés received the B.Sc. degree in computer engineering and the M.Sc. degree in computer science from the Department of Informatics, Carlos III University of Madrid, Spain, where he received his Ph.D. degree in 2012. Now, he is a lecturer at the Department of Telematics of the Technical University of Madrid (UPM). His main research subjects, within the GISAI groups at UPM, are internet of things, cyber-physical systems, physically unclonable functions (PUFs), blockchain, knowledge management, information retrieval, and research methods.