



# Training Feedforward Neural Network Using Enhanced Black Hole Algorithm: A Case Study on COVID-19 Related ACE2 Gene Expression Classification

Elham Pashaei<sup>1</sup> · Elnaz Pashaei<sup>2</sup>

Received: 13 September 2020 / Accepted: 7 December 2020 / Published online: 23 January 2021  
© King Fahd University of Petroleum & Minerals 2021

## Abstract

The aim of this paper is twofold. First, black hole algorithm (BHA) is proposed as a new training algorithm for feedforward neural networks (FNNs), since most traditional and metaheuristic algorithms for training FNNs suffer from the problem of slow coverage and getting stuck at local optima. BHA provides a reliable alternative to address these drawbacks. Second, complementary learning components and Levy flight random walk are introduced into BHA to result in a novel optimization algorithm (BHACRW) for the purpose of improving the FNNs' accuracy by finding optimal weights and biases. Four benchmark functions are first used to evaluate BHACRW's performance in numerical optimization problems. Later, the classification performance of the suggested models, using BHA and BHACRW for training FNN, is evaluated against seven various benchmark datasets: iris, wine, blood, liver disorders, seeds, Statlog (Heart), balance scale. Experimental result demonstrates that the BHACRW performs better in terms of mean square error (MSE) and accuracy of training FNN, compared to standard BHA and eight well-known metaheuristic algorithms: whale optimization algorithm (WOA), biogeography-based optimizer (BBO), gravitational search algorithm (GSA), genetic algorithm (GA), cuckoo search (CS), multiverse optimizer (MVO), symbiotic organisms search (SOS), and particle swarm optimization (PSO). Moreover, we examined the classification performance of the suggested approach on the angiotensin-converting enzyme 2 (ACE2) gene expression as a coronavirus receptor, which has been overexpressed in human rhinovirus-infected nasal tissue. Results demonstrate that BHACRW-FNN achieves the highest accuracy on the dataset compared to other classifiers.

**Keywords** Multilayer perceptron (MLP) · Black hole optimization algorithm (BHA) · Neural Network (FNN) training · Levy flight

## 1 Introduction

Artificial neural network (ANN) is a nonparametric computational model based on the biological process of the human nervous system. The ANN is capable of modeling complex nonlinear processes with high accuracy and handling a large amount of data [1]. Subsequently, ANN, as a

powerful tool, has been commonly used in various application domains such as classification [2], function approximation [3], pattern recognition [4], signal processing [5], computer vision [6], and so on [7, 8].

The ANN is mainly divided into three groups: feedforward neural network (FNN), recurrent neural network (RNN), and Convolutional neural network. The most common types of the FNN are single layer perceptron (SLP), multilayer perceptron (MLP), and radial basis function (RBF), while the Hopfield network and Kohonen self-organizing maps (SOM) are two examples of the RNN. Among different types of ANN, the FNN especially MLP is the most popular neural network in practical applications because of its learning and generalization capacity, nonlinearly, and robustness to noise [9–11]. The MLP is composed of three or more layers of different size nodes, including one input layer, one output layer, and one or more hidden layers [12].

✉ Elham Pashaei  
epashaei@gelिसim.edu.tr

Elnaz Pashaei  
elnazpashaei@aydin.edu.tr

<sup>1</sup> Department of Computer Engineering, Istanbul Gelisim University, Istanbul, Turkey

<sup>2</sup> Department of Software Engineering, Istanbul Aydin University, Istanbul, Turkey



Recently, the learning techniques have received considerable attention in machine learning such as [13, 14]. The performance of the ANN can be greatly improved based on the learning algorithm that is used for network training [9]. The training process aims to identify the best weights and biases values that minimize the cost function, known as the mean squared error (MSE) cost function. The supervised learning techniques of ANN are classified as gradient-based methods and metaheuristic-based approaches [15]. The backpropagation (BP) algorithm and its improved versions are the most widely applied gradient-based learning methods in the literature [16]. These learning methods, however, suffer from several shortcomings including slow convergence, the tendency of trapping into local optima, and heavy reliance on the initial solution. In this regard, metaheuristic algorithms (MHAs) have been proposed as reliable alternatives to the gradient-based FNNs training methods [11]. The population-based MHAs demonstrate better performance in avoiding local minima than gradient-based methods [2, 9]. Evolutionary and swarm-based algorithms, the most widely used population-based MHAs, generate a random number of solutions and update them until the best solution is obtained or convergence is reached. Such randomness provides a mechanism to move out of the local search for a global search [9].

It is worth mentioning that the population-based MHAs in a neural network can be used not only in weights and biases optimization but also in learning rule determination and neural network structure design [9, 10]. However, simultaneous optimization of network structure, weights, biases, and learning rules can drastically increase the number of parameters, which results in large-scale optimization problems [9, 15]. In this study, the MLP with a single hidden layer is optimized by concentrating on discovering the optimal set of weight and bias values.

Various MHAs have been used to solve a wide variety of complex problems such as text clustering [17–23], image enhancement [24], gene selection [25], task scheduling problems [26], and training neural networks. The genetic algorithm (GA) is one of the earliest and best-known optimization algorithms. It has been widely used in FNNs connection weight training [27], designing its architecture [28], and simultaneously optimizing FNNs architecture and weights [29]. A hybrid of GA and BP algorithm was presented in [30] for optimization of the neural network connection weights, while the power of GA as feature selection and component optimization of FNN was examined for breast cancer diagnosis in [31]. Two other types of evolutionary algorithm called differential evolution (DE) [32, 33], and biogeography-based optimization (BBO) [34], have been used for training multilayer FNNs as well. A hybrid of DE with Levenberg–Marquardt (LM) PB method for optimizing FNN's weights and biases was introduced in [35]. Swarm intelligence algorithms are another well-known population-based MHA, which are

inspired by the species' social behavior in nature. Instead of reproduction operators in evolutionary algorithms, swarm-based algorithms employ some mathematical models for updating the randomly generated solutions. In the literature for training MLP neural network and its various applications, several optimization algorithms have been studied. Some of them are artificial bee colony (ABC) [36], gray wolf optimizer (GWO) [37, 38], grasshopper optimization algorithm (GOA) [15, 39], Krill-Herd algorithm (KHA) [40], multi-verse optimizer algorithm (MOA) [2, 41], whale optimization algorithm (WOA) [9], social spider optimization algorithm (SSO) [42], lightning search algorithm (LSA) [11], cuckoo search (CS) [43, 44], bat algorithm [45], artificial ant colony optimization (ACO) [46], invasive weed optimization [47], particle swarm optimization (PSO) [48], symbiotic organisms search (SOS) algorithm [10], hybrid of ABC and dragonfly algorithm (DA) [49], moth–flame optimization (MFO) [50, 51], salp swarm algorithm (SSA) [52], ant lion optimizer (ALO) [53], and group search optimizer (GSO) [54].

Despite the rapid development of the new MHAs, their success, and widespread application, the research is still open for new designs and improvements to existing ones due to duality between exploitation and exploration capabilities of optimization algorithms. There are few comparative studies in the mathematical analysis of these algorithms in terms of coverage and efficiency [55], and a global outcome is not yet available. Moreover, according to the no free lunch theorem, no optimization method is able to beat every other MHA to solve all problems [9, 15]. Besides, the problem of trapping into local optima and slow convergence remains partly unsolved in training FNN [12].

Motivated by these reasons, the black hole algorithm (BHA) and a novel enhanced version of BHA (BHACRW) are introduced to train FNN as new alternatives to existing MHAs and conventional approaches, aim to increase FNN's accuracy. The excellent performance of the BHA in escaping local optima, high-speed convergence, its' parameter-lessness, and simplicity, encouraged us to use the BHA for MLP training. The BHA, proposed by Hatamlou in 2012 [56], is an efficient MHA, inspired by the real black hole behavior in space. Although the original BHA demonstrates superior results in various optimization problems, in certain datasets it lacks exploration capabilities. To address the exploration dilemma, Levy flight random walk [57] and opposition-based (complementary) learning [58] components are introduced in the original BHA's framework result in a new optimization algorithm, called "BHACRW". The BHA and its modified versions [59–61] have shown promising results in solving various optimization and engineering problem, such as pattern recognition [62], function optimization [63], electric power systems [64], management system [65], and more [61, 66, 67]. However, to the best of our knowledge,

training feedforward neural networks using BHA and its improved versions is still untouched.

It is also worth noting that a hybrid of Levy flight and BHA, called LBH [61], has been considered in the literature for empowering the BHA’s global search capability to solve function and clustering problems. In LBH, levy flight is used in the updating of all-stars’ positions without any constraints, while in the BHACRW, complementary learning operator and levy flight are utilized sequentially for updating the position of half-worst solutions, if the BHA get stuck in local optima.

The main contributions of this research are: (1) BHA is proposed as a new stochastic training algorithm for FNNs. (2) For the first time, the high exploration and exploitative capability of the BHA are evaluated in determining the best values for the weight and bias parameters of FNN. (3) To boost the accuracy of FNNs, complementary learning components and Levy flight random walk are integrated into BHA to establish the new BHACRW algorithm. (3) The suggested algorithm’s performance is examined against four benchmark global optimization functions and compared with nine MHAs, including Levy firefly algorithm (LFFA) [68], gravitational search algorithm (GSA) [69], cat swarm algorithm (CSA) [70], Big Bang-Big Crunch (BB-BC) [71], ABC, PSO, GWO, BHA, and LBH. On the other hand, the classification performance of FNN using the BHACRW optimizer is evaluated against original BHA and eight well-known MHAs trainer, including GA, PSO, CS, BBO, WOA, MVO, SOS, and GSA, using mean square error (MSE) and classification accuracy metrics. The obtained results indicate that the suggested BHACRW-FNN approach produces better and, at the least, competitive results in comparison to those MHAs. (4) As a case study, BHACRW-FNN is applied to classify angiotensin-converting enzyme 2 (ACE2) gene expression for patients with asthma to identify subgroups at risk for COVID-19. The proposed classifier achieves 80%

prediction accuracy on the dataset, which is the best performance compared to other conventional classifiers.

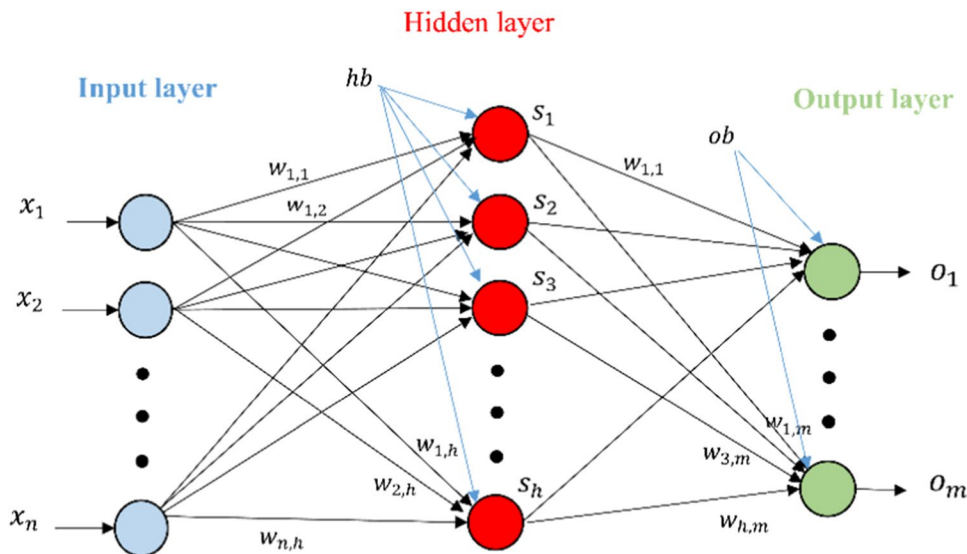
The remainder of the paper is arranged as follows: The FNN is introduced in Sect. 2. Section 3 explains the details of the suggested BHACRW method and its components including BHA, Levy flight random walk, and complementary learning approach. The FNN optimization model is explained in Sect. 4. In Sect. 5, the assessment experiments, parameter settings, and discussion have been provided. Lastly, Sect. 6 summarizes the paper’s key findings and points out potential directions for future work.

## 2 Feedforward Neural Network

A feedforward neural network (FNN) model is the simplest form of an ANN composed of three classes of layers: input, output, and hidden layers. Layers consist of a certain number of neurons (or nodes). Using weighted interconnections or links, each layer’s nodes have a full connection with all nodes in the adjacent layer [12].

A popular class of the FNN model which is mostly used for data classification is the multilayer perceptron (MLP) [9, 10, 48]. Figure 1 shows a single hidden layer MLP network, where  $x = (x_1, x_2, \dots, x_n)^T$  represents an input vector that holds  $n$  neuron values in the input layer.  $s = (s_1, s_2, \dots, s_h)^T$  indicates  $h$  neurons output vector in the hidden layer, while  $o = (o_1, o_2, \dots, o_m)^T$  shows  $m$  neurons output vector in the output layer.  $hb$  and  $ob$  represent hidden neurons biases, and output neurons biases, respectively. When a neural network structure is designed, the number of nodes and hidden layers should be specified. A larger number of nodes and hidden layers lead to greater network complexity [9, 15]. A fixed three-layer MLP was employed in this study, where a single layer was allocated for the network’s input, hidden, and output sections.

Fig. 1 MLP structure with a single hidden layer



## 2.1 Coding Schema

In the neural network, the choice of the neurons number inside of each layer depends on the problem. The number of the features and class labels of the dataset is equal to neurons number within the input layer and output layers, respectively [40, 72]. Moreover, neurons number inside the hidden layer is estimated with the Kolmogorov theorem [9, 10], defined as:

$$H = 2 * \text{Input} + 1 \quad (1)$$

Assume  $n$ ,  $h$ , and  $m$  indicate neurons number within the input, hidden and output layers, while  $hb$  and  $ob$  show the biases number inside the hidden and output layer, respectively. To utilize the suggested methods to find the optimal weight and bias values for MLP, each solution's dimension is computed as follows:

$$D = (n * h) + (h * m) + hb + ob \quad (2)$$

A population with  $N$  solution is initialized  $P = (P_1, P_2, \dots, P_N)$ , in which each solution  $P_i = \{nw, hb, hw, ob\} (i = 1, 2, \dots, N)$  provides a set of weight and bias values for MLP. In a population-based optimization algorithm, the MLP's weight and bias values inside of search agents can be represented as a vector, matrix, or binary encoding schema. The matrix encoding approach was used in the suggested BHA-MLP and BHACRW-MLP to represent each candidate solution, which is the most suitable encoding schema for the neural network training process [48]. The following equations describe the matrix encoding strategy of the aforementioned Fig. 1:

$$\text{solution}(:,i) = [W_1, hb, W_2', ob] \quad (3)$$

$$W_1 = \begin{bmatrix} w_{1,1} & \dots & w_{n,1} \\ \vdots & \ddots & \vdots \\ w_{1,h} & \dots & w_{n,h} \end{bmatrix}, hb = \begin{bmatrix} hb_1 \\ \vdots \\ hb_h \end{bmatrix},$$

$$W_2' = \begin{bmatrix} w_{1,1} & \dots & w_{h,1} \\ \vdots & \ddots & \vdots \\ w_{1,m} & \dots & w_{h,m} \end{bmatrix}, ob = \begin{bmatrix} ob_1 \\ \vdots \\ ob_h \end{bmatrix} \quad (4)$$

where  $W_1$  and  $W_2'$  represent weight matrix and the transpose of the weight matrix of the neurons between input-hidden layers and hidden-output layers, and  $hb$  and  $ob$  denote bias matrices of neurons in hidden and output layers, respectively.

## 2.2 Fitness Value

Two steps are used to calculate the MLP performance value. In the first step, the weighted summation of each hidden neuron is computed using Eq. (5) as follows:

$$s_j = \sum_{i=1}^n w_{ij}x_i + hb_j \quad j = 1, 2, \dots, h \quad (5)$$

where  $w_{ij}$  describes the connection weight between the input layer's  $i$ th node, called  $x_i$ , and the hidden layer's  $j$ th node. The bias of the hidden layer's  $j$ th node is expressed by  $hb_j$ .

An activation function is employed in the second stage to generate the hidden neuron's output. The weighted sum of each hidden neuron (Eq. 5) is passed on to the sigmoid function, which is illustrated in the equation as follows:

$$f(s_j) = \frac{1}{1 + e^{-s_j}} \quad (6)$$

On MLP layers various types of activation functions can be applied. However, the sigmoid function is the most widely used activation function in previous studies [2, 5], in which optimization algorithms are utilized for training the FNN. For the sake of comparison, the same activation function has been chosen in the MLP. After determining the outputs of the hidden layer nodes, the outputs of the neurons in the output layer are computed as follows:

$$o_k = \sum_{j=1}^h w_{jk}f(s_j) + ob_k \quad k = 1, 2, \dots, m \quad (7)$$

$$o_k = f(o_k) = \frac{1}{1 + e^{-o_k}} \quad (8)$$

where  $w_{jk}$  presents the connection weight between the hidden layer's  $j$ th neuron and  $k$ th neuron of the output layer.  $ob_k$  shows the bias of the output layer's  $k$ th neuron. The weight and bias matrices are passed to the MLP, and the fitness value of each solution in the suggested solution is computed using the mean squared error (MSE). The MSE is considered as the fitness function of proposed algorithms, which is expressed in Eq. (9)

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n (c - o)^2 \quad (9)$$

where  $n$  defines the number of training samples,  $o$  denotes predicted values of the neural network, and  $c$  symbolize the actual class labels. In addition to the MSE criterion, the classification accuracy is used to assess the classification performance of MLP on unseen data, which is determined as:

$$\text{Accuracy} = \frac{\tilde{Z}}{Z} \quad (10)$$

where  $\tilde{Z}$  and  $Z$  denote the number of samples which the classifier correctly identifies, and total sample size in the test dataset, respectively.

### 3 Proposed Methodology

The proposed BHACRW schema is explored in more detail in this section. It comprises three important parts which are BHA, the complementary-based learning components (C), and the Levy flight random walk (RW) learning method. The BHA’s random initialization is used at the first stage of the proposed method to generate candidate solutions for training NN. If the fitness values of the best global solution remain unchanged in three sequential iterations, new candidate solutions will be generated from the half-worst population using the complementary module. If complementary-based solutions have better fitness value than previous solutions, the old solutions will be replaced with new opposite solutions. Then, BHA will continue with the new population. However, if the new complementary-based solution has a worse fitness value than the origin, the Levy flight random walk module is preformed to reinitialize the solution without any fitness

comparison, and BHA continues to find optimal solutions by using the newly generated solutions. Algorithm 1 shows the suggested BHACRW pseudo-code, and Fig. 2 illustrates the main difference between the basic BHA and the improved BHA.

#### 3.1 Black Hole Algorithm

The BHA [56] belongs to the family of the physical-based optimization algorithms suggested by Hatamlou [56], which simulates the phenomenon of the black hole inside space. A mass-filled point in the space indicates a black hole (BH). All the mass around the BH falls into its center, known as the singularity. There is a strong gravitational force around the BHs singularity, called an event horizon that prevents anything from escaping [73].

**Algorithm 1** Pseudocode of BHACRW

```

01: Initialize the population with randomly generated stars
02: for  $i = 1$  to the total number of stars
03:     calculate the  $i$ th star’s fitness value and save in an array ( $f$ )
04: end
05: Determine the black hole which is the star with the highest fitness value
06: While (Criteria for max iteration T or convergence are not met ) do
07:     for  $i = 1$  to number of stars
08:          $x_i^{new} = x_i^{old} + rand * (x_{BH} - x_i^{old})$  /*The position of the stars change */
09:         Assess the new star’s fitness value ( $x_i^{new}$ )
10:         if the new star’s fitness value ( $x_i^{new}$ ) < fitness value of  $x_{BH}$  then
11:              $x_{BH} = x_i^{new}$  /* when a star enters a lower-cost position than the black hole (BH),
12:                 the location of the BH is changed with the star*/
13:         end
14:         Update the star’s fitness value ( $f_i$ ) with the new value
15:         Update the fitness value in the array ( $f$ )
16:         Calculate the radius of the event horizon  $R = \frac{f_{BH}}{\sum_{i=1}^{n_{stars}} f_i}$ 
17:         if  $\sqrt{(x_{BH} - x_i^{new})^2} < R$  Then
18:             Reinitialize the star randomly in the search space
19:         end
20:     end /* end of original BHA algorithm*/
21:     if the fitness value of  $x_{BH}$  remains unchanged in three iteration
22:         for  $i = 1$  to half number of the stars with the worst fitness values
23:              $x_i^{opposite} = x_{max} + x_{min} - x_i^{old}$  /*The position of the worst stars change using complementary */
24:             if the complementary based star’s fitness value ( $x_i^{opposite}$ ) < fitness value of  $x_i^{old}$  then
25:                  $x_i^{new} = x_i^{opposite}$ 
26:             else
27:                  $x_i^{new} = x_i^{old} + \alpha.s$ 
28:             End
29:         end
30:     end

```

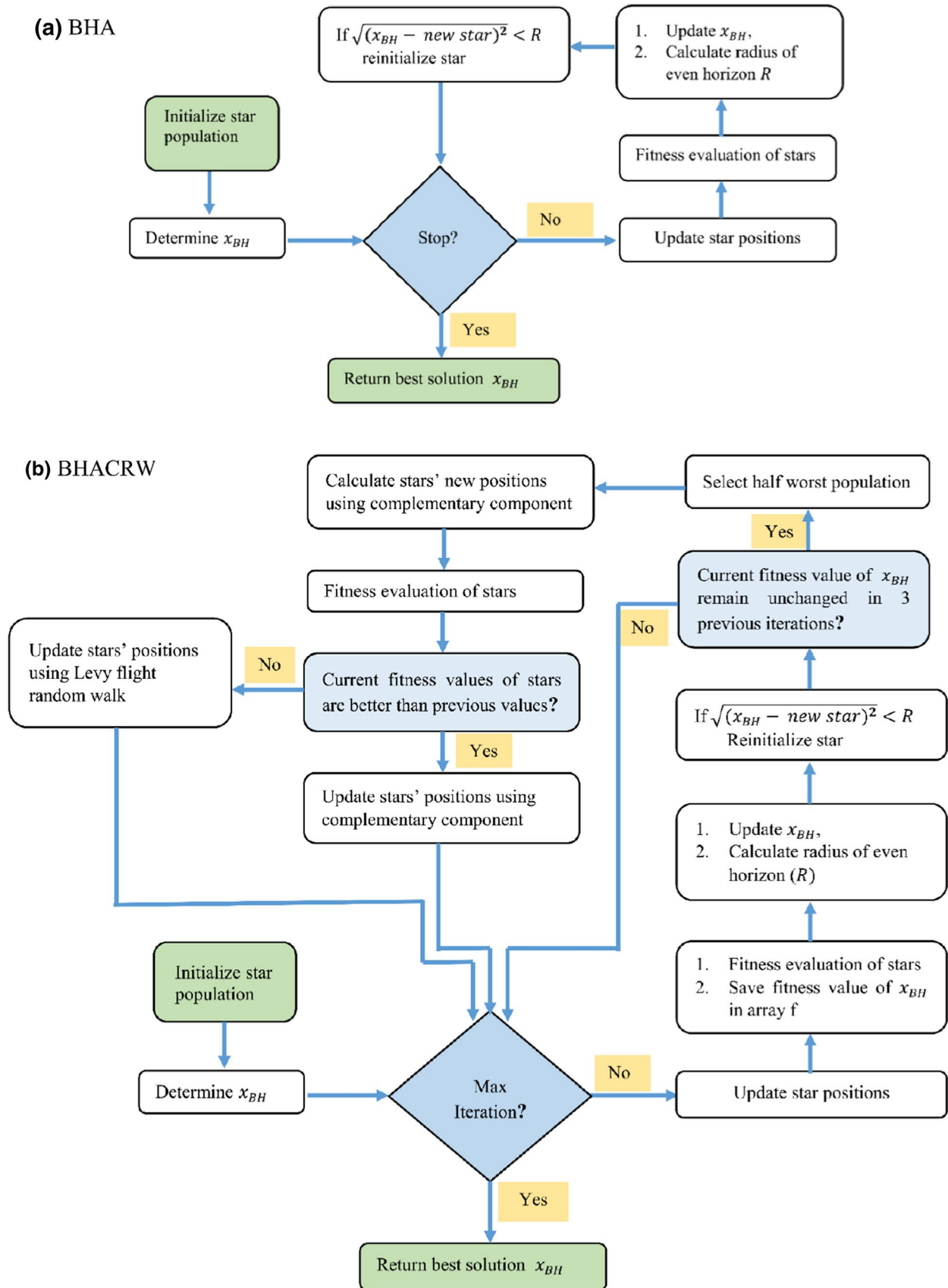


Fig. 2 The flowchart of the **a** basic black hole algorithm and **b** improved black hole algorithm

In the BH algorithm, an individual solution is called a star. The BHA begins with a random generating of the stars (population) in the search space. Then, it looks at all these stars to find the best individual solution, which is defined as a black hole (BH). Since the best location belongs to the BH in the search space, all stars begin moving toward the BH and update their positions, which are carried out through the following equation:

$$x_i^{new} = x_i^{old} + rand * (x_{BH} - x_i^{old}) \tag{11}$$

where

- $x_i^{old}$  and  $x_i^{new}$  represent the  $i$  th star position (candidate solution) for iterations  $t$  and  $t + 1$ , respectively
- $rand$  describes a random value between 0 and 1.
- $x_{BH}$  represents the black hole’s position (best solution) in search space.

The black hole starts drawing the stars into itself. However, if a star is in a better position than the BH as it moves toward the black hole, the  $x_{BH}$  changes its current position and moves to the star’s position, and the algorithm continues [74]. Moreover, the black hole immediately swallows the stars that cross the event horizon during the process of their movement, and thus it allows the creation of new stars at random in the search space. The event horizon radius ( $R$ ) is calculated using Eq. (12):

$$R = \frac{f_{BH}}{\sum_{i=1}^{popsize} f_i} \tag{12}$$

where

- $f_{BH}$  is the black hole fitness value
- $f_i$  is the  $i$  th star’s fitness value
- $popsize$  is the total star population

The star disappears from the space solution when its distance from the black hole is smaller than the event horizon radius [56]. It is important to note that a fitness function determines the goodness of each star based on position. Optimization algorithms are essentially iterative procedures, in which some updating formulas are used to improve candidate solutions in each iteration. To learn the optimization algorithm, the iterative updating process must be integrated with a fitness function. The BHA’s learning process is conducted via Eqs. (11) and (12) in an iterative manner, in which candidate solutions (stars) move toward the optimal solution (BH). Each solution stands for a candidate for FNN’s parameters. Figure 2a illustrates the flowchart of BHA.

Because of the simple structure, flexibility, high local optima avoidance, and the BHA’s parameter-less nature, various optimization and engineering problems have been successfully solved utilizing the BHA. This inspired us to employ the BHA for training FNNs due to the difficulties of the learning process. However, the BHA fails to show its excellent performance in some datasets, so an improved version of BHA has been suggested to enhance its global search capability using the opposition-based learning components and Levy flight random walking.

### 3.2 The Complementary-Based Learning Components

The suggested algorithm utilizes the complementary learning components when the best global solution remains unchanged in three sequential iterations, as it is believed that better performance could be obtained using opposite positions [58]. To continue with a better population, the half-worst stars are selected, and the opposite location of each of them is computed. Then, the fitness value of the current location of the star is compared with its new opposite location. If the star’s opposite location has a higher fitness value, the star’s original position will be updated with its complementary position. The complementary position of stars is computed using the following equation:

$$x_i^{opposite(new)} = x_{max} + x_{min} - x_i^{old} \tag{13}$$

where  $[x_{min}, x_{max}]$  presents the initial interval of the star position.

### 3.3 The Levy Flight Random Walk

If the abovementioned complementary stage fails to produce better solutions, the random walk is utilized to generate new solutions. The random walk is a stochastic process describing a path with a series of random steps, in which the steps are described by the step-size following certain probability distribution. Levy flight is a type of random walk, where its step-size is based on Levy distribution. Levy flight enhances the efficiency of the BHA by improving BHA’s global search capability and avoiding local optima [61]. The stars’ new position using Levy flight random walk can be calculated by:

$$x_i^{new} = x_i^{old} + \alpha.s \tag{14}$$

$\alpha$  is an array of normally distributed random numbers, and  $s$  indicates the steps of Levy flight which is calculated as follows:

$$s = 0.01 * \left( \frac{\mu}{|v|^{\frac{1}{\beta}}} \right) * (x_i^{\text{old}} - x_{\text{BH}})$$

$$\beta = 2/3; u \sim N(0; \sigma_u^2); v \sim N(0; \sigma_v^2) \tag{15}$$

where  $u$  and  $v$  are drawn from a normal distribution with zero mean and standard deviations  $\sigma_u^2$  and  $\sigma_v^2$ . Here,  $\sigma_v = 1$  and  $\sigma_u$  follow Levy distribution, given by:

$$\sigma_u = \left\{ \frac{\text{gamma}(1 + \beta) * \sin\left(\frac{\pi * \beta}{2}\right)}{\text{gamma}\left(\left(\frac{1 + \beta}{2}\right) * \beta * 2^{\frac{\beta - 1}{2}}\right)} \right\}^{\frac{1}{\beta}} \tag{16}$$

Finally, the proposed algorithm generates the best candidate solution for the weight and bias of FNN. The following section describes the mechanisms for training FNNs using the BHA and BHACRW algorithms, called BHA-FNN and BHACRW-FNN.

### 4 Training MLP with the BHACRW

The suggested black hole-based algorithm for training a single layer MLP is described in this section. To decrease the MLP’s overall error and subsequently increase its accuracy, the BHA and BHACRW are used to train MLP by determining the best weights and biases values. In other words, we can learn the neural network, by learning its weights and biases. The equation numbers (5–8) demonstrate the FNN model, which correlates the inputs to the outputs. Candidate solutions contain a set of weight and bias values, which are initialized randomly at the first stage. The MSE is considered as the cost function expressed by Eq. (9), in which Eq. (5–8) are employed to calculate the neural network output. The minimization of the cost function in an iterative manner is referred to as neural network learning. The candidate weights and biases of MLP are updated at each iteration which leads to cost minimization.

The suggested BHACRW-MLP approach can be described in detail in the following steps:

Step. 1 A population of  $N$  solutions (stars) is randomly generated, while each solution encodes a set of MLP’s weights and biases values.

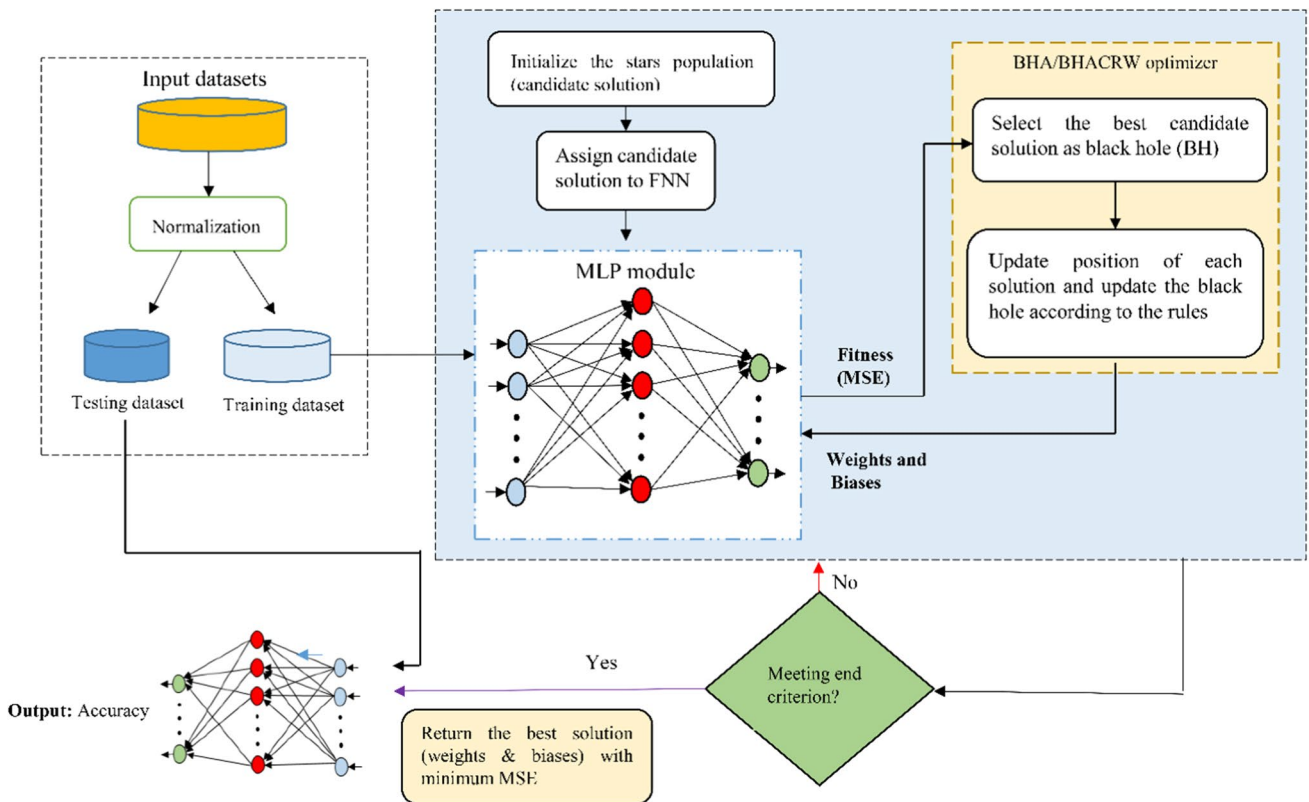


Fig. 3 The framework of the BHACRW-MLP approach



**Table 1** Benchmark test functions

Fun	Name	Test	D	Range	Opt
$f_1$	Sum square	$f_1(x) = \sum_{i=1}^N x_i^4$	30	[- 10, 10]	0
$f_2$	Quartic	$f_2(x) = \sum_{i=1}^N ix_i^4 + \text{random}(0, 1)$	30	[- 1.28, 1.28]	0
$f_3$	Ackley	$f_3(x) = -20e^{-0.02} \sqrt{D^{-1} \sum_{i=1}^D x_i^2} - e^{D^{-1} \sum_{i=1}^D \cos(2\pi x_i)} + 20 + e$	30	[- 32, 32]	0
$f_4$	Alpine No.1	$f_4(x) = \sum_{i=1}^D  x_i \sin(x_i) + 0.1x_i $	30	[- 10, 10]	0

Step. 2 The goodness of each solution is assessed using MSE criteria. The MSE of MLP is calculated for each star on a given training dataset. The star with the minimum MSE is chosen as the best solution ( $x_{BH}$ ).

Step. 3 Each star’s location and the best global are updated iteratively, and the event horizon is calculated to produce new solutions

Step. 4 A counter is set to count the number of iterations in which the best fitness value of the global solution stays unchanged.

Step. 5 If the counter reaches a predefined number of no improvement (here 3 times), the opposite positions for half number of the stars with the worst fitness values are calculated to establish a better population. Their fitness values are computed and compared with the stars’ previous positions. For each star, if the opposite position of the star has better fitness value, it will be inserted into the population and jump to step 7, else go to step 6.

Step. 6 For the stars whom their complementary positions are not better than their original positions, the Levy

**Table 2** Experimental results for test functions

Algorithms	Fun1			Fun2		
	Best	Mean	Std	Best	Mean	Std
Artificial Bee Colony (ABC)	2.79e-16	2.72e-16	8.51e-12	0.11531	0.19593	0.05549
Particle Swarm Optimization (PSO)	2.13485	4.98451	3.94512	1.3389	6.96060	0.64770
Levy Firefly Algorithm (LFFA)	0.00774	0.21006	0.34752	0.00409	0.02542	0.02312
Gray Wolf Optimizer (GWO)	0.00000	0.00000	0.00000	0.00284	0.00379	0.00134
Gravitational Search Algorithm (GSA)	0.00156	0.02943	0.08790	0.06348	0.08815	0.04413
Cat Swarm Algorithm (CSA)	4.97e-04	0.00105	4.41e-04	0.01741	0.02845	0.00148
Big Bang-Big (BB-BC)	4.1458	5.9475	2.1354	3.45892	5.48953	0.83211
Black Hole Algorithm (BHA)	3.34e-04	0.00348	3.12e-03	0.02348	0.03154	0.00284
Levy Flight Black Hole Algorithm (LBH)	0.00000	0.00000	0.00000	0.00014	0.00091	0.00053
BHACRW	2.60e-17	6.29e-16	1.46e-15	7.29e-05	0.00052	0.00019
Algorithms	Fun3			Fun4		
	Best	Mean	Std	Best	Mean	Std
Artificial Bee Colony (ABC)	0.02058	0.15442	0.00000	0.00042	0.28568	0.62473
Particle Swarm Optimization (PSO)	1.98770	2.94390	0.03719	0.00425	2.67570	12.3490
Levy Firefly Algorithm (LFFA)	0.06340	1.99940	0.00013	0.00024	0.00029	0.00037
Gray Wolf Optimizer (GWO)	0.06920	0.03660	5.49e-10	0.00116	0.10797	0.25769
Gravitational Search Algorithm (GSA)	0.00002	0.00027	0.55632	0.00493	0.02171	0.00928
Cat Swarm Algorithm (CSA)	1.22930	3.18530	0.02419	0.00005	0.00248	0.00048
Big Bang-Big (BB-BC)	1.5829	3.8331	1.0422	0.00064	1.06309	1.79308
Black Hole Algorithm (BHA)	0.02058	0.06922	0.01944	0.00481	0.08741	0.03847
Levy Flight Black Hole Algorithm (LBH)	0.00582	0.03833	0.01042	0.00004	0.00024	0.00031
BHACRW	5.43e-05	0.00012	5.09e-05	3.21e-05	5.79e-05	3.17e-05

**Table 3** Description of datasets

Dataset	#Attributes	#Class	#Training sample	#Testing sample	MLP structure
(Input- Hidden- output)					
Iris	4	3	100	50	4-9-3
Wine	13	3	118	60	13-27-3
Blood	4	2	494	254	4-9-2
Liver disorders	6	2	228	117	6-13-2
Seeds	7	3	140	70	7-15-3
Statlog (Heart)	13	2	179	91	13-27-2
Balance scale	4	3	413	212	4-9-3

flight random walk is utilized to specify their new positions.

**Step. 7** After constructing a new population, the algorithm goes to step 2, and the process is repeated until reaching the maximum number of iteration.

**Step. 8** In the end, the solution with the lowest MSE is selected. The weights and biases in the best solution are assigned to MLP and its performance is assessed on the test data.

Figure 3 shows the general steps of the BHACRW-MLP procedure.

## 5 Results and Discussions

To examine the effectiveness of the suggested BHACRW approach, experiments are conducted in three parts. In the first part, the BHACRW is applied to numerical optimization problems, and its performance is compared with the result of other MHAs. The second part evaluates BHA and BHACRW performance in training MLP neural networks and a comparison of results with other well-known algorithms is provided. This part is also divided further into three subsections: experimental setup, experimental results

of seven benchmark datasets, and statistical analysis. In the third part, the application of the proposed BHACRW-MLP is investigated on a real coronavirus related gene expression dataset and its performance evaluated against several conventional classification methods.

### 5.1 Evaluation on Numerical Function Optimization

Four standard test functions were used to assess the efficiency of BHACRW. Table 1 lists these benchmark functions with their ranges and dimensions. To properly apply BHACRW in function minimization, first, we removed the condition for testing the fixed number of iteration, second, we used regular random walk (step = 1) and used Eq. (17) instead of Eqs. (14) and (15).

$$s = 0.01 * \text{step} * \text{randn} \quad (17)$$

where randn returns a random number drawn from the standard normal distribution.

All experiments were conducted on 10 separate runs, with 250 iterations per run. The population was set at 25. Under the same conditions as our experiments, the parameters setting and experimental results for other evolutionary

**Table 4** Experimental results for the iris dataset

Algorithms	MSE				Accuracy (%)		
	Best	Worst	Mean	STD	Best	Worst	Mean
WOA-MLP	0.0197	0.0261	0.0229	0.0032	–	–	91.3333
GA-MLP	0.119	0.6510	0.3660	0.1700	98.0392	7.8431	56.0784
CS-MLP	0.0101	0.0784	0.0532	0.0193	98.0392	52.9412	85.0000
PSO-MLP	0.0471	0.1760	0.1090	0.0355	98.0392	52.9412	91.4706
BBO-MLP	0.0170	0.4650	0.0691	0.1110	98.0392	29.4100	82.9400
MVO-MLP	0.0245	0.0275	0.0258	0.0008	98.0392	98.0392	98.0392
GSA-MLP	0.0441	0.2310	0.0683	0.0407	98.0392	33.3300	93.7255
SOS-MLP	5.1e-08	0.0267	0.0142	0.0088	98.0392	64.7059	92.0588
BHA-MLP	0.0176	0.0213	0.0191	0.0013	98.0392	98.0392	98.0392
BHACRW-MLP	0.0173	0.0201	0.0190	0.0007	98.0392	98.0392	98.0392



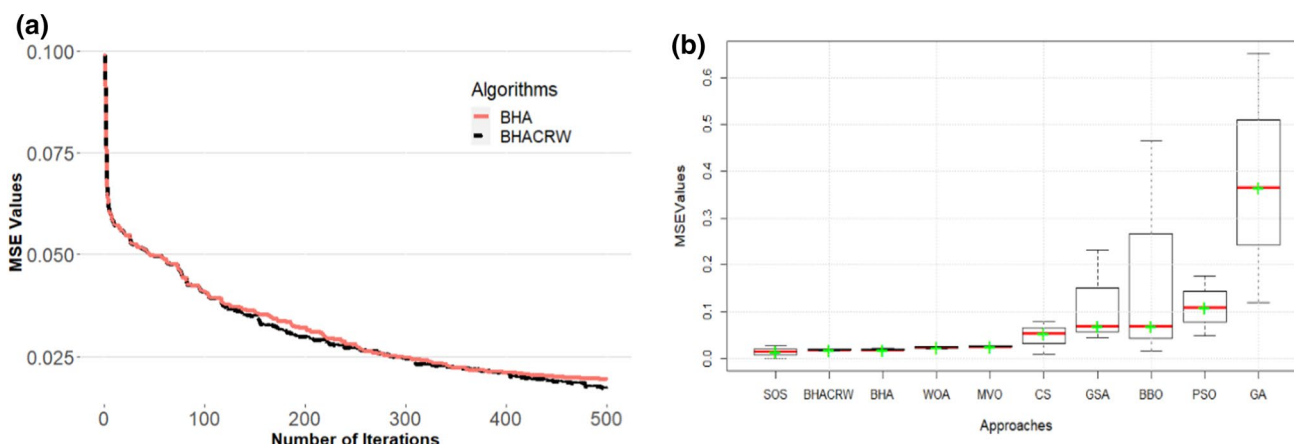


Fig. 4 Convergence curve of BHA-MLP and BHACRW-MLP **a** and box plot chart **b** for the iris dataset

algorithms, including ABC, PSO, LFFA, GWO, GSA, CSA, BB-BC, BHA, and LBH, were derived from Ref. [61]. The experimental results are tabulated in Table 2. Statistically speaking, for the best value and the average of 10 runs on 4 test functions, BHACRW is the best on 3 functions, LBH and GWO algorithms are the best on 1 function. Therefore, BHACRW in most function minimization outperforms all the evolutionary algorithms.

### 5.2 Evaluation in Training MLP Neural Network

From the University of California at Irvine (UCI) Machine Learning Repository seven standard classification datasets were employed to evaluate the efficiency of the two BHA-based trainers. The main characteristics of these datasets have been shown in Table 3, in which the size of the features, classes, and samples of the training and testing data have been reported. As can be observed, the chosen datasets contain different sizes of features and samples. It was conducted to monitor the performance of training algorithms under various conditions, which makes the process more difficult.

#### 5.2.1 Experimental Setup

To maintain the class distribution as much as possible, all datasets were divided into two subsets using stratified sampling, where 66% was allocated to the training set and the remaining 34% was assigned to the testing set. Besides, all datasets were standardized at the interval  $[-1, 1]$  using the min–max normalization to reduce the impact of attributes with different scales [2, 10]. The min–max normalization can be obtained by the following formula:

$$v' = 2 * \frac{v - \min}{\max - \min} - 1 \tag{18}$$

where  $v'$  represents the standardized value of  $v$  within the  $[\min, \max]$  range.

All experiments were carried out on 10 separate runs, with 500 iterations in each run [10]. The BHA and its enhanced version have not any parameters for adjustment, except the population size which is considered 200. Experimental results for other evolutionary algorithms, including GA-MLP [27], PSO-MLP [48], CS-MLP [44], BBO-MLP

Table 5 Experimental result for wine dataset

Algorithms	MSE				Accuracy (%)		
	Best	Worst	Mean	STD	Best	Worst	Mean
WOA-MLP	0.0400	0.0600	0.0500	–	95.45	–	88.94
GA-MLP	0.5600	0.8950	0.7030	0.0859	49.18	18.03	32.62
CS-MLP	0.0256	0.162	0.0959	0.0348	91.80	78.68	83.36
PSO-MLP	0.00974	0.0803	0.0339	0.0155	100	88.52	96.06
BBO-MLP	5.62e–12	0.342	0.0517	0.1250	100	59.01	90.49
MVO-MLP	1.34e–06	2.91e–05	5.41e–06	6.23e–06	98.36	96.72	97.95
GSA-MLP	0.000587	0.0190	0.0032	0.0039	100	91.80	96.14
SOS-MLP	66.35e–11	0.0085	0.0004	0.0019	98.36	91.80	95.49
BHA-MLP	0.0031	0.0034	0.0038	0.0003	100	98.33	99.16
BHACRW-MLP	0.0033	0.0037	0.0040	0.0004	100	98.33	99.16

[75], GSA-MLP [48], MVO-MLP [2], SOS-MLP [10], and WOA-MLP [9], were derived from Ref. [9, 10] under the same conditions as our experiments. The neurons number in

the hidden layer was computed by Eq. 1 [2, 9, 10, 75]. The structure of MLP for each dataset has been shown in Table 3.

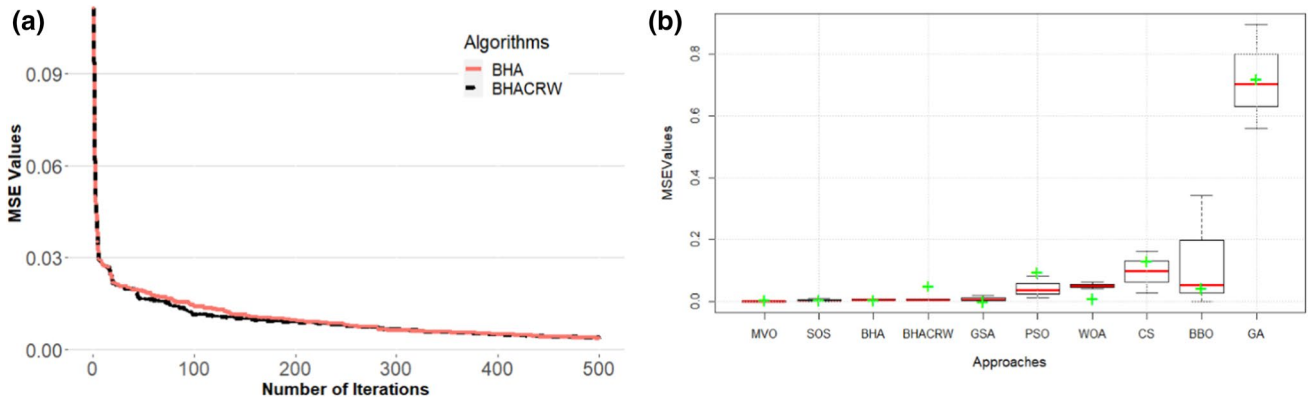


Fig. 5 Convergence curve of BHA-MLP and BHACRW-MLP **a** and box plot chart **b** for the wine dataset

Table 6 Experimental result for blood dataset

Algorithms	MSE				Accuracy (%)		
	Best	Worst	Mean	STD	Best	Worst	Mean
WOA-MLP	0.153	0.154	0.153	–	79.61	–	78.67
GA-MLP	0.330	0.417	0.378	0.0285	76.86	67.05	72.78
CS-MLP	0.311	0.321	0.317	0.0029	78.82	74.50	76.96
PSO-MLP	0.307	0.314	0.310	0.0024	80.39	77.64	79.21
BBO-MLP	0.300	0.392	0.318	0.0284	81.17	72.54	77.52
MVO-MLP	0.304	0.307	0.305	0.0007	81.17	80.00	80.74
GSA-MLP	0.310	0.333	0.323	0.0066	78.03	33.33	74.17
SOS-MLP	0.295	0.305	0.301	0.0024	82.74	77.64	79.80
BHA-MLP	0.1528	0.1541	0.1537	0.0002	82.28	77.66	79.52
BHACRW-MLP	0.1528	0.1532	0.1530	0.0001	82.28	81.88	81.38

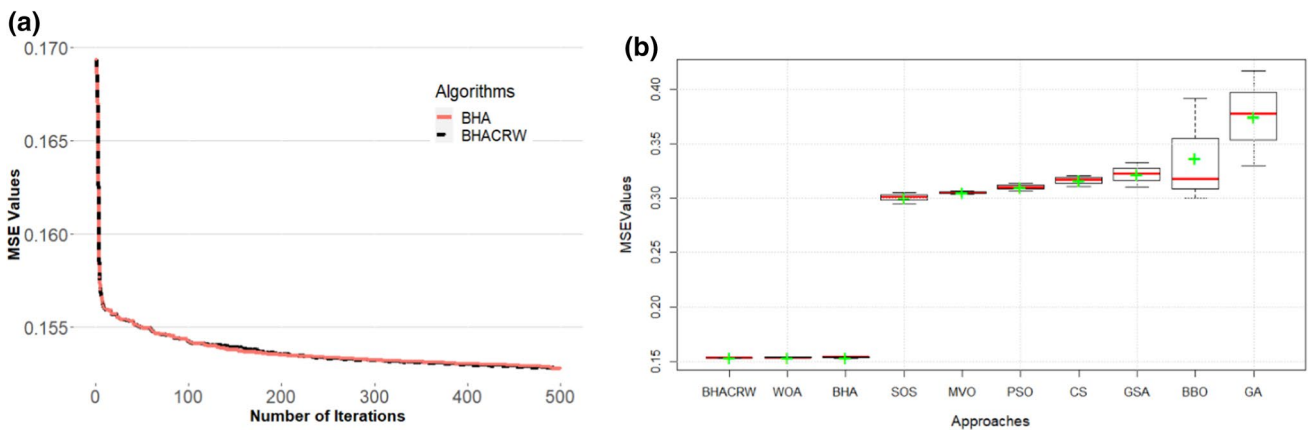


Fig. 6 Convergence curve of BHA-MLP and BHACRW-MLP **a** and box plot chart **b** for the blood dataset

### 5.2.2 Experimental Result

**5.2.2.1 Experiment 1: Iris Dataset** Iris dataset has 4 features and 3 classes, so we used an FNN with a 4-9-3 structure to find the best optimal values for weights and biases parameters of MLP. Table 4 displays the experimental results for the proposed BHA-MLP, BHACRW-MLP, and other MHAs-based trainers on the iris dataset. The performance values of the MSE assessment criteria were reported in the table, which contains the best, worst, average, and standard deviation, and were symbolized as Best, Worst, Average, and STD in the table accordingly.

Based on the results obtained, it can be seen that in most measurements the results of the BHACRW, BHA, MVO, and SOS are quite better than those of other algorithms. In Fig. 4, the average convergence trends of BHA and BHACRW and a box plot of all MSE algorithms are shown in detail. When analyzing the convergence curves (Fig. 4a), it is obvious that the BHA and BHACRW can quickly reach the minimum MSE. The comparison between the convergence curves clearly shows that at the 250th iteration the

proposed algorithms can find the best values for parameters of MLP. Table 4 and Fig. 4a also show that, in neural network training, the performance of the proposed BHACRW is marginally better than simple BHA, especially in terms of best-case MSE, STD, and the worst-case MSE.

By investigating the plot box, the proposed trainers are shown to have the smallest MSE averages after SOS and one of the most compact boxes, which shows the suggested method’s stability.

**5.2.2.2 Experiment 2: Wine Dataset** The wine dataset has 13 features and 3 categories, so we used a 13-27-3 FNN model to find the best weight and bias values for MLP. Table 5 shows the results of MHAs-based MLP networks in the wine dataset. The results show that proposed algorithms could achieve the highest classification accuracy ratio of 99.16 percent. Besides, in terms of MSE measurement, they can achieve competitive results. Whereas the SOS and GSA algorithms show better results in terms of MSE compared to BHA, both of them have a high MSE variability (std) and therefore show poor stability. In this dataset, MVO shows

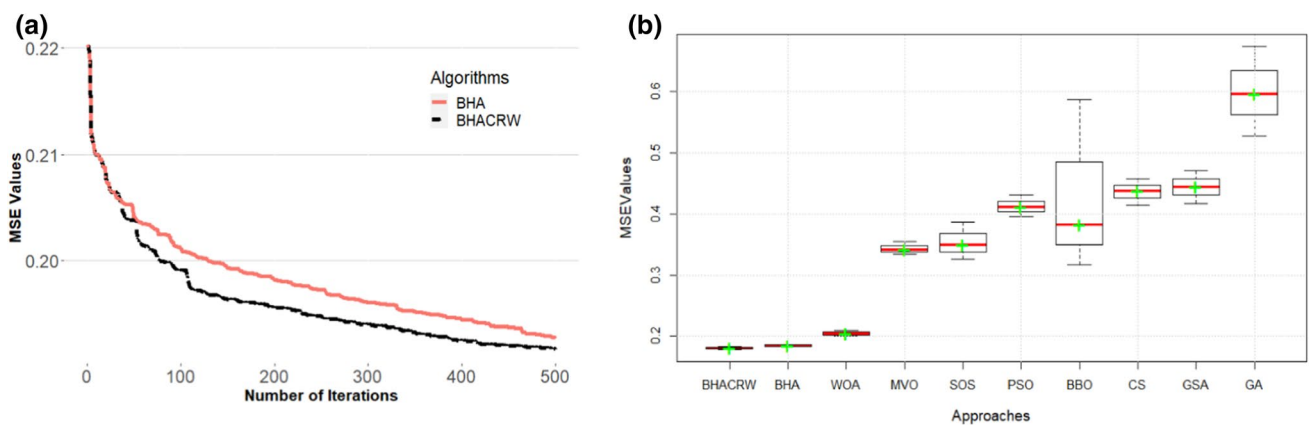


Fig. 7 Convergence curve of BHA-MLP and BHACRW-MLP a and box plot chart b for the Liver Disorder dataset

Table 7 Experimental result for liver disorders dataset

Algorithms	MSE				Accuracy (%)		
	Best	Worst	Mean	STD	Best	Worst	Mean
WOA-MLP	0.200	0.210	0.205	–	73.73	–	69.58
GA-MLP	0.527	0.673	0.595	0.043	55.08	27.11	43.05
CS-MLP	0.414	0.457	0.437	0.0112	67.79	47.45	56.14
PSO-MLP	0.396	0.431	0.411	0.0106	72.033	59.322	66.483
BBO-MLP	0.316	0.587	0.382	0.0547	72.88	45.76	65.12
MVO-MLP	0.333	0.355	0.341	0.00617	76.27	72.03	74.19
GSA-MLP	0.417	0.471	0.444	0.0132	64.40	6.77	48.30
SOS-MLP	0.326	0.386	0.349	0.0165	75.42	66.101	71.05
BHA-MLP	0.1834	0.1866	0.1845	0.0014	70.08	67.52	68.72
BHACRW-MLP	0.1788	0.1833	0.1808	0.0017	73.73	67.52	69.58

superior performance compared to other nature-inspired algorithms.

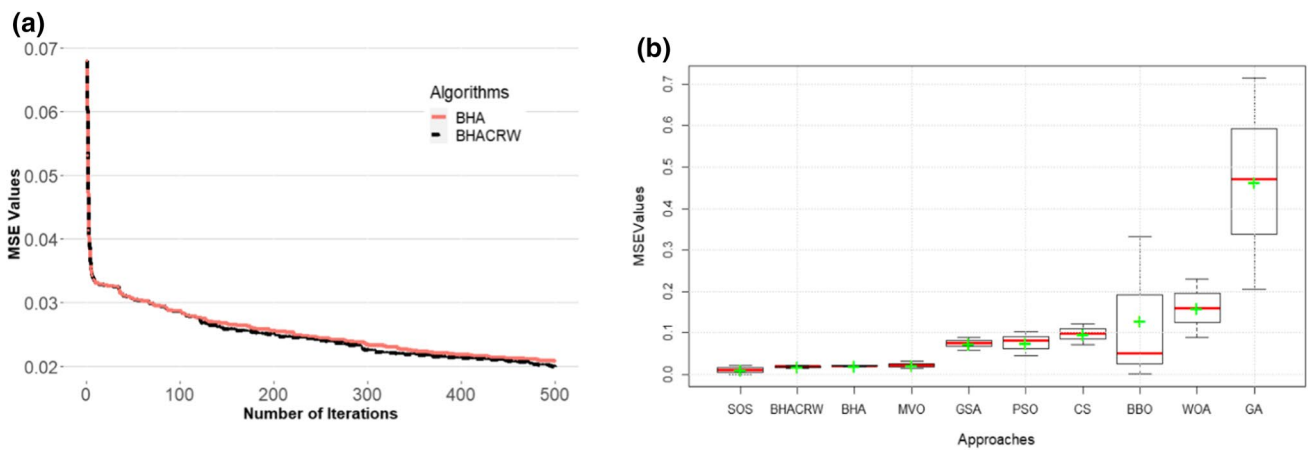
The plot box shown in Fig. 5 indicates the competitive error and acceptable stability of the BHA. For the wine dataset, the results of simple BHA is better than BHACRW. The comparison of the average convergence curves reveals that

the two curves are very close, which means that the algorithm can achieve an optimum MSE value at most times.

**5.2.2.3 Experiment 3: Blood Dataset Result** The blood dataset includes four features and two categories, so we employed the 4-9-3 FNN model to find the best weight and

**Table 8** Experimental result for seeds dataset

Algorithms	MSE				Accuracy (%)		
	Best	Worst	Mean	STD	Best	Worst	Mean
WOA-MLP	0.0900	0.2300	0.1600	–	93.06	–	89.86
GA-MLP	0.2050	0.7140	0.4710	0.1200	67.60	28.16	51.12
CS-MLP	0.0719	0.1220	0.0980	0.0134	91.54	77.46	82.25
PSO-MLP	0.0449	0.1020	0.0817	0.0153	92.95	78.87	87.46
BBO-MLP	0.0020	0.3320	0.0523	0.0963	94.36	61.97	87.32
MVO-MLP	0.0144	0.0330	0.0221	0.0063	95.77	90.14	93.45
GSA-MLP	0.0597	0.0887	0.0765	0.0088	94.36	85.91	90.35
SOS-MLP	0.0009	0.0226	0.0111	0.0053	95.77	87.32	91.33
BHA-MLP	0.0196	0.0237	0.0211	0.0016	95.77	81.69	91.54
BHACRW-MLP	0.0145	0.0231	0.0203	0.0036	95.77	91.54	94.36



**Fig. 8** Convergence curve of BHA-MLP and BHACRW-MLP **a** and box plot chart **b** for seeds dataset

**Table 9** Experimental result for the Statlog (Heart) dataset

Algorithms	MSE				Accuracy (%)		
	Best	Worst	Mean	STD	Best	Worst	Mean
WOA-MLP	–	–	–	–	–	–	–
GA-MLP	0.452	0.697	0.553	0.072	70.65	39.13	50.54
CS-MLP	0.231	0.309	0.261	0.018	83.69	68.47	77.44
PSO-MLP	0.172	0.220	0.193	0.013	86.95	77.17	82.88
BBO-MLP	0.080	0.165	0.127	0.023	80.43	66.30	75.54
MVO-MLP	0.050	0.081	0.064	0.009	88.04	77.17	82.60
GSA-MLP	0.135	0.192	0.162	0.014	86.95	33.69	79.18
SOS-MLP	0.0898	0.126	0.109	0.010	85.86	77.17	82.22
BHA-MLP	0.0669	0.081	0.0720	0.0036	87.64	81.31	83.73
BHACRW-MLP	0.0647	0.071	0.0679	0.0027	87.64	83.51	84.06

bias for MLP. Table 6 displays the blood dataset test results. In this dataset, the BHACRW has the highest accuracy and the lowest MSE compared to all nature-inspired algorithms.

The BHA has the third-highest accuracy with a slight difference from the SOS. As can be seen, after BHACRW, BHA and WOA achieved the lowest error in terms of MSE.

The results of their evaluation were very close and competitive. According to the boxplots Fig. 6b, the BHACRW, BHA, and WOA have shown a low variability. Figure 6a also shows that the BHA-based trainer can maintain a stable balance between exploration and exploitation tendencies in

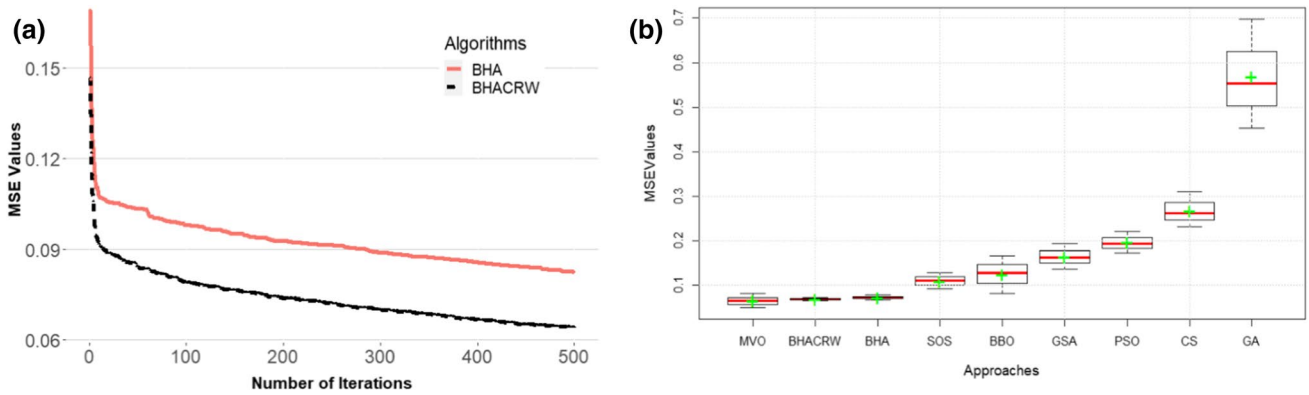


Fig. 9 Convergence curve of BHA-MLP and BHACRW-MLP a and box plot chart b for Statlog (Heart) dataset

Table 10 Experimental result for balance scale dataset

Algorithms	MSE				Accuracy (%)		
	Best	Worst	Mean	STD	Best	Worst	Mean
WOA-MLP	–	–	–	–	–	–	–
GA-MLP	0.297	0.801	0.465	0.1160	80.28	38.2	59.76
CS-MLP	0.170	0.214	0.189	0.0118	90.61	82.15	86.31
PSO-MLP	0.140	0.188	0.172	0.0132	89.20	83.56	86.71
BBO-MLP	0.085	0.124	0.102	0.0094	91.54	88.26	90.16
MVO-MLP	0.080	0.104	0.086	0.0060	92.95	89.20	91.43
GSA-MLP	0.137	0.166	0.152	0.0094	91.07	85.91	87.74
SOS-MLP	0.070	0.129	0.105	0.0133	92.01	86.85	90.02
BHA-MLP	0.0428	0.0459	0.0446	0.0011	93.39	87.26	90.02
BHACRW-MLP	0.0420	0.0440	0.0434	0.00097	93.39	89.15	91.22

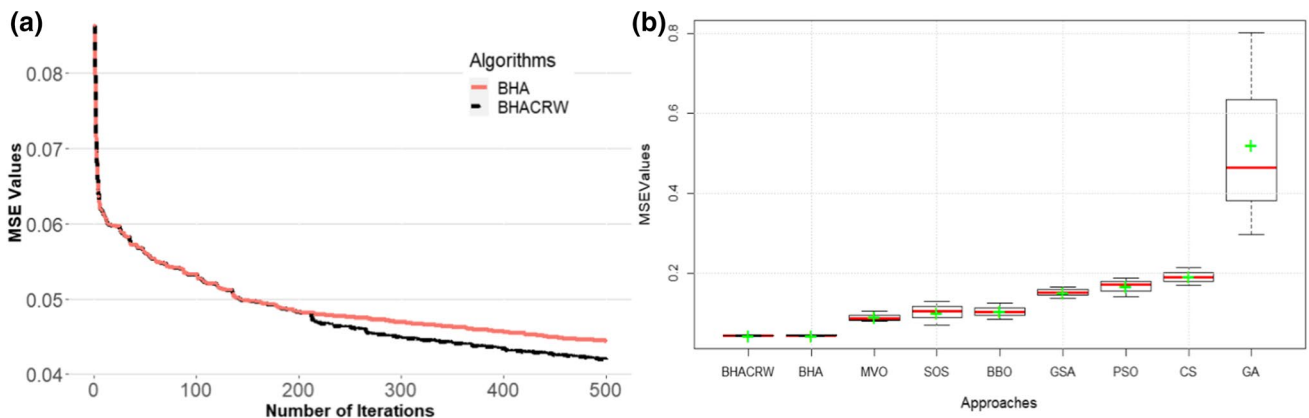


Fig. 10 Convergence curve of BHA-MLP and BHACRW-MLP a and box plot chart b for the balance scale dataset

terms of convergence curves, and most of the time can reach to least MSE.

**5.2.2.4 Experiment 4: Liver Disorders Dataset** The liver disorder dataset has 6 attributes and 2 classes, thus we utilized a 6-13-2 FNN model to find the best MLP weight and bias. The BHA and BHACRW techniques in competition with all other trainers could achieve better results with an MSE value of 0.1845 and 0.1808, respectively. Other competitive algorithms in terms of accuracy are SOS, MVO, and WOA; but, except for WOA, these algorithms demonstrated high variability in box plots. The box plot shows that the BHA and BHACRW-based trainers have the lowest MSE average and one of the most compact boxes indicating the consistency of the suggested methods for training. The convergence curve in Fig. 7a and the result of experiments in Table 7 confirm that the BHA and BHACRW have the fastest convergence trends as they can reach the lowest MSE before iteration 100. The result of the BHACRW is promising than BHA in terms of both MSE and accuracy.

**5.2.2.5 Experiment 5: Seeds Dataset** We used a 7-15-3 FNN model to figure out the best weight and bias values for MLP as the seeds dataset has 13 features and 3 categories. The findings of the evaluation of the seeds dataset are reported in Table 8. As can be seen, in terms of MSE and accuracy, BHACRW, BHA, and SOS obtained the best results. While their evaluation results have been similar and competitive, BHA has shown consistent outcomes as the standard deviation is the lowest. The box plot shows that SOS, BHACRW, and BHA have more compact boxes, but SOS has the lowest error in terms of MSE, and BHACRW has the highest accuracy. Compared to most of the MHAs'-based optimization algorithms, the MSE error of BHA and BHACRW is less than 0.03 before 100 iterations (Fig. 8a), indicating the faster convergence rate of these algorithms.

**5.2.2.6 Experiment 6: Statlog (Heart) Dataset** The Statlog dataset consists of thirteen features and two categories, so we used the 13-27-2 FNN model to determine the best

**Table 11** Average rankings of accuracy values among 10 algorithms on seven classification datasets using Friedman test

BHACRW-MLP	1.714286
MVO-MLP	2.142857
BHA-MLP	3.14285
SOS-MLP	4.214286
PSO-MLP	5.571429
GSA-MLP	6.142857
BBO-MLP	7
WOA-MLP	7.214286
CS-MLP	8.142857
GA-MLP	9.714286

**Table 12** Post-hoc Holm test using Stacking as a control method

Comparison	Adjusted <i>P</i> -values	Result
BHACRW-MLP vs MVO-MLP	0.7984509	$H_0$ is not rejected
BHACRW-MLP vs BHA-MLP	0.4658680165	$H_0$ is not rejected
BHACRW-MLP vs SOS-MLP	0.20226031	$H_0$ is not rejected
BHACRW-MLP vs PSO-MLP	0.0447693	$H_0$ is rejected
BHACRW-MLP vs GSA-MLP	0.02308965	$H_0$ is rejected
BHACRW-MLP vs BBO-MLP	0.006117633	$H_0$ is rejected
BHACRW-MLP vs WOA-MLP	0.005070	$H_0$ is rejected
BHACRW-MLP vs CS-MLP	0.0008004	$H_0$ is rejected
BHACRW-MLP vs GA-MLP	0.0000345	$H_0$ is rejected

weight and bias for MLP. The results of the assessment for the Statlog (heart) dataset are shown in Table 9. The BHACRW shows the highest average accuracy ratio and the second-lowest MSE ratio. Although the MVO algorithm shows the lowest average MSE compared to BHACRW, it has low stability by considering the compactness of the box plot (Fig. 9b) and the standard deviation of MSE. The lowest error value ( $<0.08$ ) can be achieved by proposed BHACRW and BHA over the course of 200 iterations, which shows the faster convergence rate of our proposed algorithms compared to other algorithms.

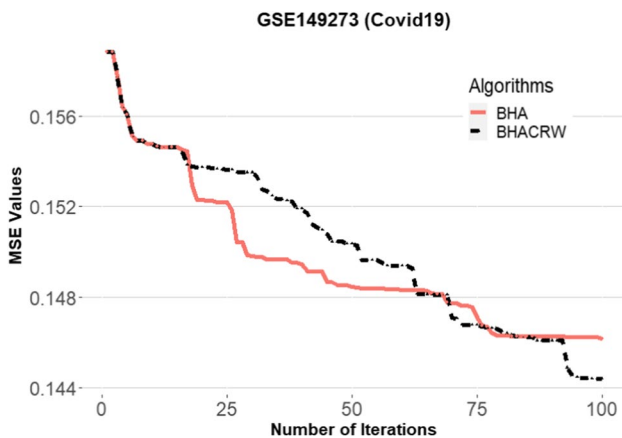
**5.2.2.7 Experiment 7: Balance Scale dataset** The balance scale dataset is composed of four features and three categories, so the 4-9-3 FNN model is used to specify the best weight and bias for MLP. The results of identifying equilibrium scale tips for the balance scale dataset are shown in Table 10. The BHACRW has the lowest MSE and standard deviation, and it is also very competitive in terms of classification accuracy. Although the MVO algorithm displays slightly better results in terms of accuracy compared to BHACRW, it has a high MSE variability and thus indicates low stability. Figure 10a also demonstrates, in terms of convergence speed, that the BHA and BHACRW-based trainers can show a faster convergence rate compared to the other algorithms since the MSE value at the 100th iteration is less than 0.05, which is lower than the best value of all algorithms. BHACRW and BHA will maintain a good balance between exploration and exploitation and show stable performance because it has a more compact box plot.

In summary, the high average accuracy and low MSE and the standard deviation obtained by both BHA and BHACRW trainers provide strong evidence that the proposed methods will consistently hamper premature convergence toward



**Table 13** Mean classification accuracy of proposed MLP approaches and other algorithms for GSE149273 (COVID-19) dataset

Dataset	Algorithms								
	KNN	NB	SVM	BP-MLP	BNN	C4.5	RF	BHA-MLP	BHACRW-MLP
COVID-19-GSE149273	53.54 ± 6.68	67.09 ± 3.53	59.99 ± 5.39	64.51 ± 10.45	54.83 ± 3.53	54.19 ± 4.2	57.81 ± 12.37	70.00 ± 2.01	80.00 ± 2.06



**Fig. 11** Convergence curve of BHA-MLP and BHACRW-MLP for the COVID-19-GSE149273 dataset

local optimization and can finding the best optimal values for MLP weights and biases.

### 5.2.3 Statistical Analysis

In this section, statistical analysis has been carried out to verify the significance of the results. In particular, due to the limited number of comparative approaches, the nonparametric Friedman test was used to assign average accuracy value rankings to each of the ten algorithms on seven classification datasets, which is shown in Table 11. As shown in Table 11, the proposed BHACRW-MLP has been placed in rank 1. In most comparisons, the *p* value suggests that the null hypothesis can be rejected. This means that the performance of the proposed approach is statistically significant vs to most of the methods. The findings obtained from the post hoc Holm test are shown in Table 12.

### 5.3 Evaluation in COVID-19 Receptor ACE2 Gene Expression Dataset

A novel virus which is known as severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2 or CV2) was identified as the cause of coronavirus disease. The CV2 required ACE2 to infect the cells. ACE2 is a protein that provides the entry point for CV2 to hook into and infect human cells. In other words, ACE2 serves as a cellular gateway or a receptor

for the COVID-19-causing virus. This COVID-19 infectious illness may be more severe in patients with asthma. Rhinovirus (RV) is the respiratory virus that is responsible for the majority of asthma exacerbations in children and adults. Recently, an analysis was conducted in [76] to examine the role of RV infection in ACE2 expression of asthmatic patients. It was found that RV infections in asthmatics lead to the overexpression of ACE2, and subsequently activate cytokine pathways which are associated with severe COVID-19 disease.

The experimental data of the above study is accessible through GSE149273. The GSE149273 is a sequence read archive (SRA) dataset that was downloaded from the gene expression omnibus database (GEO) and composed of 4056 common genes and three categories (RVA, RVC, Control). After downloading the count data some preprocessing steps such as (removing lowly expressed genes, converting counts to Differentially Expressed Genes (DGEList), quality control, normalization for composition bias, finding all common genes differentially expressed in (RVA, control) and (RVC, Control), extracting ACE2 gene with 90 samples) has been done. We only consider the expression of one gene, ACE2, so the 1-3-3 FNN model is used to specify the best weight and bias for MLP. For this dataset, the population size and number of iterations were considered as 50 and 100.

The performance of different classifiers including K-nearest neighbor (KNN), Naïve Bayes (NB), support vector machine (SVM), backpropagation (BP)-MLP, Bayesian neural network (BNN), decision tree (C4.5), random forest (RF), BHA-MLP, and BHACRW-MLP for the GSE149273 (ACE2 gene) dataset are shown in Table 13. Compared to all classifiers, the BHACRW-MLP achieved the highest accuracy. The BHACRW has the lowest average MSE (0.1432 ± 0.0020) and highest classification accuracy (80.00 ± 2.06) compare to BHA (0.1443 ± 0.0016, 70.00 ± 2.01). Figure 11 also demonstrates that BHACRW-based trainer has a faster convergence rate compared to BHA.

### 5.4 Discussion

The results of the experiments over several test functions, various benchmark classification datasets, and a real COVID-19 related gene expression dataset allow us to draw some significant conclusions.

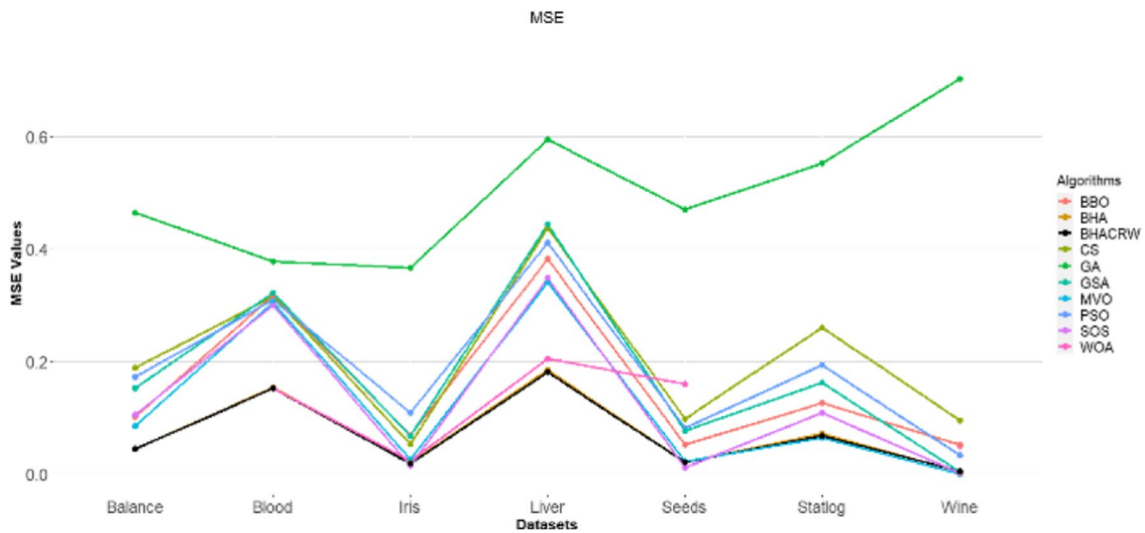


Fig. 12 MSE performance graphs for each dataset using different methods

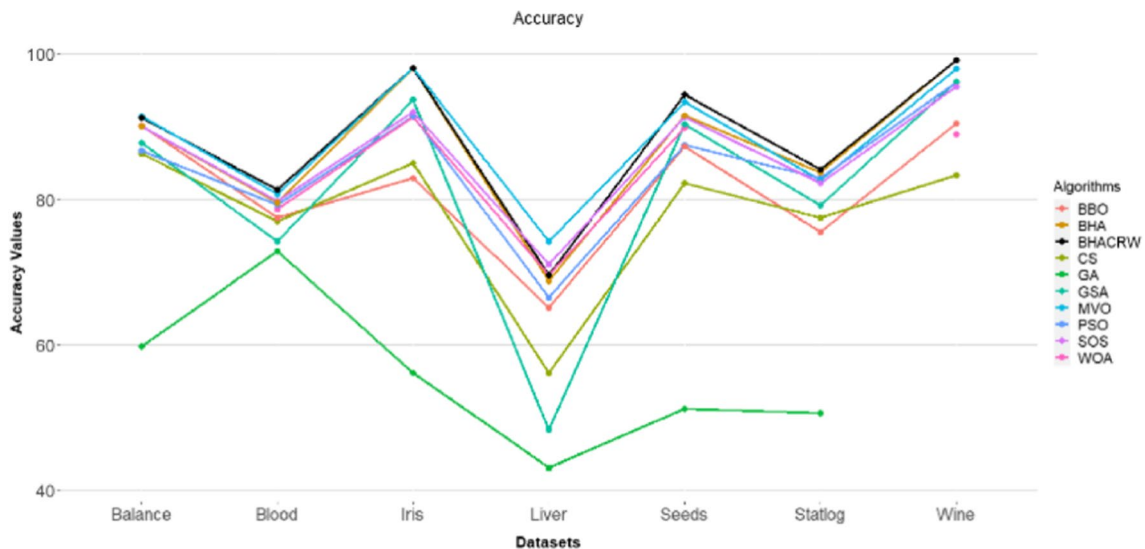


Fig. 13 Accuracy performance graphs for each dataset using different methods

First, adding complementary and Levy flight operators improves BHA's efficiency and lead this enhanced BHA to show better optimization ability for various numerical test functions. Second, in neural network training, the proposed BHACRW algorithm provides high average classification accuracy and superior local avoidance in benchmark iris, wine, blood, seeds, and Statlog datasets. For dataset wine, BHA, and BHACRW's average classification accuracy are 99.16% which means that there is only one case that cannot be properly classified in the testing dataset. It is noteworthy that while MVO has the highest accuracy in the liver disorders, and balance scale datasets, BHA exhibits comparable results. However, the accuracy demonstrated in GA is the

lowest across the algorithms inquired. The low difference between the best and worst values of the classification accuracy acquired by proposed trainers provides clear evidence that these methods are reliably capable of preventing premature convergence toward local optimization and achieving the best optimum values for MLP weights and biases especially BHACRW.

Third, based on the average of MSE, BHACRW is the most successful approach among the comparable trainers in three datasets: Balance, liver disorders, and blood. While MVO and SOS acquired the minimum average MSE regarding the Statlog (heart) and seeds datasets, BHACRW was the second best method, with very close results to MVO



and SOS. In the case of the remaining datasets, BHA and BHACRW demonstrate very competitive performance compared to MVO, SOS, and WOA approaches. In regard to box plot compactness and standard deviation, we have observed that the proposed approaches especially BHACRW perform well in most datasets, demonstrating their robustness and consistency as compared with other algorithms. Fourth, by considering the convergence curves and lowest error values, the proposed approaches have the fastest convergence speed for training all the given datasets and they can reach the lowest MSE in the middle of iterations. Figures 12 and 13 are graphical representations of the experimental results shown in Tables 4, 5, 6, 7, 8, 9, 10. From these figures, it is concluded that the proposed methods show well results as compared with different state-of-the-art methods for test benchmark classification datasets. It is worth discussing that the reason for the poor performance of metaheuristic algorithms is not only related to low efficiency and easily dropping into local optimal but also depends on the presence of many different tuning parameters. By contrast, BHA and BHACRW use only one population size parameter, thus removes the possibility of compromised performance due to inappropriate parameter tuning, and increases performance stability.

Lastly, the application of BHACRW-MLP on the real ACE2 gene expression dataset prove that the suggested approach is exceedingly effectual in solving real and complex classification problems.

In summary, while other papers have already shown that BHA is working well on complex combinatorial optimization problems and some real-world problems, this study shows that the BHA and proposed BHACRW are very capable of training FNN. The outcomes achieved are at the least, perceptive and competitive, at the most, superior to those got by most of the nature-inspired algorithms especially GA, PSO, CS, BBO, WOA, and GSA in all benchmark datasets.

## 6 Conclusion

In this study, a new modified BHA was suggested by the integration of complementary learning components and Levy flight random walk in BHA's updating phase. The main aim of this paper is to employ both BHA and suggested BHACRW as new FNN trainers to find optimal values for weight and bias parameters. The BHA's excellent performance in avoiding local optima, high convergence speed, and having no controlling parameter motivated us to employ the BHA for training MLPs. Furthermore, BHACRW was implemented in order to increase the efficiency and performance of the origin BHA. To benchmark the efficiency of the developed BHACRW approach, experiments were carried out in three steps. In the first

stage of the experiments, the BHACRW was evaluated against 4 benchmark global optimization functions and its performance was compared with the original BHA, and eight other metaheuristic algorithms: ABC, PSO, LFFA, GWO, GSA, CSA, BB-BC, and LBH. Experimental results demonstrated BHACRW's outstanding performance in most evaluation cases. In the second stage, seven UCI classification datasets with different characteristics were used to examine the performance of the suggested method for training MLP. The obtained result confirms the success of the proposed approach in FNN training, compared to other optimization techniques, including GA, PSO, CS, BBO, WOA, MVO, SOS, and GSA. The suggested BHACRW method performs better and in some datasets shows comparable results in the term of classification accuracy and convergence, due to its high convergence rate, and high local optima avoidance. In the third stage, the performance of the suggested BHACRW-MLP was tested on the classification of ACE2 gene expression that was overexpressed in asthmatic patients in order to recognize subgroups at risk for COVID-19. The result showed that the BHACRW-MLP classifier performed much more accurately than other conventional classification methods.

In conclusion, the present findings confirm the efficiency of the suggested approach in solving numerical functions and training FNNs for the classification of various datasets. However, there is still improvement needed in coverage speed and performance dependency of the proposed method on datasets.

For future work, the optimal structure of FNNs can be identified by BHA and BHACRW, including the number of nodes and hidden layers. Training other kinds of ANNs such as radial basis function (RBF) using the BHA and its variants is also recommended. Moreover, the suggested BHACRW approach may be applied to other applications, such as text document clustering.

## Compliance with Ethical Standards

**Conflict of interest** There is no conflict of interest between the authors to publish this manuscript.

## References

1. Cao, W.; Wang, X.; Ming, Z.; Gao, J.: A review on neural networks with random weights. *Neurocomputing*. **275**, 278–287 (2018). <https://doi.org/10.1016/j.neucom.2017.08.040>
2. Faris, H.; Aljarah, I.; Mirjalili, S.: Training feedforward neural networks using multi-verse optimizer for binary classification problems. *Appl. Intell.* **45**, 322–332 (2016). <https://doi.org/10.1007/s10489-016-0767-1>
3. Schweidtmann, A.M.; Mitsos, A.: Deterministic global optimization with artificial neural networks embedded. *J. Optim.*



- Theory Appl. **180**, 925–948 (2019). <https://doi.org/10.1007/s10957-018-1396-0>
4. Trujillo, M.C.R.; Alarcón, T.E.; Dalmau, O.S.; Zamudio Ojeda, A.: Segmentation of carbon nanotube images through an artificial neural network. *Soft. Comput.* **21**, 611–625 (2017). <https://doi.org/10.1007/s00500-016-2426-1>
  5. Kiranyaz, S.; Ince, T.; Abdeljaber, O.; Avci, O.; Gabbouj, M.: 1-D Convolutional neural networks for signal processing applications. In: ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings. pp. 8360–8364. Institute of Electrical and Electronics Engineers Inc. (2019)
  6. Yang, X.; Sun, H.; Sun, X.; Yan, M.; Guo, Z.; Fu, K.: Position detection and direction prediction for arbitrary-oriented ships via multitask rotation region convolutional neural network. *IEEE ACCESS.* **6**, 50839–50849 (2018). <https://doi.org/10.1109/ACCESS.2018.2869884>
  7. Bermejo, J.F.; Fernández, J.F.G.; Polo, F.O.; Márquez, A.C.: A review of the use of artificial neural network models for energy and reliability prediction. A study of the solar PV, hydraulic and wind energy sources. *Appl. Sci.* **9**, 1844 (2019). <https://doi.org/10.3390/app9091844>
  8. Pashaei, E.; Aydin, N.: Frequency difference based DNA encoding methods in human splice site recognition. In: 2nd Int. Conf. Comput. Sci. Eng. UBMK 2017. 586–591 (2017). <https://doi.org/10.1109/UBMK.2017.8093471>
  9. Aljarah, I.; Faris, H.; Mirjalili, S.: Optimizing connection weights in neural networks using the whale optimization algorithm. *Soft. Comput.* **22**, 1–15 (2018). <https://doi.org/10.1007/s00500-016-2442-1>
  10. Wu, H.; Zhou, Y.; Luo, Q.; Basset, M.A.: Training feed-forward neural networks using symbiotic organisms search algorithm. *Comput. Intell. Neurosci.* (2016). <https://doi.org/10.1155/2016/9063065>
  11. Faris, H.; Aljarah, I.; Al-Madi, N.; Mirjalili, S.: Optimizing the learning process of feedforward neural networks using lightning search algorithm. *Int. J. Artif. Intell. Tools.* **25**, 16500330 (2016). <https://doi.org/10.1142/S0218213016500330>
  12. Ojha, V.K.; Abraham, A.; Snášel, V.: Metaheuristic design of feedforward neural networks: a review of two decades of research. *Eng. Appl. Artif. Intell.* **60**, 97–116 (2017). <https://doi.org/10.1016/j.engappai.2017.01.013>
  13. Hamidzadeh, J.; Sadeghi, R.; Namaei, N.: Weighted support vector data description based on chaotic bat algorithm. *Appl. Soft Comput. J.* **60**, 540–551 (2017). <https://doi.org/10.1016/j.asoc.2017.07.038>
  14. Hamidzadeh, J.; Namaei, N.: Belief-based chaotic algorithm for support vector data description. *Soft. Comput.* **23**, 4289–4314 (2019). <https://doi.org/10.1007/s00500-018-3083-3>
  15. Heidari, A.A.; Faris, H.; Aljarah, I.; Mirjalili, S.: An efficient hybrid multilayer perceptron neural network with grasshopper optimization. *Soft. Comput.* **23**, 7941–7958 (2019). <https://doi.org/10.1007/s00500-018-3424-2>
  16. Wang, L.; Zeng, Y.; Chen, T.: Back propagation neural network with adaptive differential evolution algorithm for time series forecasting. *Expert Syst. Appl.* **42**, 855–863 (2015). <https://doi.org/10.1016/j.eswa.2014.08.018>
  17. Abualigah, L.M.Q.: Feature selection and enhanced Krill Herd algorithm for text document clustering. Springer, Cham (2019)
  18. Abualigah, L.M.; Khader, A.T.: Unsupervised text feature selection technique based on hybrid particle swarm optimization algorithm with genetic operators for the text clustering. *J. Supercomput.* **73**, 4773–4795 (2017). <https://doi.org/10.1007/s11227-017-2046-2>
  19. Abualigah, L.M.; Khader, A.T.; Hanandeh, E.S.: Hybrid clustering analysis using improved krill herd algorithm. *Appl. Intell.* **48**, 4047–4071 (2018). <https://doi.org/10.1007/s10489-018-1190-6>
  20. Abualigah, L.M.; Khader, A.T.; Hanandeh, E.S.: A new feature selection method to improve the document clustering using particle swarm optimization algorithm. *J. Comput. Sci.* **25**, 456–466 (2018). <https://doi.org/10.1016/j.jocs.2017.07.018>
  21. Abualigah, L.M.; Khader, A.T.; Hanandeh, E.S.: A combination of objective functions and hybrid Krill herd algorithm for text document clustering analysis. *Eng. Appl. Artif. Intell.* **73**, 111–125 (2018). <https://doi.org/10.1016/j.engappai.2018.05.003>
  22. Abualigah, L.M.; Khader, A.T.; Hanandeh, E.S.; Gandomi, A.H.: A novel hybridization strategy for krill herd algorithm applied to clustering techniques. *Appl. Soft Comput. J.* **60**, 423–435 (2017). <https://doi.org/10.1016/j.asoc.2017.06.059>
  23. Qasim Abualigah, L.M.; Hanandeh, S.E.: Applying genetic algorithms to information retrieval using vector space model. *Int. J. Comput. Sci. Eng. Appl.* **5**, 19–28 (2015). <https://doi.org/10.5121/ijcsea.2015.5102>
  24. Pashaei, E.; Pashaei, E.; Aydin, N.: Hybrid Krill Herd Algorithm with Particle Swarm Optimization for Image Enhancement. In: Kahraman, C., Cevik Onar, S., Oztaysi, B., Sari, I., Cebi, S., Tolga, A. (eds.) International Conference on Intelligent and Fuzzy Systems (INFUS 2020), pp. 1431–1439. Springer, Cham (2020)
  25. Pashaei, E.; Pashaei, E.: Gene selection using intelligent dynamic genetic algorithm and random forest. In: 2019 11th International Conference on Electrical and Electronics Engineering (ELECO). pp. 470–474 (2019)
  26. Abualigah, L.; Diabat, A.: A novel hybrid antlion optimization algorithm for multi-objective task scheduling problems in cloud computing environments. *Cluster Comput.* (2020). <https://doi.org/10.1007/s10586-020-03075-5>
  27. Seiffert, U.: Multiple layer perceptron training using genetic algorithms. *Eur. Symp. Artif. Neural Networks.* 159–164 (2001)
  28. Arifovic, J.; Gençay, R.: Using genetic algorithms to select architecture of a feedforward artificial neural network. *Phys. A Stat. Mech. its Appl.* **289**, 574–594 (2001). [https://doi.org/10.1016/S0378-4371\(00\)00479-9](https://doi.org/10.1016/S0378-4371(00)00479-9)
  29. Rivero, D.; Dorado, J.; Fernández-Blanco, E.; Pazos, A.: A genetic algorithm for ANN design, training and simplification. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). pp. 391–398 (2009)
  30. Ding, S.; Su, C.; Yu, J.: An optimizing BP neural network algorithm based on genetic algorithm. *Artif. Intell. Rev.* **36**, 153–162 (2011). <https://doi.org/10.1007/s10462-011-9208-z>
  31. Ahmad, F.; Mat Isa, N.A.; Hussain, Z.; Osman, M.K.; Sulaiman, S.N.: A GA-based feature selection and parameter optimization of an ANN in diagnosing breast cancer. *Pattern Anal. Appl.* **18**, 861–870 (2015). <https://doi.org/10.1007/s10044-014-0375-9>
  32. Slowik, A.; Bialko, M.: Training of artificial neural networks using differential evolution algorithm. In: 2008 Conference on Human System Interaction, HSI 2008. pp. 60–65 (2008)
  33. Zhang, L.; Li, H.; Kong, X.G.: Evolving feedforward artificial neural networks using a two-stage approach. *Neurocomputing.* **360**, 25–36 (2019). <https://doi.org/10.1016/j.neucom.2019.03.097>
  34. Rodan, A.; Faris, H.; Alqatawna, J.: Optimizing feedforward neural networks using biogeography based optimization for E-mail spam identification. *Int. J. Commun. Netw. Syst. Sci.* **09**, 19–28 (2016). <https://doi.org/10.4236/ijcns.2016.91002>
  35. Sarangi, P.P.; Sahu, A.; Panda, M.: A hybrid differential evolution and back-propagation algorithm for feedforward neural network training. *Int. J. Comput. Appl.* **84**, 1–9 (2013). <https://doi.org/10.5120/14641-2943>
  36. Bullinaria, J.A.; Alyahya, K.: Artificial bee colony training of neural networks. *Stud. Comput. Intell.* **512**, 191–201 (2014). [https://doi.org/10.1007/978-3-319-01692-4\\_15](https://doi.org/10.1007/978-3-319-01692-4_15)



37. Mirjalili, S.: How effective is the Grey Wolf optimizer in training multi-layer perceptrons. *Appl. Intell.* **43**, 150–161 (2015). <https://doi.org/10.1007/s10489-014-0645-7>
38. Amirsadri, S.; Mousavirad, S.J.; Ebrahimpour-Komleh, H.: A Levy flight-based grey wolf optimizer combined with back-propagation algorithm for neural network training. *Neural Comput. Appl.* **30**, 3707–3720 (2018). <https://doi.org/10.1007/s00521-017-2952-5>
39. Abualigah, L.; Diabat, A.: A comprehensive survey of the Grasshopper optimization algorithm: results variants and applications. *Neural Comput Appl* (2020). <https://doi.org/10.1007/s00521-020-04789-8>
40. Kowalski, P.A.; Lukasik, S.: Training neural networks with Krill Herd algorithm. *Neural Process. Lett.* **44**, 5–17 (2016). <https://doi.org/10.1007/s11063-015-9463-0>
41. Abualigah, L.: Multi-verse optimizer algorithm: a comprehensive survey of its results variants and applications. *Neural Comput. Appl.* (2020). <https://doi.org/10.1007/s00521-020-04839-1>
42. Mirjalili, S.Z.; Saremi, S.; Mirjalili, S.M.: Designing evolutionary feedforward neural networks using social spider optimization algorithm. *Neural Comput. Appl.* **26**, 1919–1928 (2015). <https://doi.org/10.1007/s00521-015-1847-6>
43. Yi, J.H.; Xu, W.H.; Chen, Y.T.: Novel back propagation optimization by cuckoo search algorithm. *Sci. World J.* (2014). <https://doi.org/10.1155/2014/878262>
44. Valian, E.; Mohanna, S.; Tavakoli, S.: Improved cuckoo search algorithm for feed forward neural network training. *Int. J. Artif. Intell. Appl.* **2**, 36–43 (2011). <https://doi.org/10.5121/ijaia.2011.2304>
45. Nawi, N.M.; Rehman, M.Z.; Khan, A.: A new bat based back-propagation (BAT-BP) algorithm. In: *Advances in Intelligent Systems and Computing*, pp. 395–404. Springer Verlag (2014)
46. Salama, K.; Abdelbar, A.M.: A Novel ant colony algorithm for building neural network topologies. *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, 8667, (2014). [https://doi.org/10.1007/978-3-319-09952-1\\_1](https://doi.org/10.1007/978-3-319-09952-1_1)
47. Huang, L.; Asteris, P.G.; Koopialipoor, M.; Armaghani, D.J.; Tahir, M.M.: Invasive weed optimization technique-based ANN to the prediction of rock tensile strength. *Appl. Sci.* **9**, 5372 (2019). <https://doi.org/10.3390/app9245372>
48. Mirjalili, S.; Mohd Hashim, S.Z.; Moradian Sardroudi, H.: Training feedforward neural networks using hybrid particle swarm optimization and gravitational search algorithm. *Appl. Math. Comput.* **218**, 11125–11137 (2012). <https://doi.org/10.1016/j.amc.2012.04.069>
49. Ghanem, W.A.H.M.; Jantan, A.: A cognitively inspired hybridization of artificial bee colony and dragonfly algorithms for training multi-layer perceptrons. *Cognit. Comput.* **10**, 1096–1134 (2018). <https://doi.org/10.1007/s12559-018-9588-3>
50. Shehab, M.; Abualigah, L.; Al Hamad, H.; Alabool, H.; Alshinwan, M.; Khasawneh, A.M.: Moth–flame optimization algorithm: variants and applications, (2019)
51. Yamany, W.; Fawzy, M.; Tharwat, A.; Hassanien, A.E.: Moth-flame optimization for training multi-layer perceptrons. In: *2015 11th International Computer Engineering Conference: Today Information Society What's Next?, ICENCO 2015*, pp. 267–272. Institute of Electrical and Electronics Engineers Inc. (2016)
52. Abualigah, L.; Shehab, M.; Alshinwan, M.; Alabool, H.: Salp swarm algorithm: a comprehensive survey. *Neural Comput. Appl.* (2020). <https://doi.org/10.1007/s00521-019-04629-4>
53. Abualigah, L.; Shehab, M.; Alshinwan, M.; Mirjalili, S.; Elaziz, M.A.: Ant lion optimizer: a comprehensive survey of its variants and applications. *Arch. Comput. Methods Eng.* **1**, 3 (2020). <https://doi.org/10.1007/s11831-020-09420-6>
54. Abualigah, L.: Group search optimizer: a nature-inspired meta-heuristic optimization algorithm with its results, variants, and applications. *Neural Comput. Appl.* (2020). <https://doi.org/10.1007/s00521-020-05107-y>
55. Darwish, A.: Bio-inspired computing: algorithms review, deep analysis, and the scope of applications. *Futur. Comput. Informatics J.* **3**, 231–246 (2018). <https://doi.org/10.1016/j.fcij.2018.06.001>
56. Hatamlou, A.: Black hole: a new heuristic optimization approach for data clustering. *Inf. Sci. (Ny)* **222**, 175–184 (2013). <https://doi.org/10.1016/j.ins.2012.08.023>
57. Yang, X.-S.: Random walks and optimization. In: *Nature-Inspired Optimization Algorithms*, pp. 45–65. Elsevier, Oxford (2014)
58. Yaghini, M.; Khoshraftar, M.M.; Fallahi, M.: A hybrid algorithm for artificial neural network training. *Eng. Appl. Artif. Intell.* **26**, 293–301 (2013). <https://doi.org/10.1016/j.engappai.2012.01.023>
59. Mohammed, S.; Mohd Faudzi, A.A.; Daniyal, H.; Muhammad, B.; Aziz, N.; Yusof, Z.; Azmi, K.; Adam, A.: Improving black hole algorithm using gravitational search, white hole operator, and local search. *Mekatronika.* **1**, 8–14 (2019). <https://doi.org/10.15282/mekatronika.v1i2.4891>
60. Xie, W.; Wang, J.S.; Tao, Y.: Improved black hole algorithm based on golden sine operator and Levy flight operator. *IEEE Access.* **7**, 161459–161486 (2019). <https://doi.org/10.1109/ACCESS.2019.2951716>
61. Abdulwahab, H.A.; Noraziah, A.; Alsewari, A.A.; Salih, S.Q.: An enhanced version of black hole algorithm via Levy flight for optimization and data clustering problems. *IEEE Access.* **7**, 142085–142096 (2019). <https://doi.org/10.1109/access.2019.2937021>
62. Yaghoobi, S.; Hemayat, S.; Mojallali, H.: Image gray-level enhancement using black hole algorithm. In: *2015 2nd International Conference on Pattern Recognition and Image Analysis, IPRIA 2015*. IEEE, Rasht, Iran (2015)
63. Farahmandian, M.; Hatamlou, A.: Solving optimization problems using black hole algorithm. *J. Adv. Comput. Sci. Technol.* **4**, 68–74 (2015). <https://doi.org/10.14419/jacst.v4i1.4094>
64. Azizpanah-Abarghooee, R.; Niknam, T.; Bavafa, F.; Zare, M.: Short-term scheduling of thermal power systems using hybrid gradient based modified teaching-learning optimizer with black hole algorithm. *Electr. Power Syst. Res.* **108**, 16–34 (2014). <https://doi.org/10.1016/j.epsr.2013.10.012>
65. Jeet, K.; Dhir, R.: Software architecture recovery using genetic black hole algorithm. *ACM SIGSOFT Softw. Eng. Notes.* **40**, 1–5 (2015). <https://doi.org/10.1145/2693208.2693230>
66. Boucekara, H.R.E.H.: Optimal design of electromagnetic devices using a black-Hole-Based optimization technique. *IEEE Trans. Magn.* **49**, 5709–5714 (2013). <https://doi.org/10.1109/TMAG.2013.2277694>
67. Boucekara, H.R.E.H.: Optimal power flow using black-hole-based optimization approach. *Appl. Soft Comput. J.* **24**, 879–888 (2014). <https://doi.org/10.1016/j.asoc.2014.08.056>
68. Yang, X.-S.: Firefly algorithm, Lévy flights and global optimization. In: *Research and Development in Intelligent Systems XXVI*, pp. 209–218. Springer London (2010)
69. Rashedi, E.; Nezamabadi-pour, H.; Saryazdi, S.: GSA: a gravitational search algorithm. *Inf. Sci. (Ny)* **179**, 2232–2248 (2009). <https://doi.org/10.1016/j.ins.2009.03.004>
70. Chu, S.-C.; Tsai, P.; Pan, J.-S.: Cat swarm optimization. Presented at the (2006)
71. Erol, O.K.; Eksin, I.: A new optimization method: big bang-big crunch. *Adv. Eng. Softw.* **37**, 106–111 (2006). <https://doi.org/10.1016/j.advengsoft.2005.04.005>
72. Tarkhaneh, O.; Shen, H.: Training of feedforward neural networks for data classification using hybrid particle swarm optimization, Mantegna Lévy flight and neighborhood search. *Heliyon.* **5**, e01275 (2019). <https://doi.org/10.1016/j.heliyon.2019.e01275>



73. Soto, R.; Crawford, B.; Olivares, R.; Taramasco, C.; Figueroa, I.; Gómez, Á.; Castro, C.; Paredes, F.: Adaptive black hole algorithm for solving the set covering problem. *Math. Probl. Eng.* (2018). <https://doi.org/10.1155/2018/2183214>
74. Harb, A.; Kassem, H.; Ghorayeb, K.: Black hole particle swarm optimization for well placement optimization. *Comput. Geosci.* (2019). <https://doi.org/10.1007/s10596-019-09887-8>
75. Mirjalili, S.; Mirjalili, S.M.; Lewis, A.: Let a biogeography-based optimizer train your Multi-Layer Perceptron. *Inf. Sci. (Ny)* **269**, 188–209 (2014). <https://doi.org/10.1016/j.ins.2014.01.038>
76. Chang, E.H.; Willis, A.L.; Romanoski, C.E.; Cusanovich, D.A.; Pouladi, N.; Li, J.; Lussier, Y.A.; Martinez, F.D.: RV infections in asthmatics increase ACE2 expression and cytokine pathways implicated in COVID-19. *Am. J. Respir. Crit. Care Med.* (2020). <https://doi.org/10.1164/rccm.202004-1343LE>

