Cluster Computing (2023) 26:1821–1843
https://doi.org/10.1007/s10586-022-03708-x

Boosting the training of neural networks through hybrid metaheuristics

Mohammed Azmi Al-Betar[1,2] · Mohammed A. Awadallah[3,4,12] · Iyad Abu Doush[5,6] · Osama Ahmad Alomari[7] · Ammar Kamal Abasi[8] · Sharif Naser Makhadmeh[1] · Zaid Abdi Alkareem Alyasseri[9,10,11]

Received: 19 October 2021 / Revised: 29 May 2022 / Accepted: 7 August 2022 / Published online: 26 August 2022

Abstract
In this paper, the learning process of multilayer perceptron (MLP) neural network is boosted using hybrid metaheuristic optimization algorithms. Normally, the learning process in MLP requires suitable settings of its weight and bias parameters. In the original version of MLP, the gradient descent algorithm is used as a learner in MLP which suffers from two chronic problems: local minima and slow convergence. In this paper, six versions of memetic algorithms (MAs) are proposed to replace gradient descent learning mechanism of MLP where adaptive β-hill climbing (AβHC) as a local search algorithm is hybridized with six population-based metaheuristics which are hybrid flower pollination algorithm, hybrid salp swarm algorithm, hybrid crow search algorithm, hybrid grey wolf optimization (HGWO), hybrid particle swarm optimization, and hybrid JAYA algorithm. This is to show the effect of the proposed MA versions on the performance of MLP. To evaluate the proposed MA versions for MLP, 15 classification benchmark problems with different size and complexity are used. The AβHC algorithm is invoked in the improvement loop of any MA version with a probability of B_r parameter, which is investigated to monitor its effect on the behavior of the proposed MA versions. The B_r setting which obtains the most promising results is then used to set the hybrid MA. The results show that the proposed MA versions excel the original algorithms. Moreover, HGWO outperforms all other MA versions in almost all the datasets. In a nutshell, MAs are a good choice for training MLP to produce results with high accuracy.

Keywords Hybrid metaheuristics · Multilayer perceptron neural network · β-hill climbing · Optimization

1 Introduction

The genuine innovation of simulating the neurons in the human brain is represented in the intelligent mathematical model of artificial neural networks (ANN). The connection between neurons empowers the biological system to manage the body's behavior through signal communication [1]. The artificial intelligence research community utilizes the neurons' biological behavior as an essential part of solving machine learning problems such as classifications, clustering, feature extractions, regressions, and predictions. The base ANN has been categorized into different types due to their learning process such as convolutional neural network [2], feedforward neural network (FNN) [3], spiking neural networks (SNN) [4], recurrent neural network (RNN) [5], and radial basis function (RBF) network [6]. Learning is the capability of gaining knowledge from experience and training. There are two types of learning: supervised where the ANN work is assisted by outdoor feedback and unsupervised where ANNs depend totally on indoor feedback.

The learning process in each type of ANN is different. The following is a brief explanation of this process in the most common ANN structures [7, 8]. The FNN obtains the information using the input layer. Then, the hidden layer processes the inputs and produces the desired result in the output layer. Each layer attempt to learn certain weights to map the input into the output. This type of network does not have feedback connections to be returned to the model. The modification of FNN to have a loop on each hidden layer would develop the RNN. This loop restriction guarantees the capturing of sequential information in the input data. Additionally, the RNN shares the parameters across different time

Extended author information available on the last page of the article

steps. Lastly, the CNN utilizes filters (i.e., kernels) to obtain from the input the relevant features by applying the convolution operation. The CNN automatically learns the filters which assist in wresting relevant features from the input data. As a result, a feature map is produced.

One of the most popular FNN is multi-layer perceptron (MLP). It is widely used to solve several types of optimization problems such as feature selection [9], classification [10, 11], predictions [12, 13], regressions [14], etc. The success of the learning process in MLP depends on the initial value of its parameters (i.e., weights and biases). These two parameters are optimized during the learning process using gradient decent optimization algorithm. However, this type of optimizer has two main shortcomings [15]: slow convergence and local optima. Therefore, several metaheuristic-based algorithms are adapted for MLP such as artificial bee colony [16], cuckoo search algorithm [17], gravitational search algorithm [18], grey wolf optimizer [19, 20], butterfly optimization algorithm [21], fish swarm algorithm [22], salp swarm algorithm [23, 24], glowworm swarm optimisation [25], genetic algorithm [26], grasshopper optimization algorithm [27, 28], dragonfly algorithm [16], krill herd algorithm [29], monarch butterfly optimization [30], social spider optimization algorithm [31], ant colony optimization [32], bat algorithm [33], biogeography based optimization [34], lightning search algorithm [35], ant lion optimizer [36], organisms search algorithm [37], and whale optimization algorithm [15].

Generally speaking, metaheuristic-based algorithms are general optimization frameworks which can be used to minimize/maximize an optimization problem using smart operators with the ability to navigate several niches in the search space [38]. The behavior of its smart operators is controlled by tuning or adapting parameters that affect the balance between exploration and exploitation processes. It is conventionally agreed that the metaheuristic-based algorithms can be classified into either population-based or local search algorithms due to their number of initial solution(s) [39]. The local search algorithms begin with a single solution. Iteratively, that solution is locally improved using the neighboring move strategy until the local minima is obtained. Local search algorithms are very efficient in navigating the niche of the initial solution, but it cannot explore several niches at the same time.

Oppositely, the population-based algorithms, such as evolutionary algorithms (EAs) and swarm intelligence, are initiated with several solutions stored in the population. Iteratively, these solutions are recombined and mutated until the population is maturely or prematurely converged. These types of algorithms can explore several search space niches because of their operators, but they cannot dig deeply in each converged niche as they can stuck in local optima [38]. Therefore, a new type of algorithm has recently proved its efficiency in tackling the optimization problem which is hybrid metaheuristic (or memetic algorithm (MA)) [40, 41].

Memetic Algorithm (MA) is the recent growing trend in evolutionary computation fields. In MA, the local search algorithm is hybridized as an operator in the iterative loop of the population-based algorithm. The main motivation behind this hybridization is to complement the advantages of both exploration and exploitation capabilities. The local search operator speeds up the convergence by concentrating on good solutions than mutating the solutions [42, 43]. The behaviour of the population-based algorithm in MA is inspired by Darwinian principles of natural selection as a gene notation. The meme notation in MA is introduced by Dawkins [44] which reflects the cultural behavior of the local search algorithm. Gene and meme units play a vital role in biological and cultural transmission. It is noted that the majority of the previous work in the training process of MLP relies on using the original version of the metaheuristic-based algorithm. Since the MLP performance substantially depends on the optimal value of its initial parameters which is refined in the later training stages. The original metaheuristic algorithms can be boosted by MA concepts.

This paper studies the effect of different MA versions as an efficient trainer to the MLP. The MLP is chosen as an efficient version of NN where this model applies feedforward to the NN. The structure of MLP is simple, easy to design, and its speed allows its fast evaluation using the proposed optimizer. Furthermore, this framework can be generalized to other NN structures. In this case, the MA algorithm enables finding the optimal configuration of the MLP to boost its performance by producing more accurate results. The following are the main contributions of the current work:

- Six MA versions were introduced which are Hybrid Flower Pollination Algorithm (HFPA), Hybrid Salp Swarm Algorithm (HSSA), Hybrid Crow Search Algorithm (HCSA), Hybrid Grey Wolf Optimizer (HGWO), Hybrid Particle Swarm Optimization (HPSO), and Hybrid JAYA algorithm (HJAYA).
- The original versions of the six population-based algorithms are used as a gene unit to refine the global search while the Adaptive β-hill climbing (AβHC) [45] is used as a meme unit for the local search process.
- The AβHC is invoked in the iterative loop of any MA version with B_r probability where $B_r \in [0, 1]$. The higher value of B_r probability leads to higher usage of local improvement (AβHC).

– The mean square error (MSE) is utilized to measure the results of the proposed MA versions where the MLP is used for classification purposes.

For evaluation purposes, fifteen classification datasets with different levels of complexity are used. Initially, the effect of B_r on the behavior of the proposed MA versions is studied. The comparative analysis is also conducted between the results of MA versions as well as those of their base algorithms. Interestingly for any MA version, the results produced are mostly better than those produced by the original algorithms. Furthermore, HGWO was able to produce the best overall results for almost all datasets used in comparison with the other proposed MA versions. In conclusion, the MA is a better choice to train MLP to produce results with high accuracy.

The remaining sections of this paper are arranged in the following order: the background about MLP and AβHC is given in Sect. 2. The proposed MA versions are thoroughly discussed in Sect. 3. The dataset description and the results discussions are given in Sect. 4. Finally, the conclusion and possible future expansions are described in Sect. 5.

2 Research background

The background section includes the basic knowledge about FNNs with a special concern for its variant used in this research (i.e., Multilayer perceptron (MLP)). Thereafter, the fundamentals of the adaptive β-hill climbing optimizer which is the local search algorithm used in the proposed MA versions are discussed.

2.1 Feedforward neural networks

Feedforward neural networks (FNN) are supervised learning algorithms that simulate the fact that the human brain is organized into a form of network architecture in which neurons are located in the layers, where there is a direct connection between each layer and the next one. In FNN structure design, the neurons are interconnected and grouped in three layers. The first layer, namely the input layer, comprises a set of neurons where it is equal to the number of input features in training data. The middle layer is known hidden layer and the last layer is known as the output layer that maps the predicted class labels in a form of output neurons in the FNN network [46].

Multilayer perceptron (MLP) is a variant of the FNN model, where its network architecture is organized by interconnected neurons distributed in the layers. The information is transferred through these connections in one way. An example of MLP network structure embeds of the single hidden layer is shown in Fig. 1. MLPs mathematical model is constructed from three parameters: input data, weights, and biases. These parameters are fed into three equation steps to compute the output of MLPs as follows:

(1) Initially, for each input in MLPs network structure, a weighted sum score is linked with them, where it is computed by using Eq. 1.

$$S_j = \sum_{i=1}^{n} (w_{ij}.X_i) - \beta_j, j = 1, 2, ..., h \tag{1}$$

where n refers to the number of input nodes in the network, w_{ij} refers to the weight vector that connects the input node i with hidden node j, X_i is the ith input, and β_j is the bias of the jth hidden node.

(2) In this step, the weighted vector output is fed into the activation function, called Sigmoid, and then the new output vector is transferred to the next layer as follows.

$$S_j = Sigmoid(S_j) = \frac{1}{1 + exp(-S_j)}, j = 1, 2, ...h \tag{2}$$

where S_j is the weighted sum score is the node j while h is the number of neurons in the hidden layer.

(3) Eventually, the final output of the network is calculated as follows:

$$\hat{y}_k = \sum_{i=1}^{m} w_{kj} f_i + b_k \tag{3}$$

where w_{jk} refers to the connection weight from hidden node j to the output node k, and b_k refers to the bias of the output node k.

It should be noted that the most significant factors in the estimation of the final output in MLPs are weight and bias vectors as demonstrated in Eqs. 1 and 3. Finding the proper values of weights and biases vectors play a vital role in generating a robust and accurate MLPs model [20].

2.2 Adaptive β-hill climbing optimizer

The β-hill climbing (βHC) optimizer is a recent local search-based algorithm proposed by Al-Betar [47]. Since its establishments, βHC has been successfully tailored to tackle several optimization algorithms such as feature selection [48, 49], classification problems [50], economic dispatch problems [51], examination timetabling problems [52], multiple-reservoir scheduling [53], generating substitution-Boxes [54], and sudoku game [55]. In specific, the βHC is hybridized with other population-based algorithms to improve their exploitation power. For example, the βHC is hybridized with bat algorithm for gene selection in [56],

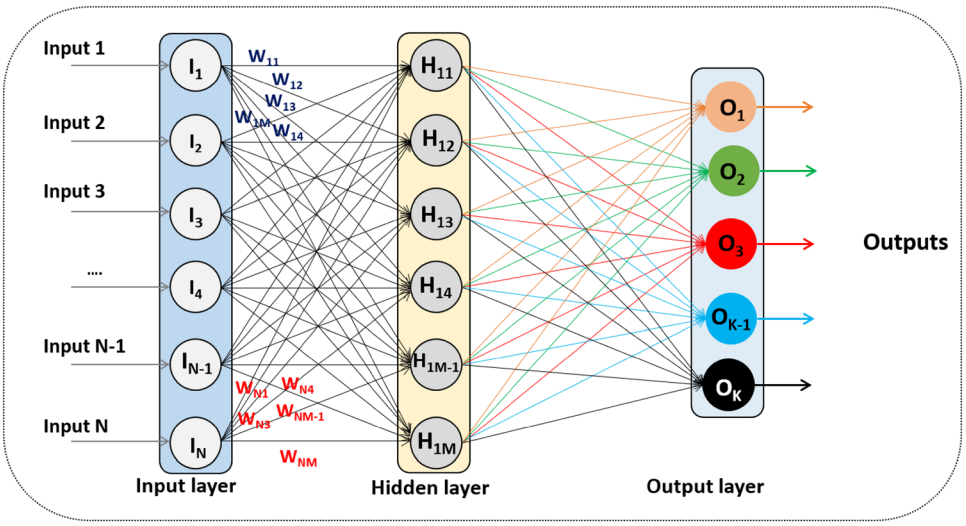


Fig. 1 Network structure of MLPs with only single one hidden layer

while it hybridized with cuckoo search for test optimization function in [57], and hybridized with polynomial harmony search algorithm [58]. Similarly, the βHC is hybridized with salp swarm optimization for text documents clustering problem [59]. The βHC is integrated within the flower pollination algorithm for person identification using EEG channel selection [60]. The βHC is hybridized with the artificial bee colony algorithm for test optimization functions in [61]. In another study, the adaptive βHC is combined within the grey wolf optimization for solving the non-convex economic load dispatch problem in [62]. The adaptive βHC is integrated within slime mould algorithm for solving numerical optimization problems [63], Finally, adaptive βHC is hybridized with salp swarm optimization for stock market prediction [64].

The main reasons behind using βHC optimizer are their features in which it is simple in concepts, easy-to-use, powerful in local refinement, speedy in convergence, and powerfully avoiding local optima. βHC optimizer has three main operators to improve the solution $\mathcal{N}$-operator, β-operator, and $\mathcal{S}$-operator. The first operator is responsible for neighbouring search controlled by $\mathcal{N}$ parameter where $\mathcal{N} \in [0, 1]$. The second operator is responsible for the random search which is controlled by β parameter where $\beta \in [0, 1]$ (i.e., similar to a uniform mutation in Genetic Algorithm). The last operator is the greedy selection process to replace the current solution with the new one, if better. Recently, An adaptive version of βHC optimizer is proposed to yield a parameter-free βHC [45]. In Adaptive βHC (i.e., AβHC), the two parameters of βHC optimizer (i.e., $\mathcal{N}$ and β) are updated during the search. Algorithm 1 shows the pseudo-code of AβHC.

As illustrated in the Algorithm 1 in the initialization step of AβHC, the values of $\beta_{min} = 0.001$, $\beta_{max} = 0.6$, and $K = 20$ parameters are assigned as suggested in [45] to determine how the parameters will be updated during the search. The initial solution $\boldsymbol{x} = (x_1, x_2, \ldots, x_d)$ is generated randomly such as $x_i = lb_i + (ub_i - lb_i) \times U(0, 1)$, $\forall i = 1, 2, \ldots, d$ where ub_i and lb_i are the upper and lower bound of variable x_i, respectively and $U(0, 1)$ generates a random uniform value between 0 and 1. The ultimate objective is to minimize $f(\boldsymbol{x})$ such that $\boldsymbol{x} \in \boldsymbol{X}$ where $X \in [\boldsymbol{lb}, \boldsymbol{ub}]$. The three operators of improvement step in AβHC can be thoroughly discussed as follows:

$\mathcal{N}$-operator: The values of the current solutions $\boldsymbol{x}$ is modified by moving to its neighbouring solution $\boldsymbol{x}'$ as follows: $x_i' = x_i \pm U(0, 1) \times \mathcal{N}(t)$ where $\mathcal{N}(t)$ is the distance bandwidth in iteration t. In AβHC, $\mathcal{N}(t)$ is adapted during searching as provided in Eq. (4).

$$\mathcal{N}(t) = 1 - C(t) \tag{4}$$

Note that $C(t)$ is calculated at time t as shown in Eq. (5).

$$C(t) = \frac{t^{\frac{1}{K}}}{\text{Max\_t}^{\frac{1}{K}}} \tag{5}$$

where K has a constant value reduced gradually to a value close to 0. Max_t is the maximum iterations.

β-operator: The value of the decision variable x_i is randomly regenerated such as $x_i'' = lb_i + (ub_i - lb_i) \times U(0, 1)$ with a probability of β where $\beta \in [0, 1]$. In AβHC, this parameter is automatically adapted during the improvement step as formulated in Eq. (6).

$$\beta(t) = \beta_{min} + t \times \frac{\beta_{max} - \beta_{min}}{\text{Max\_t}} \tag{6}$$

where $\beta(t)$ represents the β value at iteration t. β_{min}, β_{max} are the minimum and maximum value of β set in advance, respectively.

$\mathcal{S}$-operator: The selection operator works in a greedy strategy where the neighbouring solution $\boldsymbol{x}''$ replaces the current one $\boldsymbol{x}$, if fitter (i.e., $(f(\boldsymbol{x}'') \leq f(\boldsymbol{x}))$).

Algorithm 1 Adaptive β–hill climbing pseudo-code

Initialize β_{min}, β_{max}, and K
$x_i = lb_i + (ub_i - lb_i) \times U(0,1), \forall i = 1, 2, \ldots, d$ {The initial solution $\boldsymbol{x}$}
Calculate($f(\boldsymbol{x})$)
$t = 0$
while ($t \leq$ Max_t) **do**
  $\boldsymbol{x}' = \boldsymbol{x}$
  $C(t) = \frac{\frac{1}{t+K}}{\frac{1}{\text{Max\_t}+K}}$
  $\mathcal{N}(t) = 1 - C(t)$ {Adaptive $\mathcal{N}$}
  $RndIndex \in (1, d)$
  $x'_{RndIndex} = x'_{RndIndex} \pm \mathcal{N}(t)$
  $\boldsymbol{x}'' = \boldsymbol{x}'$
  $\beta(t) = \beta_{min} + t \times \frac{\beta_{max} - \beta_{min}}{\text{Max\_t}}$ {Adaptive β}
  for $i = 1, \cdots, d$ **do**
    if $(U(0,1) \leq \beta(t)$ **then**
      $x''_i = lb_i + (ub_i - lb_i) \times U(0,1)$
    end if
  end for
  if $(f(\boldsymbol{x}'') \leq f(\boldsymbol{x})$ **then**
    $\boldsymbol{x} = \boldsymbol{x}'$
    $f(\boldsymbol{x}) = f(\boldsymbol{x}'')$
  end if
  $t = t + 1$
end while

3 MA versions for MLP training

As aforementioned, the MLP has two main dilemmas affect its performance: slow convergence and local optima. The is occurred due to the stochastic configuration of its parameters (i.e., weight and biases). In order to find the proper values of its initial parameters, the MA versions for MLP are proposed in this section.

In general, the population-based algorithms have a sequence of commonly known steps: the first step is the initialization step where the optimization problem shall be defined in a Genotype form. The objective function to evaluate the solution, as well as the value range for each variable, shall be defined. Furthermore, the population initialization where a set of solutions are randomly generated, and their evaluation functions are computed. The second step is the improvement step where iteratively, the population undergoes refinements using different operators controlled by tuned or adapted parameters until a stopping criterion is met. The final step is collecting and filtering the final results where the genotype representation of the best-obtained solution is transformed to phenotype representation. These steps, as visualized in Fig. 2, are thoroughly discussed for each MA version proposed with the application as a trainer for MLP as follows:

3.1 Initialization step

The MLP as a version of FNN is mostly used to tackle the classification problems. Each classification solution is normally represented as a vector of weights and biases such as $\boldsymbol{x} = (x_1, x_2, \ldots, x_d)$ where the variables in the classification solution $\boldsymbol{x}$ is divided into two consecutive groups: weights $\boldsymbol{w} = (w_1, \ldots, w_n)$ of n weights and biases $\boldsymbol{b} = (b_1, \ldots, b_m)$ of m biases in which $d = n + m$. There is no direct formula for calculating the number of hidden layers in MLP [65]. As a result, the MA-MLP applied a fixed structure MLP [15, 66] based on the classification data, the value of n and m can be calculated as in Eq. (7):

$$\begin{aligned} h &= 2 \times K + 1 \\ n &= K \times h + h \times o \\ m &= h + o \end{aligned} \tag{7}$$

Where the number of neurons h in MLP should be calculated based on the number of features K in the classification data. The number of output o resulting from MLP should be determined.

The objective function used to evaluate the classification solution is the mean square error (MSE) function. The MSE computes the distance between the actual classification value (i.e., y) and the predicted classification value (i.e., $\hat{y}$) found by MA-based MLP algorithm of all training instances. MA-based MLP will iteratively minimize the MSE based on the optimized weight-biases vector represented by the classification solution. In Eq. (8).

$$MSE = \sum_{t=1}^{T} \frac{\sum_{i=1}^{o} (y_i^t - \hat{y}_i^t)^2}{T} \tag{8}$$

where T refers to the total number of used instances in the

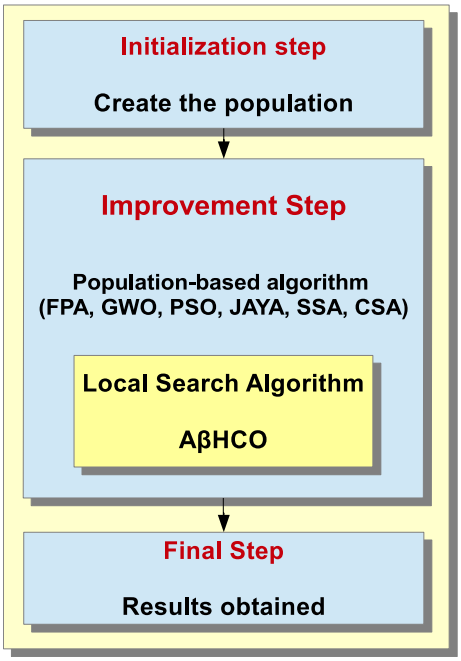


Fig. 2 The Steps of the proposed MA versions

training dataset. The small t is the training instance. The ultimate objective function is formulated in Eq. (9).

$$\min f(\boldsymbol{x}) = MSE \tag{9}$$

3.1.1 Initialize parameters of MA versions

In the initial step of the population-based algorithm, there are common algorithmic parameters which are population size (N) and the maximum number of iterations M_t. The control parameters are normally different from one algorithm to another. The population-based algorithm used are FPA, SSA, CSA, GWO, PSO, and JAYA. Note that most of the control parameters of the population-based algorithms used in the proposed MA versions are adaptively updated during runs such as FPA, SSA, GWO, and JAYA where no control parameter shall be initialized. There are two population-based algorithms used in MA versions that should tune their parameters such as Awareness probability (AP) and Flight length (fl) in the CSA and intra-weight (w), acceleration coefficients (c_1 and c_2) in PSO. For all MA versions, there is a control parameter called B_r where $B_r \in [0, 1]$ determines the volume of invoking the AβHC in the improvement loop of the population-based algorithm used.

3.1.2 Initialize the population

The population-based algorithms are normally initiated with a population of random solutions. These solutions are stored in a matrix of size $d \times N$ as shown in Eq. (10). In MAP matrix, Each row represents a solution while each column represents a decision variable. The value range of each decision variable $x_i \in [lb_i, ub_i]$ where lb_i is the lower bound and ub_i is the upper bounds of variable x_i (i.e., MLP weights and biases acceptable range).

$$\boldsymbol{MAP} = \begin{bmatrix} x_1^1 & x_2^1 & \cdots & x_d^1 \\ x_1^2 & x_2^2 & \cdots & x_d^2 \\ \vdots & \vdots & \cdots & \vdots \\ x_1^N & x_2^N & \cdots & x_d^N \end{bmatrix}. \tag{10}$$

The objective function values of the whole populations are calculated using Eq. (9). Among the solutions stored in MAP, the best solution (i.e., $\boldsymbol{x}^b$), the worst solution (i.e., $\boldsymbol{x}^w$) are usually defined and updated during the improvement step. In PSO, the local best (i.e., $\boldsymbol{x}^{lb}$) and global best (i.e., $\boldsymbol{x}^{gb}$) are distinguished in which $\boldsymbol{x}^{lb}$ is defined every iteration while $\boldsymbol{x}^{gb}$ is the overall best solution found in the whole iterations. In GWO, the three best solutions are defined where the best solution is $\boldsymbol{x}^{\alpha}$, second-bast is $\boldsymbol{x}^{\beta}$, and third-best is $\boldsymbol{x}^{\delta}$.

3.2 Improvement step

In general, each population-based algorithm has an improvement loop based on intelligence operators guided by control parameters. This is an iterative process where the current population is updated. The main difference between the population-based algorithms is the behaviour of the operators during their exploration and exploitation processes. Therefore, the operators in the improvement loop of population-based algorithms used in the proposed MA versions and the AβHC as a new operator are discussed in the following subsection.

3.2.1 The improvement step in hybrid flower pollination algorithm (HFPA)

Yang [67] proposed a new algorithm inspired by the plants blooming behavior which is called flower pollination algorithm (FPA). In FPA, there are three essential operators to update each solution. In the proposed HFPA, each generated solution is passed to AβHC algorithm as a new local refinement operator with a probability of B_r. The flowchart of HFPA is given in Fig. 3. The operators of HFPA are discussed as follows:

Operator#1: Global Search (biotic)

In each iteration (say t), any solution $\boldsymbol{x}^i(t)$ in the population will be updated using biotic operator with probability of p as formulated in Eq. (11)

$$\boldsymbol{x}^i(t+1) = \boldsymbol{x}^i(t) + L(\boldsymbol{x}^{gb} - \boldsymbol{x}^i(t)) \tag{11}$$

Note that the $\boldsymbol{x}^i(t)$ represents the solution vector $\boldsymbol{x}^i$ at iteration t. The best solution in the current iteration is x^{gb}. The L parameter identify the step size. In order to mimic the step size, the Levy distribution is modelled in Eq. (12).

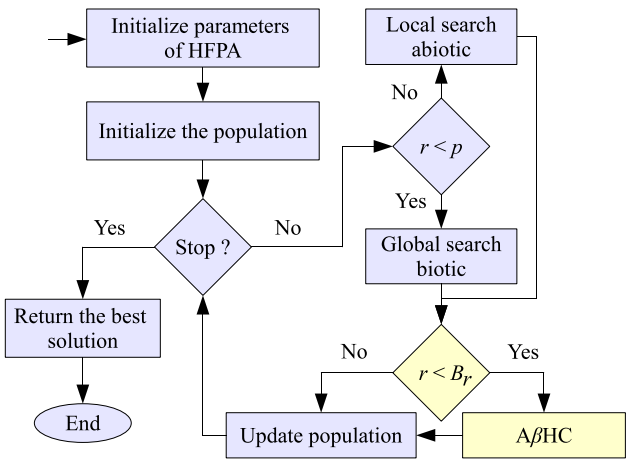


Fig. 3 Flowchart of the proposed HFPA

$$L \sim \frac{\lambda\Gamma(\lambda)sin(\pi\lambda/2)}{\pi}\frac{1}{s^{1+\lambda}}, (s >> s_0 > 0) \quad (12)$$

where L >0, and $\Gamma(\lambda)$ is the standard gamma function, and $s > 0$ is the big steps that help in reaching the desired distribution. Note that the commonly used λ value is 1.5 [68].

Operator#2: Local Search of FPA (abiotic)

With probability range of $1 - p$, each solution $\boldsymbol{x}^i(t)$ is updated using abiotic operator as shown in Eq. (13).

$$\boldsymbol{x}^i(t+1) = \boldsymbol{x}^i(t) + \epsilon(\boldsymbol{x}^j(t) - \boldsymbol{x}^k(t)) \quad (13)$$

where $\boldsymbol{x}^j(t)$ and $\boldsymbol{x}^k(t)$ are two selected randomly neighbouring solutions to the current solution. The main motivation behind abiotic operator is to emulate the restricted neighborhood of the flower constancy. ϵ is randomly selected from the range [0,1].

Operator#3: AβHC local search

To boost the exploitation capability of HFPA, each solution generated by the original operator of FPA is passed with a probability of B_r. The discussion of AβHC is mentioned in Sect. 2.

Operator#4: Update population

Each updated solution $\boldsymbol{x}^i(t+1)$ is evaluated to confirm if it is fitter than its current solution $\boldsymbol{x}^i(t)$ to be replaced such as $\boldsymbol{x}^i(t) = \boldsymbol{x}^i(t+1)$ where $f(\boldsymbol{x}^i(t+1)) \leq f(\boldsymbol{x}^i(t))$.

3.2.2 The improvement step in hybrid salp swarm algorithm (HSSA)

The salp swarm algorithm (SSA) is a population-based technique introduced to simulate the salp swarming as a unique fish behaviour in which the salps move in the sea in form of a transparent barrel-shaped body [69]. In the original version of SSA, there are three essential operators to reconstruct each solution in the population (i.e., leader rule, follower rules, and Update population). In the proposed HSSA, each reconstructed solution is passed to AβHC algorithm as a new local refinement operator. The flowchart of HSSA is given in Fig. 4. These four operators are discussed as follows:

Operator#1: Leader rule

The global best solution (i.e., $\boldsymbol{x}^{gb}$) is used to generate the decision variables of the group leader solution (i.e., $\boldsymbol{x}^1$) in which each decision variable (x_i^1) is generated as shown in Eq. (14)

$$x_i^1 \leftarrow \begin{cases} x_i^{gb} + r_1((ub_i - lb_i)r_2 + lb_i) & r_3 \geq 0.5 \\ x_i^{gb} - r_1((ub_i - lb_i)r_2 + lb_i) & r_3 < 0.5 \end{cases} \quad (14)$$

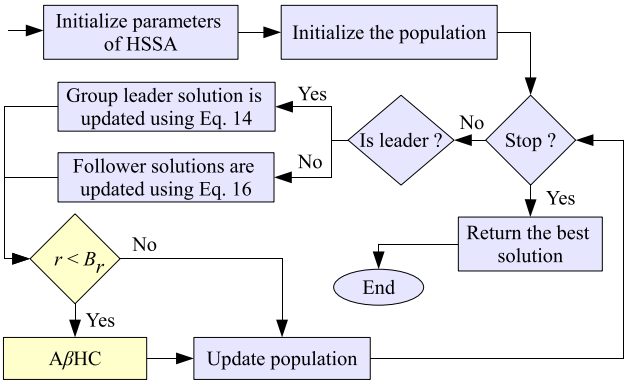


Fig. 4 Flowchart of the proposed HSSA

where x_i^{gb} is the decision variable i value in the global best solution. r_2 and r_3 values are two random values within the range [0,1]. The value of r_1 is calculated using Eq. (15) which used to find the suitable trade-off between exploration and exploitation.

$$r_1 = 2e^{-\left(\frac{4t}{M_t}\right)^2} \quad (15)$$

Recall that in Eq. (15), t is the current iteration while M_t is the maximum number of iteration.

Operator#2: Follower rules

In this operator, the other solutions in the population are called followers. The decision variable values of the follower solutions are updated based on updated Newton's law of motion as shown in Eq. (16).

$$x_j^i(t+1) = \frac{1}{2}\left(x_j^i(t) + x_j^{i-1}(t)\right) \quad (16)$$

where $i \geq 2$ and $x_j^i(t)$ shows the solution $\boldsymbol{x}^i$ follower in jth dimension.

Operator#3: AβHC local search

In order to boost the exploitation capability of HSSA, each solution generated by the original operator of SSA is passed with a probability of B_r. The discussion of AβHC is given in Sect. 2.

Operator#4: Update population

Each solution $\boldsymbol{x}^i(t)$ is replaced by updated solution $\boldsymbol{x}^i(t+1)$, if better (i.e., $f(\boldsymbol{x}^i(t+1)) \leq f(\boldsymbol{x}^i(t))$).

3.2.3 The improvement step in hybrid crow search algorithm (HCSA)

Crow Search Algorithm (CSA) is a population-based algorithm that imitates the natural phenomenon where the crows keep their excess food in hiding places in which the food is retrieved when needed [70]. In the original version of CSA, there are two essential operators to reconstruct each solution in the population (i.e., generate a new

position and update the population). In the proposed HCSA, each reconstructed solution is passed to AβHC algorithm as a new local refinement operator. The flowchart of HCSA is given in Fig. 5. The HCSA operators will be discussed as follows:

Operator#1: Generate new positions

In order to update the position of the solutions in the current population, the Eq. (17) is utilized. Two parameters are used to control the CSA (i.e., AP and fl). The parameter AP is used to control the updating positions of the current solution based on either neighbouring solution or generating a random solution (i.e., $\boldsymbol{x}_r$). The parameter AP is normally initialized with a set of values as many as population size (N) which takes very small values between 0 and 1. In most cases, $AP = 0.05$. The parameter fl determines the effect of local search or global search. large values of fl leads more to exploration while small value of fl leads to exploitation. As shown in Eq. (17), for any solution $\boldsymbol{x}^i$, a random solution $\boldsymbol{x}^k$ from the population is selected. The updated positions are affected by the different between the current solution $\boldsymbol{x}^i$ and the random solution $\boldsymbol{x}^k$.

$$\boldsymbol{x}^i(t+1) = \begin{cases} \boldsymbol{x}^i(t) + r_1 \times fl^i(t) \times (\boldsymbol{x}^k(t) - \boldsymbol{x}^i(t)) & r_2 \geq AP^i(t) \\ \boldsymbol{x}_r & r_2 < AP^i(t) \end{cases} \tag{17}$$

Note that r_1 and r_2 are two values generated from a uniform random distribution between 0 and 1.

Operator#2:AβHC local search

In order to boost the exploitation capability of HSSA, each solution generated by the original operator of SSA is passed with a probability of B_r. The discussion of AβHC is given in Sect. 2.

Operator#3:update the population

Each solution $\boldsymbol{x}^i(t)$ is replaced by updated solution $\boldsymbol{x}^i(t+1)$, if better (i.e., $f(\boldsymbol{x}^i(t+1)) \leq f(\boldsymbol{x}^i(t))$).

3.2.4 The improvement step in hybrid grey wolf optimizer (HGWO)

Grey Wolf Optimizer (GWO) is a population-based algorithm stimulated by how grey wolf packs lead and hunt [71]. In the proposed HGWO there are four essential operators to update each solution and the updated solution will be passed to AβHC algorithm as a new local refinement operator with a probability of B_r. The flowchart of HGWO is presented in Fig. 6. These four operators will be discussed as follows:

Operator#1: Social hierarchy

The grey wolf packs have a social hierarchy in which the three wolves with the highest fitness in $\boldsymbol{x}$ are choosed in the hope of obtaining the global optimal solution. The best, second best, and third best solutions are $\boldsymbol{x}_\alpha$, $\boldsymbol{x}_\beta$, $\boldsymbol{x}_\delta$, respectively. Note that other solutions are symbolized as $\boldsymbol{x}_\omega$. In each iteration the group of wolves in $\boldsymbol{x}_\omega$ are pulled to $\boldsymbol{x}_\alpha$, $\boldsymbol{x}_\beta$, $\boldsymbol{x}_\delta$ location.

Operator#2: Encircling prey

The intelligent hunting behaviour will be observed after settling the social hierarchy of the grey wolf packs. It includes three steps: (1) The packs track, chase and approach the prey; (2) They encircle and harass the prey to exhaust it; (3) The packs attack the prey. These steps are symbolized mathematically as follows:

$$\boldsymbol{x}^i(t+1) = \boldsymbol{x}^i(t) - \boldsymbol{A} \times \boldsymbol{D} \tag{18}$$

Note that the wolf next position is denoted by $\boldsymbol{x}^i(t+1)$, the current position is $\boldsymbol{x}^i(t)$, the coefficient matrix is $\boldsymbol{A}$ and the vector $\boldsymbol{D}$ depends on the prey position ($\boldsymbol{x}_p$) which is calculated as follows:

$$\boldsymbol{D} = \left| \boldsymbol{C} \times \boldsymbol{x}_p(t) - \boldsymbol{x}^i(t) \right| \tag{19}$$

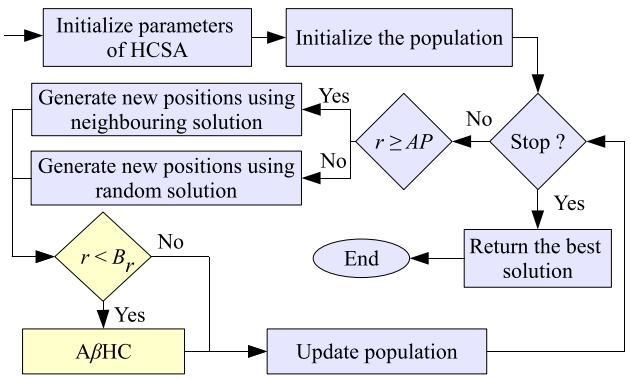


Fig. 5 Flowchart of the proposed HCSA

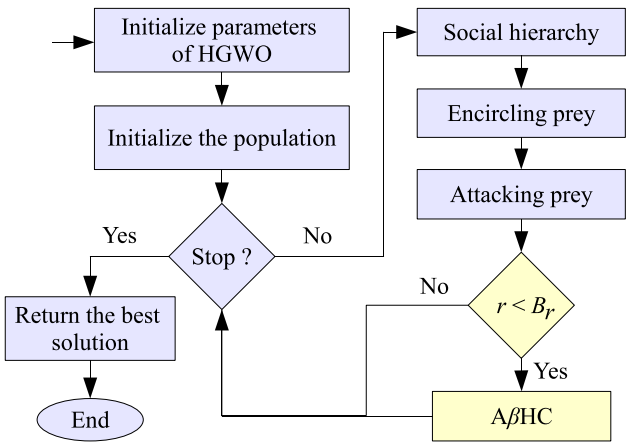


Fig. 6 Flowchart of the proposed HGWO

$$\boldsymbol{C} = 2 \times \boldsymbol{r}_2 \tag{20}$$

where $\boldsymbol{r}_2$ is a vector that is generated randomly in the range [0,1].

Operator#3: Attacking prey

As mentioned before, the fittest three solutions are denoted as $\boldsymbol{x}_\alpha$, $\boldsymbol{x}_\beta$, $\boldsymbol{x}_\delta$. These solutions will be guiding other wolves to update their locations as follows:

$$\boldsymbol{x}^i(t+1) = \frac{1}{3}\boldsymbol{x}_1 + \frac{1}{3}\boldsymbol{x}_2 + \frac{1}{3}\boldsymbol{x}_3 \tag{21}$$

where $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ and $\boldsymbol{x}_3$ are computed using Eq. (22).

$$\begin{aligned} \boldsymbol{x}_1 &= \boldsymbol{x}^i_\alpha(t) - \boldsymbol{A}_1 \times \boldsymbol{D}_\alpha \\ \boldsymbol{x}_2 &= \boldsymbol{x}^i_\beta(t) - \boldsymbol{A}_2 \times \boldsymbol{D}_\beta \\ \boldsymbol{x}_3 &= \boldsymbol{x}^i_\delta(t) - \boldsymbol{A}_3 \times \boldsymbol{D}_\delta \end{aligned} \tag{22}$$

$\boldsymbol{D}_\alpha$, $\boldsymbol{D}_\beta$ and $\boldsymbol{D}_\delta$ are computed by applying Eq. (23).

$$\begin{aligned} \boldsymbol{D}_\alpha &= |\boldsymbol{C}_1 \times \boldsymbol{x}_\alpha - \boldsymbol{x}| \\ \boldsymbol{D}_\beta &= |\boldsymbol{C}_2 \times \boldsymbol{x}_\beta - \boldsymbol{x}| \\ \boldsymbol{D}_\delta &= |\boldsymbol{C}_3 \times \boldsymbol{x}_\delta - \boldsymbol{x}| \end{aligned} \tag{23}$$

Operator#4: AβHC local search

The HGWO exploitation is enhanced by iterating through each developed solution of GWO with B_r probability. The discussion of AβHC is mentioned in Sect. 2.

3.2.5 The improvement step in hybrid particle swarm optimization (HPSO)

Particle swarm optimization (PSO) is inspired by the flocking birds' behavior when they search for the source of adequate food [72]. In HPSO, there are three essential operators to update each solution and the generated solution will be passed to AβHC algorithm as a new local refinement operator to improve the generated solution. The flowchart of HPSO is illustrated in Fig. 7. These three operators are as follows:

Operator#1:update local best and global best solutions

At each iteration, the local best solution (i.e., $\boldsymbol{x}^{lb}$) which is the best solution obtained in the current iteration must be determined. Furthermore, the global best solution (i.e., $\boldsymbol{x}^{gb}$) is updated.

Operator#2:update the positions

Before each position in any solution is updated, the velocity vector $v^i(t)$ is initially updated as shown in Eq. (25).

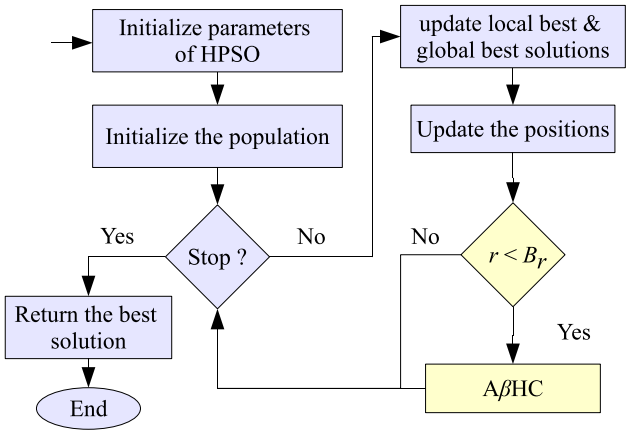


Fig. 7 Flowchart of the proposed HPSO

$$\begin{aligned} v^i(t+1) = w(t) \times v^i(t) + c_1 \times \left(r_1(t) \times \left(\boldsymbol{x}^{lb} - \boldsymbol{x}^i(t)\right)\right) \\ + c_2 \times \left(r_2 \times \left(\boldsymbol{x}^{gb} - \boldsymbol{x}^i(t)\right)\right) \end{aligned} \tag{24}$$

where $w(t) = (w_{max} - t) \times (w_{max} - w_{min})/M_t$ represents the inertia weight parameter that decreases linearly within the range 0.9 to 0.4. Furthermore, the particles velocities are embedded in this parameter. Also, the acceleration coefficients c_1 and c_2 are user defined parameters that can be set as follows: $0 \le c_1 \le 2$ and $0 \le c_2 \le 2$. Finally, r_1 and r_2 are random values in the range [0,1] that are utilized to update the velocity.

Thereafter, based on the updated velocity $v^i(t+1)$, the positions of the new solution is updated as shown in Eq. (25):

$$\boldsymbol{x}^i(t+1) = \boldsymbol{x}^i(t) + v^i(t+1) \tag{25}$$

Operator#3: AβHC local search

The HPSO exploitation is enhanced by iterating through each developed solution of PSO with B_r probability. The discussion of AβHC is mentioned in Sect. 2.

3.2.6 The improvement step in hybrid JAYA algorithm (HJAYA)

JAYA is a Sanskrit term that means *victory*. The JAYA algorithm is a population-based optimization technique that applied the principle of "survival of the fittest" [73]. The solutions generated by JAYA are moving towards the global best solutions while moving away from the worst solutions. The algorithm is considered easy to utilize as it does not have specific parameters. The proposed HJAYA algorithm uses three essential operators to update each solution and the generated solution. The developed solutions are then passed to AβHC algorithm as a new local

refinement operator to improve the generated solution with a probability of B_r. The flowchart of HJAYA is presented in Fig. 8. These three operators will be discussed as follows:

Operator#1: JAYA Evolution process

The evolution process of JAYA is conducted using Eq. (26).

$$x_j^i(t+1) = x_j^i(t) + r_1 \times (x_j^1(t) - |x_j^i(t)|) - r_2 \times (x_j^N(t) - |x_j^i(t)|) \quad (26)$$

where the new updated solution is $\boldsymbol{x}^i(t+1)$, and the current solution is $\boldsymbol{x}^i(t)$. The random value in the range of [0,1] is denoted as r_1 and r_2. These values are used to obtain the equilibrium between exploration and exploitation. Also, the decision variable j in the best solution is symbolized as $\boldsymbol{x}_j^i(t)$ and the decision variable j in the worst solution is symbolized as $\boldsymbol{x}_j^N(t)$.

Operator#2: AβHC local search

The HJAYA exploitation is enhanced by iterating through each developed solution of JAYA with B_r probability. The discussion of AβHC is mentioned in Sect. 2.

Operator#3:update population

At each iteration the fitness value of the new solution $\boldsymbol{x}^i(t+1)$ is computed. The new solution $\boldsymbol{x}^i(t+1)$ will replace the current solution $\boldsymbol{x}^i(t)$ if $f(\boldsymbol{x}^i(t+1)) \leq f(\boldsymbol{x}^i(t))$.

4 Experiments and results

In this section, the proposed hybridized adaptive β-hill climbing (AβHC) on the six MA versions for MLP is evaluated. The proposed MA versions are hybrid flower pollination algorithm (HFPA), hybrid salp swarm algorithm (HSSA), hybrid crow search algorithm (HCSA), hybrid grey wolf optimization (HGWO), hybrid particle swarm optimization (HPSO), and hybrid JAYA algorithm (HJA) by utilizing 15 datasets with different sizes and number of classes. The datasets details are presented in Sect. 4.1. The configuration of the experiments is portrayed in Sect 4.2. The performance of the different proposed MA versions on training MLP is explained Sect.4.3.

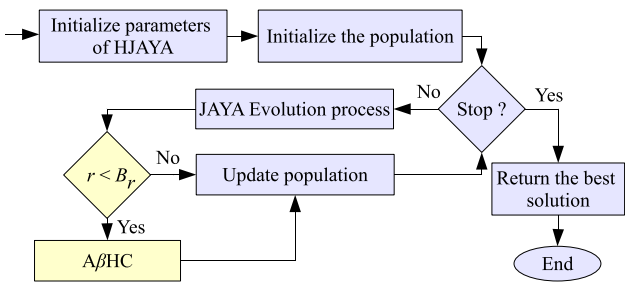


Fig. 8 Flowchart of the proposed HJAYA

4.1 Test datasets

In this section, the performance of proposed hybrid methods, which are HCSA, HFPA, HSSA, HGWO, HJA, and HPSO, is investigated and evaluated using 15 benchmark classification optimization problems. Such optimization problems are provided by UCI Machine Learning Repository [74]. All datasets characteristics, including the number of classes, features, instances, hidden layers, and MLP structures are presented in Table 1. These datasets are selected due to their different class labels; i.e., two, three, four, six, and ten labels, to investigate the proposed methods efficiently.

This study utilises min-max normalization for all datasets to minimize the features' effect with different difficulty levels. Features difficulty levels are mathematically formalized as follow:

$$x_{nor} = \frac{x_i - F_{min}}{F_{max} - F_{min}} \quad (27)$$

where x_{nor} denotes the normalized value of x_i in the range $[F_{min}, F_{max}]$. F_{min} and F_{max} are the minimum and maximum values of the features, respectively.

The number of hidden layers can be calculated on the basis of various methods. Accordingly, this study calculates neurons number in each hidden layer using the method utilized in [75, 76]. The utilized method is mathematically formulated as follow:

$$h = 2 \times N + 1; \quad (28)$$

where h denotes the neurons number in the hidden layer and N is the number of features in the dataset. Accordingly, the input-hidden-output form is presenting the complete MLP structure of each dataset. For instance, the MLP structure is 8-17-10 for the Yeast dataset, 8, 17, and 10 denote the number of input features, hidden layers, and output class labels, respectively.

For testing and training, all datasets are divided into 30% and 70%, respectively. The *stratified way* is used to split each dataset [77]. This technique calculates each class's ratio and then meets the train/test split rate based on the calculated ratio for each dataset. The use of this strategy helps preserve the share of every class in the data split and increases the representation of the classes of minorities. As a result, a balanced number of classes will be assigned for the train/test portions.

Table 1 Benchmark datasets' characteristics

No	1	2	3	4	5
Dataset	Monk	Baloon	Cancer	Heart	Vertebral
#Classes	2	2	2	2	2
#Features	6	4	9	22	6
#Instances	556	20	699	80	310
#Hidden layers	13	9	19	45	13
MLP structure	6–13–2	37,503	9–19–2	22–45–2	6–13–2
No	6	7	8	9	10
Dataset	Blood	Ionosphere	German	Titanic	Parkinson
#Classes	2	2	2	2	2
#Features	4	33	24	3	22
#Instances	748	351	1000	2201	195
#Hidden layers	9	67	49	7	45
MLP structure	37,503	33–67–2	24–49–2	37,440	22–45–2
No	11	12	13	14	15
Dataset	Iris	Seeds	Vehicle	Glass	Yeast
#Classes	3	3	4	6	10
#Features	4	7	18	9	8
#Instances	150	210	846	214	1484
#Hidden layers	9	15	37	19	17
MLP structure	37,868	7–15–3	18–37–4	9–19–6	8–17–10

4.2 Experimental settings

The proposed six MA versions CSA, HFPA, HSSA, HGWO, HJA, and HPSO are compared using the same datasets. The specifications of the machine used for experimental results are presented in Table 2. The settings for the parameters of some MA versions are summarized in Table 3. Note that the B_r parameter values used are studied in the following section for all MA versions.

The algorithms are implemented for each data set over 30 independent runs. The number of epochs is a parameter that defines the total number of iterations the learning algorithm will run through the training dataset [78]. The number of epochs is 250. It is selected based on what was previously used in the literature [75, 76] which reduces the error from the model when used on similar datasets.

Table 2 Initial parameters of the comparative algorithms

Name	Setting
Hardware	
CPU	Intel R Xeon Silver 1.8 GHz
RAM	6 GB
Hard Drive	200 GB
Software	
Language	MATLAB version 9.7.0
Cloud service	Microsoft Azure server

4.3 Comparison results

The performance of the different proposed MA versions (i.e., HCSA, HGWO, HSSA, HPSO, HFPA, and HJAYA) on training MLP are tested in this section. These

Table 3 Initial parameters of the comparative algorithms

Algorithm	Parameter	Settings
All algorithms	Population size	30
	Maximum iterations	250
	Runs	30
HCSA	Awareness probability (AP)	0.05
	Flight length (fl)	2
HPSO	Maximum inertia weight (w_{max})	0.9
	Minimum inertia weight (w_{min})	0.4
	Acceleration coefficient (c_1)	2
	Acceleration coefficient (c_2)	2

algorithms are utilized to evolve the weights of networks as well as to select the optimal number of hidden neurons. Remarkably, each of the proposed MA versions is designed by hybridizing the original version of the population-based algorithm with an adaptive β-hill climbing optimizer (AβHC). Recall that the parameter settings of these algorithms are given in Sect. 4.2, while the characteristics of the datasets used for evaluation purposes are provided in Sect. 4.1.

Table 4 illustrates the results of running the proposed MA versions on 15 datasets using different settings of the B_r parameter. These results are summarized in terms of the mean, the standard deviation (STD), and the best-obtained classification accuracy. From Table 4, it can be seen that each of the proposed MA versions had four versions based on the value of the B_r parameter. The B_r parameter is studied using different settings (B_r= 0, B_r= 0.01, B_r= 0.1, and B_r= 0.3). The higher value of B_r parameter leads to a higher rate of calling the AβHC algorithm in the population-based algorithm, and thus a higher rate of exploitation during the search process. For instance, when the value of the B_r is equal to zero, this means that the AβHC is neglected, and thus the original versions of these algorithms (i.e., CSA, HWO, SSA, PSO, FPA, and JAYA). It should be noted that the best results are highlighted in **bold** font.

The experimental results shown in Table 4 are analyzed and discussed in two phases: I) comparing the results obtained by the different four versions of each MA, and II) comparing the best results obtained by all of the proposed MA versions. In general, it can be seen from the results recorded in Table 4 that the accuracy is gradually enhanced when the value of B_r is increased. Furthermore, the performance of the algorithm is worse than the other versions of the same algorithm (i.e., the hybrid version) when the value of $B_r = 0$ (i.e., using the original algorithm).

From Table 4, it can be seen that the performance of the HCSA with $B_r = 0.3$ outperforms three other versions of the same MA by obtaining the best results on 9 datasets in terms of the mean of the results, and in 10 datasets in terms of best results. On the other hand, the HCSA with B_r= 0.3 failed to obtain the best results for any of the datasets in the terms of mean results, while it got the best results on 2 datasets (i.e., Baloon, and Iris) in terms of best results. These two datasets are the easiest datasets where the Baloon dataset has 4 features, 9 samples, and 2 classes, while the Iris dataset has 4 features, 9 samples, and 3 classes. However, the HCSA with B_r= 0.01 achieved the best results on 2 datasets (i.e., Cancer, and Heart) in terms of the mean of the results, and 5 datasets in terms of best results. While the HCSA with B_r= 0.1 get the best results on 5 and 6 datasets in terms of mean and best results respectively.

Similarly, the results of studying the impact of the B_r parameter on the performance of the proposed HGWO algorithm are recorded in Table 4. Apparently, the performance of the HGWO is gradually enhanced, when the value of the B_r is increased based on the mean of the accuracy. Whereas, the HGWO with B_r= 0.3, HGWO with B_r= 0.1, HGWO with B_r= 0.01, and HGWO with B_r= 0.01 algorithms are got the best results in terms of the mean of accuracy on 7, 5, 4, and 2 datasets respectively. On the other hand, the effect of the B_r parameter on the performance of the HHWO algorithm in terms of best results is limited. This is because all versions of HGWO achieved almost the same number of the best results. The HGWO with B_r= 0.3, HGWO with B_r= 0.1, HGWO with B_r= 0.01, and HGWO with B_r= 0.01 algorithms are obtained the best results in terms of best accuracy on 5, 6, 7, and 5 datasets respectively. In addition, it can be observed from Table 4 that the HGWO with B_r= 0.3 is more robust than the three other versions of HGWO by getting the lowest STD values for all datasets.

The influence of the B_r parameter on the performance of the proposed HSSA is studied using four different values. The results of running the HSSA are recorded in Table 4. Clearly, the performance of the HSSA with B_r= 0.3 is better than the three other versions of HSSA by getting the best results on 8 datasets in terms of the mean of accuracy. Furthermore, the HSSA obtained the best results on 7 datasets in terms of the best accuracy. On the other hand, the performance of the HSSA with B_r= 0 performs better than the other three versions of HSSA by obtaining the best results on 4 datasets (i.e., Baloon, Cancer, Titanic, and Parkinson) in terms of the mean of the accuracy. While the HSSA with B_r= 0 is achieved the best results on 3 datasets (i.e., Baloon, Glass, and Vehicle) in terms of the best accuracy. The HSSA with B_r= 0.01 outperforms the three other versions of HSSA on 2 datasets (i.e., Heart, and Blood) and 4 datasets (i.e., Baloon, Iris, Cancer, and Blood) in terms of mean and best accuracy. While the results of the HSSA with B_r= 0.1 are better than those of the three other versions of the HSSA on 3 datasets (Baloon, Iris, and Seeds)and 5 datasets (i.e., Baloon, Iris, Heart, Seeds, and Parkinson) in terms of the mean and the best accuracy. Based on the above discussion, it can be observed that the proposed HSSA with B_r= 0.3 performs better than the three other versions of HSSA on the hardest datasets with the highest number of classes like Yeast, German, Ionosphere, Glass, Vertebral, and Monk datasets. This proves the efficiency of using the AβHC as a local search algorithm within the SSA algorithm to enhance the exploitation capability of the HSSA and thus guide the search process of the HSSA to achieve better results for the hardest datasets.

Table 4 The accuracy results of the HCSA, HGWO, HSSA, HPSO, HFPA, and HJAYA algorithms on 15 datasets (Bold is the best)

Algorithm	Monk			Baloon			Iris			Cancer			Heart		
	Mean	STD	Best	Mean	STD	Best	Mean	STD	Best	Mean	STD	Best	Mean	STD	Best
HCSA (B_r= 0)	74.02	3.24	80.72	91.67	10.50	**100.00**	85.63	13.68	97.78	94.70	0.77	96.17	64.03	7.13	75.00
HCSA (B_r= 0.01)	77.21	4.12	83.73	93.33	13.56	**100.00**	94.00	1.67	95.56	95.87	0.93	97.61	**72.50**	8.38	87.50
HCSA (B_r= 0.1)	82.99	5.20	93.98	**100.00**	0.00	**100.00**	96.44	1.11	97.78	94.67	0.73	95.69	66.39	6.74	83.33
HCSA (B_r= 0.3)	86.75	5.40	96.39	**100.00**	0.00	**100.00**	96.81	1.39	97.78	95.61	1.02	98.09	57.64	5.48	66.67
HGWO (B_r= 0)	94.72	5.71	**100.00**	**100.00**	0.00	**100.00**	91.78	2.81	95.56	94.51	1.11	96.65	63.19	5.02	70.83
HGWO (B_r= 0.01)	96.14	5.59	**100.00**	**100.00**	0.00	**100.00**	96.89	1.25	**100.00**	95.74	0.65	97.13	54.03	5.63	66.67
HGWO (B_r= 0.1)	97.49	4.37	**100.00**	**100.00**	0.00	**100.00**	**99.93**	0.41	**100.00**	96.01	1.06	98.09	65.56	7.33	83.33
HGWO (B_r= 0.3)	**99.38**	0.11	99.40	**100.00**	0.00	**100.00**	95.33	0.68	95.56	95.81	0.60	97.13	59.58	6.49	75.00
HSSA (B_r= 0)	81.00	7.83	98.19	**100.00**	0.00	**100.00**	87.93	12.08	97.78	97.26	0.88	98.56	60.69	7.06	75.00
HSSA (B_r= 0.01)	75.96	7.32	95.18	77.22	26.80	**100.00**	91.11	5.60	**100.00**	96.28	1.11	**99.04**	67.64	7.56	79.17
HSSA (B_r= 0.1)	87.31	5.89	99.40	**100.00**	0.00	**100.00**	98.74	1.39	**100.00**	95.15	0.53	96.17	67.22	7.95	83.33
HSSA (B_r= 0.3)	97.75	2.28	**100.00**	**100.00**	0.00	**100.00**	97.78	0.00	97.78	96.75	0.67	97.61	56.39	6.54	75.00
HPSO (B_r= 0)	93.69	7.64	**100.00**	**100.00**	0.00	**100.00**	92.52	5.07	97.78	96.83	0.97	98.56	63.89	7.45	79.17
HPSO (B_r= 0.01)	95.82	5.19	**100.00**	82.22	16.91	**100.00**	96.00	0.90	97.78	96.70	0.98	98.56	65.28	9.11	79.17
HPSO (B_r= 0.1)	88.61	8.92	**100.00**	**100.00**	0.00	**100.00**	97.93	0.81	**100.00**	97.21	0.87	98.56	70.97	6.33	83.33
HPSO (B_r= 0.3)	93.88	6.14	**100.00**	81.11	16.80	**100.00**	96.89	1.11	97.78	96.56	0.83	**99.04**	55.42	8.70	66.67
HFPA (B_r= 0)	66.22	3.73	75.30	94.44	12.63	**100.00**	87.11	6.94	97.78	94.82	0.85	96.17	64.58	9.65	79.17
HFPA (B_r= 0.01)	69.48	4.14	77.71	84.44	15.74	**100.00**	93.48	4.81	**100.00**	96.76	0.74	98.09	70.69	7.13	83.33
HFPA (B_r= 0.1)	70.82	4.33	79.52	96.11	12.13	**100.00**	94.44	2.59	**100.00**	**97.53**	0.77	**99.04**	68.33	7.94	79.17
HFPA (B_r= 0.3)	78.19	3.33	84.34	**100.00**	0.00	**100.00**	95.78	2.50	**100.00**	96.30	0.55	97.13	62.50	7.42	75.00
HJAYA (B_r= 0)	68.25	3.49	74.70	91.67	12.18	**100.00**	89.93	3.81	97.78	97.07	0.78	98.56	62.50	8.19	79.17
HJAYA (B_r= 0.01)	66.43	4.95	78.92	93.33	10.36	**100.00**	90.96	5.02	97.78	97.18	0.66	98.56	70.56	8.88	**91.67**
HJAYA (B_r= 0.1)	69.72	3.75	77.71	96.67	10.17	**100.00**	95.26	3.18	**100.00**	96.28	0.51	97.13	61.81	6.20	75.00
HJAYA (B_r= 0.3)	70.72	5.52	78.92	95.56	11.52	**100.00**	96.81	2.78	**100.00**	95.42	0.82	96.65	65.83	8.29	83.33

Algorithm	Vertebral			Blood			Seeds			Glass			Ionosphere		
	Mean	STD	Best	Mean	STD	Best	Mean	STD	Best	Mean	STD	Best	Mean	STD	Best
HCSA (B_r= 0)	86.67	2.66	90.32	76.00	1.06	78.13	81.16	12.12	93.65	41.93	10.66	57.81	84.06	3.92	91.43
HCSA (B_r= 0.01)	82.51	2.07	87.10	78.82	0.67	79.91	91.48	7.57	96.83	49.38	5.31	57.81	86.29	3.74	95.24
HCSA (B_r= 0.1)	88.14	1.80	**91.40**	79.61	0.55	80.80	94.50	1.99	96.83	56.88	6.72	65.63	86.98	3.07	91.43
HCSA (B_r= 0.3)	86.67	1.80	90.32	80.25	0.62	81.25	87.72	2.39	92.06	62.29	3.75	70.31	91.08	2.83	97.14
HGWO (B_r= 0)	87.28	1.83	90.32	78.59	0.61	79.46	95.24	0.83	96.83	58.23	7.92	68.75	**95.02**	1.60	**98.10**
HGWO (B_r= 0.01)	80.39	1.34	82.80	**82.99**	0.76	84.38	95.13	1.31	98.41	61.56	4.81	70.31	89.87	2.24	94.29
HGWO (B_r= 0.1)	82.22	1.29	83.87	80.24	0.55	80.80	**96.30**	1.58	**100.00**	55.26	3.47	62.50	89.81	3.05	95.24
HGWO (B_r= 0.3)	89.32	1.44	**91.40**	81.44	0.70	83.04	96.19	1.36	98.41	**66.30**	4.76	**73.44**	92.00	2.13	94.29
HSSA (B_r= 0)	81.40	2.08	84.95	78.72	0.55	79.46	83.07	7.10	90.48	55.94	10.28	71.88	88.22	3.03	94.29
HSSA (B_r= 0.01)	84.19	1.85	87.10	80.91	0.62	81.70	89.68	3.14	93.65	54.48	7.55	67.19	87.78	2.75	93.33
HSSA (B_r= 0.1)	85.05	2.00	88.17	77.41	0.46	78.57	94.29	1.70	96.83	56.72	5.54	65.63	87.65	2.93	94.29
HSSA (B_r= 0.3)	**89.35**	0.99	90.32	76.10	0.72	77.23	91.11	1.42	93.65	60.42	3.02	65.63	92.16	2.13	95.24
HPSO (B_r= 0)	85.91	1.79	89.25	79.29	0.79	80.36	83.33	10.37	92.06	47.76	9.13	62.50	86.95	3.17	92.38
HPSO (B_r= 0.01)	86.42	1.58	88.17	80.58	0.59	81.25	93.12	2.81	98.41	53.85	7.22	64.06	87.97	3.20	93.33
HPSO (B_r= 0.1)	82.69	2.35	86.02	78.50	0.84	80.36	90.11	2.52	95.24	60.73	9.31	71.88	88.67	3.80	94.29
HPSO (B_r= 0.3)	83.44	2.05	88.17	78.68	1.29	81.25	92.70	1.85	95.24	52.34	6.95	64.06	84.35	3.34	88.57
HFPA (B_r= 0)	78.85	2.19	83.87	78.11	0.60	79.02	87.99	4.58	93.65	45.31	3.74	53.13	86.10	3.66	92.38
HFPA (B_r= 0.01)	79.85	1.69	83.04	82.15	1.80	**86.02**	89.42	2.51	93.65	45.83	5.37	57.81	84.44	3.02	91.43
HFPA (B_r= 0.1)	82.29	1.80	86.02	77.32	0.97	79.91	94.60	2.45	**100.00**	53.49	6.89	64.06	85.71	3.63	93.33
HFPA (B_r= 0.3)	86.42	1.34	89.25	80.61	0.83	83.04	91.32	2.82	96.83	56.35	4.30	64.06	84.41	4.16	91.43

Table 4 (continued)

Algorithm	Vertebral			Blood			Seeds			Glass			Ionosphere		
	Mean	STD	Best	Mean	STD	Best	Mean	STD	Best	Mean	STD	Best	Mean	STD	Best
HJAYA (B_r= 0)	81.83	2.94	87.10	77.98	1.11	79.91	83.39	5.60	92.06	47.03	5.23	54.69	81.33	4.90	87.62
HJAYA (B_r= 0.01)	82.40	3.41	89.25	78.99	1.74	82.14	88.62	6.61	96.83	46.61	5.69	59.38	80.79	3.57	87.62
HJAYA (B_r= 0.1)	79.25	2.02	83.87	79.78	1.73	83.48	92.80	2.59	96.83	48.33	6.25	64.06	82.41	3.41	87.62
HJAYA (B_r= 0.3)	78.78	3.07	84.95	80.01	1.60	82.59	90.16	2.65	95.24	51.51	6.18	64.06	84.70	3.50	90.48
Algorithm	German			Titanic			Vehicle			Parkinson			Yeast		
	Mean	STD	Best	Mean	STD	Best	Mean	STD	Best	Mean	STD	Best	Mean	STD	Best
HCSA (B_r= 0)	80.79	1.96	84.00	79.42	0.20	79.85	34.80	9.51	55.34	83.79	3.09	89.66	36.48	4.26	44.49
HCSA (B_r= 0.01)	81.44	2.62	85.67	78.01	0.57	78.64	39.00	8.92	53.75	85.86	3.98	93.10	40.43	6.16	48.54
HCSA (B_r= 0.1)	79.47	2.03	84.33	80.38	0.21	80.91	45.76	10.48	58.50	87.13	2.85	93.10	44.66	3.78	53.48
HCSA (B_r= 0.3)	82.91	1.25	85.67	78.48	0.00	78.48	50.97	7.30	62.85	86.15	3.77	91.38	48.68	3.89	**56.63**
HGWO (B_r= 0)	84.83	1.87	88.33	77.10	0.48	77.58	60.59	8.08	**72.73**	84.71	2.55	91.38	40.06	3.94	49.66
HGWO (B_r= 0.01)	85.29	1.48	88.33	78.80	0.25	78.94	53.70	8.00	70.75	**90.34**	3.28	**96.55**	43.21	4.90	54.38
HGWO (B_r= 0.1)	83.90	1.52	87.00	78.42	0.31	79.24	59.41	7.62	71.94	89.77	3.13	**96.55**	43.66	3.76	52.13
HGWO (B_r= 0.3)	84.27	1.45	86.67	**80.87**	0.21	**81.21**	**61.38**	6.51	71.54	89.08	2.61	93.10	45.59	3.85	55.06
HSSA (B_r= 0)	81.42	2.18	85.67	79.87	0.14	80.00	50.61	8.21	70.75	86.72	2.23	91.38	37.51	3.33	44.04
HSSA (B_r= 0.01)	78.86	1.96	83.67	79.03	0.12	79.09	43.82	7.95	57.31	83.05	2.94	87.93	39.40	4.19	48.09
HSSA (B_r= 0.1)	83.72	1.53	87.00	77.12	0.00	77.12	51.12	6.59	59.29	83.16	3.75	93.10	48.48	3.80	53.71
HSSA (B_r= 0.3)	**85.93**	1.71	**88.67**	79.72	0.11	80.30	57.69	5.62	64.03	84.66	2.73	89.66	**50.55**	3.23	55.73
HPSO (B_r= 0)	81.87	1.88	86.67	77.76	0.27	77.88	43.39	8.77	62.45	87.01	3.64	94.83	36.02	3.62	42.47
HPSO (B_r= 0.01)	81.00	1.89	84.33	77.23	0.47	77.58	40.87	10.36	57.71	86.38	2.91	91.38	37.46	3.47	44.04
HPSO (B_r= 0.1)	83.63	1.46	86.67	78.62	0.49	79.55	44.28	8.23	60.47	86.78	2.88	94.83	39.05	3.41	46.52
HPSO (B_r= 0.3)	84.28	1.45	86.67	79.00	0.47	79.39	46.64	6.52	65.61	85.00	3.21	89.66	37.97	3.86	45.84
HFPA (B_r= 0)	79.40	2.48	83.67	78.99	0.47	80.15	31.42	6.67	42.69	82.93	3.77	87.93	33.45	2.77	37.53
HFPA (B_r= 0.01)	78.17	1.97	82.00	80.10	0.63	81.21	33.48	8.45	47.43	85.80	2.99	91.38	34.07	3.17	40.45
HFPA (B_r= 0.1)	77.70	1.63	80.33	78.81	0.35	79.39	39.58	5.88	49.41	81.67	3.40	87.93	34.88	4.20	43.15
HFPA (B_r= 0.3)	79.82	2.02	83.33	79.32	0.23	79.39	46.77	4.60	57.31	85.06	2.65	89.66	39.21	4.19	46.07
HJAYA (B_r= 0)	77.81	2.54	80.67	78.72	0.64	79.39	34.49	5.99	45.06	84.89	3.22	91.38	31.26	3.01	36.85
HJAYA (B_r= 0.01)	79.32	2.61	84.67	77.92	0.74	79.09	34.35	5.39	47.43	86.55	2.09	89.66	31.11	2.81	39.55
HJAYA (B_r= 0.1)	82.09	2.12	85.00	77.21	0.41	77.88	38.80	7.09	50.20	85.63	4.73	93.10	32.99	3.06	38.43
HJAYA (B_r= 0.3)	80.27	1.91	83.67	77.07	0.52	77.88	40.28	4.10	45.85	85.69	4.75	94.83	33.20	3.19	39.78

Table 4 illustrates the results of running the proposed HPSO with different configurations of B_r parameter. Remarkably, the proposed HPSO with B_r= 0.1 outperforms the three other versions of HPSO by obtaining the best results on 7 and 10 datasets in terms of the mean and the best accuracy respectively. While the HPSO with B_r= 0, HPSO with B_r= 0.01, and HPSO with B_r= 0.3 are achieved the best results in terms of the mean accuracy on 2, 4, and 3 datasets respectively. Furthermore, the HPSO with B_r= 0, HPSO with B_r= 0.01, and HPSO with B_r= 0.3 are got the best results in terms of the best accuracy on 5, 4, and 6 datasets respectively. Notably, all versions of HPSO achieved the same best results in terms of the best accuracy for Monk and Baloon datasets. From Table 4, it can be seen that the proposed HPSO with B_r= 0.3 can obtain the best results in terms of the mean of accuracy for the hardest datasets with a higher number of classes like German, Titanic, and Vehicle. While the HPSO with B_r= 0.3 fails to obtain the best results for any of the easiest datasets like Baloon. This leads us to conclude that the higher rate of the B_r parameter is useful for the hardest dataset, while the B_r parameter with considerable value is useful for solving the easiest datasets.

The results of studying the impact of the B_r parameter on the performance of the HFPA are also illustrated in Table 4. It can be seen from the results that the

performance of the HFPA with B_r= 0.3 is better than the three other versions of HFPA by getting the best results in terms of accuracy mean on 8 datasets. In addition, it obtained the best results in terms of best accuracy on 7 datasets. The HFPA with B_r= 0 got the best results in terms of the mean of the accuracy for the Ionosphere dataset, while it obtained the best results in terms of the best accuracy on Baloon, and German datasets. Furthermore, the performance of the HFPA with B_r= 0 is worse than the three other versions of HFPA by getting the worst results for the remaining datasets. This is due to the fact that the value of the B_r parameter is equal to zero, and thus the calling the AβHC is neglected during the search. The HFPA with B_r= 0.01 obtained the best results in terms of the mean of the accuracy on 4 datasets (i.e., Heart, Blood, Titanic, and Parkinson), while it achieved the best results in terms of the best of the accuracy on 6 datasets. Finally, the HFPA with B_r= 0.1 got the best results in terms of the mean of the accuracy on Cancer and Seeds datasets. In addition, the HFPA with B_r= 0.1 obtained the best results in terms of the best accuracy on 6 datasets. From another perspective, it can be seen that the performance of the HFPA with B_r= 0.3 is more robust than the three other versions of HFPA by obtaining the smallest STD values for almost all datasets.

From Table 4, it can be shown that the proposed HJAYA with B_r= 0 is obtained the best results in terms of the mean of the accuracy on the Titanic dataset, and it is got the best results in terms of the best of the accuracy on 3 datasets. In contrast, the HJAYA with B_r= 0 performs worse than the three other versions of HJAYA for the remaining datasets. The HJAYA with B_r= 0.01, HJAYA with B_r= 0.1, and HJAYA with B_r= 0.3 achieved the best results in the terms of the mean of the accuracy on 4, 3, and 7 datasets respectively. in addition, the HJAYA with B_r= 0.01, HJAYA with B_r= 0.1, and HJAYA with B_r= 0.3 are got the best accuracy results on 6, 7, and 7 datasets. Clearly, no difference in the performance of the HJAYA algorithms when the value of the B_r parameter is bigger than zero in the terms of the best accuracy. Furthermore, the performance of the HJAYA B_r= 0.3 is better than the other versions of HJAYA, while the HJAYA B_r= 0.1 is more robust than the other versions of HJAYA in almost all datasets. This is because the higher value of the B_r parameter leads the HJAYA to converge quickly and thus get stuck in the problem of local optima in the early stages of the search process.

Figure 9 illustrates the behavior of the proposed HCSA, HGWO, HSSA, HPSO, HFPA, and HJAYA algorithms using different settings of the B_r parameter on navigating the search space of the Monk and Vehicle datasets. It should be noted that these datasets are different in the number of features, samples, and classes to visualize the differences between the behavior of the different versions of each algorithm in solving these datasets. The *x-axis* and *y-axis* represents the correlation between the fitness values (MSE) and the iteration number. The slope of the curves represents convergence rates. Apparently, the convergence of the proposed MA versions when the value of the $B_r > 0$ is faster than the convergence of them with $B_r = 0$. In addition, the convergence of hybrid-based algorithms with B_r= 0.3 is better than the other versions of each algorithm in almost all cases when it is used to solve Monk and Vehicle datasets. However, it can be seen that the behavior of the HJAYA with B_r= 0.01 is better than the three other versions of HJAYA in solving the Monk dataset. This is because the three other versions of the HJAYA algorithms converge fast and thus get stuck in local optima in the early stages of the search process. The convergence of the HCSA with B_r= 0.1 is gradually enhanced till the last stages of the search process when it is utilized to solve the Monk and Vehicle datasets. Finally, the convergence of the HPSO with B_r= 0.1 is better than the other versions of HPSO when it is applied to solving the Vehicle dataset.

In order to highlight which method has the best performance, the performance of the proposed MA versions (i.e., HCSA, HGWO, HSSA, HPSO, HFPA, and HJAYA) are compared against each other in Table 4. Firstly, the comparison in terms of the mean of the accuracy, it can be seen that the performance of the proposed HGWO performs better than the other comparative methods by obtaining the best results on 9 datasets. In addition, the performance of the HGWO is similar to some of the other comparative methods by obtaining the optimal results on Baloon datasets. The performance of the HSSA is better than the other algorithms on 3 datasets (i.e., Vertebral, German, and Yeast), while it is similar to other algorithms on the Baloon dataset. The HFPA obtains the best results on two datasets (i.e., Cancer, and Baloon). The HCSA performs similarly to or better than the other algorithms by obtaining the best results on Heart and Baloon datasets. The HPSO yields the best results on the Baloon dataset, while the HJAYA failed to obtain the best result for any of the datasets. The algorithms comparison in terms of the best accuracy shows that the proposed HGWO outperforms other algorithms on 4 datasets (i.e., Glass, Ionosphere, Vehicle, and Parkinson), while the performance of the HGWO is similar to some algorithms on 6 datasets (i.e., Titanic, Seeds, Vertebral, Iris, Baloon, and Monk). The HCSA performs better than other algorithms on Yeast datasets, while the performance of the HCSA is similar to some of the other algorithms on two datasets (i.e., Baloon, and Vertebral). The performance of the HPSO is similar to other algorithms by getting the same best results on four datasets (i.e., Monk, Baloon, Iris, and Cancer). The HSSA performs better than other algorithms on the German

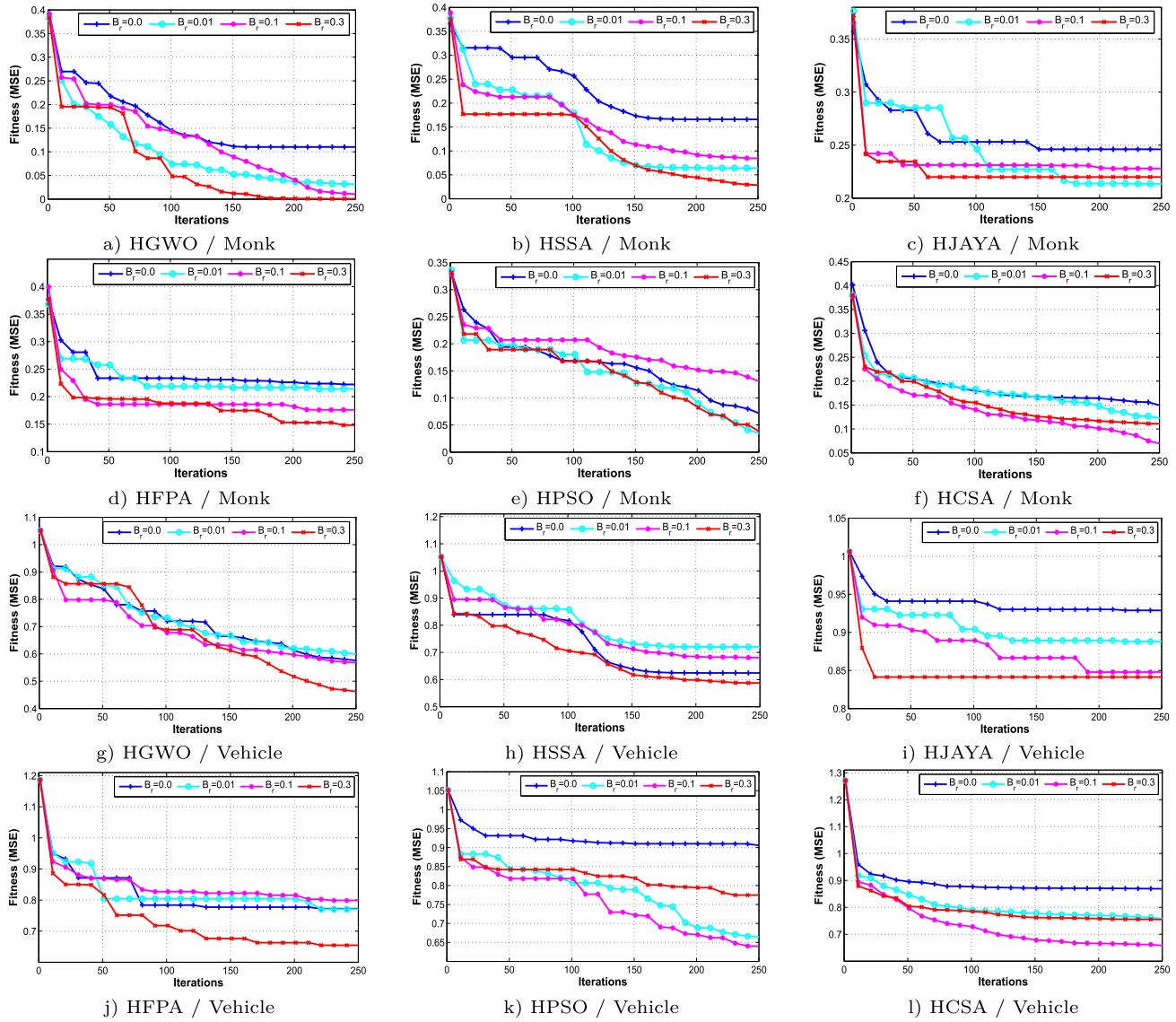


Fig. 9 Convergence results of HGWO, HSSA, HJAYA, HFPA, HPSO, and HCSA on Monk and Vehicle datasets

dataset, while its performance is similar to some of the other algorithms on 4 datasets (i.e., Monk, Baloon, Iris, and Cancer). The HJAYA yields the best results on the Heart dataset, while the performance of the HJAYA is similar to other algorithms on two datasets (i.e., Baloon, and Iris). The HFPA performs better than the other algorithms on the Blood dataset, while it obtains the same best results as the other algorithms on 5 datasets (i.e., Baloon, Iris, Cancer, Seeds, and Titanic). Based on the above discussions, we can conclude that the performance of the hybrid-based algorithms is better than the performance of the original versions of the algorithms. Furthermore, the performance of the proposed HGWO is better than the performance of the other hybrid-based algorithms on almost all of the datasets. This is since the GWO algorithm utilizes the three fittest solutions in the current population at each iteration to guide the search process. That leads to the solutions in the population following the fittest solutions and thus produces fast convergence.

Figure 10 illustrates the notched boxplot that is used to visualize the distributions of the MSE results for running the proposed MA versions, as well as the original versions of the algorithms on all datasets. It should be noted that the smallest distance between the best, median, and worst of the MSE results demonstrates the robustness of the algorithm.

4.3.1 Friedman's statistical test

This section provides a statistical study based on Friedman's statistical test that shows the average rankings of all compared methods and their MA versions, including

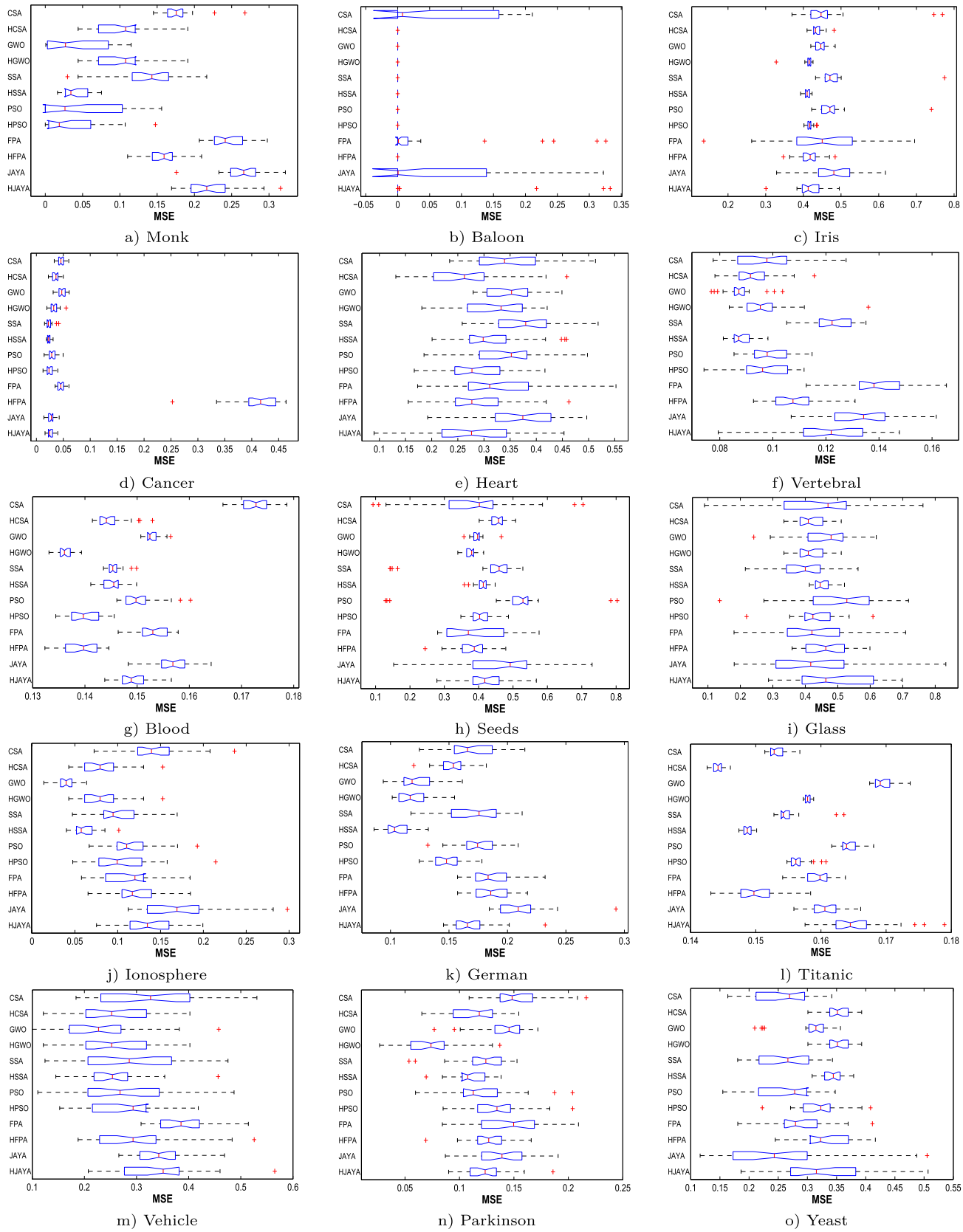


Fig. 10 Boxplot charts of MSE results of CSA, HCSA, GWO, HGWO, SSA, HSSA, PSO, HPSO, FPA, HFPA, JAYA, and HJAYA

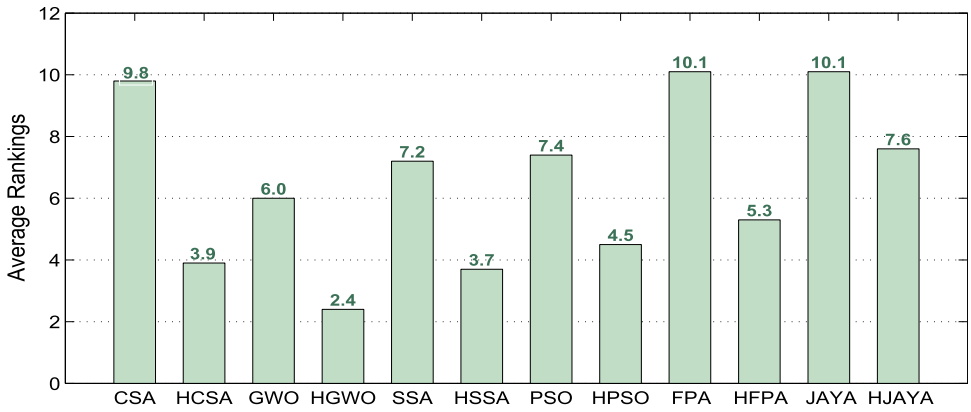


Fig. 11 Average rankings of the comparative algorithms using Friedman's statistical test

GWO, HGWO, SSA, HSSA, JAYA, HJAYA, FPA, HFPA, PSO, HPSO, CSA, and HCSA. Such average rankings are calculated using the results provided in Table 4. In Friedman's statistical test, the null hypothesis (H_0) and alternative hypothesis (H_1) are used to investigate the behavior of the compared methods. H_0 implies that the comparative methods' behaviors are similar, whereas H_1 indicates that the comparative methods' behaviors are different. In addition, the p-value term is used to investigate whether a significant difference in the compared methods' results, where a significant difference exists between the methods if p-value $\leq$ 0.05. Otherwise, no significant difference.

Figure 11 shows the average rankings of all compared methods. The figure proves the robust performance of HGWO in addressing the problem, where it obtained the lowest average rankings. However, JAYA and FPA achieved the worst results by obtaining the highest average rankings. In addition, significant differences between the comparative algorithms are proved, where the p-value obtained by Friedman's test is less than 0.05. Accordingly, the H_0 is rejected.

The Holm and Hochberg procedures are then used as *post-hoc* techniques to demonstrate whether the controlled method's results are significantly different from other methods' results based on the adjusted ρ-value. Figure 11 shows that the controlled method is HGWO, as it achieves the lowest average rankings among all compared methods. Holm's procedure rejects the null hypothesis H_0 when the ρ-value $\leq$0.0125, while Hochberg's procedure rejects the null hypothesis H_0 when the ρ-value $\leq$0.01. The difference between the HGWO method and seven other methods (FPA, JAYA, CSA, HJAYA, PSO, SSA, and GWO) is significant, as shown in Table 5. Conversely, the difference between the HGWO method and four other methods (HFPA, HPSO, HCSA, and HSSA) is not significant. This section demonstrates that the HGWO algorithm is a new alternative that can solve such problems successfully.

The proposed memetic computing framework can be used as a general structure for enhancing the performance of population-based optimization algorithms by distilling local search techniques. This improvement can empower the efficiency of MLP and thus produce more accurate results when utilized in different MLP-based applications. By means of hybridization of local search with the population-based optimization algorithm. The trade-off between local exploitation and global exploration can be achieved during the search.

Table 5 Holm/Hochberg results between the HGWO algorithm and other algorithms

Order	Algorithm	Holm/Hochberg	Adjusted ρ-value	Null Hypotheses H_0
11	FPA	0.0045	4.958E-9	Rejected
10	JAYA	0.005	5.771E-9	Rejected
9	CSA	0.0055	1.901E-8	Rejected
8	HJAYA	0.0062	9.657E-5	Rejected
7	PSO	0.0071	1.788E-4	Rejected
6	SSA	0.0083	3.241E-4	Rejected
5	GWO	0.01	0.0072	Rejected
4	HFPA	0.0125	0.0276	Not rejected
3	HPSO	0.01666	0.1106	Not rejected
2	HCSA	0.025	0.2762	Not rejected
1	HSSA	0.05	0.3359	Not rejected

5 Conclusion and future work

Using local-based search algorithms to hybridize population-based algorithms can boost their performance in terms of exploitation capability. This can help population-based search algorithms to have a balance between exploration and exploitation while examining the problem search space. In this paper, adaptive β-hill climbing (AβHC) is utilized as a local search refinement to hybridize six population-based metaheuristics for training MLP such as hybrid crow search algorithm (HCSA), hybrid flower pollination algorithm (HFPA), hybrid salp swarm algorithm (HSSA), hybrid grey wolf optimization (HGWO), hybrid JAYA algorithm (HJA), and hybrid particle swarm optimization (HPSO). The algorithms are exploring the search space to find the optimal values of MLP weights and biases.

The performance of the proposed six MA versions is evaluated using 15 datasets with different sizes and with a different number of classes. These datasets are normalized and then split into test and train sets 30% and 70% respectively. A stratified way is applied when splitting the dataset to ensure the presence of small classes when training MLPs. Each dataset is trained using MLP configured with a different number of inputs, hidden, and output neurons.

The tuning of the B_r parameter of AβHC shows that the accuracy results obtained by the proposed MA versions are gradually enhanced when this parameter value increases. This is because it empowers the exploitation capability of the algorithm. The obtained results show that all of the hybridized algorithms outperform the original version of the algorithms. In addition, the HGWO excels all other MA versions. Based on the statistical analysis results the following algorithms show competitive results as well for HFPA, HPSO, HCSA, and HSSA.

As a future direction, the proposed algorithm will be utilized to tackle real-world applications that have more complex search space. The proposed algorithms can be refined to have other local search techniques to improve their exploitation abilities. Several researchers investigated the sensitivity of the MLP to the number of added layers. For example, Zeng and Yeung [79] investigated ten MLPs with different layers starting with 2-1 and each time add a layer with two neurons into the previous MLP. In the future, the upper bounds for the number of layers in MLP can be investigated.

Author contributions MAA-B: Conceptualization, Methodology, Software, Formal analysis, Investigation, Writing—original draft, Writing—review and editing. MAA: Programming, Methodology, Writing—original draft, Writing—review and editing. IAD: Methodology, Writing—original draft, Writing—review and editing. OAA: Statistical test, Writing—original draft, Writing—review and editing. AKA: Formal analysis, Investigation, Writing—original draft. SNM: Statistical test, Writing—original draft. ZAAA: Formal analysis, Investigation, Writing—original draft.

Funding This work is supported by the Deanship of Scientific Research & Innovation at Al-Zaytoonah University of Jordan granted to the second author (Grant No. 2022-2021/08/17).

Data availability The data that support the findings of this study are available from the corresponding author upon reasonable request.

Declarations

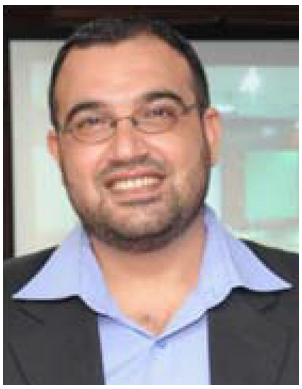
Conflict of interest The authors declare that they have no conflict of interest.

References

1. Hassoun, M.H., et al.: Fundamentals of Artificial Neural Networks. MIT press, Cambridge (1995)
2. Lawrence, S., Giles, C.L., Tsoi, A.C., Back, A.D.: Face recognition: a convolutional neural-network approach. IEEE Trans. Neural Netw. **8**(1), 98–113 (1997)
3. Bebis, G., Georgiopoulos, M.: Feed-forward neural networks. IEEE Potentials **13**(4), 27–31 (1994)
4. Ghosh-Dastidar, S., Adeli, H.: Spiking neural networks. Int. J. Neural Syst. **19**(04), 295–308 (2009)
5. Medsker, L.R., Jain, L.C.: Recurrent neural networks. Des. Appl. **5**, 64 (2001)
6. Orr, M.J.L. et al.: Introduction to radial basis function networks (1996)
7. Yong, Y., Si, X., Changhua, H., Zhang, J.: A review of recurrent neural networks: Lstm cells and network architectures. Neural Comput. **31**(7), 1235–1270 (2019)
8. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT press, Cambridge (2016)
9. Verikas, A., Bacauskiene, M.: Feature selection with neural networks. Pattern Recogn. Lett. **23**(11), 1323–1335 (2002)
10. She, F.H., Kong, L.X., Nahavandi, S., Kouzani, A.Z.: Intelligent animal fiber classification with artificial neural networks. Textile Res. J. **72**(7), 594–600 (2002)
11. Savalia, S., Emamian, V.: Cardiac arrhythmia classification by multi-layer perceptron and convolution neural networks. Bioengineering **5**(2), 35 (2018)
12. Meshram, S.G., Ghorbani, M.A., Shamshirband, S., Karimi, V., Meshram, C.: River flow prediction using hybrid Psogsa algorithm based on feed-forward neural network. Soft Comput. **23**(20), 10429–10438 (2019)
13. Doush, I.A., Sawalha, A.: Automatic music composition using genetic algorithm and artificial neural networks. Malays. J. Comput. Sci. **33**(1), 35–51 (2020)
14. Belciug, S.: Logistic regression paradigm for training a single-hidden layer feedforward neural network. application to gene expression datasets for cancer research. J. Biomed. Inform. **102**, 103373 (2020)
15. Aljarah, I., Faris, H., Mirjalili, S.: Optimizing connection weights in neural networks using the whale optimization algorithm. Soft Comput. **22**(1), 1–15 (2018)
16. Ghanem, W.A.H.M., Jantan, A.: A cognitively inspired hybridization of artificial bee colony and dragonfly algorithms for

training multi-layer perceptrons. Cogn. Comput. **10**(6), 1096–1134 (2018)
17. Valian, E., Mohanna, S., Tavakoli, S.: Improved cuckoo search algorithm for feedforward neural network training. Int. J. Artif. Intell. Appl. **2**(3), 36–43 (2011)
18. Mirjalil, S., Hashim, S.Z.M., Sardroudi, H.M.: Training feedforward neural networks using hybrid particle swarm optimization and gravitational search algorithm. Appl. Math. Comput. **218**(22), 11125–11137 (2012)
19. Faris, H., Mirjalili, S., Aljarah, I.: Automatic selection of hidden neurons and weights in neural networks using grey wolf optimizer based on a hybrid encoding scheme. Int. J. Mach. Learn. Cybern. **10**(10), 2901–2920 (2019)
20. Mirjalili, S.: How effective is the grey wolf optimizer in training multi-layer perceptrons. Appl. Intell. **43**(1), 150–161 (2015)
21. Jalali, S.M.J., Ahmadian, S., Kebria, P.M., Khosravi, A., Lim, C.P., Nahavandi, S.: Evolving artificial neural networks using butterfly optimization algorithm for data classification. In: International Conference on Neural Information Processing, pp. 596–607. Springer (2019)
22. Chen, H., Wang, S., Li, J., Li, Y.: A hybrid of artificial fish swarm algorithm and particle swarm optimization for feedforward neural network training. In: International Conference on Intelligent Systems and Knowledge Engineering 2007. Atlantis Press (2007)
23. Bairathi, D., Gopalani, D.: Salp swarm algorithm (ssa) for training feed-forward neural networks. In: Soft computing for problem solving, pp. 521–534. Springer (2019)
24. Yin, Y., Tu, Q., Chen, X.: Enhanced salp swarm algorithm based on random walk and its application to training feedforward neural networks. Soft Comput. **24**, 14791 (2020)
25. Alboaneen D.A., Tianfield H., Zhang, Y.: Glowworm swarm optimisation for training multi-layer perceptrons. In: Proceedings of the Fourth IEEE/ACM International Conference on Big Data Computing, Applications and Technologies, pp. 131–138 (2017)
26. Montana, D.J., Davis, L.: Training feedforward neural networks using genetic algorithms. In: IJCAI, vol. 89, pp. 762–767 (1989)
27. Moayedi, H., Nguyen, H., Foong, L.K.: Nonlinear evolutionary swarm intelligence of grasshopper optimization algorithm and gray wolf optimization for weight adjustment of neural network. Eng. Comput. **37**, 1265 (2019)
28. Heidari, A.A., Faris, H., Aljarah, I., Mirjalili, S.: An efficient hybrid multilayer perceptron neural network with grasshopper optimization. Soft Comput. **23**(17), 7941–7958 (2019)
29. Faris, H., Aljarah, I., Alqatawna, J.: Optimizing feedforward neural networks using krill herd algorithm for e-mail spam detection. In: 2015 IEEE Jordan Conference on Applied Electrical Engineering and Computing Technologies (AEECT), pp. 1–5. IEEE (2015)
30. Faris, H., Aljarah, I., Mirjalili, S.: Improved monarch butterfly optimization for unconstrained global search and neural network training. Appl. Intell. **48**(2), 445–464 (2018)
31. Mirjalili, S.Z., Saremi, S., Mirjalili, S.M.: Designing evolutionary feedforward neural networks using social spider optimization algorithm. Neural Comput. Appl. **26**(8), 1919–1928 (2015)
32. Socha, K., Blum, C.: An ant colony optimization algorithm for continuous optimization: application to feed-forward neural network training. Neural Comput. Appl. **16**(3), 235–247 (2007)
33. Jaddi, N.S., Abdullah, S., Hamdan, A.R.: Multi-population cooperative bat algorithm-based optimization of artificial neural network model. Inf. Sci. **294**, 628–644 (2015)
34. Zhang, Y., Phillips, P., Wang, S., Ji, G., Yang, J., Jianguo, W.: Fruit classification by biogeography-based optimization and feedforward neural network. Expert Systems **33**(3), 239–253 (2016)
35. 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**(06), 1650033 (2016)
36. Heidari, A.A., Faris, H., Mirjalili, S., Aljarah, I., Mafarja, M.: Ant lion optimizer: theory, literature review, and application in multi-layer perceptron neural networks. In: Mirjalili, S., Song Dong, J., Lewis, A. (eds.) Nature-Inspired Optimizers, pp. 23–46. Springer, Cham (2020)
37. Wu, H., Zhou, Y., Luo, Q., Basset, M.A.: Training feedforward neural networks using symbiotic organisms search algorithm. Comput. Intell. Neurosci. https://doi.org/10.1155/2016/9063065 (2016)
38. Al-Betar, M.A., Khader, A.T., Zaman, M.: University course timetabling using a hybrid harmony search metaheuristic algorithm. IEEE Trans. Syst. Man Cybern. Part C Appl. Rev. **42**(5), 664–681 (2012)
39. Blum, C., Puchinger, J., Raidl, G.R., Roli, A.: Hybrid metaheuristics in combinatorial optimization: a survey. Appl. Soft Comput. **11**(6), 4135–4151 (2011)
40. Ong, Y.-S., Lim, M.-H., Zhu, N., Wong, K.-W.: Classification of adaptive memetic algorithms: a comparative study. IEEE Trans. Syst. Man Cybern. Part B Cybern. **36**(1), 141–152 (2006)
41. Dokeroglu, T., Sevinc, E., Kucukyilmaz, T., Cosar, A.: A survey on new generation metaheuristic algorithms. Comput. Ind. Eng. **137**, 106040 (2019)
42. Eiben, A.E., Smith, J.E., et al.: Introduction to Evolutionary Computing, vol. 53. Springer, Berlin (2003)
43. Sörensen, K.: Metaheuristics' the metaphor exposed. Int. Trans. Opera. Res. **22**(1), 3–18 (2015)
44. Dawkins, R.: The Selfish Gene. Oxford University Press, Oxford (1967)
45. Al-Betar, M.A., Aljarah, I., Awadallah, M.A., Faris, H., Mirjalili, S.: Adaptive β-hill climbing for optimization. Soft Comput. **23**(24), 13489–13512 (2019)
46. Sun, K., Huang, S.-H., Shan-Hill-Wong, D., Jang, S.-S.: Design and application of a variable selection method for multilayer perceptron neural network with lasso. IEEE Trans. Neural Netw. Learn. Syst. **28**(6), 1386–1396 (2016)
47. Al-Betar, M.A.: β-hill climbing: an exploratory local search. Neural Comput. Appl. **28**(1), 153–168 (2017)
48. Al-Betar, M.A., Hammouri, A.I., Awadallah, M.A., Doush, I.A.: Binary β-hill climbing optimizer with s-shape transfer function for feature selection. J. Ambient Intell. Humaniz. Comput. **12**, 7637 (2020)
49. Ahmed, S., Ghosh, K.K., Garcia-Hernandez, L., Abraham, A., Sarkar, R.: Improved coral reefs optimization with adaptive β-hill climbing for feature selection. Neural Comput. Appl. **33**, 6467 (2020)
50. Alweshah, M., Al-Daradkeh, A., Al-Betar, M.A., Almomani, A., Oqeili, S.: β-hill climbing algorithm with probabilistic neural network for classification problems. J. Ambient Intell. Humaniz. Comput. **11**, 3405 (2019)
51. Al-Betar, M.A., Awadallah, M.A., Doush, I.A., Alsukhni, E., ALkhraisat, H.: A non-convex economic dispatch problem with valve loading effect using a new modified β-hill climbing local search algorithm. Arabian J. Sci. Eng. **43**(12), 7439–7456 (2018)
52. Al-Betar, M.A.: A β-hill climbing optimizer for examination timetabling problem. J. Ambient Intell. Humaniz. Comput. **12**, 653–666 (2021)
53. Alsukni, E., Arabeyyat, O.S., Awadallah, M.A., Alsamarraie, L., Abu-Doush, I., Al-Betar, M.A.: Multiple-reservoir scheduling using β-hill climbing algorithm. J. Intell. Syst. **28**(4), 559–570 (2019)
54. Alzaidi, A.A., Ahmad, M., Doja, M.N., Al Solami, E., Beg, M.M.S.: A new 1d chaotic map and β-hill climbing for generating substitution-boxes. IEEE Access **6**, 55405–55418 (2018)

55. Al-Betar, M.A., Awadallah, M.A., Bolaji, A.L., Alijla, B.O.: β-hill climbing algorithm for sudoku game. In: 2017 Palestinian International Conference on Information and Communication Technology (PICICT), pp. 84–88. IEEE (2017)
56. Alomari, O.A., Khader, A.T., Al-Betar, M.A., Awadallah, M.A.: A novel gene selection method using modified mrmr and hybrid bat-inspired algorithm with β-hill climbing. Appl. Intell. **48**(11), 4429–4447 (2018)
57. Abed-alguni, B.H., Alkhateeb, F.: Intelligent hybrid cuckoo search and β-hill climbing algorithm. J. King Saud Univ. Comput. Inf. Sci. **32**(2), 159–173 (2020)
58. Doush, I.A., Santos, E.: Best polynomial harmony search with best β-hill climbing algorithm. J. Intell. Syst. **30**(1), 1–17 (2020)
59. Abasi, A.K., Khader, A.T., Al-Betar, M.A., Alyasseri, Z.A.A., Makhadmeh, S.N., Al-laham, M., Naim, S.: A hybrid salp swarm algorithm with β-hill climbing algorithm for text documents clustering. In: Aljarah, I., Faris, H., Mirjalili, S. (eds.) Evolutionary Data Clustering: Algorithms and Applications, p. 129. Springer, Singapore (2021)
60. Alyasseri, Z.A.A., Khader, A.T., Al-Betar, M.A., Alomari, O.A.: Person identification using EEG channel selection with hybrid flower pollination algorithm. Pattern Recogn. **105**, 107393 (2020)
61. Jarrah, M.I., Jaya, A.S.M., Alqattan, Z.N., Azam, M.A., Abdullah, R., Jarrah, H., Abu-Khadrah, A.I.: A novel explanatory hybrid artificial bee colony algorithm for numerical function optimization. J. Supercomput. **76**, 9330 (2020)
62. Al-Betar, M.A., Awadallah, M.A., Krishan, M.M.: A non-convex economic load dispatch problem with valve loading effect using a hybrid grey wolf optimizer. Neural Comput. Appl. **32**, 12127–12154 (2020)
63. Sun, K., Jia, H., Li, Y., Jiang, Z.: Hybrid improved slime mould algorithm with adaptive β hill climbing for numerical optimization. J. Intell. Fuzzy Syst. **40**(1), 1667–1679 (2021)
64. Sarkar, R.: An improved salp swarm algorithm based on adaptive β-hill climbing for stock market prediction. In: Machine Learning and Metaheuristics Algorithms, and Applications: Second Symposium, SoMMA 2020, Chennai, India, October 14–17, 2020, Revised Selected Papers, vol. 1366, p. 107. Springer (2021)
65. Stathakis, D.: How many hidden layers and nodes? Int. J. Remote Sens. **30**(8), 2133–2147 (2009)
66. Aljarah, I., Faris, H., Mirjalili, S., Al-Madi, N., Sheta, A., Mafarja, M.: Evolving neural networks using bird swarm algorithm for data classification and regression applications. Cluster Comput. **22**(4), 1317–1345 (2019)
67. Yang, X.-S.: Flower pollination algorithm for global optimization. In: International Conference on Unconventional Computing and Natural Computation, pp. 240–249. Springer (2012)
68. Alyasseri, Z.A.A., Khader, A.T., Al-Betar, M.A., Awadallah, M.A., Yang, X.S.: Variants of the flower pollination algorithm: a review. In: Yang, X.S. (ed.) Nature-Inspired Algorithms and Applied Optimization, pp. 91–118. Springer, Cham (2018)
69. Mirjalili, S., Gandomi, A.H., Mirjalili, S.Z., Saremi, S., Faris, H., Mirjalili, S.M.: Salp swarm algorithm: a bio-inspired optimizer for engineering design problems. Adv. Eng. Softw. **114**, 163–191 (2017)
70. Askarzadeh, A.: A novel metaheuristic method for solving constrained engineering optimization problems: crow search algorithm. Comput. Struct. **169**, 1–12 (2016)
71. Mirjalili, S., Mirjalili, S.M., Lewis, A.: Grey wolf optimizer. Adv. Eng. Softw. **69**, 46–61 (2014)
72. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of ICNN'95-International Conference on Neural Networks, vol. 4, pp. 1942–1948. IEEE (1995)
73. Rao, R.: Jaya: a simple and new optimization algorithm for solving constrained and unconstrained optimization problems. Int. J. Ind. Eng. Comput. **7**(1), 19–34 (2016)
74. UCI Machine Learning Repository. https://archive.ics.uci.edu/ml/index.php (2021). Accessed 6 June 2021
75. Wdaa, A.S.I., Sttar, A.: Differential evolution for neural networks learning enhancement. PhD thesis, Universiti Teknologi Malaysia Johor Bahru (2008)
76. Mirjalili, S., Mirjalili, S.M., Lewis, A.: Let a biogeography-based optimizer train your multi-layer perceptron. Inf. Sci. **269**, 188–209 (2014)
77. Cano, J.-R., Garcia, S., Herrera, F.: Subgroup discover in large size data sets preprocessed using stratified instance selection for increasing the presence of minority classes. Pattern Recogn. Lett. **29**(16), 2156–2164 (2008)
78. Srinivasan, P.A., Guastoni, L., Azizpour, H.: PHILIPP Schlatter, and Ricardo Vinuesa Predictions of turbulent shear flows using deep neural networks. Phys. Rev. Fluids. **4**(5), 054603 (2019)
79. Zeng, X., Yeung, D.S.: Sensitivity analysis of multilayer perceptron to input and weight perturbations. IEEE Trans. Neural Netw. **12**(6), 1358–1366 (2001)

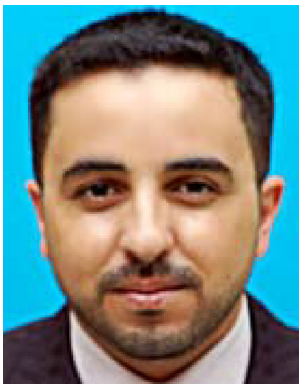


Mohammed Azmi Al-Betar has received his PhD in Artificial Intelligence, 2010 from University of Science Malaysia. He was also haired as a post-doctoral research fellow in the same university for 3 years and invited as a visiting researcher two times. He is currently the Head of Artificial Intelligence Research Center (AIRC) and full time faculty member in the Master of Artificial Intelligence (MSAI) program at Ajman University since 2020. He is also the Head of the Evolutionary Computation Research Group (ECRG) which publish more than 175 scientific publications in high quality and well-reputed journals and conferences. Therefore, Dr. Al-Betar ranked in Stanford study of the world's top 2% of scientists.

Mohammed A. Awadallah is an Associate Professor of artificial intelligence, in the department of computer & information sciences, Al-aqsa University, Gaza, Palestine. He was appointed as adjunct research associate (ARA) at Artificial Intelligence Research Center (AIRC) - Ajman University, United Arab Emirates since February 2021. The main research interest in the area of applied computing where many real-world complex problems such as nurse rostering, timetabling, economic load dispatch, feature selection, gene selection etc are being solve using metaheuristic-based

optimization methods like harmony search, artificial bee colony, grey wolf optimizer, bat algorithm, krill herd algorithm and so on.

Iyad Abu Doush is an Associate Professor in the department of Computer Science and Information Systems at American University of Kuwait. He obtained his PhD from the Computer Science Department at New Mexico State University / USA in 2009. Dr. Abu Doush completed his B.Sc. in computer science from Yarmouk University, Jordan, and his M.Sc. in Computer Science and Information Systems from Yarmouk University, Jordan. Dr. Abu Doush has supervised, advised and referred senior projects, master theses and number of journals. Dr. Abu Doush served as coach and committee member in the ACM Jordanian Collegiate Programming Contest for three years. Dr. Abu Doush has been funded several times to conduct research in his areas of expertise from different agencies including: USAID, Microsoft, King Abdullah II Design & Development Bureau, Deanship of Research & Graduate Studies at Yarmouk University and Jordanian Scientific Research Support Fund. Dr. Abu Doush has published more than 40 articles in international journals and conferences. Dr. Abu Doush was selected to serve as a visiting researcher in universities of Malaysia and Lithuania. His research interests include: Evolutionary Algorithms, Optimization, Accessibility, and Human Computer Interaction.



Osama Ahmad Alomari received his M.Sc. in computer science from the Universiti Kebangsaan Malaysia (UKM), Malaysia in 2014. He is also pursued a PhD from the school of Computer Sciences at Universiti Sains Malaysia. His research interests include evolutionary algorithms, nature-inspired computation, and their applications to optimization problems. Now he is employee at MLALP Research Group, University of Sharjah, Sharjah, United Arab Emirates.

Ammar Kamal Abasi received his PhD from the school of Computer Sciences at Universiti Sains Malaysia. His research interests include evolutionary algorithms, nature-inspired computation, and their applications to optimization problems.

Sharif Naser Makhadmeh received his PhD from the school of Computer Sciences at Universiti Sains Malaysia. He is assistant professor at College of Engineering and Information Technology, Ajman University, Ajman, United Arab Emirates. His research interests include evolutionary algorithms, nature-inspired computation, and their applications to optimization problems.

Zaid Abdi Alkareem Alyasseri received the B.Sc. degree in computer science from Babylon University, Babylon-Iraq, in 2007, and the M.Sc. degree in computer science from University Science Malaysia (USM), Penang-Malaysia, in 2013, and the Ph.D. from USM in the field of Artificial Intelligence (Brain-Inspired Computing) in 2019. His main research interests include Optimization, Pattern recognition, EEG, Brain-Computer Interface, Signal and Image Processing, Machine and Deep Learning.

Authors and Affiliations

Mohammed Azmi Al-Betar[1,2] · Mohammed A. Awadallah[3,4,12] · Iyad Abu Doush[5,6] · Osama Ahmad Alomari[7] · Ammar Kamal Abasi[8] · Sharif Naser Makhadmeh[1] · Zaid Abdi Alkareem Alyasseri[9,10,11]

✉ Mohammed Azmi Al-Betar
mohbetar@bau.edu.jo

Mohammed A. Awadallah
ma.awadallah@alaqsa.edu.ps

Iyad Abu Doush
idoush@auk.edu.kw

Osama Ahmad Alomari
oalomari@sharjah.ac.ae

Ammar Kamal Abasi
ammar.abasi@mbzuai.ac.ae

Sharif Naser Makhadmeh
s.makhadmeh@ajman.ac.ae

Zaid Abdi Alkareem Alyasseri
zaid.alyasseri@uokufa.edu.iq

[1] Artificial Intelligence Research Center (AIRC), College of Engineering and Information Technology, Ajman University, Ajman, United Arab Emirates

[2] Department of Information Technology, Al-Huson University College, Al-Balqa Applied University, Irbid, Jordan

[3] Department of Computer Science, Al-Aqsa University, P.O. Box 4051, Gaza, Palestine

[4] Artificial Intelligence Research Center (AIRC), Ajman University, Ajman, United Arab Emirates

[5] Department of Computing, College of Engineering and Applied Sciences, American University of Kuwait, Salmiya, Kuwait

[6] Computer Science Department, Yarmouk University, Irbid, Jordan

[7] MLALP Research Group, University of Sharjah, Sharjah, United Arab Emirates

[8] Machine Learning Department, Mohamed Bin Zayed University of Artificial Intelligence (MBZUAI), Abu Dhabi, United Arab Emirates

[9] Information Technology Research and Development Center (ITRDC), University of Kufa, Najaf, Iraq

[10] Department of Business Administration, College of Administration and Financial Sciences, Imam Ja'afar Al-Sadiq University, Baghdad, Iraq

[11] ECE Department, Faculty of Engineering, University of Kufa, Najaf, Iraq

[12] Faculty of Science and Information Technology, Al-Zaytoonah University of Jordan, Amman, Jordan