Arabian Journal for Science and Engineering (2022) 47:9557–9581
https://doi.org/10.1007/s13369-021-06286-z

RESEARCH ARTICLE-COMPUTER ENGINEERING AND COMPUTER SCIENCE

An Improved Animal Migration Optimization Algorithm to Train the Feed-Forward Artificial Neural Networks

Şaban Gülcü[1]

Received: 9 December 2020 / Accepted: 5 October 2021 / Published online: 10 November 2021

Abstract
The most important and demanding part of the artificial neural network is the training process which involves finding the most suitable values for the weights in the network architecture, a challenging optimization problem. Gradient approaches and the meta-heuristic approaches are two methods extensively used to optimize the weights in the network. Gradient approaches have serious disadvantages including getting stuck in local optima, inadequate exploration, etc. To overcome these disadvantages, meta-heuristic approaches are preferred in training the artificial neural network instead of gradient methods. Therefore, in this study, an improved animal migration optimization algorithm with the Lévy flight feature was proposed to train the multilayer perceptron. The proposed hybrid algorithm is named IAMO-MLP. The main contributions of this article are that the IAMO algorithm was developed, the IAMO-MLP algorithm can successfully escape from local optima, and the initial positions did not affect the performance of the IAMO-MLP algorithm. The enhanced algorithm was tested and validated against a wider set of benchmark functions and indicated that it substantially outperformed the original implementation. Afterward, the IAMO-MLP was compared with ten algorithms on five classification problems (xor, balloon, iris, breast cancer, and heart) and one real-world problem in terms of mean squared error, classification accuracy, and nonparametric statistical Friedman test. According to the results, the IAMO was successful in training the multilayer perceptron.

Keywords Animal migration optimization algorithm · Artificial neural networks · Civil engineering · Lévy flight · Multilayer perceptron · Training of artificial neural networks

1 Introduction

The artificial neural network (ANN) is one of the popular topics in machine learning and artificial intelligence. The design of ANNs was inspired by the working principle of the biological nervous system in the 1940s. It has been used in many areas as a result of studies that gained speed in the 1980s, and its success has been proven [1]. Nowadays, the ANN is a method frequently used in engineering studies due to its effective problem-solving strategy for complex and difficult problems. It is formed by connecting many artificial process elements (neurons) in its layers. ANNs learn from the samples of the given problem and then decides by using the information it has obtained when it encounters samples of the problem that it has never been seen before [2]. The method has been successfully used in areas such as classification [3], system modeling [4], face recognition [5], speech recognition [6], and optimization [7].

The most important feature of ANNs is the ability to make inferences for different conditions using the experience gained by learning from the information. ANNs consist of two stages, training (learning) and testing. In the first step, ANN is trained with training data. Then, the network is tested using the test data to evaluate the performance of the trained ANN [8]. The most important and demanding part of the method is the training process of the network which involves finding the most suitable values for the weights in the network architecture, a challenging optimization problem. As emphasized in [9], two approaches are extensively used to optimize weights in the network: gradient approaches and meta-heuristic approaches. There are some disadvantages in gradient approaches: (i) They can easily get stuck in local optima. (ii) The value of the learning rate in gradient approaches is very important and affects the performance of the algorithm because if the value of the learning rate is too

✉ Şaban Gülcü
sgulcu@erbakan.edu.tr

[1] Computer Engineering Department, Necmettin Erbakan University, Konya, Turkey

small, the training process takes a long time and can get stuck, and if it is too large, the training process takes a short time and the algorithm converges prematurely. (iii) The wider the search space is, the worse the gradient approaches generate results. (iv) The gradient approaches depend heavily on the initial values of the weights. Moreover, the same initial values in different runs generate the same results. Because of these problems, meta-heuristic approaches can be preferred in training ANNs instead of gradient methods. Computer scientists have developed many meta-heuristic algorithms inspired by certain behaviors of creatures in nature to find solutions to optimization problems. The animal migration optimization (AMO) algorithm was designed inspired by the animal migration behavior of individuals that can be found in all major animal groups including birds, mammals, and insects. The developer of the AMO algorithm showed that this algorithm is able to improve the initial random population, converge toward the global optimum, provide very competitive results compared to other well-known algorithms in the literature, and solve different kinds of optimization problems. Thanks to the success of the AMO algorithm, it has been applied to many different optimization problems such as association rule mining [10], clustering [11], unmanned aerial vehicles placement [12], mobile ad hoc networks [13], the optimal power flow problem [14], the traveling salesman problem [15], reinforcement of bridges [16], and multilevel image thresholding [17]. Therefore, these applications motivated our attempts to employ the AMO algorithm for training ANNs. In this study, the improved animal migration optimization (IAMO) algorithm with the Lévy flight feature was developed and used in the process of finding optimum weights of the network to increase the success of ANN.

The most important and demanding part of ANNs is the training process of the network. To increase the success of the network, the network should be updated with optimum weights. In the literature, the training of ANNs has been carried out using some different optimization algorithms. The success of these optimization algorithms in training ANNs was determined by comparing them in the solution of different world problems. Ibrahim Aljarah et al. [18] proposed the whale optimization algorithm for training ANNs. This algorithm was proven to be able to solve a wide variety of optimization problems and surpass existing algorithms. Twenty datasets with different difficulty levels were used to test the success of the algorithm in the training of the feed-forward ANN. The success of the algorithm was determined by comparing it with the backpropagation (BP) algorithm and six different evolutionary optimization algorithms. It was shown that the proposed model performs better than the other algorithms in terms of both local optimum avoidance and convergence speed. Ilyas Benmessahel et al. [19] proposed an advanced detection approach by combining the multi-verse optimization algorithm and an ANN for the intrusion detection system. The main idea of the study was to train feed-forward multilayer ANN using the multi-verse optimization algorithm to detect new intrusions. NSL-KDD and UNSW-NB15 datasets were used to test the success of the approach, and the results for UNSW-NB15 were better than the results for NSL-KDD. Moreover, the proposed method outperforms the ANN trained using the particle swarm optimization (PSO) algorithm. Sankhadeep Chatterjee et al. [20] proposed an ANN trained using the PSO algorithm to solve the problem of predicting the failure probability of multi-story concrete structures by determining the causes of their structural failure. In experimental studies, a database of multi-story reinforced concrete structures consisting of 150 multistoried buildings was used. The proposed method was compared to the multilayer feed-forward ANN model. The proposed method demonstrated a better success rate than the multilayer feed-forward ANN model. Thus, the success of the proposed method was proved. Gülay Tezel et al. [21] used the artificial algae algorithm (AAA) as a tool to optimize the weights of the ANN. The ANN and AAA were combined in such a way that the training phase of the ANN was performed by the AAA. The proposed model was tested in three data sets (iris, thyroid, and dermatology) taken from the University of California, Irvine (UCI) machine learning database. The results were compared with the multilayer perceptron algorithm with backpropagation in terms of mean absolute error. It has been stated that the models where the proposed method would be applied can provide a reduction of up to 96% in mean squared error. In [22], the moth flame optimization algorithm is proposed for training feed-forward multilayer ANN. The algorithm is used to produce weight and bias values that ensure to obtain minimum error and a high classification rate. Five classification datasets were used to evaluate the performance of the proposed method which was compared with the genetic algorithm (GA), PSO, ant colony optimization (ACO), and evolution strategy. The experimental results proved that the moth flame optimization algorithm solves the local minima problem and achieves high accuracy. Najmeh Sadat Jaddi et al. [23] proposed a hybrid method based on the bat optimization algorithm (BAT) and the ANN. The BAT algorithm produces the weight and bias values of the ANN with minimum error and high classification success. To test the performance of the proposed approach in terms of classification and prediction accuracy, six classification and two time series data sets were used. The statistical tests showed that the proposed method produces good results. The method was applied to a real-world problem to predict future values of rain data, and the results showed the method's success. In [24], the particle swarm optimization algorithm was used in training the ANN. In the experimental studies, four datasets from the UCI machine learning database were used. In [25], the grey wolf optimization (GWO) algorithm

was applied to train a multilayer perceptron (GWO-MLP). Eight datasets were used in the experiments, and the GWO-MLP algorithm was compared to the PSO-MLP based on the PSO algorithm, the GA-MLP based on the GA algorithm, the ACO-MLP based on the ACO algorithm, the ES-MLP based on the evolution strategy algorithm, and the PBIL-MLP based on the population-based incremental learning algorithm. In [26], the states of matter search (SMS) algorithm was used to train the ANN. In the experimental studies, five datasets from the UCI machine learning database were used, and the SMS-MLP algorithm was compared to six algorithms in the literature. According to the experimental results, the SMS-MLP algorithm outperformed the six algorithms. In [27], an improved version of the beetle antennae search algorithm was proposed. The improved beetle antennae search algorithm outperformed the original implementation on the benchmark functions. The improved beetle antennae search algorithm is used to optimize the parameters of the adaptive neuro-fuzzy inference system and to improve the performance of the prediction model. The improved beetle antennae search algorithm was evaluated for COVID-19 case prediction using the World Health Organization's official data. The overfitting problem of convolutional neural networks was overcome by means of properly selecting a regularization parameter known as a dropout in the context of convolutional neural networks using meta-heuristic-driven techniques. In [28], the overfitting problem of convolutional neural networks was overcome by means of properly selecting a regularization parameter known as a dropout in the context of convolutional neural networks using four distinct meta-heuristic techniques (particle swarm optimization, bat algorithm, cuckoo search and firefly algorithm). The results of four optimization methods were compared with the default dropout-less and the default dropout ratio. The experiments were carried out over four public datasets in the context of image classification. The experimental results showed that the meta-heuristic-based dropout convolutional neural network is very promising.

In this study, we propose the IAMO algorithm which is used in the process of finding the optimum weights of the network to increase the success of the ANN. The contribution of this article is that the IAMO algorithm has the Lévy flight strategy. The proposed hybrid algorithm is called IAMO-MLP, and 13 benchmark functions, five classification problems (xor, balloon, iris, breast cancer, heart) and one real-world problem (a prediction problem in civil engineering) are used in the experiments. On the benchmark functions, the IAMO algorithm was compared with the original AMO algorithm. On the classification problems, the results of the IAMO-MLP algorithm were compared in detail with the results of the AMO-MLP algorithm, the BAT-MLP based on the bat optimization algorithm, the SMS-MLP [26] based on the states of matter search optimization [29] algorithm, and the BP algorithm. The IAMO-MLP algorithm was also compared to the GWO-MLP, ACO-MLP, GA-MLP, PBIL-MLP, PSO-MLP, and ES-MLP algorithms in [25]. On the real-world problem, the results of the IAMO-MLP algorithm were compared with the results of the AMO-MLP, BAT-MLP, SMS-MLP, and BP. Moreover, the algorithms were run using different numbers of neurons in the hidden layer. The experimental results showed that the proposed IAMO-MLP algorithm is more efficient than the others.

The main contributions of this article are as follows: (1) The proposed IAMO algorithm has the Lévy flight feature. (2) This article employs the AMO and IAMO to train the ANN for the first time, and the proposed algorithm is called IAMO-MLP. (3) The IAMO-MLP algorithm has the ability to escape successfully from local optima. (4) The initial parameters and positions do not affect the performance of the IAMO-MLP algorithm. (5) The features of the IAMO-MLP algorithm are the simplicity, requiring only a few parameters, and solving a wide array of problems.

This article is organized as follows. Information about the training process of ANN and some meta-heuristic algorithms employed in training ANNs is provided in the Introduction section. Information about the ANN algorithm, the AMO algorithm, the proposed IAMO algorithm, and the IAMO-MLP algorithm (training ANN using IAMO algorithm) is provided in the Material and Methods section. The experimental results of the algorithms on the benchmark functions, the classification problems and the real-world problem are given in the Experimental Results section. Finally, the results obtained are evaluated, and suggestions about future studies are presented in the Conclusion section.

2 Material and Method

This section gives detailed information about the ANN algorithm, the AMO algorithm, the proposed IAMO algorithm and the IAMO-MLP algorithm.

2.1 Artificial Neural Networks

The first studies on ANN were carried out in the late nineteenth century and early twentieth century. The first study looked at physics, psychology, and neuropsychology [30] emphasizing the general theory of learning, perception, and conditioning. Over time, new developments such as the BP algorithm which is used to train multilayer networks further strengthened the studies on ANN. Over the years, many articles have been written about ANNs, and developed ANN models have been used in many distinct areas.

ANN is frequently mentioned in the area of machine learning and artificial intelligence nowadays. The main areas in which ANN is used are classification, clustering, pattern

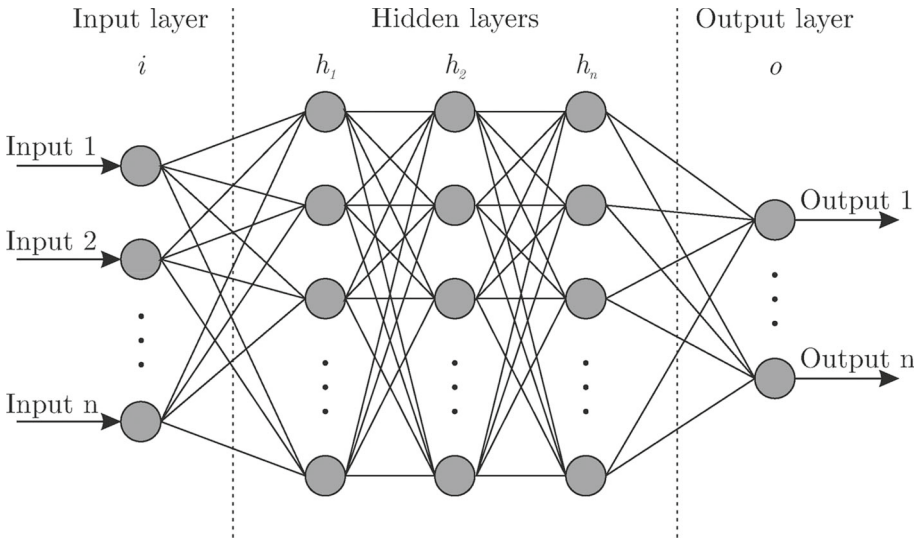

Fig. 1 Structure of the MLP [37]

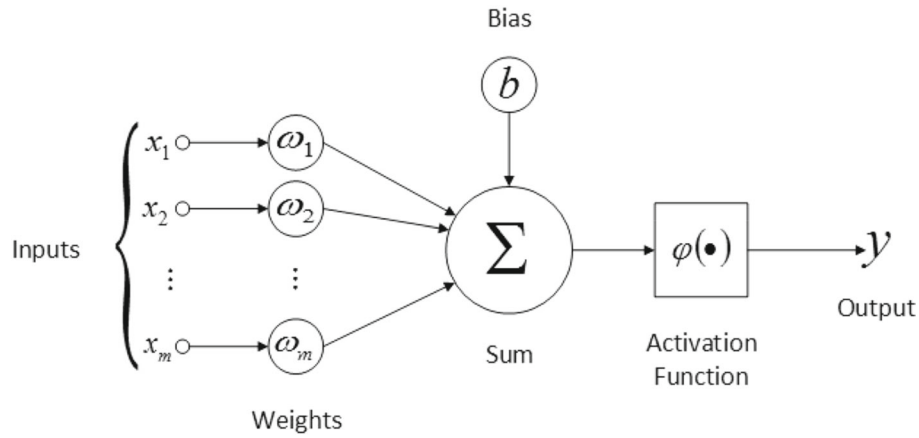

Fig. 2 Neuron in the MLP [38]

recognition, estimation, and optimization. ANN is used for problem-solving not only in engineering but also in many other fields including finance [31], medicine [32], physics [33], transportation [34], statistic [35], and mathematics [36]. ANN has many features that cause it to be used in many different areas including the ability to produce nonlinear models, the ability to learn and generalize, and the applicability to different problems.

ANN is a problem-solving strategy that consists of artificial nerve cells similar to the structure of biological nerve cells. The multilayer perceptron (MLP) is a version of ANN which consists of three main layers, namely input, hidden, and output. The general structure of MLP is given in Fig. 1. The input layer is where the input data in the data set of the problem are given to the MLP. In the input layer, there are as many cells as the input data in the dataset of the problem. The data given to the input layer are transmitted to the next layer in order to process the data. The hidden layer processes the data taken from the input layer and transmits the results to the output layer. While some MLPs have only one hidden layer, some have more than one. The number of neurons in the hidden layer is independent of the number of neurons in the input layer and the output layer. The complexity of the algorithm and the solution duration of the problem increase along with the increase in the number of neurons in the hidden layer. But this situation enables the ANN to be used in solving more complex problems. The output layer is the layer where the output of the network is produced by processing the data coming from the hidden layers [2].

MLP consists of artificial nerve cells, and an artificial nerve cell consists of five main parts: inputs, weights, addition (aggregation) function, activation function, and outputs. The data coming to neurons are called input. Weights are used to adjust the effect of inputs to artificial nerve cells on the output of the problem.

The value of a weight can be a positive value, a negative value, or zero. If the weight value is 0, the inputs do not affect the output of the neuron. The input data of artificial nerve cells are multiplied by the weights of the connections, and the net input is calculated using the addition function. The bias value is also added to the net. The activation function produces the output of the artificial nerve cell by process-

ing the net input value obtained from the addition function. This process is shown in Fig. 2. When determining the activation function, nonlinear activation functions are generally preferred. Another point to consider when determining the activation function is that the derivative of the function should be easily calculated. The sigmoid activation function is generally preferred in the MLP model, which is widely used today. The sigmoid activation function given in Eq. (1) is a continuous, nonlinear, and easy derivative function. This function generates a value between 0 and 1 for each input value. The value obtained using the activation function corresponds to the output value of the artificial nerve cell [2]. In addition, all datasets are normalized by using the min—max normalization function given in Eq. (2) to eliminate the effect of attributes that may have different effective rates on the classification [18].

$$\text{Sigmoid}(x) = \frac{1}{1 + e^{-x}} \tag{1}$$

$$x' = \frac{(x - x_{\min})}{(x_{\max} - x_{\min})} \tag{2}$$

where x' is the normalized value of x which is in the range $[x_{\min}, x_{\max}]$. The normalized value x' will be in the range [0, 1].

2.2 The Animal Migration Optimization Algorithm

Migration, is a common animal behavior arising out of the animal's survival efforts, is a behavioral movement that transports animals to new habitats. Animal migrations are the movements of individuals over long distances, usually seasonally. Migration is a vital activity found in all animal groups including birds, mammals, and insects. Climate and insufficient food are the main reasons that force animals to migrate. During the migration process, individuals in animal groups act by following three main instructions: (1) move in the same direction as neighbors, (2) stay close to neighbors, and (3) avoid colliding with neighbors. Recent studies on starlings have shown that each bird changes its position in a direct relationship to six or seven animals around it, regardless of how close or how far the animals are. These interactions between starlings in a flock are based on a topological rule [39]. Inspired by these rules, a new swarm-based algorithm called AMO has been proposed by Li, Zhang, and Yin [40].

The main idea of the AMO algorithm is applied through concentric regions around each animal. In the thrust zone, the animal concerned will try to distance itself from its neighbors to avoid the collision. Moving away a little, the animal will try to align its direction of movement with its neighbors in the zone of harmony. In the outermost attraction zone, the animal concerned will try to move toward its neighbor.

The AMO algorithm is a swarm-based optimization algorithm developed to solve global optimization problems inspired by animal migration behavior that can be found in all large animal groups such as birds and fish. Two idealized assumptions are used to describe the basic function of the algorithm: (1) The animal with the highest quality in the herd will be defined as the leader animal, and the leader animal will be protected in future generations. (2) The number of animals in the herd is fixed, and each animal will be replaced with a new individual with probability P_a. In this case, the animal will leave the group, and then a new animal will join the group.

The AMO algorithm consists of two processes: the migration process and the population updating process. The migration process covers how the animals move from the current location to the new location. Animals must obey three topological rules of migration in this process. (1) move in the same direction as neighbors, (2) stay close to neighbors, and (3) avoid colliding with neighbors. When these rules are followed, animals migrate in an optimized way. The migratory animal population consists of a range of animal herds as follows. This migration population is shown in Eq. (3),

$$\text{population} = \{X_1, X_2, \ldots, X_{NP}\} \tag{3}$$

where NP and X_i represent the population size and an individual in the population, respectively. Each individual in the population consists of a d-dimensional vector accepted as the elements of a solution within the maximum and minimum limits in the search space. At the beginning of the algorithm, a value within the maximum and minimum limits is assigned to the d-dimensional vector of each individual in the population using Eq. (4),

$$X_i = X_{\min} + \text{rand} * (X_{\max} - X_{\min}) \tag{4}$$

where X_i, $X_{\min}$ and $X_{\max}$ represent an individual in the population, the minimum bounds in the search space, and maximum bounds in the search space, respectively. $rand$ is a uniformly distributed random number between 0 and 1.

A local neighbor cluster is needed to determine the new location of each individual in the population. The ring topology scheme given in Fig. 3 is used to define this cluster.

In Fig. 3, the length of the adjacent cluster is set to be five for each dimension of the individual. If the animal index is i, the neighbor cluster will be created with animals having the indices $i-2$, $i-1$, i, $i + 1$, and $i + 2$. If the animal index is 1, the neighbor cluster will be created with animals having indices NP-1, NP, 1, 2, and 3. NP represents the total number of individuals in the migration population. Each individual whose neighbor cluster is determined calculates the new position according to the position of the individuals in the neighbor cluster by obeying the rules to be considered

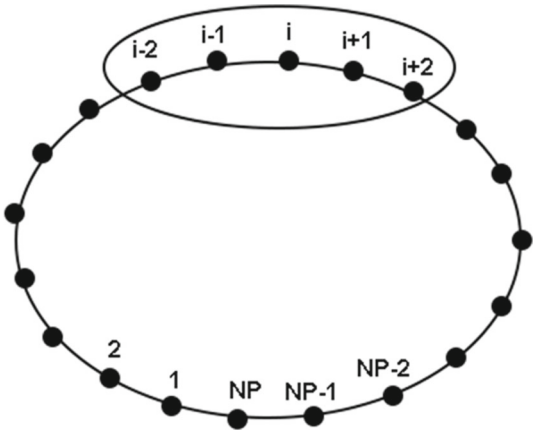

Fig. 3 Concept of the local neighborhood of an individual

during the migration. The new position of the individual is calculated using Eq. (5),

$$X_i^{t+1} = X_i^t + \delta * \left(X_{\text{neighbor}}^t - X_i^t \right) \tag{5}$$

where X_{neighbor}^t is the current position of the neighbor selected from the cluster. X_i^t and X_i^{t+1} are the positions of the ith individual in the iterations t and $t + 1$, respectively, and δ is a random value between 0 and 1. This value may vary according to different problems in the real world.

The population updating process covers how some animals left the herd and how new animals are added to the herd. A probability value is given to each individual in the population according to their fitness value. While the probability value for the most compatible individual in the population is 1, this value is $1/NP$ for the most incompatible individual, and NP is the population size. When the probability value of the individual is less than the randomly generated value, a new individual is created using Eq. (6),

$$X_i^{t+1} = X_{r1}^t + \text{rand} * \left(X_{\text{best}}^t - X_i^t \right) + \text{rand} * \left(X_{r2}^t - X_i^t \right) \tag{6}$$

where X_i^t and X_i^{t+1} are the positions of the ith individual in the iterations t and $t + 1$, respectively. X_{r1}^t and X_{r2}^t are randomly selected individuals from the population, X_{best}^t is the leading animal with the high quality of the position, and *rand* is a random value between 0 and 1. If the quality of the new individual is better than the current individual, then the current individual is removed from the population and the new individual is added to the population. The flowchart of the AMO algorithm is shown in Fig. 4, and the pseudo-code of the AMO algorithm is presented in [40].

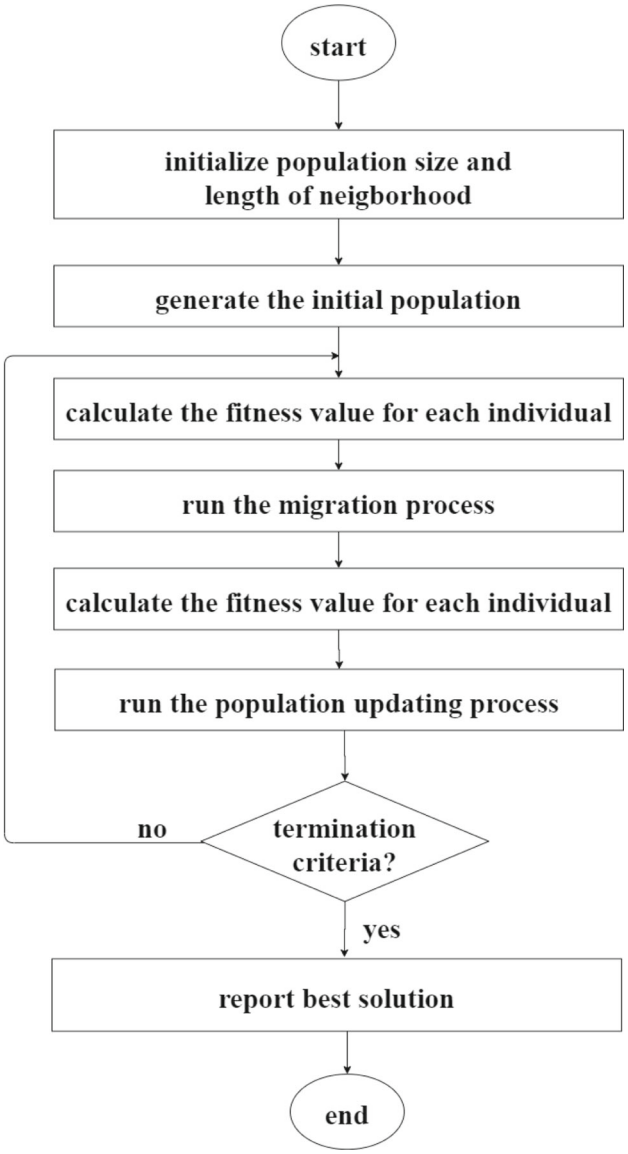

Fig. 4 Flowchart of the AMO algorithm

2.3 The Improved Animal Migration Optimization Algorithm

Although the AMO algorithm is one of the recent metaheuristic algorithms and shows good performance in solving optimization problems, it has some bottlenecks. According to [41], the performance of AMO is degraded rapidly when the dimensionality is larger than 30. According to [42], the bottlenecks of the AMO algorithm are premature convergence and falling into local optima. In order to overcome these bottlenecks, we have proposed the IAMO algorithm which has the Lévy flight strategy.

The Lévy flight developed by Paul Lévy is a version of the random walk model. It is based on the Lévy distribution which is a continuous probability distribution. Studies show

that the distance traveled by many animals, including bees, ants and fish, in foraging behavior corresponds to the Lévy distribution [43]. The advantage of the Lévy flight is that it optimizes the distance in foraging. Therefore, we applied the Lévy flight strategy to the individuals in the IAMO algorithm and as a result of this, the Lévy flight improved the diversification and intensification in the IAMO algorithm. Normally, the position of an individual is updated using Eq. (5). But in IAMO, if an individual cannot improve its position in several consecutive iterations, this individual updates its position using Eq. (7) which contains the Lévy flight strategy.

$$X_i^{t+1} = X_i^t + \text{Levy}(D) \times X_i^t \tag{7}$$

where X_i^t and X_i^{t+1} are the positions of the ith individual in the iterations t and $t+1$, respectively. D is the dimension of the position X_i^t. The Lévy flight is calculated using Eq. (8),

$$\text{Levy}(x) = 0.01 \times \frac{r_1 \times \sigma}{|r_2|^{1/\beta}} \tag{8}$$

where β is a constant (1.5), and r_1 and r_2 are the random numbers between 0 and 1. σ is calculated using Eq. (9),

$$\sigma = \left(\frac{\Gamma(1+\beta) \times \sin\left(\frac{\pi\beta}{2}\right)}{\Gamma\left(\frac{1+\beta}{2}\right) \times \beta \times 2^{\left(\frac{\beta-1}{2}\right)}} \right)^{1/\beta} \tag{9}$$

where Γ is calculated using Eq. (10).

$$\Gamma(x) = (x-1)! \tag{10}$$

In contrast to AMO, each individual in IAMO has also a *counter* variable that records the number of consecutive iterations where the individual cannot be improved. There is also a *threshold* variable in IAMO that controls the activation of the Lévy flight. If the value of the *counter* variable of an individual exceeds the *threshold* value, then the Lévy flight strategy is applied to this individual using Eq. (7).

Figure 5 shows the flowchart of the IAMO algorithm. The pseudo-code of the IAMO algorithm is shown in Fig. 6. Firstly, the parameters are initialized, and the population is randomly generated. The fitness value of each individual is calculated, and the global best position is determined. Secondly, the migration process of the algorithm, which covers how the animals move from the current location to the new location, starts. An individual updates its position with the help of its neighbors. If an individual cannot improve its fitness value in several consecutive iterations, this individual updates its position using the Lévy flight strategy, and the *counter* value of the individual is reset. The fitness value of the new position of each individual is calculated, and, if the new position is better than the current position, then the migration of the individual occurs. Otherwise, the individual keeps using its current position, and the *counter* value is incremented. After the migration process is finished, the global best position in the herd is updated. Thirdly, the population updating process, which covers how some animals left the herd and how new animals are added to the herd, starts. The probability *Pa* of each individual is calculated according to their fitness value. While the probability value for the most compatible individual in the population is 1, and this value is 1/*NP* for the most incompatible individual. *NP* is the population size. When the probability value of the individual is less than a randomly generated value between 0 and 1, then a new individual is created using Eq. (6). The fitness values of all individuals are calculated. If the position of the new individual is better than the position of the current individual,

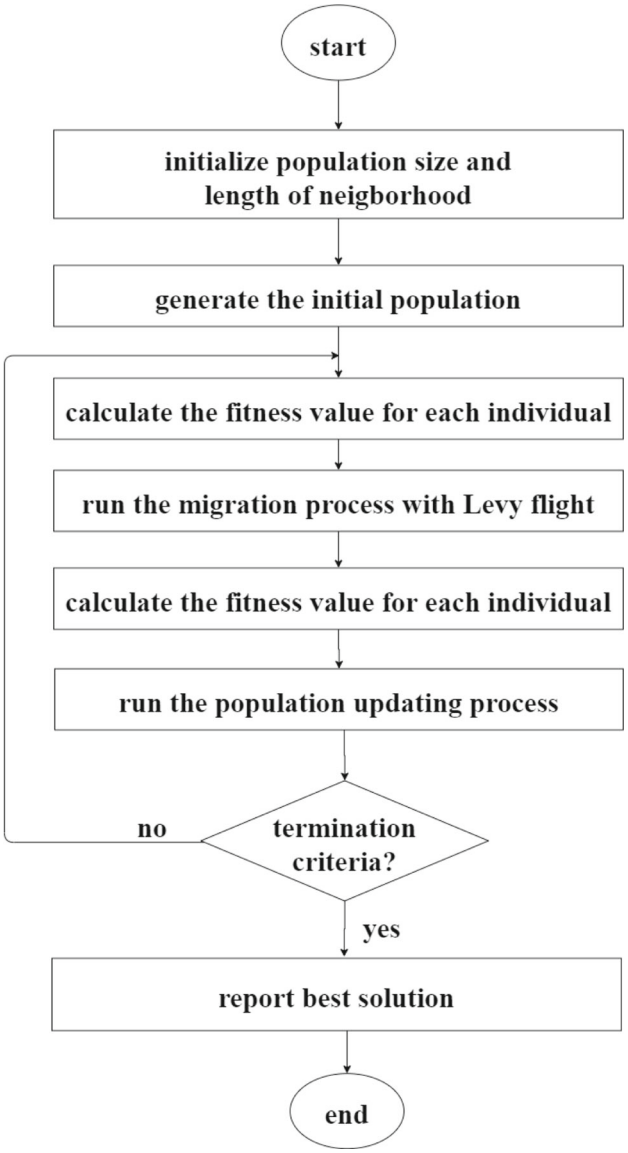

Fig. 5 Flowchart of the IAMO algorithm

Algorithm 1: The IAMO Algorithm

```
Initialize the parameters NP, D, threshold, counter
Initialize population using Eq. (4)
Calculate fitness for each individual
int t=0;                    // iteration counter
repeat:
    for i=1 to NP do
        if counter[i] <= threshold then
            Find the neighbors of the individual i
            Select randomly an individual X_neighbor^t from its neighbors
            Calculate new position X_i^{t+1} of the individual i using Eq. (5)
        else
            Calculate new position X_i^{t+1} of the individual i using Eq. (7) /* Lévy flight */
            counter[i]=0;
        end if
    end for
    for i=1 to NP do
        Calculate the new fitness of the individual i
        if X_i^{t+1} is better than X_i^t then
            X_i^t = X_i^{t+1};
            counter[i]=0;
        else
            counter[i]= counter[i]+1;
        end if
    end for
    Run population updating process using Eq. (6)
    for i=1 to NP do
        Calculate the new fitness of the individual i
        if X_i^{t+1} is better than X_i^t then
            X_i^t = X_i^{t+1};
            counter[i]=0;
        else
            counter[i]= counter[i]+1;
        end if
    end for
    Update the best solution achieved so far
    t=t+1;
until the stopping criterion is met
return the best solution
```

Fig. 6 Pseudo-code of the IAMO algorithm

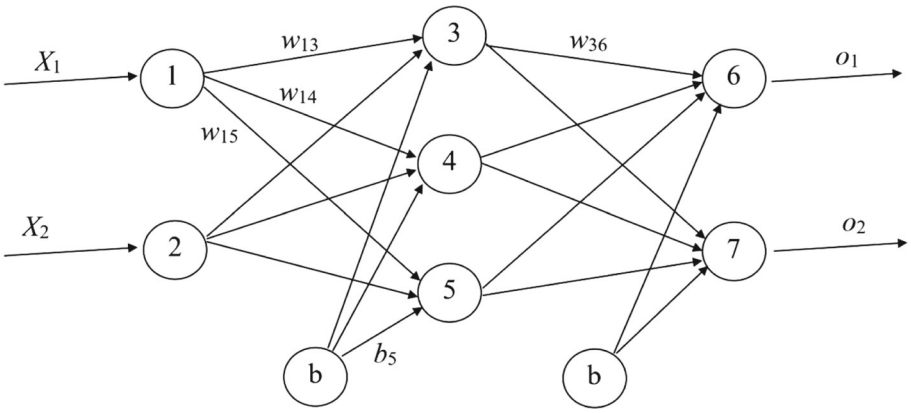

Fig. 7 MLP with the 2-3-2 structure

w_{13}	w_{14}	w_{15}	w_{23}	w_{24}	.	.	.	b_3	b_4	b_5	b_6	b_7

Fig. 8 Position vector of the individual for the weights and biases

then the current individual leaves the herd, and the new individual is added to the herd. Otherwise, the current individual keeps staying within the herd, and the *counter* value is incremented. After the population updating process is finished, the global best position in the herd is updated. Thus, one iteration is completed. The migration and population updating

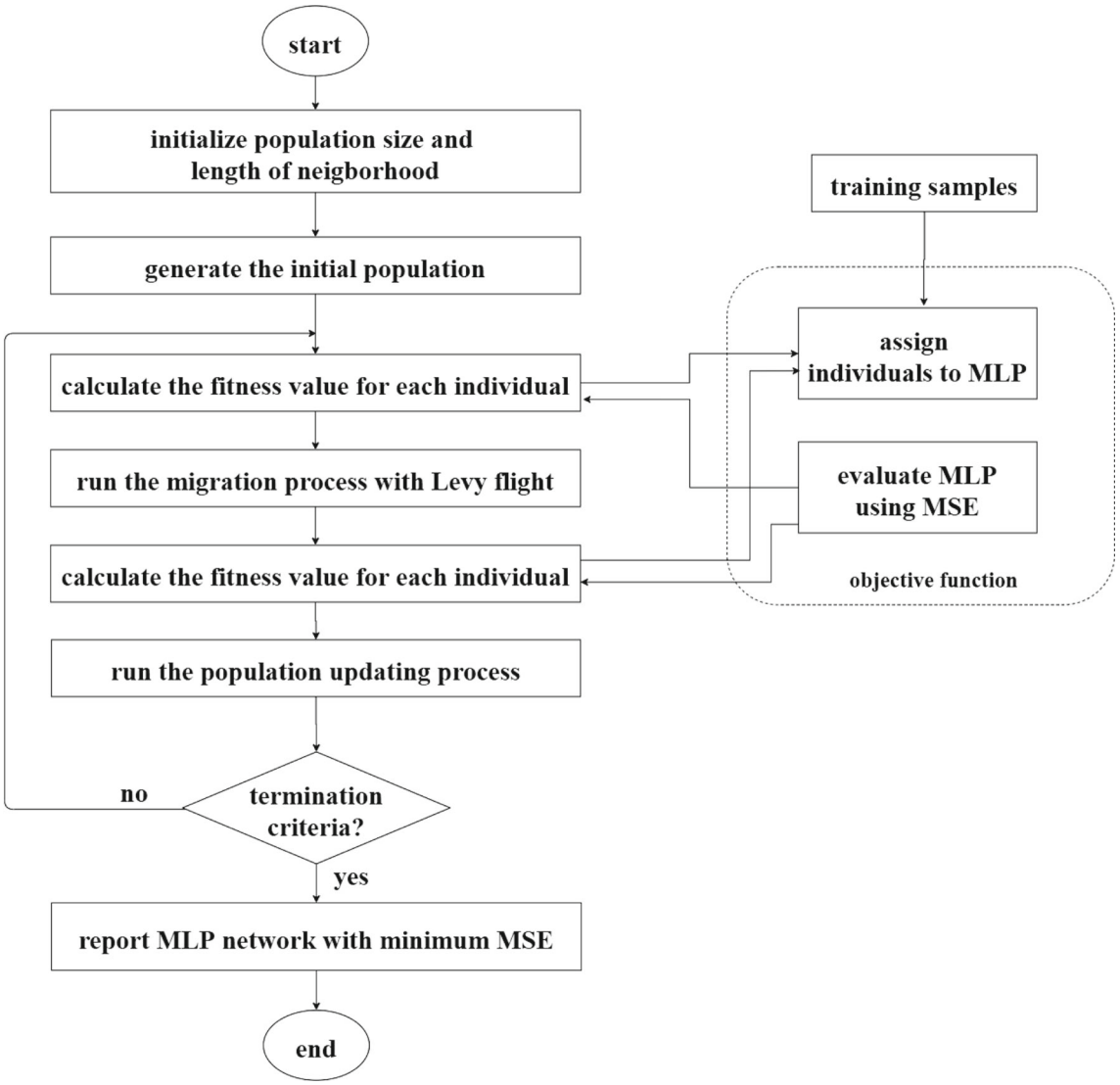

Fig. 9 Flowchart of the IAMO-MLP algorithm

processes are run consecutively until the termination criteria are met. Finally, the global best position is reported at the end of the algorithm.

2.4 Training Multilayer Perceptron using Improved Animal Migration Optimization Algorithm

In this work, a hybrid algorithm (IAMO-MLP) is proposed to train the multilayer perceptron (MLP) using the IAMO algorithm. In the proposed IAMO-MLP algorithm, the IAMO algorithm optimizes the weights and biases of the MLP. In the proposed IAMO-MLP algorithm, each individual with a d-dimensional solution in the migration population represents a candidate solution which is a d-dimensional vector. This candidate solution vector (position vector) represents the structure of the MLP, namely the weights and biases of the MLP. The position vector consists of four parts. The first part contains the weight values between the neurons in the input layer and the hidden layer. The second part contains the weight values between the neurons in the hidden layer and the output layer. The third part contains the bias values of the neurons in the hidden layer. The fourth part contains the bias values of the neurons in the output layer. An example is shown in Fig. 8. Figure 7 shows an exemplar of the MLP with the 2-3-2 structure, and Fig. 8 shows the position vector for this MLP. In Fig. 7, X_i, o_i, w_ij and b_i represent the inputs, the outputs, the weights, and the biases of the MLP, respectively. The candidate solution vector is the same size for all individuals in the population and is equal to the total number

Table 1 Benchmark functions, dimensions, global minimums, and search ranges

Function	D	$f_{\min}$	Range
$f_1 = \sum_{i=1}^{D} x_i^2$	30	0	[− 100, 100]
$f_2 = \sum_{i=1}^{D} \lvert x_i \rvert + \prod_{i=1}^{D} \lvert x_i \rvert$	30	0	[− 10, 10]
$f_3 = \sum_{i=1}^{D} \left(\sum_{j=1}^{i} x_j \right)^2$	30	0	[− 100, 100]
$f_4 = \max_i \{ \lvert x_i \rvert, 1 \le i \le D \}$	30	0	[− 100, 100]
$f_5 = \sum_{i=1}^{D-1} \left[100 \left(x_{i+1} - x_i^2 \right)^2 + (x_i - 1)^2 \right]$	30	0	[− 30, 30]
$f_6 = \sum_{i=1}^{D} (x_i + 0.5)^2$	30	0	[− 100, 100]
$f_7 = \sum_{i=1}^{D} i x_i^4 + \text{random}[0, 1)$	30	0	[− 1.28, 1.28]
$f_8 = \sum_{i=1}^{D} -x_i \sin\left(\sqrt{\lvert x_i \rvert} \right)$	30	− 418.9829*D	[− 500, 500]
$f_9 = \sum_{i=1}^{D} \left[x_i^2 - 10 \cos(2\pi x_i) + 10 \right]$	30	0	[− 5.12, 5.12]
$f_{10} = -20 \exp\left(-0.2 \times \sqrt{\frac{1}{D} \sum_{i=1}^{D} x_i^2} \right) - \exp\left(\frac{1}{D} \sum_{i=1}^{D} \cos(2\pi x_i) \right) + 20 + e$	30	0	[− 32, 32]
$f_{11} = \frac{1}{4000} \sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D} \cos\left(\frac{x_i}{\sqrt{i}} \right) + 1$	30	0	[− 600, 600]
$f_{12} = \frac{\pi}{D} \left\{ 10 \sin(\pi y_i) + \sum_{i=1}^{D-1} (y_i - 1)^2 \left[1 + 10 \sin^2(\pi y_{i+1}) \right] + (y_D - 1)^2 \right\} + \sum_{i=1}^{D} u(x_i, 10, 100, 4)$ $y_i = 1 + \frac{x_i + 1}{4}$ $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, & x_i > a \\ 0, & -a < x_i < a \\ k(-x_i - a)^m, & x_i < a \end{cases}$	30	0	[− 50, 50]
$f_{13} = 0.1 \left\{ \sin^2(3\pi x_1) + \sum_{i=1}^{D} (x_i - 1)^2 \left[1 + \sin^2(3\pi x_i + 1) \right] + (x_D - 1)^2 \left[1 + \sin^2(2\pi x_D) \right] \right\} + \sum_{i=1}^{D} u(x_i, 5, 100, 4)$	30	0	[− 50, 50]

of weights and biases that make up the network. The length of the candidate solution vector is calculated using Eq. (11)

$$L = (k * l) + l + (l * m) + m \tag{11}$$

where L, k, l, and m represent the length of the vector, the number of neurons in the input layer, the number of neurons in the hidden layer, and the number of neurons in the output layer, respectively.

The IAMO-MLP algorithm optimizes the weights and biases of MLP according to the inputs–outputs pattern. To find the optimum values, the IAMO-MLP algorithm tries to minimize the error between the real outputs and predicted outputs. The mean squared error (MSE) shown in Eq. (12) is used as the objective function to calculate the fitness values of the solution vector. The IAMO-MLP algorithm aims to minimize the MSE.

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{K}\left(r_j^i - p_j^i\right)^2 \tag{12}$$

where N is the number of the training samples, K is the number of neurons in the output layer, and r_j^i and p_j^i are the real output and predicted output of the neuron j for the ith instance of the training samples, respectively.

The flowchart of the IAMO-MLP algorithm is shown in Fig. 9. The IAMO-MLP algorithm works as follows: Each individual in the algorithm represents an animal and offers a solution. Each individual has a fitness value calculated by the objective function in Eq. (12). Firstly, the population size and the length of the neighborhood are initialized and the population is randomly generated using Eqs. (3) and (4). Then, the fitness value for each individual is calculated by the objective function. In this stage, each individual is assigned to MLP and evaluates MLP using MSE on the training samples. Then, the migration process with the Lévy flight is run. In the migration process with the Lévy flight, each individual seeks better solutions in the search space with the help of their neighbors and the Lévy flight strategy using Eqs. (5) and (7). Then, the population updating process is run in which animals with low fitness value are removed from the population and new individuals with high fitness value are added to the population using Eq. (6). Thus, each individual searches for better solutions in the search space. This process is continued until the termination criteria are met. At the end of the algorithm, the best position, namely the best MLP with minimum MSE, is reported as the output of the algorithm. Thus, the IAMO-MLP algorithm finds the most appropriate values of the weights and biases according to the inputs–outputs pattern. Consequently, the IAMO algorithm updates the values of the weights, and biases of the MLP to minimize the MSE until the termination criteria of the training process are met.

The efficiency of an algorithm may be demonstrated using a theoretical analysis or an empirical analysis [44]. In an empirical analysis of IAMO-MLP, Table 6 presents the average computational time in seconds. In theoretical analysis, the worst-case complexity of the algorithm is generally computed. The worst-case complexity of IAMO-MLP depends on the number of iterations, the number of animals, the structure of the MLP, the number of training instances, the number of attributes of training instances, the migration process, and the population updating process. So, the worst-case complexity of IAMO-MLP is as follows:

$$\begin{aligned} O(\text{IAMO}-\text{MLP}) = O(g(&O(\text{migration}) + O(\text{MLP}) \\ &+O(\text{population updating}) + O(\text{MLP}))) \end{aligned} \tag{13}$$

where g is the number of iterations. The computational complexity of the migration phase is $O(\text{Nd})$, where N is the number of individuals and d is the dimension of the position vectors of an individual. The computational complexity of an MLP with N animals, i input nodes, h hidden nodes, o output nodes, and t training instances is equal to $O(\text{Nt}((ih) + (ho)))$. The migration phase and the population updating phase have the same computational complexity, namely $O(\text{Nd})$. Therefore, the final computational complexity of the IAMO-MLP algorithm is as follows:

$$\begin{aligned} O(\text{IAMO}-\text{MLP}) = O(g(&O(\text{Nd}) + O(\text{Nt}((ih) + (ho)))) \\ &+O(\text{Nd}) + O(\text{Nt}((ih) + (ho)))) \end{aligned} \tag{14}$$

where g is the number of iterations, N is the number of individuals, t is the number of training instances, i is the number of input nodes, h is the number of hidden nodes, o is the number of output nodes, and d is the dimension of the position vectors of an individual.

3 Experimental Results

In this section, to verify the accuracy and robustness of the proposed IAMO and IAMO-MLP algorithms, the experimental studies were carried out on datasets with different difficulty levels and different features. These datasets are 13 benchmark functions, five classification datasets taken from UCI Machine Learning Repository, and a real-world problem taken from [45]. The specifications of the hardware and software used in the experiments are as follows: Intel(R) Core(TM) i5-3330 3.00 GHz, 4 GB memory, and Microsoft Windows 10. The algorithms were implemented in MATLAB R2015a. All statistical analyses in this study were performed with Microsoft Excel 2013 software.

Table 2 Minimization result of the benchmark functions for different algorithms

		IAMO	AMO [40]	PSO [46]	DE [47]	BBO [48]	CS [49]	FA [50]	GSA [51]	ABC [52]
f_1	Mean	**7.83E−47**	8.65E−40	3.33E−10	5.60E−14	3.74E−01	5.66E−06	1.70E−03	3.37E−18	2.99E−20
	StdDev	**6.03E−47**	1.04E−39	7.04E−10	4.41E−14	1.18E−01	2.86E−06	4.06E−04	8.09E−19	2.15E−20
f_2	Mean	**2.16E−34**	8.23E−32	6.66E−11	4.73E−10	1.66E−01	2.00E−03	4.53E−02	8.92E−09	1.42E−15
	StdDev	**1.01E−34**	3.41E−32	9.26E−11	1.78E−10	3.42E−02	8.10E−04	3.38E−02	1.33E−09	5.53E−16
f_3	Mean	4.57E−09	8.89E−04	2.98E+00	**2.80E−11**	1.60E+03	1.40E−03	1.82E−02	1.13E−01	2.40E+03
	StdDev	1.47E−08	8.73E−04	2.28E+00	**3.68E−11**	8.34E+02	6.10E−04	6.40E−03	1.27E−01	6.57E+02
f_4	Mean	**1.29E−42**	2.86E−05	8.00E+00	2.22E−01	7.97E+00	3.24E+00	5.54E−02	9.93E−10	1.85E+01
	StdDev	**4.82E−42**	2.35E−05	2.54E+00	2.43E−01	2.66E+00	6.64E−01	1.01E−02	1.19E−10	4.25E+00
f_5	Mean	3.99E−01	4.18E+00	4.69E+01	2.66E−01	6.47E+01	8.01E+00	3.81E+01	2.01E+01	**4.41E−02**
	StdDev	1.18E+00	2.16E+00	3.80E+01	1.03E+00	3.63E+01	1.92E+00	3.04E+01	1.72E−01	**7.07E−02**
f_6	Mean	**0**	**0**	3.69E−10	4.50E−14	3.70E−01	5.43E−06	1.70E−03	3.34E−18	3.09E−20
	StdDev	**0**	**0**	6.37E−10	2.33E−14	1.12E−01	2.24E−06	4.16E−04	5.68E−19	4.01E−20
f_7	Mean	1.81E−03	**1.70E−03**	1.35E−02	4.20E−03	3.00E−03	9.60E−03	8.20E−03	3.90E−03	3.24E−02
	StdDev	5.31E−04	**4.71E−04**	4.10E−03	1.40E−03	1.20E−03	2.80E−03	9.30E−03	1.30E−03	5.90E−03
f_8	Mean	**−1.26E+04**	**− 1.26E+04**	− 8.83E+03	− 1.13E+04	− 1.26E+04	− 9.15E+03	− 6.22E+03	− 3.05E+03	− 1.25E+04
	StdDev	**1.85E−12**	**1.24E−07**	6.11E+02	1.81E+03	5.76E−01	2.53E+02	7.72E+02	3.39E+02	6.11E+01
f_9	Mean	**0**	**0**	1.83E+01	1.35E+02	3.85E−02	5.12E+01	2.35E+01	7.28E+00	**0**
	StdDev	**0**	**0**	4.80E+00	2.89E+01	1.54E−02	8.11E+00	8.37E+00	1.90E+00	**0**
f_{10}	Mean	**4.44E−15**	**4.44E−15**	3.87E−06	7.47E−08	1.95E−01	2.38E+00	9.40E−03	1.47E−09	1.19E−09
	StdDev	**0**	**0**	2.86E−06	3.11E−08	4.61E−02	1.12E+00	1.40E−03	1.44E−10	5.01E−10
f_{11}	Mean	**0**	**0**	1.68E−02	**0**	2.77E−01	4.49E−05	2.50E−03	1.27E−02	**0**
	StdDev	**0**	**0**	2.05E−02	**0**	7.96E−02	8.96E−05	4.69E−04	2.16E−02	**0**
f_{12}	Mean	**1.57E−32**	**1.57E−32**	8.30E−03	4.71E−15	2.00E−03	5.07E−01	8.87E−06	2.04E−20	1.19E−21
	StdDev	**5.89E−35**	**2.81E−48**	2.87E−02	3.26E−15	2.30E−03	2.66E−01	2.80E−06	4.53E−21	1.08E−21
f_{13}	Mean	**1.35E−32**	**1.35E−32**	4.67E−07	3.16E−14	2.18E−02	4.70E−04	1.28E−04	5.70E−33	2.30E−20
	StdDev	**5.57E−48**	**2.81E−48**	1.37E−06	2.28E−14	9.60E−03	2.99E−04	4.15E−05	6.26E−33	2.29E−20

The bold text refers to the best results

Table 3 Properties of the classification datasets

Dataset	Attribute number	Class number	Training	Test	MLP architecture	Vector size
Xor	3	2	8	8	3-7-1	36
Balloon	4	2	20	20	4-9-1	55
Iris	4	3	150	150	4-9-3	75
Breast Cancer	9	2	599	100	9-19-1	210
Heart	22	2	80	187	22-45-1	1081

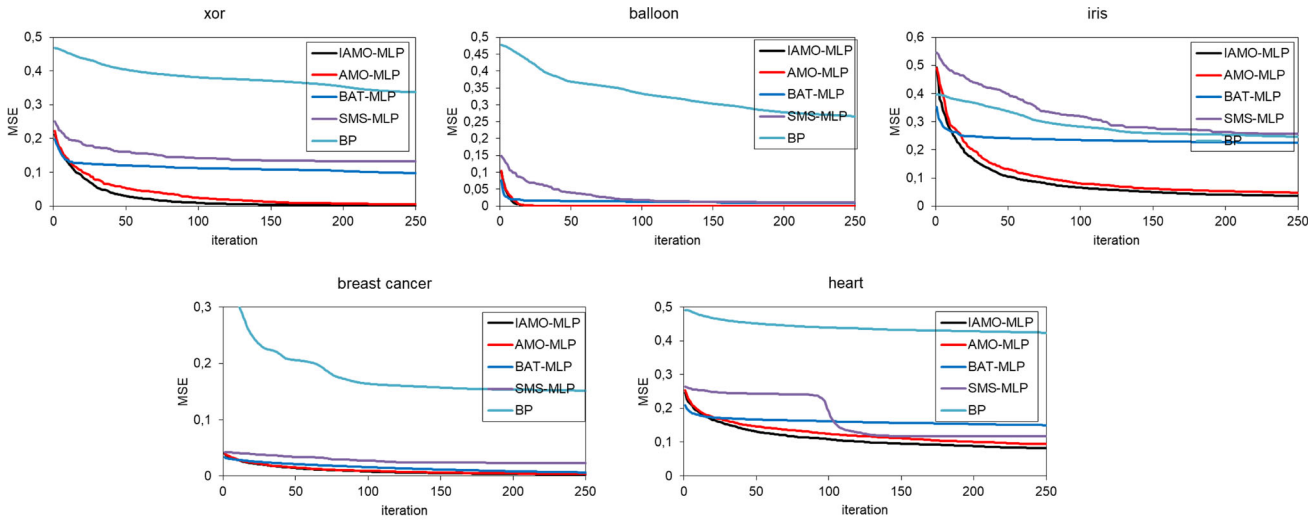

Fig. 10 Convergence graphs of the algorithms on training data

Table 4 Parameters of the algorithms

Algorithm	Parameter	Value
AMO/IAMO	Lower bound	− 10
	Upper bound	10
	Length of neighborhood	5
	Threshold in IAMO	5
	Population size	50 for xor and balloon, 200 for the others
	Maximum Fes	12,500 for xor and balloon, 50,000 for the others
BAT	Lower bound	− 10
	Upper bound	10
	Loudness	0.5
	Pulse rate	0.5
	Minimum frequency	0
	Maximum frequency	2
	Population size	50 for xor and balloon, 200 for the others
	Maximum Fes	12,500 for xor and balloon, 50,000 for the others
SMS	Lower bound	− 10
	Upper bound	10
	α, β, H, ρ (gas state)	0.3, 0.9, 0.9, 0.85
	α, β, H, ρ (liquid state)	0.05, 0.5, 0.2, 0.35
	α, β, H, ρ (solid state)	0, 0.1, 0, 0.1
	Population size	50 for xor and balloon, 200 for the others
	Maximum FEs	12,500 for xor and balloon, 50,000 for the others
BP	Lower bound	− 10
	Upper bound	10
	Learning rate	0.3
	Momentum	0.8
	Maximum epoch	12,500 for xor and balloon, 50,000 for the others

Table 5 Average and standard deviation of the MSE results on training data

Dataset	IAMO-MLP	AMO-MLP	BAT-MLP	SMS-MLP [26]	BP
Xor	**9.28E−04 ± 9.68E−04**	4.74E–03 ± 4.24E–03	9.76E–02 ± 6.74E–02	1.33E–01 ± 3.39E–02	1.38E–01 ± 1.65E–01
Balloon	**4.24E−10 ± 1.95E−09**	3.47E–09 ± 6.22E–09	8.59E–03 ± 2.19E–02	1.11E–02 ± 1.31E–02	7.68E–02 ± 1.71E–01
Iris	**3.59E−02 ± 4.82E−03**	4.75E–02 ± 4.42E—03	2.24E–01 ± 1.32E—01	2.58E–01 ± 5.12E–02	7.25E–02 ± 1.57E–01
Breast Cancer	**3.06E−03 ± 5.37E−04**	3.81E–03 ± 5.64E–04	6.57E–03 ± 6.59E–03	2.27E–02 ± 4.33E–03	4.85E–02 ± 1.67E–01
Heart	**8.18E−02 ± 8.63E−03**	9.37E–02 ± 9.27E–03	1.50E–01 ± 3.56E–02	1.18E–01 ± 3.13E–02	2.84E–01 ± 1.27E–01

The bold text refers to the best results

Table 6 Average classification accuracy on test data and the average computational time of the algorithms

Dataset	Classification Accuracy (%)					Time (s)				
	IAMO-MLP	AMO-MLP	BAT-MLP	SMS-MLP	BP	IAMO-MLP	AMO-MLP	BAT-MLP	SMS-MLP	BP
Xor	**100.00**	**100.00**	87.92	83.75	84.58	13.9	13.9	4.7	5.5	**1.6**
Balloon	**100.00**	**100.00**	99.17	99.17	92.33	46.7	43.7	19.5	20.2	**1.7**
Iris	**98.00**	97.27	83.22	86.78	86.84	1592.2	1443.0	798.9	881.6	**9.7**
Breast Cancer	**96.50**	96.37	96.00	85.67	91.83	5338.8	5172.3	2657.1	2830.3	**29.5**
Heart	70.32	70.86	**71.30**	68.57	63.40	2033.5	1868.6	972.1	892.4	**16.6**

The bold text refers to the best results

Table 7 Results of the performance metrics (Sensitivity, Specificity, Precision, F_1-Score)

Algorithm	Sensitivity (%)	Specificity (%)	Precision (%)	F_1-Score	Sensitivity (%)	Specificity (%)	Precision (%)	F_1-Score
	Xor				Balloon			
IAMO-MLP	**100.00**	**100.00**	**100.00**	**1.0000**	**100.00**	**100.00**	**100.00**	**1.0000**
AMO-MLP	**100.00**	**100.00**	**100.00**	**1.0000**	**100.00**	**100.00**	**100.00**	**1.0000**
BAT-MLP	86.67	89.17	90.89	0.8738	99.17	99.17	98.89	0.9894
SMS-MLP	81.70	86.70	88.90	0.8349	99.20	99.20	98.80	0.9894
BP	84.17	85.00	83.22	0.8222	91.67	92.78	91.63	0.9011
	Iris				Breast Cancer			
IAMO-MLP	**98.00**	**99.00**	**98.03**	**0.9800**	91.59	97.81	91.88	**0.9149**
AMO-MLP	97.49	98.74	97.54	0.9749	91.75	97.59	91.18	0.9115
BAT-MLP	83.22	91.61	77.44	0.7896	88.25	**98.06**	**92.60**	0.8944
SMS-MLP	82.10	91.10	81.00	0.8002	50.20	95.10	83.70	0.5828
BP	86.84	93.42	83.25	0.8406	**92.70**	91.60	85.36	0.8700
	Heart							
IAMO-MLP	70.12	**72.67**	**96.73**	0.8113				
AMO-MLP	70.81	71.33	96.61	0.8163				
BAT-MLP	**71.63**	67.56	96.23	**0.8192**				
SMS-MLP	68.39	70.67	96.41	0.7980				
BP	64.22	54.00	91.52	0.7311				

The bold text refers to the best results

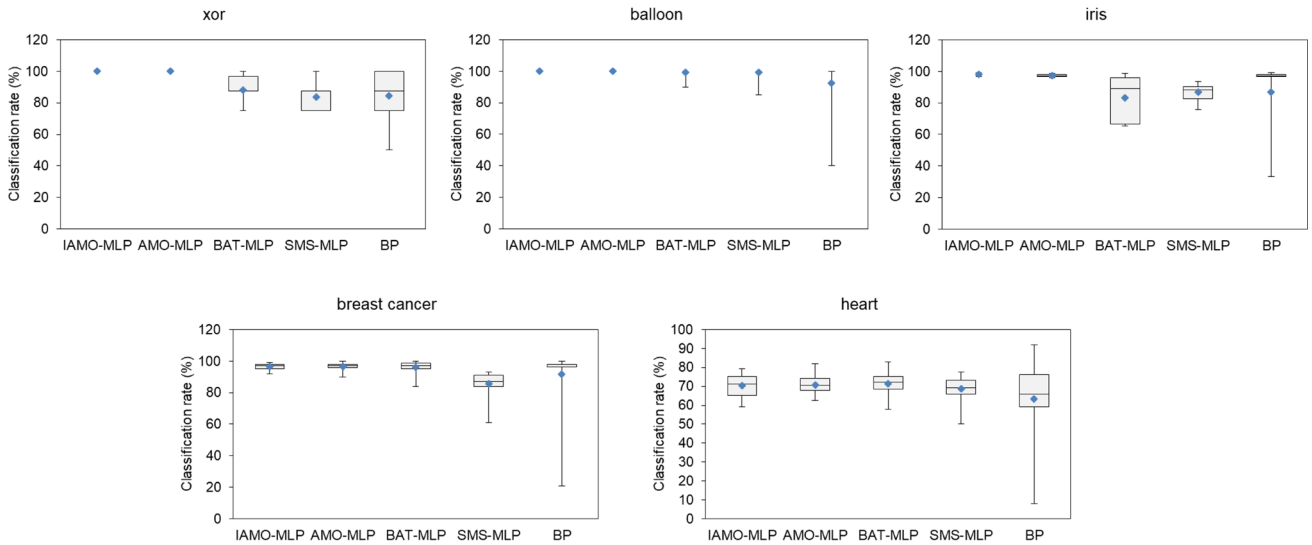

Fig. 11 Boxplot charts of the classification rate results on test data

3.1 Benchmark Functions

To determine the success of the proposed IAMO algorithm, 13 benchmark functions shown in Table 1 were used in the experiments. These functions have been widely used in the literature. Functions $f_1 - f_5$ are unimodal functions, and the rest of the functions are multimodal. Moreover, Table 1 also shows the dimension, the global minimum values, and the search ranges.

The performance of the proposed IAMO algorithm was compared with the performance of the following algorithms: AMO [40], particle swarm optimization (PSO) [46], differential evolution (DE) [47], biogeography-based optimization (BBO) [48], cuckoo search (CS) [49], the firefly algorithm (FA) [50], the gravitational search algorithm (GSA) [51], and artificial bee colony (ABC) [52]. The results of these algorithms were taken from [40] and presented in Table 2. To fairly compare the algorithms, each algorithm runs the same function evaluations (FEs) at each run: 150,000 FEs for f_1,

f_6, f_{10}, f_{12} and f_{13}; 200,000 FEs for f_2 and f_{11}; 300,000 FEs for f_7, f_8 and f_9; and 500,000 FEs for f_3, f_4 and f_5. The *threshold* and population size parameters of the IAMO algorithm are set to 5 and 50, respectively. The IAMO algorithm was independently run 30 times on each function. The average values (mean) of the results and their standard deviations (stdDev) in 30 runs are provided in Table 2. According to the results of the algorithms on the benchmark functions in Table 2, it is evident that the IAMO algorithm outperforms the other algorithms in the majority of the test cases. The IAMO algorithm provides the best results on ten of the benchmark functions. It has also the second-best results on two of the benchmark functions. According to Table 2, it is obvious that the proposed IAMO algorithm has better performance than the canonical AMO algorithm.

3.2 Classification Problems

To determine the success of the proposed IAMO-MLP algorithm, five classification datasets were used in the experiments: xor, balloon, iris, breast cancer, and heart. These classification datasets are selected according to the different levels of difficulty to effectively test the performance of the IAMO-MLP algorithm. The properties (number of attributes and classes, the number of training and test samples) of these five datasets are shown in Table 3. The training and test subsets of the datasets are taken from the website www.seyedalimirjalili.com and also used in [25]. Additionally, Table 3 shows the architecture of the MLP and the vector size of the candidate solutions. The number of neurons in the hidden layer was determined by $2 \times n + 1$ where n is the number of the neurons in the input layer. The length of the vector is calculated using Eq. (11). The IAMO-MLP algorithm was independently run 30 times on each dataset. 12,500 function evaluations (FEs) for xor and balloon, and 50,000 FEs for the remaining datasets were carried out at each run. The initial values of the weights and biases of MLP were randomly determined in the range of $[-10, 10]$.

The proposed IAMO-MLP algorithm is also compared with the algorithms AMO-MLP, BAT-MLP, SMS-MLP, and BP. Table 4 shows the parameters of these algorithms. To fairly compare all the algorithms, they have the same number of FEs.

Table 5 shows the average and standard deviation of the MSE results of the algorithms on training data. The lower the MSE is, the better the performance of the algorithm is. According to the results in Table 5, the IAMO-MLP algorithm has the lowest MSE value for each dataset. Figure 10 shows the convergence graphs of the algorithms in terms of the MSE value at each iteration. When the convergence graphs are analyzed, it is seen that the IAMO-MLP algorithm exhibits very good performance.

Table 6 shows the average classification accuracy on test data and the average computational time of the algorithms. The IAMO-MLP algorithm has a better classification accuracy than the other algorithms on the datasets xor, balloon, iris, and breast cancer. On the dataset heart, the BAT-MLP algorithm has a better classification accuracy than the other algorithms. According to the average computational time, the fastest algorithm is the BP algorithm, and the slowest algorithm is the IAMO-MLP algorithm. Table 7 shows the results of the performance metrics sensitivity, specificity, precision, and F_1-Score. According to the results, the IAMO-MLP algorithm has the highest percentage of sensitivity, specificity, and precision on xor, balloon, and iris datasets among all the algorithms. Additionally, the IAMO-MLP algorithm has the highest percentage of specificity and precision on the heart dataset among all the algorithms. In terms of the F_1-Score, the IAMO-MLP algorithm has better results than the other algorithms on xor, balloon, iris and breast cancer datasets. Overall, the IAMO-MLP algorithm is successful according to the performance metrics. Figure 11 shows the boxplot charts of the classification rate results of the algorithms on test data. Boxplot charts are an easy way to visually show the distribution of data, particularly used to summarize data in terms of the central location, spread, skewness, and kurtosis and to identify outliers. Boxplot charts present the minimum value, first quartile, median value, mean value, third quartile, and maximum value of data. According to the boxplot charts in Fig. 11, the IAMO-MLP algorithm generates more robust results than the AMO-MLP, BAT-MLP, SMS-MLP, and BP algorithms on each dataset, although the BAT-MLP algorithm has a better average classification accuracy than the IAMO-MLP and AMO-MLP algorithms on the heart dataset.

The IAMO-MLP algorithm is also compared with the GWO-MLP, PSO-MLP, GA-MLP, ACO-MLP, ES-MLP, and PBIL-MLP algorithms whose results are taken from [25]. Table 8 shows this comparison according to the best classification accuracy (%). The results show that the IAMO-MLP algorithm outperforms the other six algorithms on all the datasets. On the xor dataset, the IAMO-MLP, GWO-MLP, and GA-MLP algorithms have 100% classification accuracy. On the balloon dataset, all the algorithms have 100% classification accuracy. On the iris dataset, the IAMO-MLP algorithm has 99.33% classification accuracy, and the GWO-MLP algorithm has the second-best result with 91.33% classification accuracy. On the breast cancer dataset, the IAMO-MLP algorithm and the GWO-MLP algorithm have the same classification accuracy, namely 99%. On the heart dataset, the IAMO-MLP algorithm has 79.14% classification accuracy, and the GWO-MLP algorithm has the second-best result with 75% classification accuracy.

The Friedman test is a nonparametric statistical test. In the Friedman test, two or more samples are used to compare the populations. This test also gives the ranking results of

Table 8 Comparison of the algorithms according to the best classification accuracy (%)

Dataset	IAMO-MLP	GWO-MLP	PSO-MLP	GA-MLP	ACO-MLP	ES-MLP	PBIL-MLP
Xor	**100.00**	**100.00**	37.50	**100.00**	62.50	62.50	62.50
Balloon	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**
Iris	**99.33**	91.33	37.33	89.33	32.66	46.66	86.66
Breast Cancer	**99.00**	**99.00**	11.00	98.00	40.00	06.00	07.00
Heart	**79.14**	75.00	68.75	58.75	00.00	71.25	45.00

The bold text refers to the best results

Table 9 Average ranking of the algorithms according to the classification rate on the classification dataset (Friedman test)

Algorithm	Ranking
IAMO-MLP	**4.4**
AMO-MLP	4.2
BAT-MLP	2.9
SMS-MLP	1.7
BP	1.8

The bold text refers to the best results

Table 10 Civil engineering dataset [45]

No	W/C	SP	GS	Fc
1	0.42	13.5	1743.02	81
2	0.44	13.5	1735.97	82.7
…	…	…	…	…
…	…	…	…	…
…	…	…	…	…
111	0.4	4.95	1410.9	19.4
112	0.47	3.25	1335	20.2

the population. Table 9 shows the average ranking of the algorithms according to the classification rate. Because the aim is to maximize the classification rate, the higher values in Table 9 are better. According to the Friedman test result in Table 9, the IAMO-MLP algorithm ranks higher than the other algorithms with a ranking score of 4.4. The AMO-MLP algorithm has the second ranking with a ranking score of 4.2. The BAT-MLP algorithm has the third ranking with a ranking score of 2.9. The BP algorithm has the fourth ranking with a ranking score of 1.8. The SMS-MLP algorithm has the last ranking with a ranking score of 1.7.

3.3 A Real-World Engineering Problem

In this section, the IAMO-MLP algorithm is applied to solve a real-world problem in the civil engineering area in which waste tires are added into cement with the result that both waste tires are recycled, and the strength of the concrete is increased [53–55]. However, the compressive strength of rubberized concrete varies according to the amount of substances added into it. To find the optimum value of the compressive strength, the amount of substances added is empirically determined. Recently, some models based on soft computing techniques have been created such as ANN [45, 56] to estimate the compressive strength. In this study, the IAMO-MLP algorithm was used to estimate the compressive strength. The dataset of the real-world problem was taken from [45]. It has three attributes (water–cement ratio *w/c*, superplasticizer *sp,* and granular squeleton *gs*), one output (comprehensive strength *fc*), and 112 instances. Some of the data are shown in Table 10. The 95 instances of them are used for training, and the remaining 17 instances are used for the test.

The IAMO-MLP algorithm was compared to the AMO-MLP, BAT-MLP, SMS-MLP [26], and BP algorithms. In the previous section, the number of neurons in the hidden layer was calculated by $2 \times n + 1$, where n is the number of neurons in the input layer. In this section, we compared the algorithms using different numbers of neurons in the hidden layer. Therefore, the numbers (H) of neurons in the hidden layer were set from 7 up to 20. We also compared the algorithms using different numbers of population size (P) and maximum iteration number (I): P 50 and I 250, P 50 and I 500, P 100, and I 250. To fairly compare the algorithms, the number of function evaluations of each algorithm was kept equal. Therefore, the maximum epoch number of the BP algorithm was 12,500 for 50 population–250 iterations, and 25,000 for 50 population–500 iterations and 100 population–250 iterations. The other parameters are shown in Table 4. The algorithms were independently run 30 times.

Table 11 shows the results of the algorithms on the training data according to the MSE for P 50 and I 250. It is seen that the IAMO-MLP algorithm outperforms the other algorithms for all H values. The IAMO-MLP algorithm has both the best average values and the best standard deviations. On the other hand, the BP algorithm has both the worst average MSE results and the worst standard deviations. Table 12 shows the results of the algorithms on the test data according to the MSE for P 50 and I 250. Figure 12 shows the boxplot charts of the MSE results on test data for P 50 and I 250. As seen from the table and boxplot charts, the IAMO-MLP algorithm outperforms the other algorithms for all H values except H 8, 11, and 15. Besides, the results of the AMO-MLP

Table 11 Comparison of the algorithms on training data according to the MSE results for *P* 50 and *I* 250

Neurons (H)	IAMO-MLP			AMO-MLP			BAT-MLP			SMS-MLP			BP		
	Avg	Std	Min	Avg	Std	Min	Avg	Std	Min	Avg	Std	Min	Avg	Std	Min
7	**50.0**	8.5	35.6	64.0	8.1	51.8	172.3	95.1	71.4	157.6	32.2	93.4	495.5	682.4	16.4
8	**52.2**	8.6	34.7	61.1	7.6	43.7	169.8	100.7	55.0	155.6	33.9	92.7	636.4	895.8	16.7
9	**51.4**	8.5	29.7	60.5	8.9	40.5	139.0	85.2	47.2	170.7	40.7	94.2	712.3	970.8	15.3
10	**48.7**	8.7	33.7	61.5	8.9	46.2	238.3	184.2	43.0	160.9	28.8	107.8	323.8	631.8	6.7
11	**49.3**	9.2	34.2	62.5	8.4	43.5	211.9	153.5	46.7	176.3	45.9	103.3	809.3	929.1	12.4
12	**50.1**	7.1	32.3	65.0	7.1	53.2	200.8	141.0	58.3	173.5	41.5	80.3	859.2	1000.5	10.9
13	**49.1**	9.5	28.6	64.1	9.2	42.0	199.2	162.6	34.1	188.2	44.1	117.3	721.9	1021.3	7.7
14	**49.4**	9.7	29.3	67.3	7.1	53.5	225.8	170.2	38.7	175.3	56.4	48.8	474.4	753.9	7.7
15	**49.7**	10.3	29.4	67.8	10.7	48.4	243.1	147.8	59.4	179.5	44.0	102.8	579.2	895.8	4.9
16	**49.8**	9.7	28.0	70.6	10.7	45.6	237.7	200.0	53.4	216.5	58.0	116.6	714.6	926.2	7.7
17	**46.6**	8.3	22.4	66.5	13.3	31.5	193.8	150.8	40.2	197.5	58.9	97.0	472.8	814.5	7.3
18	**54.4**	10.4	29.4	71.2	10.3	56.0	206.7	137.6	47.2	204.9	55.6	61.7	697.8	869.9	6.6
19	**51.6**	9.4	29.4	70.1	15.1	30.6	237.9	175.0	25.3	200.9	54.4	101.7	921.1	962.0	6.9
20	**53.3**	9.9	31.3	71.3	11.5	51.4	283.5	227.2	39.5	214.1	54.1	127.2	634.8	780.6	7.0

The bold text refers to the best results
Avg average, *Std* standard deviation, *Min* minimum

Table 12 Comparison of the algorithms on test data according to the MSE results for *P* 50 and *I* 250

Neurons (H)	IAMO-MLP			AMO-MLP			BAT-MLP			SMS-MLP			BP		
	Avg	Std	Min	Avg	Std	Min	Avg	Std	Min	Avg	Std	Min	Avg	Std	Min
7	**112.4**	36.5	64.2	121.8	42.7	51.3	217.8	99.3	96.5	196.0	71.6	72.6	547.9	665.3	55.6
8	105.8	37.7	48.9	**104.8**	27.4	50.3	243.2	122.5	63.0	198.0	86.9	77.1	751.7	969.6	57.0
9	**116.8**	38.3	67.1	118.7	45.5	42.7	193.7	104.4	87.2	216.4	100.8	70.2	826.8	1051.4	65.8
10	**102.2**	28.6	45.4	114.5	35.9	69.2	271.9	182.6	84.9	180.5	70.6	28.5	408.8	618.0	68.0
11	126.7	35.9	73.8	**115.7**	39.2	41.4	259.2	165.2	40.4	211.9	86.0	96.7	924.5	1002.5	65.9
12	**117.2**	34.9	40.9	120.2	48.8	66.0	268.4	170.5	80.8	237.4	106.0	103.7	993.1	1073.3	74.4
13	121.9	43.8	44.3	**116.2**	45.9	46.6	269.7	207.7	45.2	220.1	79.0	76.0	876.7	1098.1	79.9
14	**118.8**	43.0	45.1	122.8	49.1	44.1	269.0	164.3	56.8	221.0	97.3	79.9	574.1	750.9	75.7
15	135.5	55.9	68.9	**131.3**	37.9	67.6	312.8	167.5	70.3	219.3	63.6	104.3	711.7	922.1	59.7
16	**133.5**	44.8	59.4	134.0	57.7	78.1	314.4	204.5	67.8	233.9	137.7	31.9	850.3	1004.4	54.6
17	**124.8**	50.6	62.6	130.9	41.2	69.4	267.5	163.8	79.9	200.8	76.9	50.5	601.6	833.1	82.7
18	131.2	42.1	66.0	**122.5**	50.7	44.4	250.4	165.5	39.4	236.5	107.6	83.9	788.4	863.1	80.2
19	**124.5**	42.3	48.1	145.0	72.8	61.9	318.4	201.1	76.7	252.1	91.8	101.2	1073.9	1015.7	99.2
20	**140.0**	54.1	52.4	143.5	64.7	46.8	328.1	206.6	77.0	252.9	115.5	102.0	733.0	754.4	87.1

The bold text refers to the best results

algorithm are better than the SMS-MLP algorithm, the BAT-MLP algorithm, and the BP algorithm. The BP algorithm has the worst results.

Table 13 shows the results of the algorithms on the training data according to the MSE for *P* 100 and *I* 250. It is seen that the IAMO-MLP algorithm outperforms the other algorithms for all *H* values. The IAMO-MLP algorithm has both the best average values and low standard deviations. On the other hand, the BP algorithm has both the worst average MSE results and the worst standard deviations. Table 14 shows the results of the algorithms on the test data according to the MSE for *P* 100 and *I* 250.

Figure 13 shows the boxplot charts of the MSE results on test data for *P* 100 and *I* 250. As seen from the table and boxplot charts, the IAMO-MLP algorithm outperforms the other algorithms for all *H* values except *H* 8, 10, 12, and 13. The BP algorithm has the worst results.

Table 15 shows the results of the algorithms on the training data according to the MSE for *P* 50 and *I* 500. It is seen that the IAMO-MLP algorithm outperforms the other algo-

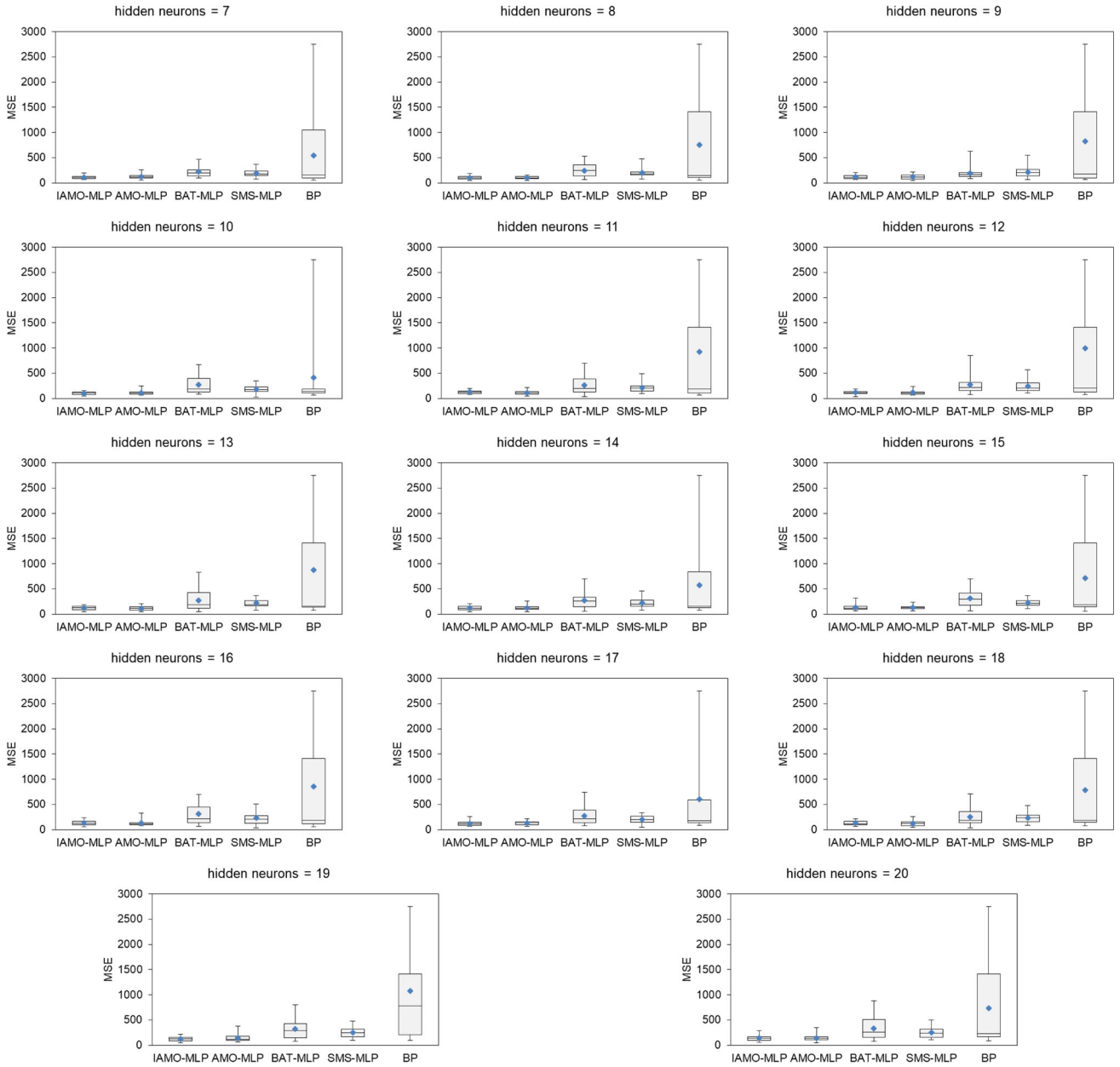

Fig. 12 Boxplot charts of the MSE results on test data for P 50 and I 250

rithms for all H values. The IAMO-MLP algorithm has both the best average values and low standard deviations. On the other hand, the BP algorithm has both the worst average MSE results and the worst standard deviations. Table 16 shows the results of the algorithms on the test data according to the MSE for P 50 and I 500. Figure 14 shows the boxplot charts of the MSE results on test data for P 50 and I 500. As seen from the table and boxplot charts, the IAMO-MLP algorithm outperforms the other algorithms for all H values except H 8, 9, and 11. The AMO-MLP algorithm also has the second-best results. Besides, the BP algorithm has the worst results.

Table 17 shows the Friedman test results. Because the goal was to minimize the MSE, the lower values in the result of the Friedman test are better. For P 50 and I 250, the IAMO-MLP algorithm ranked higher than the other algorithms according to the Friedman test results with a ranking score of 1.36. The AMO-MLP algorithm has the second ranking with a ranking score of 1.64. The SMS-MLP algorithm has the third ranking with a ranking score of 3.07. The BAT-MLP algorithm has the fourth ranking with a ranking score of 3.93. The BP algorithm has the last ranking with a ranking score of 5.0.

For P 100 and I 250, the IAMO-MLP algorithm ranked higher than the other algorithms with a ranking score of 1.36.

Table 13 Comparison of the algorithms on training data according to the MSE results for P 100 and I 250

Neurons (H)	IAMO-MLP			AMO-MLP			BAT-MLP			SMS-MLP			BP		
	Avg	Std	Min	Avg	Std	Min	Avg	Std	Min	Avg	Std	Min	Avg	Std	Min
7	**44.8**	9.5	25.4	57.7	5.5	48.7	87.8	52.7	19.3	140.1	31.4	71.5	210.8	545.0	11.8
8	**48.7**	5.9	39.8	56.8	6.8	40.9	74.4	52.6	17.4	140.8	27.0	104.4	569.8	925.6	8.7
9	**45.6**	9.3	23.8	57.1	7.8	39.7	78.3	60.8	20.1	140.4	32.9	77.0	746.2	982.3	10.4
10	**43.3**	9.6	23.3	56.0	6.4	43.6	84.8	67.5	18.0	147.8	31.2	67.8	305.3	680.1	7.5
11	**43.6**	9.0	24.8	57.4	6.2	41.2	87.4	108.0	14.2	152.8	28.5	93.2	600.5	857.0	6.8
12	**45.7**	7.7	33.9	58.3	6.0	46.7	98.1	98.8	16.4	159.8	38.2	66.5	561.8	930.3	7.4
13	**45.3**	9.4	28.2	58.8	6.2	43.8	90.7	83.1	11.0	150.7	33.2	81.3	533.8	889.5	6.0
14	**44.4**	8.9	30.6	58.9	7.8	44.7	112.0	106.7	17.6	157.2	37.3	92.6	509.7	785.4	5.6
15	**46.1**	7.9	28.6	59.3	8.4	44.6	84.0	66.2	22.4	164.2	39.2	100.2	686.3	921.2	5.2
16	**49.3**	9.4	32.1	61.7	8.5	39.0	129.8	123.1	11.4	160.5	38.6	80.7	394.0	773.1	6.4
17	**44.4**	10.3	24.4	61.0	7.3	47.9	78.1	89.0	15.5	171.6	45.1	117.2	274.7	654.6	4.4
18	**43.2**	7.6	19.1	60.9	8.3	48.1	80.0	77.6	10.5	163.8	35.0	114.8	823.3	971.3	5.3
19	**48.1**	9.4	30.5	61.9	5.2	47.9	113.2	104.6	21.5	180.2	39.3	79.9	602.3	893.4	4.9
20	**46.3**	9.6	21.1	59.7	8.3	38.6	134.5	143.3	16.3	168.0	39.1	93.6	445.5	820.7	4.7

The bold text refers to the best results

Table 14 Comparison of the algorithms on test data according to the MSE results for P 100 and I 250

Neurons (H)	IAMO-MLP			AMO-MLP			BAT-MLP			SMS-MLP			BP		
	Avg	Std	Min	Avg	Std	Min	Avg	Std	Min	Avg	Std	Min	Avg	Std	Min
7	**105.3**	23.9	65.3	106.8	45.9	38.2	155.2	74.3	62.0	167.0	70.7	40.1	289.4	560.0	58.5
8	122.5	35.3	50.0	**106.3**	31.2	57.0	140.0	67.4	68.8	173.4	62.4	62.8	700.8	991.1	51.9
9	**105.2**	31.6	40.0	122.7	45.3	36.0	146.5	81.2	31.0	169.4	62.8	68.6	869.4	1047.6	72.6
10	112.5	31.9	56.3	**104.1**	33.1	33.6	143.6	87.4	50.5	157.5	70.2	35.0	431.5	704.5	65.5
11	**110.1**	29.2	51.6	116.0	37.1	63.1	166.1	107.3	39.4	180.5	65.9	65.7	703.0	861.1	70.0
12	124.5	50.9	56.2	**114.7**	40.0	52.0	174.1	98.7	49.4	200.5	79.2	90.8	703.4	989.2	70.5
13	140.6	54.4	48.9	**125.2**	42.8	58.5	183.9	121.4	68.9	201.0	86.2	78.8	679.3	962.1	55.1
14	**119.4**	36.7	48.0	147.9	64.5	59.1	193.7	106.9	65.7	179.8	84.6	51.1	602.9	772.6	59.9
15	**115.5**	36.6	36.8	132.1	54.9	47.4	185.5	96.7	45.7	189.1	90.6	62.4	814.1	955.0	70.1
16	**114.6**	43.3	37.4	129.3	36.1	56.8	229.9	175.9	29.9	212.6	92.6	70.6	527.0	812.4	72.6
17	**124.1**	43.1	37.4	139.2	46.4	61.6	188.3	99.5	69.6	208.5	102.4	75.8	418.0	684.8	84.2
18	**119.3**	41.0	62.7	144.0	65.8	38.2	166.3	85.8	57.7	201.4	72.7	70.9	976.6	1045.0	60.1
19	**117.8**	33.0	53.5	142.0	57.7	61.5	199.3	127.4	47.0	231.2	104.1	83.8	733.1	917.0	111.8
20	137.4	43.3	45.0	**131.3**	64.9	34.4	204.1	165.9	55.2	201.4	99.6	77.1	590.0	833.6	72.5

The bold text refers to the best results

The AMO-MLP algorithm has the second ranking with a ranking score of 1.64. The BAT-MLP algorithm has the third ranking with a ranking score of 3.21. The SMS-MLP algorithm has the fourth ranking with a ranking score of 3.79. The BP algorithm has the last ranking with a ranking score of 5.0.

For P 50 and I 500, the IAMO-MLP algorithm ranked higher than the other algorithms with a ranking score of 1.21. The AMO-MLP algorithm has the second ranking with a ranking score of 1.79. The SMS-MLP algorithm has the third ranking with a ranking score of 3.14. The BAT-MLP algorithm has the fourth ranking with a ranking score of 3.86. The BP algorithm has the last ranking with a ranking score of 5.0.

Finally, the IAMO-MLP algorithm achieves good performance on the real-world problem according to the MSE, the boxplot charts, and Friedman test results. From the experimental results, it is demonstrated that the IAMO algorithm is successful in training the MLP, and the IAMO-MLP algorithm has the ability to escape successfully from local optima. Moreover, the independent 30 runs prove that the randomly generated initial positions do not affect the performance of

Table 15 Comparison of the algorithms on training data according to the MSE results for *P* 50 and *I* 500

Neurons (H)	IAMO-MLP			AMO-MLP			BAT-MLP			SMS-MLP			BP		
	Avg	Std	Min	Avg	Std	Min	Avg	Std	Min	Avg	Std	Min	Avg	Std	Min
7	**41.5**	9.0	24.6	49.4	7.7	26.1	120.7	81.0	34.2	145.6	26.0	96.1	312.9	669.5	10.3
8	**39.8**	6.3	22.0	50.6	6.2	36.9	126.4	125.0	25.4	143.5	28.8	78.9	359.3	771.2	7.3
9	**41.6**	7.3	22.8	48.5	7.7	33.6	126.7	166.2	16.0	141.1	32.6	82.7	742.3	931.7	6.4
10	**36.7**	7.3	19.4	47.8	6.6	34.0	153.0	146.4	23.4	134.3	25.7	83.9	457.0	814.7	5.4
11	**37.7**	7.8	22.1	51.7	6.6	35.3	134.4	93.5	18.7	137.3	28.2	89.3	592.3	889.1	7.5
12	**39.3**	7.9	25.5	50.0	7.5	31.6	119.1	87.3	20.0	144.1	29.3	91.6	607.3	941.4	5.5
13	**39.1**	7.2	21.4	51.3	5.3	35.8	86.8	70.7	25.6	133.2	26.6	79.2	506.1	869.8	5.2
14	**38.8**	6.6	22.3	51.7	7.8	33.6	133.2	140.0	20.8	148.0	35.0	90.7	596.7	780.3	5.4
15	**38.8**	7.8	21.6	51.7	8.4	40.3	151.9	154.1	16.0	145.1	37.6	78.8	576.2	897.7	6.1
16	**37.4**	6.9	24.1	52.6	6.6	40.3	164.2	148.4	14.5	140.5	32.0	79.9	504.4	870.7	5.6
17	**38.0**	7.5	16.2	52.2	6.8	41.0	196.7	203.4	19.6	154.1	30.3	106.8	667.5	838.2	7.1
18	**38.3**	7.2	23.5	53.8	6.9	36.6	173.9	179.6	15.4	151.7	27.1	95.1	446.8	820.0	4.7
19	**38.7**	8.2	23.4	54.7	8.2	37.3	150.7	145.8	19.2	149.1	39.3	78.5	1113.7	1077.3	4.5
20	**40.0**	7.6	25.0	54.6	8.0	40.8	181.0	190.6	25.9	161.7	41.0	81.3	755.2	982.5	4.3

The bold text refers to the best results

Table 16 Comparison of the algorithms on test data according to the MSE results for *P* 50 and *I* 500

Neurons (H)	IAMO-MLP			AMO-MLP			BAT-MLP			SMS-MLP			BP		
	Avg	Std	Min	Avg	Std	Min	Avg	Std	Min	Avg	Std	Min	Avg	Std	Min
7	**99.0**	32.7	43.0	106.4	33.6	39.3	180.6	103.2	69.9	175.0	60.9	40.9	406.9	706.7	60.4
8	111.4	31.6	65.5	**103.1**	26.5	60.5	187.6	130.3	53.9	175.1	74.2	73.7	471.1	824.3	61.8
9	118.6	51.0	64.1	**107.8**	35.3	59.4	209.1	162.4	57.5	188.6	77.6	57.6	839.4	986.6	65.5
10	**111.3**	34.4	68.1	112.6	39.0	55.9	206.9	134.1	58.3	164.9	58.9	64.8	569.8	843.7	74.5
11	123.6	45.0	38.0	**110.5**	37.2	44.7	184.2	103.3	36.9	187.9	63.0	99.1	701.7	929.4	53.7
12	**119.0**	41.0	62.1	119.1	38.9	62.6	192.5	120.4	46.8	183.5	66.5	57.4	746.6	990.9	88.7
13	**117.8**	25.7	73.3	130.1	42.6	63.0	170.9	82.4	49.2	185.5	76.3	69.4	643.5	912.4	73.0
14	**122.5**	41.2	58.4	124.2	49.2	46.5	195.9	132.0	64.9	179.3	85.4	57.1	673.1	769.6	58.4
15	**117.5**	47.3	44.2	120.6	38.6	62.5	230.0	130.0	50.2	169.7	81.4	49.5	698.0	930.0	98.2
16	**109.4**	32.1	58.8	143.1	57.4	66.3	226.7	128.8	76.9	190.8	89.0	66.3	645.5	910.8	82.9
17	**114.1**	44.2	33.2	145.8	65.8	55.7	272.7	205.0	46.0	188.9	102.5	60.5	768.1	837.1	78.8
18	**121.0**	38.7	74.7	124.4	40.3	42.9	244.9	170.8	59.8	197.5	98.1	53.6	578.9	839.3	79.1
19	**114.6**	39.7	60.3	135.3	59.3	62.1	251.9	146.0	79.4	182.1	80.4	58.2	1307.9	1160.3	85.4
20	**124.8**	35.4	64.7	130.6	48.1	54.6	294.6	266.4	63.9	182.5	86.7	56.2	903.7	1028.0	115.3

The bold text refers to the best results

Table 17 Average ranking of the algorithms according to the MSE on the civil engineering dataset (Friedman test)

Algorithm	*P* 50—*I* 250	*P* 100—*I* 250	*P* 50—*I* 500
IAMO-MLP	**1.36**	**1.36**	**1.21**
AMO-MLP	1.64	1.64	1.79
BAT-MLP	3.93	3.21	3.86
SMS-MLP	3.07	3.79	3.14
BP	5.00	5.00	5.00

The bold text refers to the best results

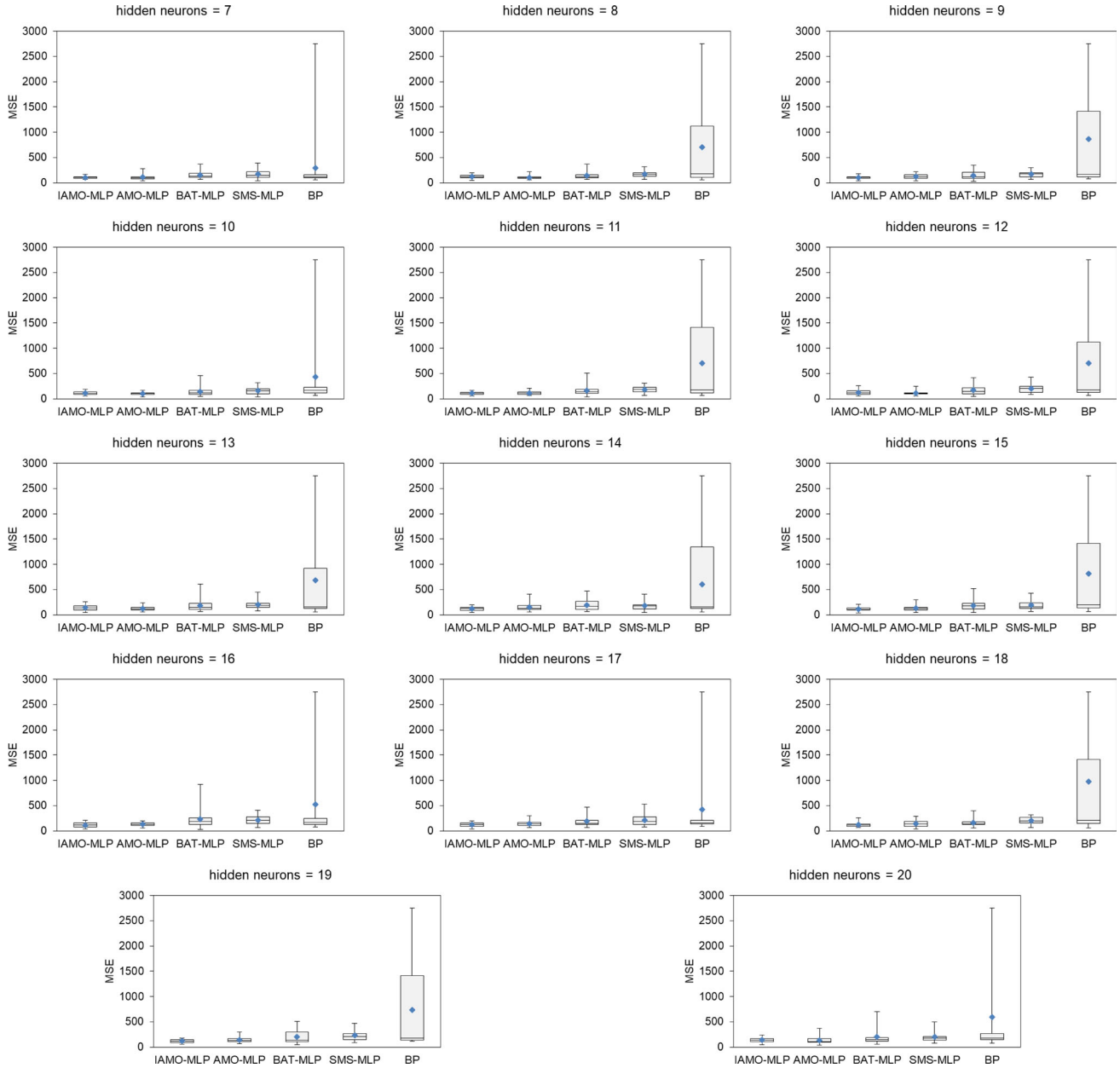

Fig. 13 Boxplot charts of the MSE results on test data for P 100 and I 250

the IAMO-MLP algorithm. Different numbers of neurons in the hidden layer were also investigated, and the performance of the IAMO-MLP algorithm is also satisfactory for different structures of the MLP. The IAMO-MLP algorithm was also used in solving a real-world problem in the civil engineering area. The results indicate that the proposed IAMO-MLP algorithm is successful in predicting the compressive strength of the rubberized concrete with an acceptable degree of accuracy.

3.4 Discussion

In order to achieve an efficient method for training neural networks, the improved animal migration optimization algorithm with the Lévy flight feature was developed. The proposed approach was tested on several datasets. The results showed that this method was able to achieve the best training performance for most of the datasets. The proposed IAMO algorithm was compared with the original AMO algorithm and seven algorithms in the literature. The experimental results proved that the IAMO outperforms other algorithms in terms of both local optima avoidance and convergence speed.

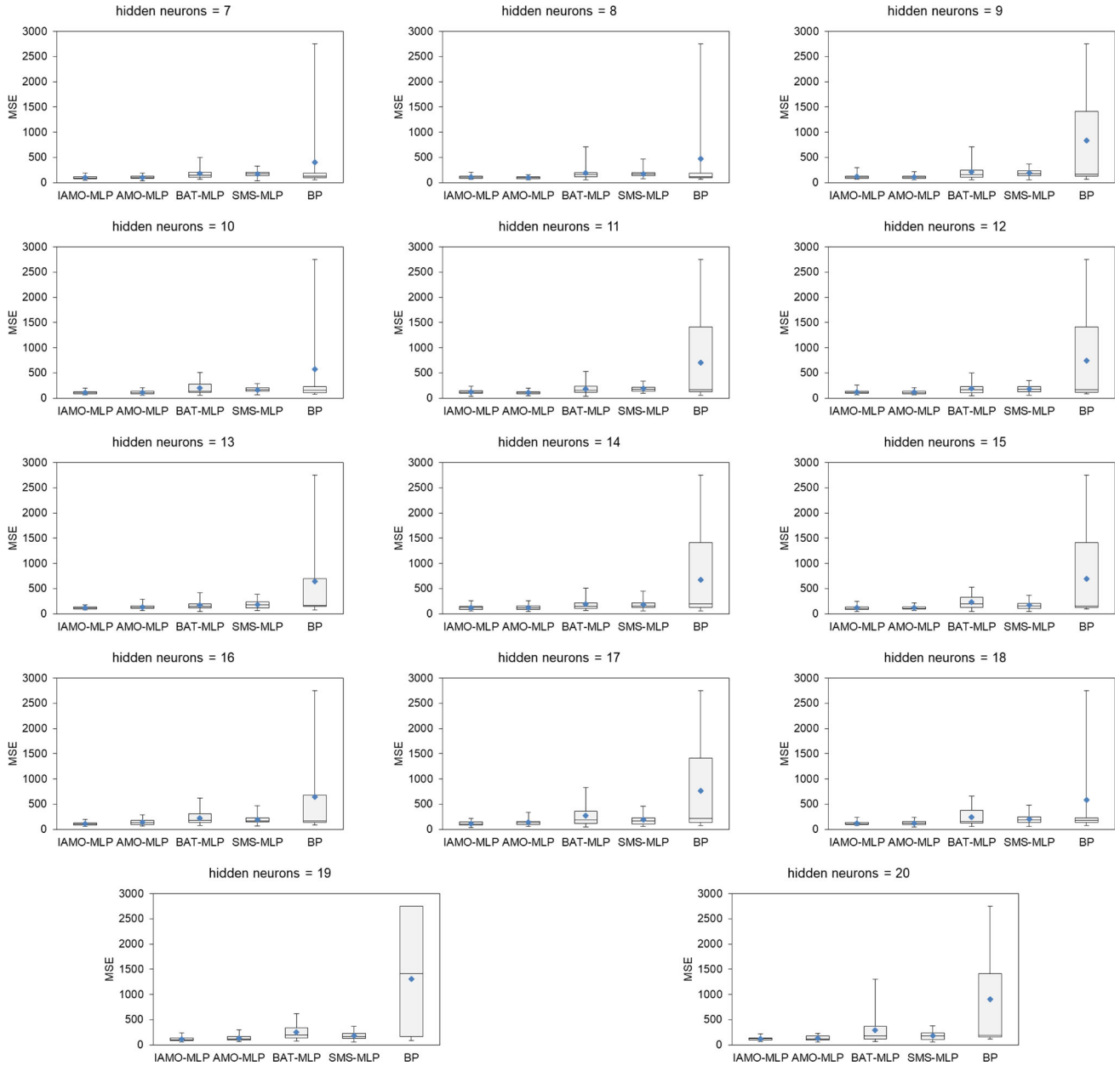

Fig. 14 Boxplot charts of the MSE results on test data for P 50 and I 500

The high local optima avoidance is due to the intense exploration of this algorithm. The Lévy flight strategy assists this algorithm in avoiding the many local solutions in the problem of training MLPs. Also, the AMO and IAMO have the neighbor cluster feature in which an individual searches the space around its neighbors. Moreover, AMO and IAMO have the population updating process which covers how some animals leave the herd and how new animals are added to the herd. Thanks to the population updating process, the AMO and IAMO have a good ability on global exploration. The superior convergence speed of the IAMO-MLP algorithm originates from the Lévy flight strategy, migration process, and population updating process of the IAMO algorithm. Therefore, the IAMO and IAMO-MLP algorithms manage to outperform other algorithms on most of the datasets. Moreover, the convergence graphs, boxplots charts and Friedman tests show that the initial positions do not affect the performance of the IAMO-MLP algorithm.

According to this comprehensive study, the IAMO algorithm is highly recommended to be used in hybrid intelligent optimization schemes such as training MLP. This recommendation is made because of its exploration behavior which results in high local optima avoidance when training MLP. The exploitation behavior is another reason why the IAMO-

MLP can converge rapidly toward the global optimum for different datasets.

The IAMO-MLP algorithm was compared with the BP algorithm in the experiments, and the results of the IAMO-MLP algorithm showed that it was very promising since it was able to obtain better results in almost all datasets. On the other hand, IAMO-MLP requires a higher computational load than BP because the fitness update of each individual in IAMO-MLP needs to be evaluated under an MLP architecture. Thus, this process during iterations takes a longer time to find the most suitable values for the weights in the network architecture. It is usually expected that more iterations and individuals would provide better results, but such a process comes at the price of a higher computational burden. Therefore, it should be noted here that IAMO is highly recommended only when the dataset and the number of attributes are very large. Small datasets with very few features can be solved much faster by gradient-based training algorithms and without extra computational cost. In contrast, the IAMO algorithm is useful for large datasets due to the extreme number of local optima that makes the conventional training algorithm almost ineffective.

4 Conclusions

The most important and demanding part of the artificial neural network (ANN) is the training process of the network. The training of the ANN is the process of finding the most suitable values for the weights in the network architecture, and this is a very difficult optimization problem. Therefore, this study set out to optimize the values of the weights and biases of the multilayer perceptron (MLP) using the proposed improved animal migration optimization (IAMO) algorithm called IAMO-MLP. The original AMO algorithm was designed inspired by the animal migration behavior of individuals that can be found in all major animal groups including birds, mammals, and insects. The main contributions of this article are: (1) The proposed IAMO algorithm has the Lévy flight strategy. (2) This article employs the AMO and IAMO algorithms for training ANN for the first time. (3) The IAMO-MLP algorithm has the ability to escape successfully from local optima. (4) The initial positions do not affect the performance of the IAMO-MLP algorithm. (5) The features of the IAMO-MLP algorithm are simplicity, requiring only a few parameters, and solving a wide array of problems.

In the experiments, datasets with different difficulty levels and different features were used. These datasets are 13 benchmark functions, five classification datasets taken from the UCI Machine Learning Repository, and a real-world problem taken from the literature. The IAMO-MLP algorithm was compared with the original AMO-MLP algorithm and nine algorithms in the literature in terms of mean squared error, classification accuracy, nonparametric statistical Friedman test, boxplot charts, and convergence graphs. The experimental results indicate that the proposed IAMO-MLP algorithm is successful not only in solving classification problems, but also in predicting the compressive strength of the rubberized concrete with an acceptable degree of accuracy. In conclusion, this study has shown that the proposed IAMO-MLP algorithm is successful in training the MLP. The proposed IAMO-MLP algorithm can be successfully used in areas such as classification, face recognition, speech recognition, pattern recognition, prediction, and optimization.

In future work, the IAMO-MLP algorithm can be applied to different datasets such as covid-19. Further research regarding the role of the activation function and threshold parameter would be worthwhile. Some of the most representative computational intelligence algorithms such as the krill herd algorithm [57], the monarch butterfly optimization [58], the earthworm optimization algorithm [59], the elephant herding optimization [60], the moth search algorithm [61], the slime mould algorithm [62], and the Harris hawks optimization [63] can be used to train the ANN in solving the civil engineering problem. Besides, the IAMO-MLP algorithm can be hybridized with another meta-heuristic algorithm to increase its success.

Acknowledgements This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Author contribution Şaban Gülcü involved in conceptualization, formal analysis, investigation, methodology, project administration, software, validation, visualization, writing—original draft, and writing—review and editing.

Declarations

Conflict of interest The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

1. Turkoglu, B.; Kaya, E.: Training multi-layer perceptron with artificial algae algorithm. Eng. Sci. Technol. Int. J. **23**, 1342–1350 (2020)
2. Öztemel, E.: Yapay sinir ağları Artificial Neural Networks. Papatya Publishing, London (2012)
3. Frimpong, E.A.; Oluwasanmi, A.; Baagyere, E.Y.; Zhiguang, Q.: A feedforward artificial neural network model for classification and detection of type 2 diabetes. J. Phys. Conf. Ser. **1734**, 012026 (2021)
4. Tümer, A.; Edebali, S.; Gülcü, Ş: Modeling of removal of chromium (VI) from aqueous solutions using artificial neural network. Iran. J. Chem. Chem. Eng. **39**, 163–175 (2020)
5. Sarkar, S.D.; KB, A.S.: Face recognition using artificial neural network and feature extraction. In: 2020 7th International Confer-

ence on Signal Processing and Integrated Networks (SPIN), IEEE, pp. 417–422 (2020)
6. Patel, P.; Doss, A.S.A.; Kalyan, L.P.; Tarwadi, P.J.: Speech recognition using neural network for mobile robot navigation. Trends Mech. Biomed. Des. **6**, 665–676 (2021)
7. Madenci, E.; Gülcü, Ş: Optimization of flexure stiffness of FGM beams via artificial neural networks by mixed FEM. Struct. Eng. Mech. **75**, 633–642 (2020)
8. Kamal, L.; Kodaz, H.: Training artificial neural network by bat optimization algorithms. Int. J. Adv. Comput. Eng. Netw. **5**, 53–56 (2017)
9. Tang, R.; Fong, S.; Deb, S.; Vasilakos, A.V.; Millham, R.C.: Dynamic group optimisation algorithm for training feed-forward neural networks. Neurocomputing **314**, 1–19 (2018)
10. Chiclana, F.; Kumar, R.; Mittal, M.; Khari, M.; Chatterjee, J.M.; Baik, S.W.: ARM–AMO: an efficient association rule mining algorithm based on animal migration optimization. Knowl.-Based Syst. **154**, 68–80 (2018)
11. Hou, L.; Gao, J.; Chen, R.: An information entropy-based animal migration optimization algorithm for data clustering. Entropy **18**, 185 (2016)
12. Duraki, S.; Demirci, S.; Aslan, S.: UAV placement with animal migration optimization algorithm. In: 2020 28th Telecommunications Forum (TELFOR), IEEE, pp. 1–4 (2020)
13. Verma, J.; Kesswani, N.: AMIGM: animal migration inspired group mobility model for mobile Ad hoc networks. Scalable Comput. Pract. Exper. **20**, 577–590 (2019)
14. Chinta, P.; Subhashini, K.; Satapathy, J.: Optimal power flow using a new evolutionary approach: animal migration optimization (2018)
15. Ülker, E.: An elitist approach for solving the traveling salesman problem using an animal migration optimization algorithm. Turk. J. Electr. Eng. Comput. Sci. **26**, 605–617 (2018)
16. Morales, A.; Crawford, B.; Soto, R.; Lemus-Romani, J.; Astorga, G.; Salas-Fernández, A.; Rubio, J.-M.: Optimization of bridges reinforcement by conversion to tied arch using an animal migration algorithm. In: International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, Springer, pp. 827–834 (2019)
17. Farshi, T.R.: A multilevel image thresholding using the animal migration optimization algorithm. Iran J. Comput. Sci. **2**, 9–22 (2019)
18. Aljarah, I.; Faris, H.; Mirjalili, S.: Optimizing connection weights in neural networks using the whale optimization algorithm. Soft. Comput. **22**, 1–15 (2018)
19. Benmessahel, I.; Xie, K.; Chellal, M.: A new evolutionary neural networks based on intrusion detection systems using multiverse optimization. Appl. Intell. **48**, 2315–2327 (2018)
20. Chatterjee, S.; Sarkar, S.; Hore, S.; Dey, N.; Ashour, A.S.; Balas, V.E.: Particle swarm optimization trained neural network for structural failure prediction of multistoried RC buildings. Neural Comput. Appl. **28**, 2005–2016 (2017)
21. Tezel, G.; Uymaz, S.A.; Yel, E.: Combining artificial Algae Algorithm to artificial neural network for optimization of weights. Data Sci. Appl. **1**, 37–44 (2018)
22. Yamany, W.; Fawzy, M.; Tharwat, A.; Hassanien, A.E.: Moth-flame optimization for training multi-layer perceptrons. In: 2015 11th International Computer Engineering Conference (ICENCO), IEEE, pp. 267–272 (2015)
23. Jaddi, N.S.; Abdullah, S.; Hamdan, A.R.: Optimization of neural network model using modified bat-inspired algorithm. Appl. Soft Comput. **37**, 71–86 (2015)
24. Erdoğan, F.; Gülcü, Ş.: Training of Artificial Neural Networks using Meta Heuristic Algorithms. In: The International Aluminium-Themed Engineering and Natural Sciences Conference (IATENS19), Konya, Turkey, pp. 124–128 (2019).
25. Mirjalili, S.: How effective is the Grey Wolf optimizer in training multi-layer perceptrons. Appl. Intell. **43**, 150–161 (2015)
26. Gülcü, Ş: Training of the artificial neural networks using states of matter search algorithm. Int. J. Intell. Syst. Appl. Eng. **8**, 131–136 (2020)
27. Zivkovic, M.; Bacanin, N.; Venkatachalam, K.; Nayyar, A.; Djordjevic, A.; Strumberger, I.; Al-Turjman, F.: COVID-19 cases prediction by using hybrid machine learning and beetle antennae search approach. Sustain. Cities Soc. **66**, 102669 (2021)
28. De Rosa, G.H.; Papa, J.P.; Yang, X.-S.: Handling dropout probability estimation in convolution neural networks using meta-heuristics. Soft. Comput. **22**, 6147–6156 (2018)
29. Cuevas, E.; Echavarría, A.; Ramírez-Ortegón, M.A.: An optimization algorithm inspired by the States of Matter that improves the balance between exploration and exploitation. Appl. Intell. **40**, 256–272 (2014)
30. Hagan, M.; Demuth, H.; Beale, M.: Neural Network Design. PWS, Boston (1996)
31. Moghaddam, A.H.; Moghaddam, M.H.; Esfandyari, M.: Stock market index prediction using artificial neural network. J. Econ. Finance Administr. Sci. **21**, 89–93 (2016)
32. Salah, M.; Altalla, K.; Salah, A.; Abu-Naser, S.S.: Predicting medical expenses using artificial neural network. Int. J. Eng. Inf. Syst. **2**, 11–17 (2018)
33. Carleo, G.; Troyer, M.: Solving the quantum many-body problem with artificial neural networks. Science **355**, 602–606 (2017)
34. Beşikçi, E.B.; Arslan, O.; Turan, O.; Ölçer, A.: An artificial neural network based decision support system for energy efficient ship operations. Comput. Oper. Res. **66**, 393–401 (2016)
35. Torabi-Kaveh, M.; Naseri, F.; Saneie, S.; Sarshari, B.: Application of artificial neural networks and multivariate statistics to predict UCS and E using physical properties of Asmari limestones. Arab. J. Geosci. **8**, 2889–2897 (2015)
36. Elkatatny, S.; Tariq, Z.; Mahmoud, M.: Real time prediction of drilling fluid rheological properties using Artificial Neural Networks visible mathematical model (white box). J. Petrol. Sci. Eng. **146**, 1202–1210 (2016)
37. Bre, F.; Gimenez, J.M.; Fachinotti, V.D.: Prediction of wind pressure coefficients on building surfaces using artificial neural networks. Energy and Buildings **158**, 1429–1441 (2018)
38. Ahire, J.: Artificial Neural Networks: the Brain behind AI, Lulu. com, (2018)
39. Braha, D.: Global civil unrest: contagion, self-organization, and prediction. PloS One **7**, e48596 (2012)
40. Li, X.; Zhang, J.; Yin, M.: Animal migration optimization: an optimization algorithm inspired by animal migration behavior. Neural Comput. Appl. **24**, 1867–1877 (2014)
41. Lai, Z.; Hu, X.; Jiang, C.: An intelligent algorithm with interactive learning mechanism for high-dimensional optimization problem based on improved animal migration optimization. Concurr. Comput. Pract. Exper. **5**, e5774 (2020)
42. Cao, Y.; Li, X.; Wang, J.: Opposition-based animal migration optimization. Math. Problems Eng. **2**, 19 (2013)
43. Humphries, N.E.; Queiroz, N.; Dyer, J.R.; Pade, N.G.; Musyl, M.K.; Schaefer, K.M.; Fuller, D.W.; Brunnschweiler, J.M.; Doyle, T.K.; Houghton, J.D.: Environmental context explains Lévy and Brownian movement patterns of marine predators. Nature **465**, 1066–1069 (2010)
44. Talbi, E.-G.: Metaheuristics: From Design to Implementation. Wiley, New York (2009)
45. Bachir, R.; Mohammed, A.M.S.; Habib, T.: Using artificial neural networks approach to estimate compressive strength for rubberized concrete. Periodica Polytechnica Civ. Eng. **62**, 858–865 (2018)
46. Kennedy, J.; Eberhart, R.: Particle swarm optimization. In: Proceedings of ICNN'95-International Conference on Neural Net-

works, Institute of Electrical and Electronics Engineers (IEEE), pp. 1942–1948 (1995)
47. Storn, R.; Price, K.: Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. J. Global Optim. **11**, 341–359 (1997)
48. Simon, D.: Biogeography-based optimization. IEEE Trans. Evol. Comput. **12**, 702–713 (2008)
49. Yang, X.-S.; Deb, S.: Cuckoo search via Lévy flights. In: IEEE 2009 World congress on nature & biologically inspired computing (NaBIC), pp. 210–214 (2009).
50. Yang, X.-S.: Firefly algorithm, Levy flights and global optimization. In: Research and development in intelligent systems XXVI, Springer, pp. 209–218 (2010)
51. Rashedi, E.; Nezamabadi-Pour, H.; Saryazdi, S.: GSA: a gravitational search algorithm. Inf. Sci. **179**, 2232–2248 (2009)
52. Karaboga, D.; Basturk, B.: A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. J. Global Optim. **39**, 459–471 (2007)
53. Raj, B.; Ganesan, N.; Shashikala, A.: Engineering properties of self-compacting rubberized concrete. J. Reinf. Plast. Compos. **30**, 1923–1930 (2011)
54. Duplan, F.; Abou-Chakra, A.; Turatsinze, A.; Escadeillas, G.; Brule, S.; Masse, F.: Prediction of modulus of elasticity based on micromechanics theory and application to low-strength mortars. Constr. Build. Mater. **50**, 437–447 (2014)
55. Gesoğlu, M.; Güneyisi, E.; Khoshnaw, G.; İpek, S.: Investigating properties of pervious concretes containing waste tire rubbers. Constr. Build. Mater. **63**, 206–213 (2014)
56. Topcu, I.B.; Sarıdemir, M.: Prediction of properties of waste AAC aggregate concrete using artificial neural network. Comput. Mater. Sci. **41**, 117–125 (2007)
57. Gandomi, A.H.; Alavi, A.H.: Krill herd: a new bio-inspired optimization algorithm. Commun. Nonlinear Sci. Numer. Simul. **17**, 4831–4845 (2012)
58. Wang, G.-G.; Deb, S.; Cui, Z.: Monarch butterfly optimization. Neural Comput. Appl. **31**, 1995–2014 (2019)
59. Wang, G.-G.; Deb, S.; Coelho, L.D.S.: Earthworm optimisation algorithm: a bio-inspired metaheuristic algorithm for global optimisation problems. Int. J. Bio-inspired Comput. **12**, 1–22 (2018)
60. Li, J.; Lei, H.; Alavi, A.H.; Wang, G.-G.: Elephant herding optimization: variants, hybrids, and applications. Mathematics **8**, 1415 (2020)
61. Wang, G.-G.: Moth search algorithm: a bio-inspired metaheuristic algorithm for global optimization problems. Memetic Comput. **10**, 151–164 (2018)
62. Li, S.; Chen, H.; Wang, M.; Heidari, A.A.; Mirjalili, S.: Slime mould algorithm: A new method for stochastic optimization. Futur. Gener. Comput. Syst. **111**, 300–323 (2020)
63. Heidari, A.A.; Mirjalili, S.; Faris, H.; Aljarah, I.; Mafarja, M.; Chen, H.: Harris hawks optimization: Algorithm and applications. Futur. Gener. Comput. Syst. **97**, 849–872 (2019)