Neural Computing and Applications (2023) 35:15923–15941
https://doi.org/10.1007/s00521-023-08577-y

ORIGINAL ARTICLE

Archive-based coronavirus herd immunity algorithm for optimizing weights in neural networks

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

Received: 9 July 2022 / Accepted: 5 April 2023 / Published online: 19 April 2023

Abstract
The success of the supervised learning process for feedforward neural networks, especially multilayer perceptron neural network (MLP), depends on the suitable configuration of its controlling parameters (i.e., weights and biases). Normally, the gradient descent method is used to find the optimal values of weights and biases. The gradient descent method suffers from the local optimal trap and slow convergence. Therefore, stochastic approximation methods such as metaheuristics are invited. Coronavirus herd immunity optimizer (CHIO) is a recent metaheuristic human-based algorithm stemmed from the herd immunity mechanism as a way to treat the spread of the coronavirus pandemic. In this paper, an external archive strategy is proposed and applied to direct the population closer to more promising search regions. The external archive is implemented during the algorithm evolution, and it saves the best solutions to be used later. This enhanced version of CHIO is called ACHIO. The algorithm is utilized in the training process of MLP to find its optimal controlling parameters thus empowering their classification accuracy. The proposed approach is evaluated using 15 classification datasets with classes ranging between 2 to 10. The performance of ACHIO is compared against six well-known swarm intelligence algorithms and the original CHIO in terms of classification accuracy. Interestingly, ACHIO is able to produce accurate results that excel other comparative methods in ten out of the fifteen classification datasets and very competitive results for others.

Keywords Coronavirus herd immunity optimizer · CHIO · Optimization · Feedforward neural networks · MLP · Archive technique

1 Introduction

Artificial neural network (ANN) is considered an intelligent mathematical model inspired by the neurons in the biological brain where the connections between neurons exchange signals to communicate their data [1]. In machine learning, the main applications of ANN are feature extraction, classifications, predictions and regressions problems [2–4]. Since their establishment in 1943 [5], ANNs different types have been developed including radial basis function (RBF) network [6], Feedforward neural network [7], convolutional neural network [8], recurrent neural network [9], and spiking neural networks [10]. The main difference between these types is the learning process. Normally, the learning process is either supervised, where ANN takes feedback from the outsource, or unsupervised, where the model discovers hidden patterns in the data by itself [11].

The multilayer perceptron (MLP) neural network is one of the most popular feedforward versions of the ANN that has been applied successfully for several classification problems [12–14]. This is because of its success in the learning process during the training stage. The MLP normally used a supervised method based on the backpropagation principle to accomplish accurate training which adjusts the weights and biases of the MLP through at least three layers (i.e., input, hidden, and output). The backpropagation algorithm is a well-known gradient decent technique [15]. Generally speaking, gradient descent techniques suffer from chronic problems related to the slow convergence and local optima trap [16, 17]. In order to

Extended author information available on the last page of the article

overcome such shortcomings, stochastic methods such as metaheuristic-based techniques came to the fore [18].

Metaheuristic-based techniques such as evolutionary algorithms and swarm-based algorithms can greatly support the MLP by accelerating the convergence as well as avoiding becoming trapped in local optima. There is a wide range of metaheuristic-based approaches used in the training process of MLP. The earliest methods are genetic algorithm (GA) [19], particle swarm optimization (PSO) [20], and differential evolution (DE) [21, 22]. Nowadays, a plethora of recent metaheuristic-based algorithm are being used for MLPs with a very successfully outcomes such as gray wolf optimizer [18, 23], salp swarm algorithm [24], glowworm swarm optimization [25], grasshopper optimization algorithm [26, 27], artificial bee colony [28], butterfly optimization algorithm [29], monarch butterfly optimization [30], social spider optimization algorithm [31], dragonfly algorithm [28], fish swarm algorithm [32], ant colony optimization [33], bat algorithm [34], biogeography-based optimization [35], gravitational search algorithm [36], krill herd algorithm [37], ant lion optimizer [38], cuckoo search algorithm [39], organisms search algorithm [40], and lightning search algorithm [41].

According to the no free lunch (NFL) theorem for optimization [42], there is no superior algorithm that can perform well and excel others for all optimization problems or even for different instances of the same optimization problem. Therefore, there is still a window for improving the MLP performance by investigating other state-of-the-art metaheuristic-based methods to function as a training method in MLP. Quite recently, a new human-based metaheuristic algorithm called coronavirus herd immunity optimizer (CHIO) has been proposed for global optimization problems [43]. The main idea of CHIO is inspired from herd immunity as a strategy to confront the pandemic. CHIO can be considered as an evolutionary algorithm which is initiated with a population of random individuals. The population has three types of individual cases: susceptible, infected, and recovered. During the improvement loop, the susceptible case can be infected based on their inherited attributes. Also, the infected cases can be either recover or die based on their improvement over a specific period (i.e., specific iterations or can be called age). The recovered cases which are the highly immuned cases are stronger than the other cases and stand as a shield to stop the pandemic. CHIO algorithm will stop when the whole population is immune based on the herd immunity strategy. The main advantage of using CHIO to tackle any optimization problem is its ability to be adapted to the optimization problem without prior-knowledge or derivative information in the initial search. It is very simple and easy-to-use as a black-box. Recently, CHIO has been successfully applied for solving several problems such as traveling salesman problems [44] and wheel motor design [45].

Archive methods have been introduced by many researchers to promote the population diversity during evolution and to store the potential optima. For example, Lacroix et al. [46] introduced an archive method to store the best-known solution in the evolving population into one collection. The collection is used as an indexer for searching space regions to identify the weak regions for exploration. Later, these regions are stored in another collection. During the evolution, the two collections are continuously updated. Archive method for sub-populations was introduced in Zhang et al. [47], Kundu et al. [48] to undergo regeneration that will eventually form an initial population of solutions for the evolving population. In [49], an archive method is implemented to store stagnant solutions. In this method, the detected stagnant solution will be reinitialized, along with its neighbors whose fitness is lower than the stagnant individual. Additionally, Sheng et al. [50], Turky and Abdullah [51] utilizes the archive to solve dynamic optimization problems. The archive aims to improve the population evolution and maintain the best potential solutions for subsequent cycles. The findings demonstrate that the method has achieved better performance than other methods in terms of locating several optimal solutions in the problem search space reliably. The external archive is used by other researchers to tackle multi-objective optimization problems [52, 53]. Such modification can increase the methods' exploration capabilities by speeding up the convergence toward their optimal/near-optimal solutions.

To improve the exploration capabilities of CHIO an archive that saves the best results is implemented. The use of archive [46, 49, 54] can help in enhancing the ability to search into more promising regions. The modified CHIO is called archive-CHIO (ACHIO). The proposed algorithm is used to improve the performance of MLP in training a single hidden layer neural network. ACHIO is used to a good initial sets of weights and biases to train MLP efficiently. In order to measure the performance of the proposed ACHIO-MLP system, the mean square error (MSE) is used as an accuracy measurement [24, 55–57]. The proposed ACHIO-MLP system is evaluated using 15

classification datasets of various complexity. The proposed algorithm is evaluated against the original CHIO and six well-known swarm intelligence algorithms (Artificial Bee Colony (ABC), Bat Algorithm (BA), Flower Pollination Algorithm (FPA), Particle Swarm Optimization (PSO), Sine Cosine Algorithm (SCA), and Harmony Search (HS)).

The remaining parts of the present paper are arranged as follows: the feedforward neural network (FNN) is discussed in Sect. 2. The proposed ACHIO-MLP is thoroughly described in Sect. 3. The experimental results and their discussions are provided in Sect. 4. Finally, the paper is concluded and the possible future developments are given in Sect. 5.

2 Feedforward neural networks

A feedforward neural networks (FFNN) is computational learning algorithm inspired by the processing units of the human brain. These processing units are called neurons; they are interconnected and grouped in three layers. The first layer, the input layer, is composed of the same number of neurons as the number of the input features corresponding vector, the middle layers are called hidden layers, and the last layer is the output layer that consists of output neurons to the predicted class labels [58]. Multilayer perceptron (MLP) is a FFNN model whose architecture is formed by neurons interconnected in layers. Through these connections, the information flows one-way. Figure 1 shows the network structure of an MLP with only one hidden layer. The mathematical model of an MLP is based on three factors: input data, weights, and biases. These factors are applied in three steps in order to calculate the output of MLPs as follows:

1. Initially, the weighted sum of the input is calculated using Eq. (1).

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

where n represents the number of the input nodes in the network, w_{ij} represents the weight on the connections between the input node i and hidden node j, X_i is the ith input, and β_j is the bias of the jth hidden node.

2. In this step, an activation function (e.g., Sigmoid, which is commonly used in MLPs) is adopted to transfer the weighted output in the hidden layer to the next layer as follows:

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

3. Finally, the last output of the network is computed as follows:

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

where w_{jk} is the weighted edge connecting hidden node j to the output node k, and the bias of the output node k is b_k.

As observed from Eqs. (1) and (3), the weights and biases are primary factors for computing the final output in MLPs. Having robust training for MLPs entails seeking the proper values for both weights and biases [23]. In the next sections, the CHIO algorithm is adapted as a trainer for MLPs.

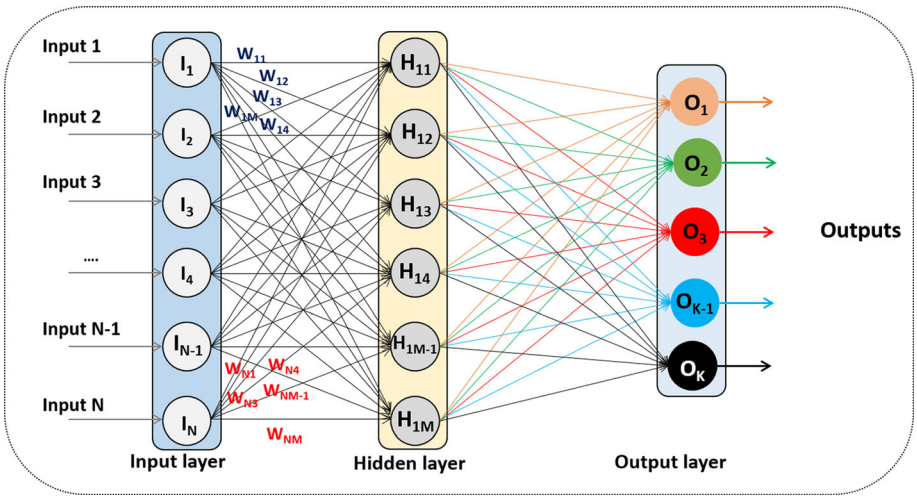

Fig. 1 Network structure of MLPs with only single one hidden layer

3 Archive-based coronavirus herd immunity optimizer for MLP training

Coronavirus herd immunity optimizer (CHIO) is a recent nature-inspired human-based optimization algorithm proposed by Al-betar et al. [43]. CHIO imitates how herd immunity can be utilized to confront the COVID-19 pandemic. The algorithm proves its effectiveness when compared with other comparative methods to tackle optimization problems [59, 60].

In CHIO, the total population is divided into three sub-populations. The susceptible case sub-populations are the solutions not infected by the virus, so they can be infected. The infected case sub-populations have the solutions where they are changed from being susceptible to being infected after they inherit values from an infected case. The immuned case sub-populations have solutions which are the cases that survived after being infected. They are the strongest portion of the population who are not affected by the infected cases.

If a susceptible individual takes attributes from an infected case and is not immuned, he will become also infected (see Eq. (14) and lines 44–45 of Algorithm 1). The individual will stay in this status until MaxAge is reached where the case becomes either immuned (recovered) or died (see lines 65-72 of Algorithm 1). Any susceptible individual who takes attributes from the infected person will become also infected. The contagion is possible only for the susceptible individual. As the infected case will become immuned (recovered) or it will die after reaching MaxAge, but the immuned case cannot become infected. Note that if a susceptible individual takes attributes from an immuned individual his status will not change (i.e., will still be susceptible) as shown in lines 57-60 of Algorithm 1.

The external archive is used to improve CHIO performance by saving the best solutions to be used in the next iterations. This enhanced version of CHIO is called ACHIO. This section describes how ACHIO can be used to train an MLP. As mentioned before, weights and biases are the variables in MLP neural networks. The proposed technique uses ACHIO to optimize the selected weights and biases of the MLP neural networks by choosing the MLPs that obtain the highest classification accuracy. In ACHIO-MLP, the MLP is used in each iteration in the process of evaluating the current solutions where the weights and biases are the input vectors.

The archive rate (A_r) indicates the percentage of the best solutions from the population that will be used (i.e., the best weights and biases for the fittest MLP). These best solutions as well as are stored in an external archive to be used as a part of the initial population in the following run. This allows ACHIO-MLP to make use of the best-obtained solutions so far, during the algorithm evolution. This process is repeated in each run. Since the initial run has no feedback from the historical runs, the population is randomly constructed in the first run. The archive will be utilized after the first run.

The procedural steps of ACHIO-MLP to train a NN are presented next. Figure 2 presents the steps of the ACHIO-MLP algorithm. Furthermore, the pseudo-code of ACHIO-MLP is given in Algorithm 1.

The proposed ACHIO-MLP algorithm has eight main steps as follows:

Step 1: Define the external archive The external archive is a matrix (*ARCH*) of size $K \times N$ (see Eq. (4)) where N is the total number of weights and biases in the solution vector while K is the number of best MLPs selected after each training session for copying to the archive. K is set in Eq. (4) by the ratio A_r. A_r is a parameter of the ACHIO algorithm that is chosen in a preliminary experiment (see Sect. 4.3). Note that in the first run, the *ARCH* will be empty and it will be updated after the first run.

$$\mathbf{ARCH} = \begin{bmatrix} w_1^1 & \cdots & w_n^1 & b_1^1 & \cdots & b_m^1 \\ w_1^2 & \cdots & w_n^2 & b_1^2 & \cdots & b_m^2 \\ \vdots & \vdots & \cdots & \vdots & & \\ w_1^K & \cdots & w_n^K & b_1^K & \cdots & b_m^K \end{bmatrix}. \tag{4}$$

$$K = \text{HIS} \times A_r \tag{5}$$

where the A_r is the archive rate which determines the rate of extracting the best solutions from the previous run. In other words, Ar is the ratio of MLPs selected from the training set of MLPs for the next version of the archive. The population size (i.e., *HIS*) determines the number of solutions where each solution consists of a vector of weights and biases for each MLP. Note that the archive size depends on the population size using the archive rate (A_r). Note that *ARCH* is only constructed at the beginning of the execution and it keeps updating after each run.

Step 2: setting ACHIO and MLP parameters In MLP neural network, the optimization problem can be represented using a one-dimensional vector that is the set of weights and biases that needs to be adjusted to increase the fitness of an MLP to the data. The number of weights and biases in the vector can be calculated using Eq. (6):

$$h = 2 \times F + 1; \tag{6}$$

$$n1 = N \times h + h \times o \tag{7}$$

$$b = h + o \tag{8}$$

where h is the number of neurons in the hidden layer, F is the number of features in the dataset, $n1$ is the number of

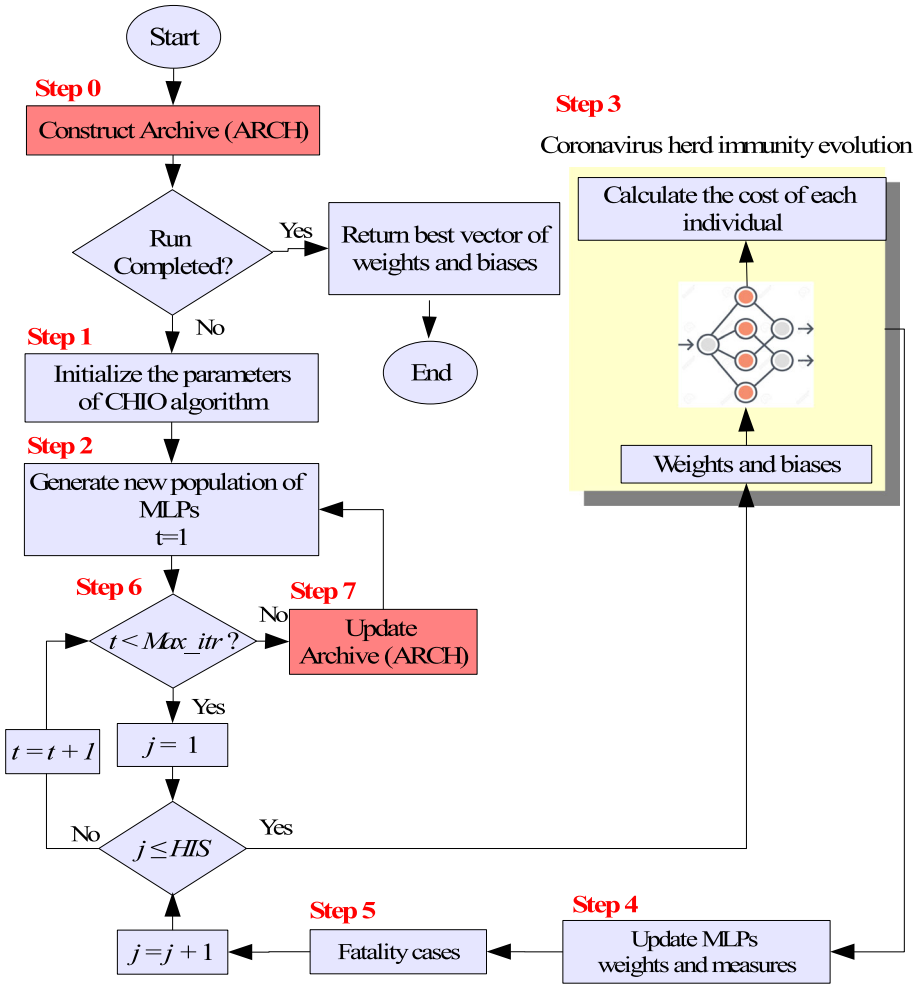

Fig. 2 The steps of the proposed ACHIO-MLP

weights, o is the number of outputs from the MLP neural network, and b is the number of biases.

The resulting MLP is applied to all instances in the dataset, and it is measured using mean square error (MSE). The MSE is the difference between the MLP output and the actual data. MSE is a common metric used that calculates the difference between the actual value and the predicted value. The equation for MSE is demonstrated in Eq. (9). Note that y represents the actual value, $\hat{y}$ represents the predicted value, and k represents the number of training samples.

$$\mathrm{MSE} = \frac{1}{k}\sum_{i=1}^{k}(y-\hat{y})^2 \tag{9}$$

The $\hat{y}$ is predicted based on feeding the current weights and biases to the MLP and identifying the correct class label for each data input. The output of MLP is compared against the actual data output to identify the quality of the MLP (i.e., the MSE against the data). In the training phase, the *MSE* value is the difference between the actual outcomes and the predicted outcomes by the generated MLPs by the proposed ACHIO-MLP algorithm.

Note that the size of the output prediction vector produced by the neural network will be depending on the number of classes. For example, in case we have 3 classes then we have an output vector of size 3 from the neural network. Now the predicted class will have the larger value of the 3 values in the vector. Assuming that the correct class is the first one and the neural network predicted it to be the second class. Then when calculating MSE the predicted value (i.e., $\hat{y}$) used for the first and the third classes will be zero. On the other hand, the actual value (i.e., y) for the second class and the third class will be zero.

In general, the classification problem can be modeled as follows:

$$\min_{x} f(\boldsymbol{x}) \quad \boldsymbol{x} \in [\boldsymbol{lb}, \boldsymbol{ub}] \tag{10}$$

where $f(\boldsymbol{x})$ is MSE value (i.e., the objective function that must be lowered) which is evaluated for the case $\boldsymbol{x} = (x_1, x_2, \ldots, x_N)$. Note that for MLP, the vector $\boldsymbol{x}$ has two parts which are the weights (n) and biases (m) such that

$\boldsymbol{x} = (w_1, w_2, \ldots, w_n, b_1, b_2, \ldots, b_m)$. Here, x_i is the decision variable indexed by i, and N is the total number of decision variables in each individual which is $N = n + m$. The weights and biases values for the MLP are within the interval $\in [lb_i, ub_i]$ where lb_i and ub_i are the smallest and highest limits of the variable x_i (i.e., the acceptable range for MLP weights and biases).

There are five algorithmic parameters for ACHIO-MLP which are as follows:

- C_0 represents how many infected cases we have initially, normally set to be one.
- Max_{Itr} represents the maximum number of iterations.
- HIS is the population size (i.e., the weights and biases vectors of MLPs).
- N represents the number of variables in the solution (i.e., the number of weights and biases for each MLP).
- A_r represents the archive rate which is the percentage of re-using the best solutions from the previous run.

In addition, ACHIO-MLP has two control parameters:

- Basic reproduction Rate (*BR*): which identifies the speed of virus spreading among individuals, where it assigned a random value in the range of [0, 1]. In other words, *BR* is the average proportion of MLP's weights and biases that are changed toward the corresponding weight or bias of another MLP at each time step (see Eq. (19)).
- Maximum infected cases age (Max_{Age}): when an infected (S = 1) (but not immune) case's age reaches Max_{Age} it will either die (i.e. removed from the population of MLPs being trained) or become immune (S = 2), depending on its MSE value compare to the average MSE value of all the MLPs in the population being trained (see Eq. (19)).

Step 3: Produce the population for MLP configuration Firstly, in the first run, ACHIO-MLP generates *HIS* random solutions and stores them in CHIO memory (*CHIOM*) as shown in Eq. (11) where in the consecutive run, the ACHIO-MLP will generate HIS $- K$ random solutions and K solution will be taken from *ARCH*. Each solution represents possible weights and biases as input for MLP. Each solution is a vector $\boldsymbol{x} = (w_1, w_2, \ldots, w_n, b_1, b_2, \ldots, b_m)$.

$$\textbf{CHIOM} = \begin{bmatrix} w_1^1 & \cdots & w_n^1 & b_1^1 & \cdots & b_m^1 \\ w_1^2 & \cdots & w_n^2 & b_1^2 & \cdots & b_m^2 \\ \vdots & \vdots & \cdots & \vdots & & \\ w_1^{\text{HIS}} & \cdots & w_n^{\text{HIS}} & b_1^{\text{HIS}} & \cdots & b_m^{\text{HIS}} \end{bmatrix}. \tag{11}$$

where each row represents a solution $\boldsymbol{x}^j$ which is a set of weights and biases. The solution is generated as follows: $x_i^j = lb_i + (ub_i - lb_i) \times U(0,1), \forall i = 1, 2, \ldots, N$. The cost function is computed using MSE as presented in Eq. (9). It is worth mentioning that after the first run, some solutions K are copied from the archive *ARCH* directly and the remaining solutions are constructed randomly to fill up *CHIOM*.

For simplicity, the term x_i^j is used next to refer to the variable i (weight or bias) of a solution vector j.

Step 4: The progress of herd immunity The ACHIO-MLP algorithm is used to improve the weights and biases of all MLP in the current population. Note that the current population is improved including the archive added from the previous population. The weight or bias (x_i^j) (i.e., $x_i^j = w_j^i$ or $x_i^j = b_j^i$) for the individual $\boldsymbol{x}^j$ stored in *CHIOM* would be changed or not by applying the following three social distancing rules based on the *BR* ratio:

$$x_i^j(t+1) \leftarrow \begin{cases} C(x_i^j(t)) & r \in \left[0, \frac{1}{3}BR\right). \quad \text{//infected case} \\ N(x_i^j(t)) & r \in \left[\frac{1}{3}BR, \frac{2}{3}BR\right). \quad \text{//susceptible case} \\ R(x_i^j(t)) & r \in \left(\frac{2}{3}BR, BR\right]. \quad \text{//immuned case} \\ x_i^j(t) & r \geq BR \end{cases} \tag{12}$$

Note that r is a random number within the range [0,1]. The following is how the weights and biases of an MLP change depending on other MLPs:

Infected case : in case $r \in [0, \frac{1}{3}BR)$, the new weight or bias value $x_i^j(t+1)$ would be based on a previous value of an infected case $\boldsymbol{x}^c$ computed as follows:

$$x_i^j(t+1) = C(x_i^j(t)) \tag{13}$$

where

$$C(x_i^j(t)) = x_i^j(t) + r \times (x_i^j(t) - x_i^c(t)) \tag{14}$$

where $x_i^c(t)$ is from a randomly chosen infected case $\boldsymbol{x}^c$.

Susceptible case: in case $r \in [\frac{1}{3}BR, \frac{2}{3}BR)$ then the new weight or bias value $x_i^j(t+1)$ would be based on a previous susceptible case $\boldsymbol{x}^m$ as follows:

$$x_i^j(t+1) = N(x_i^j(t)) \tag{15}$$

where

$$N(x_i^j(t)) = x_i^j(t) + r \times (x_i^j(t) - x_i^m(t)) \tag{16}$$

where $x_i^m(t)$ is from a randomly chosen susceptible case $\boldsymbol{x}^m$.

Immuned case: in case $r \in [\frac{2}{3}BR, BR)$, the new weight or bias value $x_i^j(t+1)$ would be based on a previous immuned case $\boldsymbol{x}^v$ as follows:

$$x_i^j(t+1) = R(x_i^j(t)) \tag{17}$$

where

$$R(x_i^j(t)) = x_i^j(t) + r \times (x_i^j(t) - x_i^v(t)) \tag{18}$$

Note that $x_i^v(t)$ is from a randomly chosen immuned case $\boldsymbol{x}^v$.

$$f(x^v) = \arg \min_{j\sim \{k|\mathcal{S}_k=2\}} f(x^j).$$

The weights and biases of $x_i^j(t+1)$ are used as input parameters for MLP. The obtained result of MLP is used in MSE which is a common metric used to evaluate the performance of the obtained result. The objective here is to find the set of weights and biases that minimize MSE using the training instances from the selected dataset.

It is worth mentioning that in each CHIO operator, the next value of any variable is calculated based on the original value plus a small distance between the current value and a randomly chosen variable value from a solution with the same type.

Step 5: Refreshing the population The cost function $f(\boldsymbol{x}^j(t+1))$ of the newly generated weights and biases vector, $\boldsymbol{x}^j(t+1)$, is computed. Then, it will replace the current case $\boldsymbol{x}^j(t)$ if better, such as $f(\boldsymbol{x}^j(t+1)) < f(\boldsymbol{x}^j(t))$ then the age vector $\mathcal{A}_j$ would be incremented one step.

For each case $\boldsymbol{x}^j$, the status value ($\mathcal{S}_j$) is modified according to the herd immune threshold represented in Eq. (19).

$$\mathcal{S}_j \leftarrow \begin{cases} 1 & f(\boldsymbol{x}^j(t+1)) < \dfrac{f(\boldsymbol{x})^j(t+1)}{\triangle f(\boldsymbol{x})} \wedge \mathcal{S}_j = 0 \wedge is\_\text{Corona}\,(\boldsymbol{x}^j(t+1)) \\ 2 & f(\boldsymbol{x}^j(t+1)) > \dfrac{f(\boldsymbol{x})^j(t+1)}{\triangle f(\boldsymbol{x})} \wedge \mathcal{S}_j = 1 \end{cases} \tag{19}$$

Note that $is\_\text{corona}\,(\boldsymbol{x}^j(t+1))$ symbolizes a binary value that is equal to one if the newly generated case $\boldsymbol{x}^j(t+1)$ was based on any infected case. Additionally, $\triangle f(\boldsymbol{x})$ is the mean value of the population immune rates which is defined as $\frac{\sum_{i=1}^{\text{HIS}} f(x_i)}{\text{HIS}}$. Note that each MLP's status value indicates its current state; for MLP_j its status is $S_j \in \{0, 1, 2\}$ where $S_j = 0$ indicates a susceptible case, $S_j = 1$ indicates an infected case, and $S_j = 2$ indicates an immuned case. The status of an MLP can change at any iteration of the training, see Step 6 below.

Step 6: Fatality cases The MLP becomes dead if it is an infected case ($\mathcal{S}_j == 1$) and its immunity rate ($f(\mathbf{x}^j(t+1))$ did not improve over a predefined number of trials determined by *age* comparable to the maximum age Max_{Age} (i.e., $\mathcal{A}_j \geq \text{Max}_{\text{Age}}$). In such a situation, the case is reconstructed as a new solution by applying $x_i^j(t+1) = lb_i + (ub_i - lb_i) \times U(0,1), \quad \forall i = 1, 2, \ldots, N$. The algorithm performs that to diversify its population (i.e., weights and biases).

Step 7: Stop and test Steps 4 to 6 are replicated until we reach the maximum number of iterations. After that, the MLP with the lowest MSE value is tested with the test dataset. In this study, all datasets are split into 30% for testing and 70% for training.

Step 8: Update the external archive At the end of each run, the solutions (i.e., the vector of weights and biases for each MLP) in the population are arranged in ascending order according to MSE values. The archive is cleared, the best A_r solutions are copied to a new *ARCH*. Even though the archive (ARCH) is emptied at the start of each training run, its K MLPs are included in the new population of MLPs to be trained, and the best A_r of the new population of MLPs after training are copied to a new version archive (ARCH), so the archive can be considered the store of accumulated knowledge.

Algorithm 1 ACHIO-MLP pseudo-code

Construct the archive $ARCH$ matrix of size $K \times N$, K=$HIS \times A_r$.
Initialize the parameters of CHIO (HIS, S_r, C_0=1, Max_{Age}, h, w, and b).
for i = 1 to $w + b$ **do**
  $lb_i = -10$
  $ub_i = 10$
end for
for run = 1 to $\#runs$ **do**
  if run==1 **then**
    for j = 1 to HIS **do**
      for i = 1 to N **do**
        $x_i^j = lb_i + (ub_i - lb_i)$ {Generate herd immunity population: first run}
      end for
      Calculate the cost value $f(x^j)$
      if j<=C0 **then**
        Set $\mathcal{S}_j$ = 1
      else
        Set $\mathcal{S}_j$ = 0
      end if
      Set $\mathcal{A}_j$ = 0
    end for
  else
    copy the K solutions from $ARCH$.
    for j = 1 to $HIS - K$ **do**
      for i = 1 to $w + b$ **do**
        $x_i^j = lb_i + (ub_i - lb_i)$ {Generate herd immunity population after first run}
      end for
      Calculate the cost value $f(x^j)$
      Set $\mathcal{S}_j$ = 0
      Set $\mathcal{A}_j$ = 0 {$\mathcal{A}_j$ is the age which is the number of iterations of the MLP learning CHIO algorithm }
    end for
  end if
  for c = 1 to C_0 **do**
    $j \in [1, HIS]$
    Set $\mathcal{S}_j$ = 1 {Set infected cases in the initial population}
  end for
  $t = 1$
  while ($t \leq Max_{itr}$) **do**
    for j = 1 to HIS **do**
      $is\_Corona(\boldsymbol{x}^j(t+1))$ = false
      for i = 1 to N **do**
        if ($r < \frac{1}{3} \times BR$) **then**
          $x_i^j(t+1)$ is assigned value using Eq. (14) {Infected case}
          $is\_Corona(\boldsymbol{x}^j(t+1))$ = true
        else if ($r < \frac{2}{3} \times BR$) **then**
          $x_i^j(t+1)$ is assigned value using Eq. (16)) {Susceptible case}
        else if ($r < BR$) **then**
          $x_i^j(t+1)$ is assigned value using Eq. (18) {Immuned case}
        else
          $x_i^j(t+1) = x_i^j(t)$ {keeping the previous value}
        end if
      end for
      if ($f(\boldsymbol{x}^j(t+1)) \leq f(\boldsymbol{x}^j(t))$) **then**
        $\boldsymbol{x}^j(t) = \boldsymbol{x}^j(t+1)$ {Update herd immunity population}
      else
        $\mathcal{A}_j = \mathcal{A}_j + 1$
      end if
      $\triangle f(\boldsymbol{x}) = \frac{\sum_{i=1}^{HIS} f(x_i)}{HIS}$ {Mean value of the population immune rates}
      if $f(\boldsymbol{x}^j(t+1)) < \frac{f(\boldsymbol{x}^j(t+1))}{\triangle f(\boldsymbol{x})} \wedge \mathcal{S}_j = 0 \wedge is\_Corona(\boldsymbol{x}^j(t+1))$ **then**
        $\mathcal{S}_j$ = 1
        $\mathcal{A}_j$ = 1
      end if
      if $f(\boldsymbol{x}^j(t+1)) > \frac{f(\boldsymbol{x}^j(t+1))}{\triangle f(\boldsymbol{x})} \wedge \mathcal{S}_j = 1$ **then**
        $\mathcal{S}_j$ = 2
        $\mathcal{A}_j$ = 0
      end if
      if (($\mathcal{A}_j \geq Max_{Age}$) $\wedge$ ($\mathcal{S}_j$ = 1)) **then**
        for i = 1 to N **do**
          $x_i^j = lb_i + (ub_i - lb_i)$ {Fatality condition}
        end for
        $\mathcal{S}_j$ = 0
        $\mathcal{A}_j$ = 0
      end if
    end for
    $t = t + 1$
  end while
  Update the Archive by feeding the $ARCH$ by the best solutions from the current run.
end for{End of Runs}

4 Experiments and results

In this section, the effectiveness and robustness of the proposed ACHIO-MLP algorithm are studied using 15 benchmark datasets with different levels of complexity. The characteristics of these datasets are presented in Sect. 4.1. The experimental settings are demonstrated in Sect 4.2. The influence of the archive rate parameter on the performance of the ACHIO-MLP is studied in Sect. 4.3. Finally, the performance of the proposed ACHIO-MLP against the classical CHIO-MLP and six other meta-heuristic algorithms in terms of classification accuracy and algorithm convergence are discussed and analyzed in Sect. 4.4.

4.1 Test datasets

The effectiveness of the proposed ACHIO-MLP is investigated using a set of experiments by utilizing 15 benchmark classification problems. These problems are selected from the UCI Machine Learning Repository.[1] The number of classes, features, and instances (or samples) of these datasets are presented in Table 1. The selected benchmark datasets have different numbers of classes, i.e. 2, 4, 6, or 10 classes.

The datasets are normalized by applying min-max normalization to improve the performance and training stability of the model. The following is the mathematical formula used to reduce the scale of the features:

$$x' = \frac{x_i - \min_F}{\max_F - \min_F} \tag{20}$$

where x' is the normalized value of x in the range $[\min_F, \max_F]$.

In the last two columns of Table 1, the number of nodes in the hidden layer and the MLP structure is presented. The number of nodes in the hidden layer can be determined using different techniques. In this paper, we followed the method presented in [61, 62] in which the number of neurons in the hidden layer can be identified using the formula demonstrated in Eq. (21).

$$h = 2 \times L + 1; \tag{21}$$

where L represents the number of features in the dataset. Therefore, the whole MLP structure of each dataset is presented in the form of input-hidden-output. For example, in the Monk dataset, the MLP structure is 6-13-2 where the number of input features is 6, the number of nodes in the hidden layers is 13, and the number of output class labels is 2.

[1] https://archive.ics.uci.edu/ml/index.php.

All datasets are split into 30% for testing and 70% for training. We used a *stratified sampling* to split the dataset [63]. This technique computes the ratio of each class and then satisfies the train/test split percentage for each dataset based on the computed ratio. Using this strategy can help in maintaining the proportion of each class in the divided data and in increasing the presence of minority classes. Such that the train/test portions will have a balanced number of classes.

4.2 Experimental settings

The proposed algorithm is compared against six swarm intelligence algorithms using the same datasets. All experiments are conducted using a Microsoft Azure server with MATLAB version 9.7.0 on a PC with Windows operating system, Intel R Xeon Silver 1.8 GHz CPU, and 6 GB of RAM. The algorithms are implemented for each dataset over 30 independent runs and the number of iterations is 250. The size of the population of MLPs to be trained for all comparative algorithms is set to 70. The parameter settings of all comparative methods are given in Table 2. These parameters are set based on the recommendation given by researchers of their original papers. Note that the proposed ACHIO-MLP is compared with other algorithms in terms of classification accuracy. Classification accuracy represents the number of correct predictions from all predictions made.

4.3 Study the influence of archive rate

The influence of using various settings of the archive rate (A_r) parameter on the performance of the proposed ACHIO-MLP is studied in this section. Note that each preliminary experiment divided the data 70% for training and 30% for testing. Three different A_r values are considered $A_r \in \{0.1, 0.2, 0.5\}$. It should be noted that the higher value of A_r leads to a higher rate of exploration.

It is worth mentioning that for choosing Ar, the data are divided into 70% for training and 30% for testing in ACHIO. However, the meta parameters of the other swarm intelligence algorithms were set using the default values as suggested in the literature.

Table 3 shows the classification accuracy of the three variants of the proposed ACHIO-MLP compared to the original CHIO-MLP in terms of the mean, the standard deviation, and the best results. The higher accuracy results mean better performance, while the lower STD values reflect the algorithm's robustness. The best results are highlighted using bold fonts.

The accuracy mean results shows that the proposed ACHIO-MLP with $A_r = 0.2$ was ranked first by achieving the best accuracy results in 9 out of 15 datasets. The

Table 1 The classification datasets and MLP structure for each dataset

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

Table 2 Parameters settings of the comparative algorithms

Algorithm	Parameter	Settings
HS [64]	Harmony memory size (HMS)	30
	Pitch adjustment rate (PAR)	0.1
	Memory consideration rate (HMCR)	0.95
	Bandwidth (bw)	0.02
PSO [65]	Population size	30
	Acceleration constants (c_1 and c_2)	[2, 2]
	Inertia weights (w)	[0.2, 0.9]
BA [66]	Population size	30
	Loudness (A_0)	0.5
	Pulse emission rate (r)	[0,1]
	Minimum frequency (f_{min})	0
	Maximum frequency (f_{max})	2
ABC [67]	Food sources	60
	Limit	0.5*Food sources*(#weights + #biases)
FPA [68]	Population size	30
	Switch probability (p)	0.8
SCA [69]	Population size	30

proposed ACHIO-MLP with $A_r = 0.5$ ranked second with the best accuracy results in 4 datasets, while the remaining two algorithms (i.e., CHIO-MLP and ACHIO-MLP with $A_r = 0.1$) ranked last with each one obtains the best accuracy results in 2 datasets.

According to the best accuracy results, it is clear that the proposed ACHIO-MLP with $A_r = 0.2$ was ranked first by obtaining the highest accuracy results in 8 datasets. In addition, the CHIO-MLP was placed second by obtaining the best results in 6 datasets. While the proposed ACHIO-MLP with $A_r = 0.5$ ranked third by getting the best results in 4 datasets. The ACHIO-MLP with $A_r = 0.1$ was placed last by obtaining the best results in 3 datasets.

Reading the standard deviation results in Table 3, it can be concluded that the performance of the three variants of the proposed ACHIO-MLP is more robust than the CHIO-MLP by getting the minimum standard deviation results in the largest number of datasets.

Table 3 shows that the proposed ACHIO-MLP with $A_r = 0.2$ ranked first by having the minimum average ranking using Friedman's test, while the two remaining variants of the proposed ACHIO-MLP (i.e., ACHIO-MLP with $A_r = 0.5$ and ACHIO-MLP with $A_r = 0.1$) are ranked second and third. The CHIO-MLP is ranked last by having the highest Friedman score. This proves the effectiveness

Table 3 The accuracy results of the proposed ACHIO-MLP using various settings of the archive rate parameter

Dataset	Measures	CHIO-MLP	ACHIO-MLP		
			$A_r = 0.1$	$A_r = 0.2$	$A_r = 0.5$
Monk	Mean	75.94	93.86	**96.12**	95.32
	STD	3.79	6.56	6.23	4.71
	Best	83.13	98.19	100.00	98.19
Balloon	Mean	**100.00**	**100.00**	**100.00**	**100.00**
	STD	0.00	0.00	0.00	0.00
	Best	100.00	100.00	100.00	100.00
Iris	Mean	94.81	97.48	**98.30**	97.70
	STD	2.21	0.96	1.51	0.41
	Best	97.78	97.78	100.00	97.78
Cancer	Mean	96.24	96.35	96.28	**97.13**
	STD	0.72	0.60	0.59	0.87
	Best	97.61	97.13	97.13	98.09
Heart	Mean	58.75	56.53	**67.08**	55.69
	STD	7.69	5.65	4.42	3.37
	Best	79.17	75.00	75.00	62.50
Vertebral	Mean	83.19	**86.99**	79.28	83.87
	STD	1.90	2.06	2.31	0.75
	Best	89.25	89.25	83.87	84.95
Blood	Mean	80.03	79.69	79.09	**80.22**
	STD	1.00	0.30	0.42	0.90
	Best	82.14	80.36	79.91	82.59
Seeds	Mean	88.15	94.71	96.77	**97.25**
	STD	3.81	3.55	0.66	1.50
	Best	93.65	98.41	98.41	98.41
Glass	Mean	54.27	56.35	**64.22**	53.54
	STD	7.32	2.84	3.75	1.48
	Best	67.19	62.50	68.75	56.25
Ionosphere	Mean	83.59	85.59	**89.62**	85.59
	STD	3.66	1.80	1.76	2.76
	Best	92.38	89.52	91.43	89.52
German	Mean	71.06	81.47	**83.28**	80.43
	STD	1.90	2.06	1.13	1.09
	Best	76.00	83.33	85.00	82.67
Titanic	Mean	**79.23**	78.79	79.07	76.68
	STD	0.42	0.65	0.06	0.33
	Best	79.55	79.24	79.09	77.12
Vehicle	Mean	43.39	46.68	**57.18**	57.10
	STD	5.28	2.28	3.19	3.23
	Best	53.75	49.80	60.87	60.47
Parkinson	Mean	85.57	84.14	**88.45**	87.53
	STD	3.75	1.31	1.76	2.20
	Best	93.10	86.21	91.38	91.38
Yeast	Mean	37.43	47.43	**48.35**	47.12
	STD	3.15	4.45	4.66	4.32
	Best	43.60	52.13	54.38	52.36
Ranking		3.2	2.6	1.8	2.4

Table 4 The accuracy results of the proposed ACHIO-MLP in comparison with other swarm-based algorithms

Dataset	Measures	ACHIO	CHIO	ABC	BA	FPA	PSO	SCA	HS
Monk	Mean	**96.12**	75.94	78.21	65.02	66.22	93.69	69.64	61.29
	STD	6.23	3.79	4.22	5.21	3.73	7.64	4.13	4.62
	Best	100.00	83.13	88.55	74.70	75.30	100.00	78.31	73.49
Balloon	Mean	**100.00**	**100.00**	97.78	87.78	94.44	**100.00**	95.56	81.11
	STD	0.00	0.00	8.46	14.47	12.63	0.00	10.66	16.22
	Best	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
Iris	Mean	**98.30**	94.81	94.07	66.15	87.11	92.52	87.04	65.48
	STD	1.51	2.21	1.96	14.99	6.94	5.07	6.78	7.82
	Best	100.00	97.78	95.56	93.33	97.78	97.78	97.78	86.67
Cancer	Mean	96.28	96.24	96.24	95.84	94.82	**96.83**	95.30	94.21
	STD	0.59	0.72	0.79	0.68	0.85	0.97	0.79	1.37
	Best	97.13	97.61	97.61	97.13	96.17	98.56	96.17	96.65
Heart	Mean	67.08	58.75	49.86	**68.06**	64.58	63.89	63.75	63.33
	STD	4.42	7.69	11.55	10.17	9.65	7.45	9.23	8.85
	Best	75.00	79.17	75.00	87.50	79.17	79.17	79.17	75.00
Vertebral	Mean	79.28	83.19	80.22	80.04	78.85	**85.91**	79.32	73.33
	STD	2.31	1.90	2.44	4.33	2.19	1.79	2.58	4.44
	Best	83.87	89.25	84.95	86.02	83.87	89.25	82.80	81.72
Blood	Mean	79.09	**80.03**	78.81	76.61	78.11	79.29	76.90	76.65
	STD	0.42	1.00	1.08	1.28	0.60	0.79	1.04	1.80
	Best	79.91	82.14	80.36	78.57	79.02	80.36	79.02	79.91
Seeds	Mean	**96.77**	88.15	92.91	66.14	87.99	83.33	78.10	58.36
	STD	0.66	3.81	1.85	13.36	4.58	10.37	6.08	10.22
	Best	98.41	93.65	96.83	93.65	93.65	92.06	88.89	76.19
Glass	Mean	**64.22**	54.27	53.59	37.81	45.31	47.76	44.95	29.79
	STD	3.75	7.32	7.92	9.06	3.74	9.13	5.83	10.74
	Best	68.75	67.19	67.19	53.13	53.13	62.50	57.81	51.56
Ionosphere	Mean	**89.62**	83.59	83.68	79.05	86.10	86.95	78.13	72.92
	STD	1.76	3.66	3.61	5.58	3.66	3.17	3.60	4.39
	Best	91.43	92.38	89.52	88.57	92.38	92.38	85.71	81.90
German	Mean	**83.28**	71.06	78.42	73.71	79.40	81.87	77.31	68.66
	STD	1.13	1.90	2.95	3.91	2.48	1.88	2.25	2.43
	Best	85.00	76.00	84.33	81.00	83.67	86.67	81.67	74.67
Titanic	Mean	79.07	**79.23**	77.53	77.87	78.99	77.76	77.43	76.61
	STD	0.06	0.42	0.47	0.83	0.47	0.27	0.86	1.17
	Best	79.09	79.55	78.03	79.39	80.15	77.88	78.94	78.33
Vehicle	Mean	**57.18**	43.39	41.75	29.47	31.42	43.39	35.18	28.26
	STD	3.19	5.28	5.83	5.87	6.67	8.77	6.34	5.58
	Best	60.87	53.75	58.10	41.11	42.69	62.45	44.27	40.71
Parkinson	Mean	**88.45**	85.57	81.72	87.30	82.93	87.01	82.76	80.29
	STD	1.76	3.75	3.83	5.17	3.77	3.64	3.84	4.22
	Best	91.38	93.10	89.66	93.10	87.93	94.83	91.38	86.21
Yeast	Mean	**48.35**	37.43	38.85	24.99	33.45	36.02	31.78	21.87
	STD	4.66	3.15	3.85	8.39	2.77	3.62	2.98	10.77
	Best	54.38	43.60	46.29	32.58	37.53	42.47	37.98	34.61

of the proposed changes to the CHIO framework when it is used for optimizing the weights of neural networks.

Note that ACHIO-MLP with $A_r = 0.2$ will be used in the next comparison section as it obtains the best results.

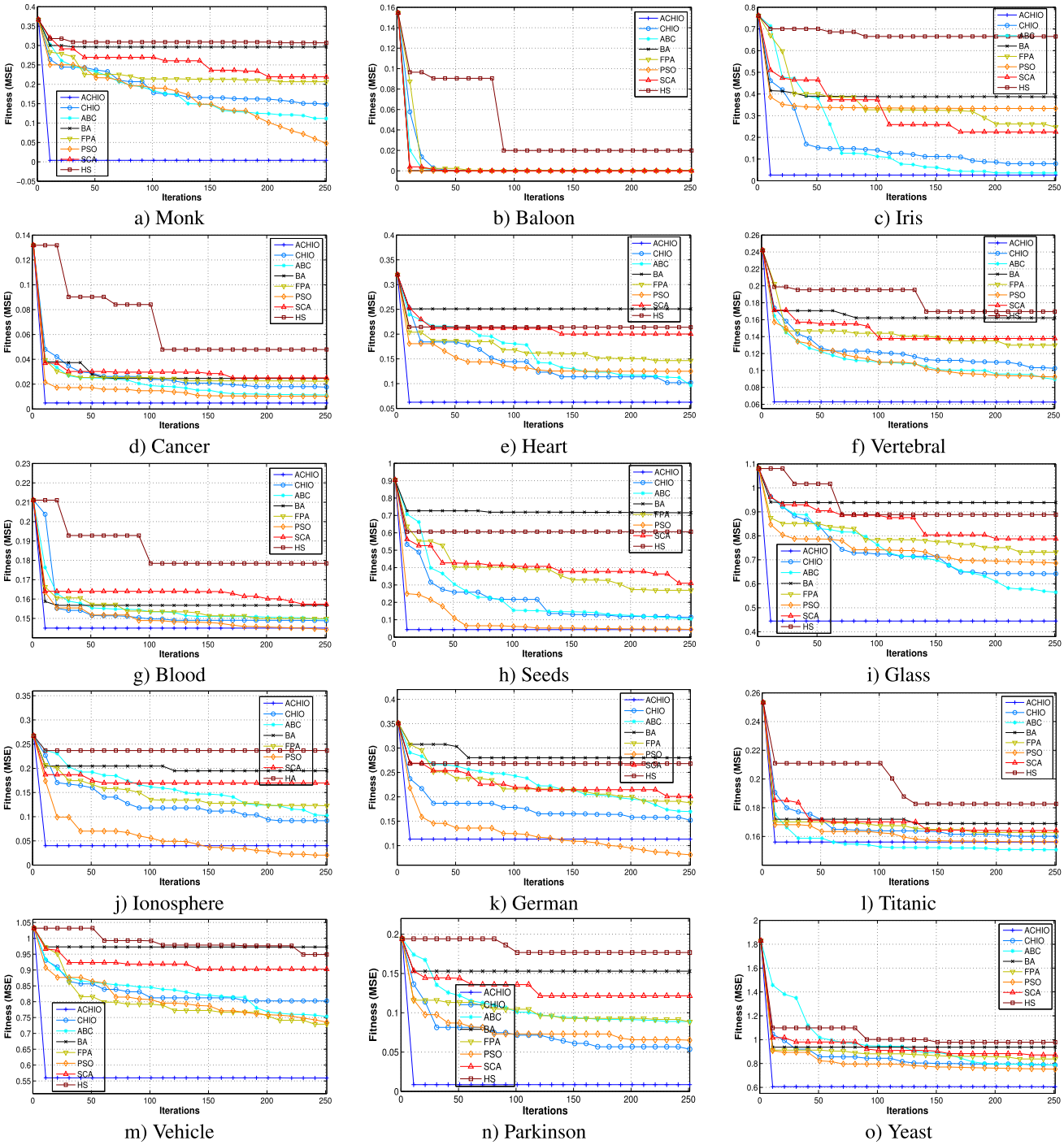

Fig. 3 Convergence results for the different algorithms

4.4 Comparison with other swarm-based optimization algorithms

In this section, the performance of the proposed ACHIO-MLP is evaluated and compared against seven swarm-based metaheuristics. These metaheuristics are the original CHIO [43], artificial bee colony (ABC) [28], bat algorithm (BA) [34], flower pollination algorithm (FPA) [68], particle swarm optimization (PSO) [20], sine cosine algorithm (SCA) [70], harmony search (HS) [71]. In order to ensure a fair comparison, all comparative algorithms are coded by the authors using the same datasets. The same parameter settings of all comparative methods are also unified as mentioned in Sect. 4.2.

The experimental results obtained by the ACHIO-MLP and all comparative methods are presented in Table 4. The

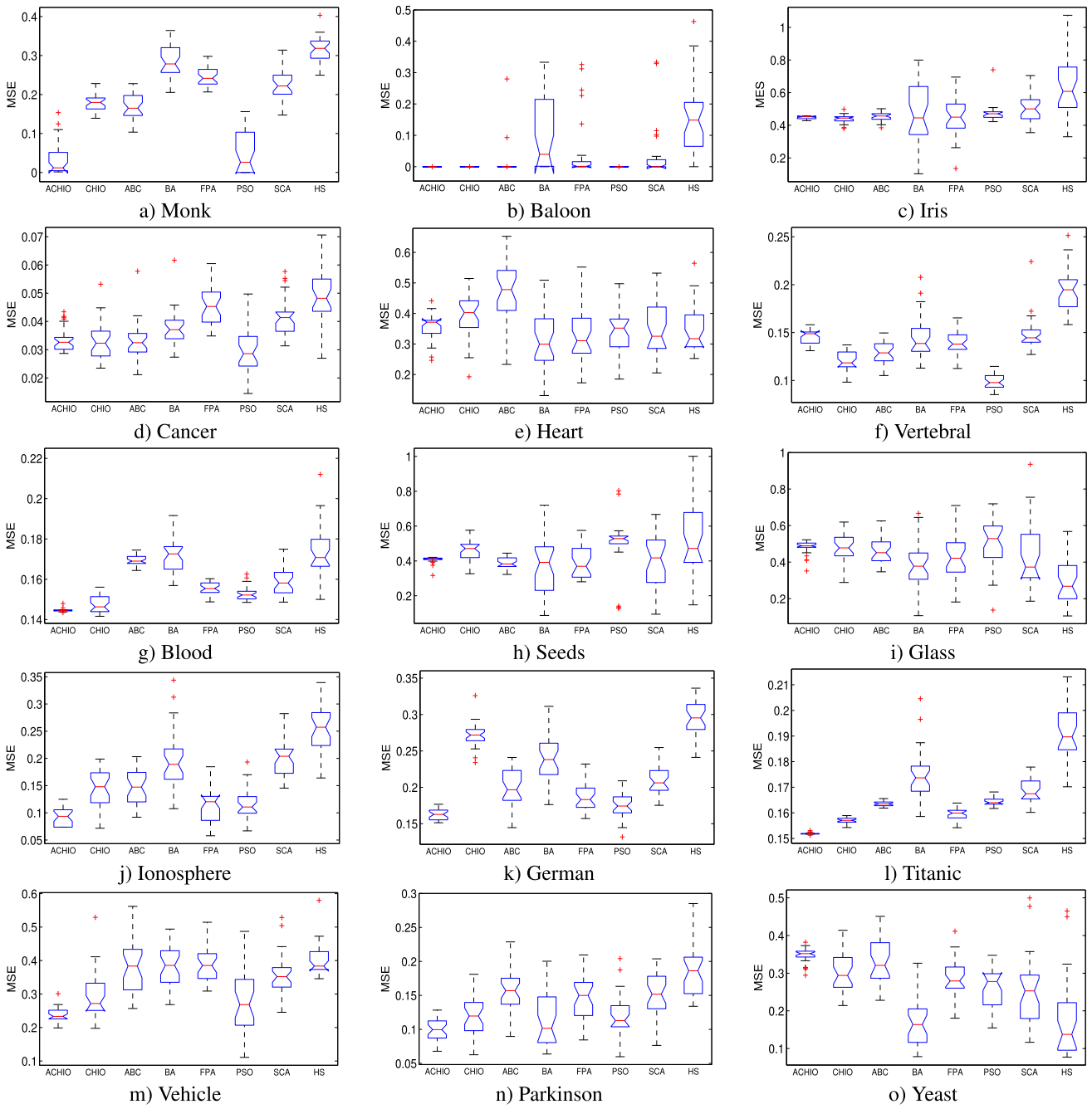

Fig. 4 Boxplot charts of MSE results for the different algorithms

results are presented in terms of the mean, standard deviation, and best classification accuracy. The bold values indicate the best value in each dataset. Note that the fittest MLPs which is the one with the lowest MSE on the training dataset are measured on the test dataset.

The best classification accuracy results are presented in Table 4. It shows that the ACHIO-MLP obtains the best classification accuracy for 6 datasets, including Monk (2), Balloon (2), Iris (3), Seeds (3), Glass (6), and Yeast (10). Note that the number of classes is shown between parentheses. Surprisingly, ACHIO-MLP excels the other comparative methods in two large datasets with six and ten classes. This shows the proposed algorithm's strength in navigating the search space in different ways and being able to achieve promising results. These high-quality results are due to the high balance between the exploration and exploitation of ACHIO-MLP. The proposed ACHIO-MLP algorithm comparison against the comparative methods shows that the ACHIO-MLP algorithm outperforms the CHIO, ABC, BA, FPA, PSO, SCA, and HS

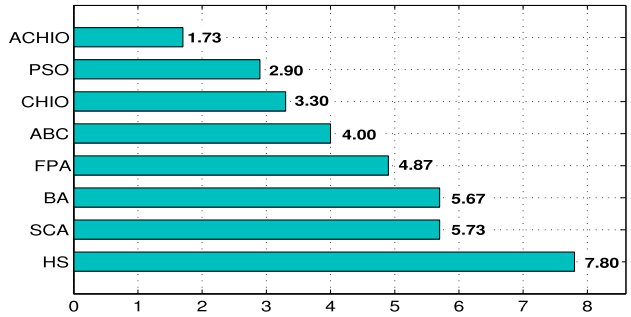

Fig. 5 Average rankings of the comparative algorithms using Friedman's statistical test

algorithms in seven, ten, nine, ten, five, twelve, and twelve datasets, respectively. On the other hand, PSO and CHIO achieve the best results in five and three datasets, respectively. Indeed, the PSO and ACHIO have common characteristics where they behave efficiently when navigating the search space of the weights and biases. They can widely explore several regions of the search space and exploit deeply each region of the MLP search space and find the local optima. Furthermore, since the MLP search space is non-convex and multimodal, the PSO and ACHIO are proven to be very efficient in dealing with the nature of this search space. Finally, the ACHIO behaves as a strong exploiter through the proposed archive-based concept. This allows it to make use of the accumulative knowledge and remember the best points in the MLP search space. Note that all of the algorithms produce the same optimal results for the Balloon dataset. Since the size of this dataset is small with only two classes, the algorithms did not require much effort to achieve the best results.

Similarly, the proposed algorithm is compared against comparative methods in terms of the mean of the classification accuracy. Table 4 shows that the performance of the ACHIO-MLP performs better than other comparative algorithms on ten datasets (i.e., Monk (2), Balloon (2), Ionosphere (2), German (2), Parkinson (2), Iris (3), Seeds (3), Vehicle (4), Glass (6), and Yeast (10)). Furthermore, the performance of the ACHIO-MLP is similar to CHIO and PSO by obtaining the best results for the Balloon dataset. The ACHIO-MLP is able to achieve the second-best results on three datasets (i.e., Cancer (2), Heart (2), and Titanic (2)). While ACHIO-MLP obtained the third-best results on the Blood (2) dataset. The strength of ACHIO-MLP is due to the behavior of its efficient operators where the infection and susceptible cases' operators can follow any random solution in the population while the recovered case operator exploits the attributes of the best solution. Furthermore, archiving the best results to be used in the next iteration improves the algorithm search. The lower standard derivation reflects the robustness of the algorithm. From Table 4, it can be observed that ACHIO-MLP has better robustness than other comparative algorithms in most datasets.

4.4.1 Convergence analysis

The performance of the comparative methods can be investigated using the convergence behavior toward the optimal solution. Accordingly, ACHIO-MLP and all the comparative methods convergence behaviors for all datasets are plotted in Fig. 3. In the figure, the iteration number is the *x-axis* and the fitness values are the *y-axis*. Notably, ACHIO-MLP significantly and rapidly converges toward its optimal solution without stagnation in local optima. This yields improvement in its achievements. In addition, ACHIO-MLP obtains the best convergence rate in ten datasets (i.e., Monk (2), Iris (3), Cancer (2), Heart (2), Vertebral (2), Seeds (3), Glass (6), Vehicle (4), Parkinson (2), and Yeast (10)) as the best MSE is reached within the defined number of iterations. It achieves the second-best in almost all other datasets. The ACHIO-MLP operators allow the algorithm to explore efficiently the search space niches and exploit each niche deeply. Using this strategy, the ACHIO-MLP owns a maneuver behavior movement strategy in the search space to escape the local optima trap during the search.

The boxplots for various classification datasets are shown in Fig. 4. Note that the MSE values obtained from MLP using the test dataset by utilizing the fittest MLP with the lowest MSE using the training dataset are plotted. This figure boxplots the obtained MSEs for the 15 classification datasets. In the boxplot, the smaller distance between the

Table 5 Holm/Hochberg outcome when having ACHIO the controlled algorithm against the other algorithms

Order	Algorithm	Adjusted ρ-value	Holm/Hochberg	Null hypotheses H_0
7	HS	1.18E-11	0.00714	Reject
6	SCA	7.74E-06	0.00833	Reject
5	BA	1.09E-05	0.01000	Reject
4	FPA	4.60E-04	0.01250	Reject
3	ABC	0.01127	0.01667	Reject
2	CHIO	0.07984	0.025	Not reject
1	PSO	0.19211	0.05	Not reject

best, median, and worst MSE demonstrates the stability of the algorithm. It is worth mentioning that the whiskers represent the farthest MSE values, while the box represents the interquartile range. The outliers are represented by the small circles, and the median value is represented by the bar in the box. In this figure, the boxplots demonstrate and explain the good performance of ACHIO-MLP for training MLP. The ACHIO-MLP shows the smallest MSE distance in twelve datasets (i.e., Balloon (2), Iris (3), Cancer (2), Heart (2), Blood (2), Seeds (3), Glass (6), German (2), Titanic (2), Vehicle (4), Parkinson (2), and Yeast (10)). In addition, the proposed ACHIO-MLP presents the second-best MSE distance in most of the rest datasets.

4.4.2 Friedman's statistical test

Figure 5 shows the ranking of the comparative algorithms using Friedman's test. It should be noted that the experimental results provided in Table 4 are used to calculate the rankings of the comparative algorithms. The null hypothesis (H_0) is that there is no significant difference between the performance of the proposed ACHIO-MLP and the alternative methods judged over all the datasets. On the other hand, the alternative hypothesis (H_1) is that there is a significant difference between the performance of the ACHIO-MLP and the alternative methods judged over all the datasets. Fig. 5 proves the high performance of the proposed method, where the ACHIO-MLP achieves the best ranking among all compared algorithms. The ρ-value calculated by Friedman's test is equal to 8.134649E-11, and this value is below the significance level ($\alpha = 0.05$). As a result, there is a significant difference between the comparative algorithms, and thus, the hypothesis H_0 is rejected.

The Holm and Hochberg procedures are used as *post-hoc* techniques to calculate the adjusted ρ-value in order to show if there is a significant difference between the controlled algorithm and other algorithms. It should be noted that the proposed ACHIO-MLP is the controlled algorithm because it obtains the first ranking as shown in Fig. 5. The null hypothesis H_0 is rejected using Holm's procedure when the ρ-value ≤ 0.01667, and the null hypothesis H_0 is rejected using Hochberg's procedure when the ρ-value ≤ 0.0125. As presented in Table 5, there is a significant difference between ACHIO-MLP and the other five comparative algorithms (HS, SCA, BA, FPA, and ABC). However, no significant difference between the behavior of the ACHIO-MLP and the two algorithms CHIO and PSO. This proves that the proposed ACHIO-MLP algorithm is a new good alternative algorithm that is able to succeed in solving such problems.

5 Conclusion and future work

CHIO is a powerful algorithm recently proposed to imitate the herd immunity treatment strategy to tackle the coronavirus pandemic. CHIO algorithm is selected because of its capabilities in finding the right trade-off between the exploration of the different search space niches and the exploitation of each search space niche. In this paper, to maintain the local optima and to preserve a good level of population diversity, an archive of best solutions is implemented. The new proposed algorithm (called ACHIO-MLP) selects and trains MLPs. The MLP training problem is mathematically modeled to minimize MSE. The decision variables are the weights and biases in MLPs for which ACHIO-MLP searches to find the elite amount for weights and biases.

In order to evaluate the performance of ACHIO-MLP, a collection of 15 classification datasets with different degrees of difficulty is utilized. Each dataset is normalized before it is used. All datasets are split into 30% for testing and 70% for training. A *stratified way* is used to split each dataset to maintain the proportion of each class in the divided data to have a balanced number of classes in the train/test split. As each dataset has a different number of features (or class labels), each MLP uses a variant number of inputs, hidden, and output nodes.

The results of the proposed method are compared against the original CHIO and six swarm optimization algorithms: HS, PSO, BA, ABC, FPA, and SCA. Interestingly, ACHIO-MLP can produce very accurate results which excel other comparative methods in ten out of fifteen classification datasets and very competitive results for other datasets. In addition to that, the results demonstrate a better convergence of the proposed algorithm. In a nutshell, the proposed ACHIO-MLP avoids local optima because of its different diversification techniques. Moreover, the results expose how fast the convergence of the proposed algorithm is when compared to other comparable methods. Finally, ACHIO-MLP can train MLPs to obtain a promising set of weights and biases that can produce better results.

In the future, the proposed algorithm will be applied to tackle real-world applications. Also, the proposed algorithm can be hybridized with other local search algorithms to improve its exploitation abilities

Funding No funding sources are applicable for this research.

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 MH et al (1995) Fundamentals of artificial neural networks. MIT press
2. Schmidhuber J (2015) Deep learning in neural networks: an overview. Neural Netw 61:85–117
3. Abiodun OI, Jantan A, Omolara AE, Dada KV, Mohamed NA, Arshad H (2018) State-of-the-art in artificial neural network applications: a survey. Heliyon 4(11):00938
4. Liao S-H, Wen C-H (2007) Artificial neural networks classification and clustering of methodologies and applications-literature analysis from 1995 to 2005. Expert Syst Appl 32(1):1–11
5. McCulloch WS, Pitts W (1943) A logical calculus of the ideas immanent in nervous activity. Bull Math Biophys 5(4):115–133
6. Orr MJ et al (1996) Introduction to radial basis function networks. Technical Report, center for cognitive science, University of Edinburgh
7. Bebis G, Georgiopoulos M (1994) Feed-forward neural networks. IEEE Potentials 13(4):27–31
8. Nowlan SJ, Platt JC (1995) A convolutional neural network hand tracker. Adv Neural Inf Process Syst, 901–908
9. Medsker LR, Jain L (2001) Recurrent neural networks. Design Appl , 5
10. Ghosh-Dastidar S, Adeli H (2009) Spiking neural networks. Int J Neural Syst 19(04):295–308
11. Samarasinghe S (2016) Neural networks for applied sciences and engineering: from fundamentals to complex pattern recognition. Crc Press
12. She FH, Kong L, Nahavandi S, Kouzani A (2002) Intelligent animal fiber classification with artificial neural networks. Textile Res J 72(7):594–600
13. Ahmadian S, Khanteymoori AR (2015) Training back propagation neural networks using asexual reproduction optimization. In: 2015 7th Conference on Information and Knowledge Technology (IKT), pp 1–6. IEEE
14. Savalia S, Emamian V (2018) Cardiac arrhythmia classification by multi-layer perceptron and convolution neural networks. Bioengineering 5(2):35
15. Zhang L, Li H, Kong X-G (2019) Evolving feedforward artificial neural networks using a two-stage approach. Neurocomputing 360:25–36
16. Nasr MB, Chtourou M (2006) A hybrid training algorithm for feedforward neural networks. Neural Process Lett 24(2):107–117
17. Ng S-C, Cheung C-C, Leung S-H (2004) Magnified gradient function with deterministic weight modification in adaptive learning. IEEE Trans Neural Netw 15(6):1411–1423
18. Faris H, Mirjalili S, Aljarah I (2019) 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
19. Ding S, Su C, Yu J (2011) An optimizing bp neural network algorithm based on genetic algorithm. Artif Intell Rev 36(2):153–162
20. Das G, Pattnaik PK, Padhy SK (2014) Artificial neural network trained by particle swarm optimization for non-linear channel equalization. Expert Syst Appl 41(7):3491–3496
21. Slowik A, Bialko M (2008) Training of artificial neural networks using differential evolution algorithm. In: 2008 Conference on Human System Interactions, pp 60–65. IEEE
22. Ilonen J, Kamarainen J-K, Lampinen J (2003) Differential evolution training algorithm for feed-forward neural networks. Neural Process Lett 17(1):93–105
23. Mirjalili S (2015) How effective is the grey wolf optimizer in training multi-layer perceptrons. Appl Intell 43(1):150–161
24. Bairathi D, Gopalani D (2019) Salp swarm algorithm (ssa) for training feed-forward neural networks. In: Soft Computing for Problem Solving, pp 521–534. Springer
25. Alboaneen DA, Tianfield H, Zhang Y (2017) 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
26. Moayedi H, Nguyen H, Foong LK (2019) Nonlinear evolutionary swarm intelligence of grasshopper optimization algorithm and gray wolf optimization for weight adjustment of neural network. Eng Comput, 1–11
27. Heidari AA, Faris H, Aljarah I, Mirjalili S (2019) An efficient hybrid multilayer perceptron neural network with grasshopper optimization. Soft Comput 23(17):7941–7958
28. Ghanem WA, Jantan A (2018) A cognitively inspired hybridization of artificial bee colony and dragonfly algorithms for training multi-layer perceptrons. Cognit Comput 10(6):1096–1134
29. Jalali SMJ, Ahmadian S, Kebria PM, Khosravi A, Lim CP, Nahavandi S (2019) Evolving artificial neural networks using butterfly optimization algorithm for data classification. In: International Conference on Neural Information Processing, pp 596–607. Springer
30. Faris H, Aljarah I, Mirjalili S (2018) Improved monarch butterfly optimization for unconstrained global search and neural network training. Appl Intell 48(2):445–464
31. Mirjalili SZ, Saremi S, Mirjalili SM (2015) Designing evolutionary feedforward neural networks using social spider optimization algorithm. Neural Comput Appl 26(8):1919–1928
32. Chen H, Wang S, Li J, Li Y (2007) 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
33. Socha K, Blum C (2007) An ant colony optimization algorithm for continuous optimization: application to feed-forward neural network training. Neural Comput Appl 16(3):235–247
34. Jaddi NS, Abdullah S, Hamdan AR (2015) Multi-population cooperative bat algorithm-based optimization of artificial neural network model. Inf Sci 294:628–644
35. Zhang Y, Phillips P, Wang S, Ji G, Yang J, Wu J (2016) Fruit classification by biogeography-based optimization and feedforward neural network. Expert Syst 33(3):239–253
36. Mirjalili S, Hashim SZM, Sardroudi HM (2012) Training feedforward neural networks using hybrid particle swarm optimization and gravitational search algorithm. Appl Math Comput 218(22):11125–11137
37. Faris H, Aljarah I, Alqatawna J (2015) 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
38. Heidari AA, Faris H, Mirjalili S, Aljarah I, Mafarja M (2020) Ant lion optimizer: theory, literature review, and application in multi-layer perceptron neural networks. Nature-Inspired Optimizers, 23–46
39. Valian E, Mohanna S, Tavakoli S (2011) Improved cuckoo search algorithm for feedforward neural network training. Int J Artif Intell Appl 2(3):36–43

40. Wu H, Zhou Y, Luo Q, Basset MA (2016) Training feedforward neural networks using symbiotic organisms search algorithm. Comput Intell Neurosci 2016
41. Faris H, Aljarah I, Al-Madi N, Mirjalili S (2016) Optimizing the learning process of feedforward neural networks using lightning search algorithm. Int J Artif Intell Tools 25(06):1650033
42. Wolpert DH, Macready WG (1997) No free lunch theorems for optimization. IEEE Trans Evol Comput 1(1):67–82
43. Al-Betar MA, Alyasseri ZAA, Awadallah MA, Doush IA (2020) Coronavirus herd immunity optimizer (chio). Neural Comput Appl, 1–32
44. Dalbah LM, Al-Betar MA, Awadallah MA, Zitar RA (2021) A coronavirus herd immunity optimization (chio) for travelling salesman problem. In: International Conference on Innovative Computing and Communications, pp 11–19. Springer
45. Kumar C, Magdalin Maryb D, Gunasekar T (2021) Mochio: A novel multi-objective coronavirus herd immunity optimization algorithm for solving brushless direct current wheel motor design optimization problem. PREPRINT (Version 1) available at Research Square
46. Lacroix B, Molina D, Herrera F (2016) Region-based memetic algorithm with archive for multimodal optimisation. Inf Sci 367:719–746
47. Zhang Y-H, Gong Y-J, Chen W-N, Zhan Z-H, Zhang J (2014) A generic archive technique for enhancing the niching performance of evolutionary computation. In: 2014 IEEE Symposium on Swarm Intelligence, pp 1–8. IEEE
48. Kundu S, Biswas S, Das S, Suganthan PN (2013) Crowding-based local differential evolution with speciation-based memory archive for dynamic multimodal optimization. In: Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation, pp 33–40
49. Wang Z-J, Zhan Z-H, Lin Y, Yu W-J, Yuan H-Q, Gu T-L, Kwong S, Zhang J (2017) Dual-strategy differential evolution with affinity propagation clustering for multimodal optimization problems. IEEE Trans Evol Comput 22(6):894–908
50. Sheng W, Wang X, Wang Z, Li Q, Chen Y (2021) Adaptive memetic differential evolution with niching competition and supporting archive strategies for multimodal optimization. Inf Sci 573:316–331
51. Turky AM, Abdullah S (2014) A multi-population harmony search algorithm with external archive for dynamic optimization problems. Inf Sci 272:84–95
52. Zhu Q, Lin Q, Chen W, Wong K-C, Coello CAC, Li J, Chen J, Zhang J (2017) An external archive-guided multiobjective particle swarm optimization algorithm. IEEE Trans Cybern 47(9):2794–2808
53. Got A, Moussaoui A, Zouache D (2020) A guided population archive whale optimization algorithm for solving multiobjective optimization problems. Expert Syst Appl 141:112972
54. Kalra S, Rahnamayan S, Deb K (2017) Enhancing clearing-based niching method using delaunay triangulation. In: 2017 IEEE Congress on Evolutionary Computation (CEC), pp 2328–2337. IEEE
55. Bhesdadiya R, Jangir P, Jangir N, Trivedi IN, Ladumor D (2016) Training multi-layer perceptron in neural network using whale optimization algorithm. Indian J Sci Technol 9(19):28–36
56. Askari Q, Younas I (2021) Political optimizer based feedforward neural network for classification and function approximation. Neural Process Lett 53(1):429–458
57. Irmak B, Karakoyun M, Gülcü Ş (2022) An improved butterfly optimization algorithm for training the feed-forward artificial neural networks. Soft Comput, 1–19
58. Sun K, Huang S-H, Wong DS-H, Jang S-S (2016) 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
59. Makhadmeh SN, Al-Betar MA, Awadallah MA, Abasi AK, Alyasseri ZAA, Doush IA, Alomari OA, Damaševičius R, Zajančkauskas A, Mohammed MA (2022) A modified coronavirus herd immunity optimizer for the power scheduling problem. Mathematics 10(3):315
60. Dalbah LM, Al-Betar MA, Awadallah MA, Zitar RA (2022) A modified coronavirus herd immunity optimizer for capacitated vehicle routing problem. J King Saud Univ Comput Inf Sci 34(8):4782–4795
61. Wdaa ASI, Sttar A (2008) Differential evolution for neural networks learning enhancement. In: PhD Thesis, Universiti Teknologi Malaysia Johor Bahru
62. Mirjalili S, Mirjalili SM, Lewis A (2014) Let a biogeography-based optimizer train your multi-layer perceptron. Inf Sci 269:188–209
63. Cano J-R, García S, Herrera F (2008) 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
64. Geem ZW, Kim JH, Loganathan GV (2001) A new heuristic optimization algorithm: harmony search. SIMULATION 76(2):60–68
65. Kennedy J, Eberhart R (1995) Particle swarm optimization. Proc ICNN'95 Int Conf Neural Netw 4:1942–1948
66. Yang X-S (2010) A new metaheuristic bat-inspired algorithm. In: Nature Inspired Cooperative Strategies for Optimization (NICSO 2010), pp 65–74. Springer
67. Karaboga D (2005) An idea based on honey bee swarm for numerical optimization. Technical report, Technical report-tr06, Erciyes university, engineering faculty, computer.
68. Yang X-S (2012) Flower pollination algorithm for global optimization. In: International Conference on Unconventional Computing and Natural Computation, pp 240–249. Springer
69. Mirjalili S (2016) Sca: a sine cosine algorithm for solving optimization problems. Knowl Based Syst 96:120–133
70. Sahlol AT, Ewees AA, Hemdan AM, Hassanien AE (2016) Training feedforward neural networks using sine-cosine algorithm to improve the prediction of liver enzymes on fish farmed on nano-selenite. In: 2016 12th International Computer Engineering Conference (ICENCO), pp 35–40. IEEE
71. Kulluk S, Ozbakir L, Baykasoglu A (2012) Training neural networks with harmony search algorithms for classification problems. Eng Appl Artif Intell 25(1):11–19

Authors and Affiliations

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

✉ Iyad Abu Doush
idoush@auk.edu.kw

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

Mohammed Azmi Al-Betar
m.albetar@ajman.ac.ae

Osama Ahmad Alomari
oalomari@sharjah.ac.ae

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

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

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

[1] College of Engineering and Applied Sciences, American University of Kuwait, Salmiya, Kuwait

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

[3] Department of Computer Science, Al-Aqsa University, Gaza, Palestine

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

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

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

[7] MLALP Research Group, University of Sharjah, Sharjah, UAE

[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