Evolving neural networks using bird swarm algorithm for data classification and regression applications

Article *in* Cluster Computing · February 2019

DOI: 10.1007/s10586-019-02913-5

CITATIONS
0

READS
382

6 authors, including:

Ibrahim Aljarah
University of Jordan
79 PUBLICATIONS **676** CITATIONS
SEE PROFILE

Hossam Faris
University of Jordan
124 PUBLICATIONS **979** CITATIONS
SEE PROFILE

Seyedali Mirjalili
Griffith University
134 PUBLICATIONS **6,001** CITATIONS
SEE PROFILE

Nailah Al-Madi
Princess Sumaya University for Technology
18 PUBLICATIONS **91** CITATIONS
SEE PROFILE

Cluster Computing
https://doi.org/10.1007/s10586-019-02913-5

Evolving neural networks using bird swarm algorithm for data classification and regression applications

Ibrahim Aljarah[1] · Hossam Faris[1] · Seyedali Mirjalili[2] · Nailah Al-Madi[3] · Alaa Sheta[4] · Majdi Mafarja[5]

Received: 24 April 2018 / Revised: 4 January 2019 / Accepted: 31 January 2019

Abstract
This work proposes a new evolutionary multilayer perceptron neural networks using the recently proposed Bird Swarm Algorithm. The problem of finding the optimal connection weights and neuron biases is first formulated as a minimization problem with mean square error as the objective function. The BSA is then used to estimate the global optimum for this problem. A comprehensive comparative study is conducted using 13 classification datasets, three function approximation datasets, and one real-world case study (Tennessee Eastman chemical reactor problem) to benchmark the performance of the proposed evolutionary neural network. The results are compared with well-regarded conventional and evolutionary trainers and show that the proposed method provides very competitive results. The paper also considers a deep analysis of the results, revealing the flexibility, robustness, and reliability of the proposed trainer when applied to different datasets.

Keywords Optimization · Neural networks · Multilayer perceptron · Bird Swarm Algorithm · Classification · Regression

1 Introduction

Classification, function approximation, and prediction using machine learning techniques has been popular applications in different fields of study. Undoubtedly, Artificial neural networks (ANNs) are among the most well-regarded techniques in this area which have been largely applied to different problems. ANNs [65, 71, 96] are non-paramateric mathematical models inspired by biological neural systems. ANNs represent a robust information processing system, which is composed of highly interconnected elements called neurons. ANNs perform simultaneous computations and data processing to solve specific problems with different complexities. ANNs have became more popular over the last decade, and have directed most of researchers' attention to apply ANNs in different fields. ANNs benefit from high performance and ease of implementation, and they are able to capture the hidden relationship between the inputs. In addition, ANNs are high scalable and can be implemented in parallel architectures, taking advantage of modern advancements and technologies in this context [93, 94]. Furthermore,

✉ Ibrahim Aljarah
i.aljarah@ju.edu.jo

Hossam Faris
hossam.faris@ju.edu.jo

Seyedali Mirjalili
seyedali.mirjalili@griffithuni.edu.au

Nailah Al-Madi
n.madi@psut.edu.jo

Alaa Sheta
alaa.sheta@tamucc.edu

Majdi Mafarja
mmafarja@birzeit.edu

[1] Department of Information Technology, King Abdullah II School for Information Technology, The University of Jordan, Amman, Jordan

[2] School of Information and Communication Technology, Griffith University, Nathan, Brisbane, QLD 4111, Australia

[3] King Hussein Faculty of Computing Sciences, Princess Sumaya University for Technology, Amman, Jordan

[4] Department of Computing Sciences, Texas A&M University, Corpus Christi, TX 78412, USA

[5] Department of Computer Science, Birzeit University, Birzeit, Palestine

Published online: 15 February 2019

ANNs have a remarkable ability to solve challenging problems such as image recognition [56, 69, 92], data classification [1, 3, 55, 97], function approximation [52], control of non-linear systems modeling [34, 85], environmental forecasting [24] and many others.

In general, ANNs consist of two main components: neurons, which represent the processing units, and connections between the neurons. Each connection carries a weight, which is used to accomplish the computational process by the neuron with its current information.

A variety of ANNs have been developed in the literature such as feedforward neural network (FNN) [15], radial basis function network (RBF) [36, 41], and recurrent neural networks [70]. Most of these ANNs have different structures to process the information inside the network and depend on how the network neurons exchange the information between each other.

FNNs consist of two main types of neural networks: single-layer perceptrons (SLP) [37] and multi-layer perceptrons (MLP) [13, 49]. SLP is proper for modeling the linear problems, while MLP is used for non-linear problems.

One of the most impact properties of a ANNs is the ability to learn. ANNs structure can be adapted by adjusting ere are four common strategies to learn the neural network, namely; supervised learning [21, 23], unsupervised learning [59, 75], and reinforcement learning [44, 83], and meta-heuristic learning [35, 47, 80, 95]. Supervised learning is used when the problem outputs are known in prior as in pattern recognition and classification problems. A common supervised learning approach used in ANNs is the back-propagation (BP) algorithm [20, 40, 78, 98], which is a gradient-based algorithm. BP has some drawbacks that make it unreliable for practical applications such slow convergence, and premature convergence to local optimum.

Unsupervised learning is used when the outputs are missing or unknown. Unsupervised learning is frequently used in text categorization and clustering based applications [61]. On the other hand, reinforcement learning is used when the problem has complex stochastic structure and very difficult to analyze, like control optimization problems.

Meta-heuristic algorithms are search strategies to find sufficiently good solution for the optimization problems [5–7]. Meta-heuristic learning has the ability to estimate optimal or semi-optimal connection weights set for ANNs with less probability to be trapped into the many local optima in the search space [4, 35, 80]. Many meta-heuristic learning algorithms have been used to train ANNs such as Genetic Algorithm (GA) [66, 76], Particle Swarm Optimization (PSO) [101], Evolutionary Strategies (ES) [90], Ant Colony Optimization (ACO) [57], Cuckoo Search (CS) [68, 86], Krill Herd Optimization (KH) [25, 48], Firefly Algorithm (FA) [19], Population-Based Incremental Learning (PBIL) [30], Differential Evolution (DE) [42, 88], Artificial Bee Colony (ABC) [45], and many others.

As mentioned in the previous paragraph, there are many meta-heuristic algorithms used for learning ANNs, which give a clear indication of the efficiency of the meta-heuristic learning algorithms. Furthermore, most of these algorithms tried to resolve the drawbacks of gradient-based methods like BP by accelerating the convergence and avoiding the local optima. Moreover, we noticed that there is no superior meta-heuristic algorithm can perfectly learn the ANNs and handle all type of the problems. This has been proven by the well-known No Free Lunch theorem (NFL) [18, 39]. As a result, all of these reasons encourage a lot of researchers to apply other meta-heuristic algorithms to train ANNs.

Bird Swarm Algorithm (BSA) [60] is one of the most recent meta-heuristic algorithm, which is a global optimization algorithm that uses strong formulation strategy to achieve optimal or semi-optimal problem solutions. BSA is similar to other meta-heuristics algorithms that use guided randomization mechanism to generate solutions with high diversity property.

This paper presents a new learning approach based on BSA to optimize the MLP. In this work, we have made the following key contributions:

- The BSA is proposed for the first time to optimize the MLP neural networks. In this approach, BSA is integrated as a learner into MLP neural network to solve different data classification and regression problems.
- The performance of the proposed approach is evaluated on thirteen real world classification datasets with different settings and characteristics to demonstrate its effectiveness and quality of solutions.
- The performance of the proposed approach is tested on three regression problems, which represent real function approximations.
- The BSA-based learner is applied to a very challenging real world problem called Tennessee Eastman (TE) chemical process reactor problem [22] as well, which is a simulation of an actual system at the Tennessee Eastman Company, USA. TE is considered as a large-scale nonlinear, open-loop unstable system with both fast and slow variable dynamics [43]. This makes it a challenge process for both system identification and control. The proposed BSA learner assists the MLP network to find the optimal chemical process models.
- The proposed approach is compared with six popular meta-heuristic learners such as GA, DE, Evolution Strategy (ES), ABC, PSO, and ACO, and two popular

standard gradient decent learning algorithms: the BP algorithm and Levenberg–Marquardt (LM).

This paper is organized as follows: several related works in the literature are presented in Sect. 2. The preliminary background concerning the BSA and MLP are presented in Sects. 3, and 4 respectively. In Sect. 5, the proposed learner and the design details of the MLP are described. In Sect. 6, the experimental results for BSA learner and other comparison are described. Finally, in Sects. 7, the general conclusions, and future directions of this research are given.

2 Related works

The learning of ANNs has received much attention in last decade to improve the efficiency of the ANNs modeling results. Due to space constraints, we focus only on closely related work of meta-heuristics algorithms that employed in the learning process of MLPs .

Genetic Algorithm (GA) is considered as one of the first meta-heuristic algorithms used for training MLPs [66]. Many authors in the literature applied GA to learn MLP networks. In [66], the authors applied the GA to find an optimal set of weights in an acceptable running time. They evaluated the GA optimizer using sonar images dataset with different forms of mutations and crossover operations. The results showed that the GA optimizer is efficient and able to outperform the standard BP learning algorithm. Another work based on GA was proposed in [17], the authors applied the GA to find the global solution of some continuous functions. In addition, more variants of GA are proposed in [11, 53, 76, 89] to enhance the MLP learning process.

In [42], Jarmo et al. applied the differential evolution (DE) optimization method in the MLP learning process. The performance results of DE as a learning algorithm were very competitive with the gradient-based methods. In addition, their work did not disclose any obvious evidence to use DE over gradient-based methods. Another work in [81] used the DE to optimize the weights of the MLP network. In this work, an adaptive mechanism to select DE control parameters was proposed to enhance the efficiency of the DE optimizer. The proposed algorithm was evaluated using the parity-p classification problem with promising results. A hybrid method in [84] was proposed that combining the DE with gradient based methods. The work was applied to solve the nonlinear system identification problem.

Christian et al. in [32] introduced a new MLP learning mechanism based on the Evolution Strategy (ES). The ES-based trainer showed better performance in many applications such as car detection and tracking problems. Another work in [90] used the ES algorithm to train MLP networks.

Particle swarm optimization (PSO) in [99] was used to evolve the MLP networks; namely, the weights and network structure. The learning process was adapted based on PSO obtained better accuracy than other optimizers. Other researches such as [31, 58, 87] implemented modified PSO variants to enhance the PSO performance in the learning process. A hybrid method in [101] combining PSO optimizer with back-propagation was proposed to learn MLP network. The hybrid method resolve the local searching limitations of PSO by back-propagation.

In [57, 82], the authors introduced an Ant Colony Optimization algorithm (ACO) to solve the continuous optimization. The work was applied to learning of MLP networks, and evaluated using different data classification problems. In addition, the ACO was combined with different gradient based methods such as Levenberg–Marquardt and back-propagation algorithms to solve large-scale classification problems.

Recently, many new meta-heuristic algorithms have been used for learning such biogeography-based optimizer (BBO) [64], Moth-flame optimization [91], multi-verse optimizer (MVO) [27], Grey Wolf optimizer (GWO) [63], and many others [8, 9, 26, 28, 29, 38].

3 Multi-layer perceptron neural networks (MLP)

Multilayer perceptron (MLP) neural networks is considered the most popular type of FNNs.An MLP maps a set of inputs onto a set of suitable outputs by applying transformation procedure to obtain the outputs. MLP is comprised of nodes called neurons distributed in different levels of layers; namely, input layer, hidden layer, and output layer. The input layer receives n data inputs and direct them to the next layer. The hidden layers form the middle point between input and the output layers. The MLP network could have more than one hidden layer, where the number depends on the type of the problem. Most of the researches used one hidden layer as a default number. The main objective of the neurons in hidden layer is to transform the inputs to desired outputs using a transfer function. The output layer collects the final results of the network. Furthermore, the number of neurons in output layer is selected based on the data classes.

Figure 1 shows a simple MLP neural network and single neuron. Figure 1a shows an MLP with input layer, single hidden layer, and output layer, and Fig. 1b shows one single neuron. The input layer contains n neurons, hidden layer has m neurons, and output layer has k neurons. The

MLP forms a well connected directed graph such as each hidden neuron is connected with n connection weights with extra one called bias weight. In each hidden neuron, two main operations are used to aggregate the final neurons output: summation and activation operations. The output of the summation operation of the neuron j is accomplished by Eq. 1. After that, the summation operation output is mapped using a special type of functions called transfer or activation functions. The activation operation is accomplished using Eq. 2.

$$Sum_j = \sum_{i=1}^{n} w_{ij} * in_i + b_j \tag{1}$$

where w_{ij} is the connection weight between the input neuron i and hidden neuron j; b_j is the bias j to hidden neuron j.

$$y_j = f(Sum_j) \tag{2}$$

where y_j is the neuron j output; $j = 1, 2, \ldots, m$; f is a Sigmoid function, and calculated using Eq. 3.

$$f(Sum_j) = \frac{1}{1 + e^{-Sum_j}} \tag{3}$$

After aggregating the outputs of all hidden nodes, the final outputs Y_j are calculated using the summation and activation operations as described in Eqs. 4 and 5:

$$Sum_j = \sum_{i=1}^{m} w_{ij} * y_j + b_j \tag{4}$$

where w_{ij} is the connection weight between the hidden neuron i and output neuron j; b_j is the bias j to output neuron j.

$$Y_j = f(Sum_j) \tag{5}$$

where Y_j is the final output j ; $j = 1, 2, \ldots, k$; f is the same Sigmoid function that used in Eq. 3.

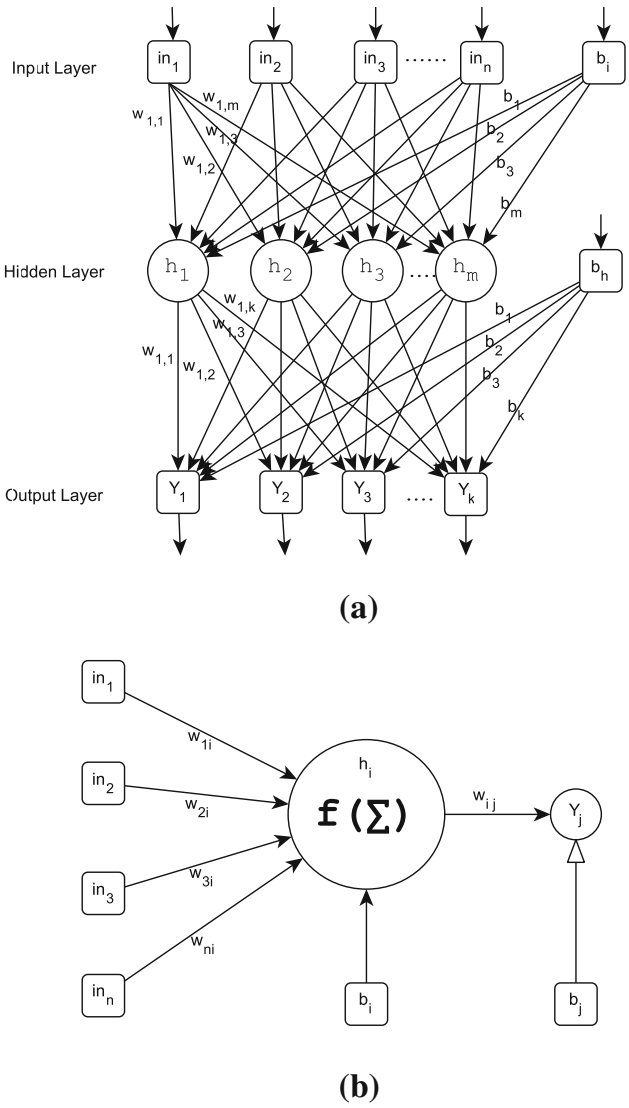


Fig. 1 **a** MLP network with a single hidden layer. **b** One single neuron

4 The Bird Swarm Algorithm

Bird Swarm Algorithm (BSA) is a new swarm intelligent and global optimization algorithm inspired by the behavior of social iteration of birds in nature. Authors in [60], proposed their BSA algorithm based on three main behaviors of birds which are foraging, vigilance and flight. The abstract idea of the algorithm can be summarized in the following five rules:

- Rule 1: Each bird can be in one of two statuses either vigilance or foraging.
- Rule 2: In the foraging status, each bird keeps tracking and memorizes its own best experience and the best experience among the swarm about food positions. This information will affect its movement and search path for food.
- Rule 3: In the vigilance status, each bird tries competitively to move toward the center, assuming that birds with higher reserves lie closer to the center of the flock. Birds in the center are less probable to be attacked by other predators.
- Rule 4: Birds keep moving from one site to another and they iteratively keep switching between producing and scrounging. The algorithm assumes that birds with highest reserves are producers while the lowest are scroungers. On the other hand, other birds are randomly assumed to be producers or scroungers.
- Rule 5: Producing birds lead the search for food while the scrounging ones randomly follow a producing bird.

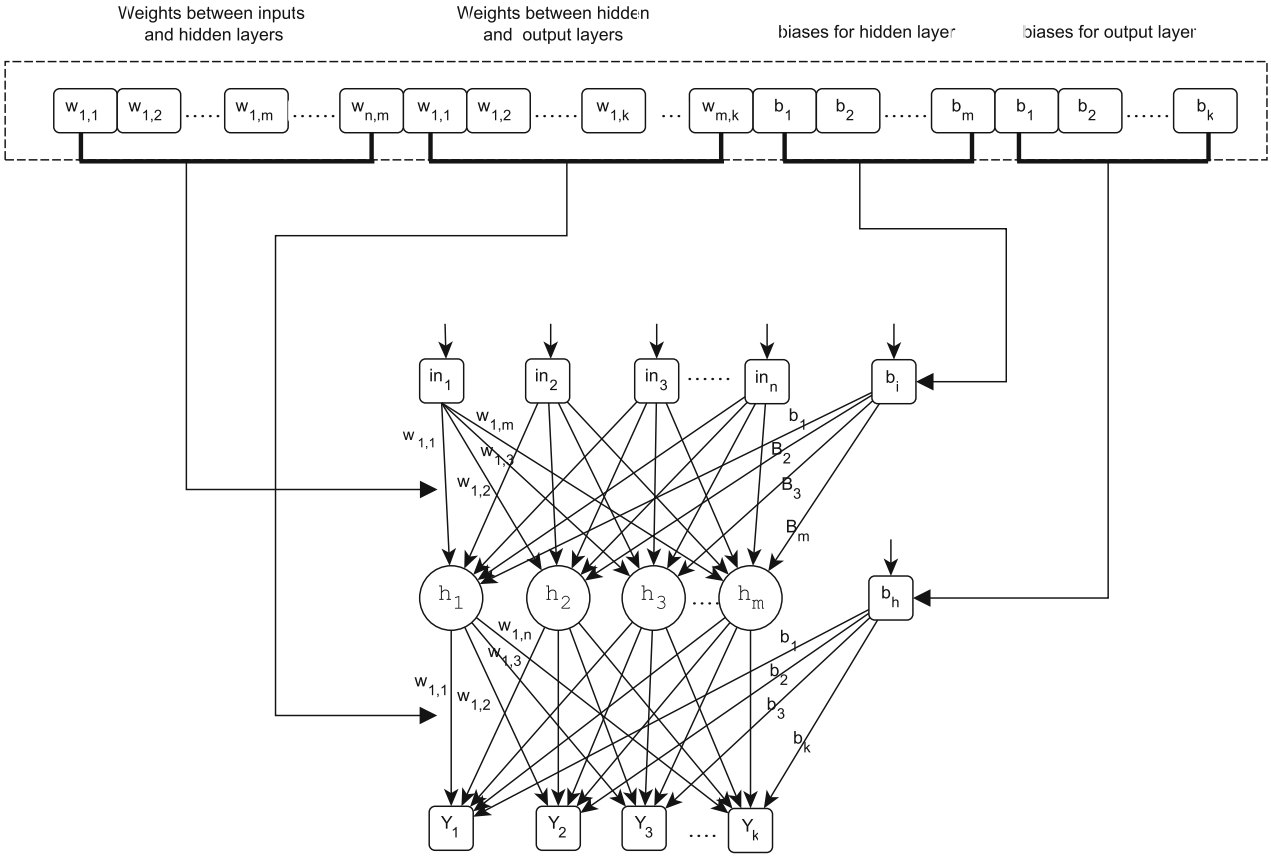


Fig. 2 Mapping a BSA individual to an MLP network

Based on these assumptions, the main operators of the BSA algorithm are modeled as follows: the algorithm starts by initializing randomly a predetermined number of N birds in a search space of D dimensions. As specified in Rule 2, each bird searches for food based on its experience and the best experience of the flock. This rule is modeled as shown in Eq. 6. $x_{i,j}^{t}$ is the value of element j of bird number i of generation t, where $i \in [1, \ldots, N]$, $j \in [1, \ldots, D]$ and $rand_a$ is random number drawn from the normal distribution in the interval [0, 1]. C and S are called the cognitive and social accelerated coefficients which are two constant positive numbers. $P_{i,j}$ and g_i represent the personal and global experience, respectively.

$$x_{i,j}^{t+1} = x_{i,j}^{t} + (p_{i,j} - x_{i,j}^{t}) \times C \times rand_a + (g_j - x_{i,j}^{t}) \times S \times rand_a \tag{6}$$

This foraging behavior is activated if a randomly generated number is larger than a threshold P. This is implemented as a simple application of Rule 1.

BSA models the movement of competing birds toward the center which was described previously in Rule 3 as given in Eqs. 7, 8 and 9.

$$x_{i,j}^{t+1} = x_{i,j}^{t} + A1(mean_j - x_{i,j}^{t}) \times rand_a + A2(p_{k,j} - x_{i,j}^{t}) \times S \times rand_b \tag{7}$$

$$A1 = a1 \times exp\left(\frac{-pFit_i}{sumFit + \varepsilon} \times N\right) \tag{8}$$

$$A2 = a2 \times exp\left(\left(\frac{pFit_i - pFit_k}{|pFit_k - pFit_i| + \varepsilon}\right) \frac{N \times pFit_k}{sumFit + \varepsilon}\right) \tag{9}$$

where $a1$ and $a2$ are positive constant integers in [0, 2], $pFit_i$ is the best fitness value of bird i, $sumFit$ is the sum of all birds' best fitness values, ε is a very small constant to avoid division by zero, $mean_j$ is the value of the jth element of the average position of all the swarm.

In the previous model, the average fitness of the swarm is used to replace the effect of the surroundings when the birds move toward the center of the swarm.

Finally, Rule 4 is modeled to represent the producing and scrounging birds after performing a flight behavior. Equations 10 and 11 represent these birds respectively:

$$x_{i,j}^{t+1} = x_{i,j}^{t} + randn \times x_{i,j}^{t} \tag{10}$$

$$x_{i,j}^{t+1} = x_{i,j}^{t} + (x_{k,j}^{t} - x_{i,j}^{t}) \times FL \times rand_a \tag{11}$$

where *randn* is a random drawn number drawn from the Gaussian distribution with a mean 0 and standard deviation of 1, $k \in [1, \ldots, N]$ and $k \neq i$, $FL \in [0, 2]$. The last model is performed every FQ iterations.

The pseudocode of the BSA optimizer can be summarized as shown in Algorithm 1.

Algorithm 1 BSA optimizer pseudocode

```
Input:
N: Number of individuals
FQ: Frequency flight behaviors (producing and scrounging)
P: Foraging probability
C, S, a1, a2, FL: Constant parameters
MaxIter: Maximum number of iterations

Evaluate (Population)
Find best solution
procedure BSA(Population, FQ, P, MaxIter, .. )
    for Iter = 1 to MaxIter do
        if t % FQ ≠ 0 then
            for i = 1 to N do
                if rand ≤ P then
                    Birds forage for food
                else
                    Birds keep vigilance
                end if
            end for
        else
            Divide the population into producers and scroungers
            for i = 1 to N do
                if i is a producer then
                    Update producing birds
                else
                    Update scrounging birds
                end if
            end for
        end if
        Evaluate (Population)
        Find best solution
    end for
    return Best solution
end procedure
```

5 BSA for learning MLP

There are many meta-heuristic algorithms in the literature that are used to enhance the learning process of the MLP network. Meng et al. in [60] proved that the BSA is an efficient optimization algorithm for continuous functions. Furthermore, BSA has distinguished properties such as swarm integration, searching strategies, population diversity, and local optima avoidance. All of these properties encouraged us to integrate BSA with MLP neural network to optimize its learning process, which is discussed in this section.

In this paper, the BSA optimizer is used to find the optimal set of the network connections (weights and biases). Because there is no standard way for choosing the number of hidden nodes, the BSA uses fixed structure of MLP network such as the number of neurons is calculated based on the following equation:

$$m = 2 \times d + 1 \tag{12}$$

where m is the number of neurons; d is the number of data features or attributes.

Therefore, the total number of weights and biases (n) is calculated based on the following equation:

$$n = (d * m) + (2 * m) + 1 \tag{13}$$

In order to integrate the BSA optimizer with MLP networks, BSA individuals (birds) represent the weights and biases fractions. The bird is represented by a vector with n floating-point numbers. The bird representation and its mapping to an MLP network is shown in Fig. 2.

MLP learning can be accomplished by BSA optimizer by integrating the BSA operators with MLP network. The flowchart of the proposed learning approach is presented in Fig. 3. This process can be summarized in the following steps:

- Initialization: The proposed method starts by specifying the MLP structure such as the number of neurons (m), and the total number of weights and biases (n). Then, a random set of MLP networks (weights and biases) is generated, which represent N birds are initialized.
- Fitness evaluation: In this step, the fitness value for each bird is calculated using a fitness function and the training dataset. In this paper, we used the mean squared errors (MSE) in Eq. 14 as a fitness function.

$$MSE = \frac{1}{k}\sum_{i=1}^{k}(y_i - \hat{y_i})^2 \tag{14}$$

  where y_i is actual output of *ith* training sample; $\hat{y_i}$ is the predicted output of *ith* training sample; k is the total number of the training samples.
- Update: In order to train MLPs, the best global fitness (best global MLP), and best personal fitness for each bird are first updated. Each bird's vector is then updated based on the bird's status (foraging or vigilance). In addition, the birds will be divided into two groups (producing and scrounging) to enhance the diversity of the population. After that, the global fitness and its related solution will be updated.
- Termination: These steps are repeated until the maximum number of iterations is reached.

It is worth mentioning here that the best global solution (best MLP network) resulted from this iterative process is used to calculate MSE using the testing samples to make sure that the resulted MLP is applicable as a predictive model.

Based on the previous proposed approach steps, the BSA algorithm creates a set of new MLP networks considering the best MLP networks found so far. The process of calculating MSEs and improving the MLPs continues

until the satisfaction of the end criterion, which is the maximum iterations in this approach. It should be noted that the average MSE is calculated when classifying all training samples in the dataset for each MLP network in the proposed BSA-based trainer. Therefore, the computational complexity is of $O(ntd)$ where n is the number of random MLP networks, t indicates the maximum iterations, and d is the number of training samples in the dataset.

6 Experiments and results

In this section the BSA algorithm is evaluated on 13 classification datasets, and three function approximation benchmark datasets. In addition, the BSA-based learner is evaluated on a very challenging real world problem called Tennessee Eastman chemical process reactor (TECPR) problem [22].

The classification benchmark datasets are obtained from the University of California at Irvine (UCI) Machine Learning Repository [54]. The function approximation datasets are a one-dimensional sigmoid, one-dimensional sine with four peaks, and two-dimensional sphere. The classification datasets are divided into fixed ratio such as [2:1]; two folds for training, and one fold for testing. In the function approximation datasets, the training-testing ratio is [1:2]; one fold for training, and two folds for testing. Note that the training-testing ratio for TE problem is [1:1].

The BSA algorithm is compared to DE, GA, PSO, ACO, ES and ABC over these benchmark datasets in order to verify its performance. Furthermore, the comparison with gradient-based methods (backpropagation (BP) and Levenberg–Marquardt (LM) methods) are discussed.

All dataset features are normalized using the min–max method to the interval [0, 1]. To make fair comparisons, 30 runs are executed for each algorithm, each run is set to 250 iterations as stopping criteria. The number of birds, and individuals is set to 100 and are randomly initialized in the range [− 1,1]. Furthermore, all parameters and their initial values as used in our experiments for all algorithms are presented in Table 1 [33, 62, 79, 100].

In order to evaluate BSA-based learner and other algorithms, different evaluation measures are used based on the type of the experiment. For classification benchmark datasets, we used the mean squared error (MSE), which is given in Eq. 14, classification rate and Wilcoxons test over the 30 runs. Another performance indicator is classification rate which measures the rate of the correctly classified samples to the actual classes. For each experiment, average (AVE), standard deviation (STD), and Best of the

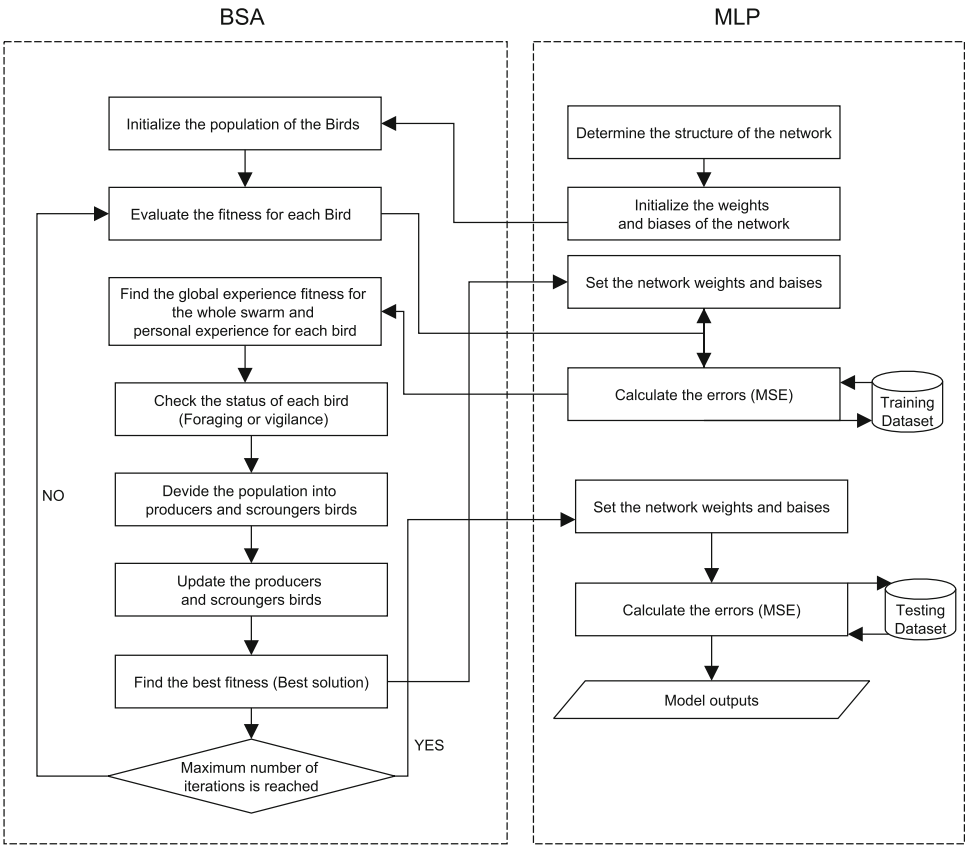


Fig. 3 Flow chart of the proposed learning algorithm (BSA-MLP)

Table 1 The initial parameters of the metaheuristic algorithms

Algorithm	Parameter	Value
GA	Crossover probability	0.9
	Mutation probability	0.1
	Selection mechanism	Stochastic sampling
DE	Crossover probability	0.9
	Differential weight	0.5
ES	λ	10
	σ	1
PSO	Acceleration constants	[2.1,2.1]
	Inertia weights	[0.9,0.6]
ACO	Initial pheromone (τ)	1e−06
	Pheromone update constant (Q)	20
	Pheromone constant (q)	1
	Global pheromone decay rate (p_g)	0.9
	Local pheromone decay rate (p_t)	0.5
	Pheromone sensitivity (α)	1
	Visibility sensitivity (β)	5
ABC	Acceleration coefficient upper bound	1

classification results are reported. Wilcoxons test is a nonparametric statistical test that used to check the significant difference of the given results. In this paper, the Wilcoxons test is calculated at 5% significance level against the calculated p-values.

For function approximation benchmark datasets, we used the MSE, test error (the mean absolute error (MAE)), and Wilcoxons test. MAE is computed using the following equation:

$$MAE = \frac{1}{k}\sum_{i=1}^{k} |y_i - \hat{y}_i| \tag{15}$$

where k is the total number of samples.

For TE problem, we used the MSE, test error (MAE), variance-accounted-for (VAF), and Wilcoxons test. These measures are used to evaluate how the predicted values are closed to the real values. VAF is computed by the following equation:

$$VAF = \left[1 - \frac{var(y_i - \hat{y}_i)}{var(y_i)}\right] \times 100\% \tag{16}$$

where var is the variance; y is the actual value; $\hat{y}$ is the estimated output value;

As the qualitative results, the algorithms' convergence curves are investigated to check the speed of the algorithms in achieving the optimal solutions. For the function approximation datasets, we draw the shape of functions approximated to qualitatively compare the training algorithms as well.

The results of benchmark datasets are illustrated and discussed in Sects. 6.1–6.3. In Sect. 6.1, BSA algorithm is evaluated on different classification datasets, Sect. 6.2 discusses the results of function approximation benchmark datasets, and Sect. 6.3 presents the results on the Tennessee Eastman chemical process reactor problem.

6.1 Classification datasets

The proposed BSA-based learning algorithm is evaluated using 13 popular classification datasets, which are selected from the UCI repository[1]. Table 2 shows the selected datasets with feature numbers, number of training and testing samples, and the used MLP structures. The evaluation results of the algorithms on these datasets are presented and discussed as follows:

- Breast dataset: the results of the BSA and other meta-heuristics learning algorithms for this dataset are presented in Table 3. The average classification rates show that BSA and PSO have the same results, and they outperform the other meta-heuristics. The standard deviations of the classification rates indicate that BSA has the best results, which means that the BSA is a robust algorithm compared to other algorithms. Furthermore, MSE results show that the results of BSA are very competitive . However, the best MSE and classification rates reported in Table 3 show that the BSA finds very closed solutions to the global optimum. In addition, the p-values of the statistical tests show that the differences between BSA results and DE, ACO, and ES are statistically significant, but not significant compared with PSO, GA, and ABC.

[1] http://archive.ics.uci.edu/ml/

Table 2 Summary of the classification datasets

No.	Dataset	#attributes	#train instances	#test instances	MLP structure
1	Breast	8	461	238	8–17–1
2	Liver	6	227	118	6–13–1
3	Diagnosis I	6	79	41	6–13–1
4	Diagnosis II	6	79	41	6–13–1
5	PlanningRelax	12	120	62	12–25–1
6	Diabetes	8	506	262	8–17–1
7	Haberman	3	201	105	3–7–1
8	Hepatitis	10	102	53	10–21–1
9	Heart	13	178	92	13–27–1
10	Phoneme	5	3566	1838	5–11–1
11	Saheart	9	304	158	9–19–1
12	Spectf	44	176	91	44–89–1
13	Vertebral	6	204	106	6–13–1

Table 3 Classification rate, p-values, and MSE results for breast cancer dataset

Algorithm	Classification rate (AVE ± STD)[Best]	p-values	MSE (AVE ± STD)[Best]
BSA	0.9703 ± 0.0054[0.9790]	*6.61E−01*	3.17E−02 ± 2.25E−03[2.75E−02]
DE	0.9478 ± 0.0153[0.9748]	1.03E−12	5.18E−02 ± 6.41E−03[3.86E−02]
GA	0.9675 ± 0.0058[0.9748]	*6.82E−02*	2.84E−02 ± 9.66E−04[2.66E−02]
PSO	**0.9704** ± 0.0075[0.9790]	N/A	3.53E−02 ± 1.98E−03[3.00E−02]
ACO	0.9246 ± 0.0335[0.9664]	1.58E−09	7.20E−02 ± 1.38E−02[4.75E−02]
ES	0.9667 ± 0.0059[0.9790]	2.53E−02	3.71E−02 ± 1.60E−03[3.36E−02]
ABC	0.9689 ± 0.0079[0.9832]	*3.88E−01*	3.37E−02 ± 1.02E−03[3.13E−02]

The best results are marked in bold

- Liver dataset: The results of learning algorithms on Liver dataset are presented in Table 4. Inspecting the results of the different measures, it is evident that BSA has the best ability to avoid local optima for this dataset. Moreover, BSA has 70.28% classification rate, which outperforms all the other learning algorithms. In addition, the p-values of the statistical tests show that the differences between BSA results and GA are not statistically significant, but they significantly outperform others combined.
- Diagnosis I and Diagnosis II datasets: The experimental results of these two datasets are given in Tables 5 and 6, respectively. According to the classification rate results, the BSA obtained 100% classification rate for the two datasets, which are similar to GA, PSO, ES, and ABC, and better than DE and ACO results. However, the p-values show that there is no statistically significant difference between BSA and other four learning algorithms. This means that BSA provides very competitive results on these two datasets.
- PlanningRelax dataset: the experimental results of this dataset are shown in Table 7. It might be seen in this table that the p-values results of BSA are significantly outperform most of the other algorithms. Moreover, BSA has the highest classification rate with comparable MSE results.
- Diabetes dataset: the evaluation results of this dataset are shown in Table 8. As per the results of classification rates in this table, BSA provides the highest results with comparable MSE results. The p-values of BSA are significantly outperform four of the other algorithms.
- Haberman dataset: The results of learning algorithms on Haberman dataset are shown in Table 9. The results on this dataset reveal that BSA has the best performances in terms of classification rates with 73.05%. The average and standard deviation results of MSEs show that the efficiency of the BSA and GA is very close, and better than others. Furthermore, the p-values of the statistical tests show that the differences between BSA results and most of the other algorithms are statistically significant.
- Hepatitis dataset: the results of this dataset are provided in Table 10. The observed results for this dataset indicate that GA has the best average classification rates, but with no statistically significant difference compared to the results of BSA. Moreover, the MSE

results of the GA and BSA are very close, and outperform all of other algorithms.

- Heart dataset: the results of this dataset are reported in Table 11. This table shows the superiority of the BSA algorithm in term of the classification rate. The lowest MSE results show the ability of the BSA to avoid local optima. Also, the p-values show that the BSA results are highly statistically significant compared with all of other algorithms.
- Phoneme dataset: the results of Phoneme dataset are presented in Table 12. It should be noted from the results of this dataset that the classification rate of the BSA is very close to ABC algorithm which has the highest rate, and both of them outperform other algorithms. Furthermore, the MSE results are very competitive to the other algorithms.
- SAheart dataset: the results of the BSA and other algorithms on SAheart dataset are shown in Table 13. The BSA has the best classification rate with 73.02%. The AVE, STD, and Best results of MSEs show that the efficiency of the BSA to avoid local optima. Furthermore, the p-values of the statistical tests show that BSA results are statistically significant in comparison with GA, DE, ACO, and ABC algorithms.
- Spectf dataset: the results of Spectf dataset are presented in Table 14. The table results show the predominance of the BSA algorithm in term of the classification rate and MSE measures. However, the p-values show that the BSA results are statistically significant compared the majority of other algorithms.
- Vertebral dataset: the results of the BSA and other meta-heuristics optimizers on this dataset are presented in Table 15. The average classification rates show that BSA outperforms other meta-heuristics. The standard deviations of the classification rates indicate that BSA provide very competitive results, which means that the BSA's performance is very stable. Furthermore, average of MSE and Best MSE results reported in Table 15 show that the BSA is able to find very accurate approximation of the global optimum. In addition, the p-values of the BSA and other algorithms show that the differences in the results are statistically significant.

As the qualitative results, Fig. 4 shows the convergence curves of BSA, DE, GA, PSO, ACO, ES, and ABC based on averages of the MSE for all classification datasets over 30 independent runs. These convergence curves prove that BSA has acceptable convergence rate on the majority of the datasets.

The BSA algorithm is compared with the two popular gradient-based learning methods as well: backpropagation (BP) and Levenberg–Marquardt (LM). These two methods are based on mathematical representation that employ the derivatives and gradients to learn the MLPs network. The BP, and LM results in terms of classification rate, MSE, and p-values are reported in the Table 16. It can be seen that the average and standard deviations results of the BSA algorithm are better than BP, and LM in all datasets. The results show that the BSA results are statistically significant compared to gradient based trainer. Also, the BSA has a superior ability to avoid the local optima and achieve close solutions to the global optimum.

6.2 Function approximation datasets

The proposed BSA-based learning algorithm is evaluated using three popular function approximation datasets as well. Table 17 shows the selected function approximations with function formula, number of training and testing samples, dimensions, and MLP structure. The evaluation results of the algorithms on these datasets are as follows:

- Sigmoid dataset: the results of MLP learning algorithms for this function approximation dataset are presented in Table 18. The results in the table show that the average MSE, and the average Test Error of the BSA algorithm outperform other algorithms. In addition, the p-values indicate that the BSA has statistically significant results and it has highly ability to avoid the local minima compared with all other learners. To qualitatively compare the algorithms, Fig. 5 is given that shows the BSA owns the most accurate approximation curve for the Sigmoid function compared with the actual curve.
- Sine dataset: the results of the BSA and other algorithms for Sine dataset are reported in Table 19. Inspecting these results, it may be observed that the Test Error of the BSA algorithm is better than other algorithms and shows competitive MSE results. Moreover, the p-values indicate that the BSA is not statistically significant superiority compared to the GA algorithm. However, BSA is statistically better than other algorithms. Ina addition, Fig. 6 show that the approximation curves of the Sine function of different algorithms.This figure shows that none of the algorithms finds an accurate shape. This is due to the difficulty of this function and several fluctuations in the curve. The accuracy of algorithm can be improved by tuning parameters and increasing the number of nodes. However, the main focus was on the comparison of the algorithms under fair conditions and fine-tuning of algorithm is out of the scope of this work.
- Sphere dataset: The results of the Sphere function dataset are shown in Table 20. The experimental results show that the BSA outperforms other algorithms in terms of Test error and MSE measures. Furthermore, the BSA results based on the p-values are statistically significant. The approximation curves in Fig. 7 verify the accuracy of BSA algorithm as well.

Table 4 Classification rate, p-values, and MSE results for liver dataset

Algorithm	Classification rate (AVE ± STD)[Best]	p-values	MSE (AVE ± STD)[Best]
BSA	**0.7028** ± 0.0340[0.7542]	N/A	2.08E−01 ± 3.96E−03[2.00E−01]
DE	0.6121 ± 0.0509[0.7288]	1.17E−12	2.31E−01 ± 5.06E−03[2.22E−01]
GA	0.6949 ± 0.0302[0.7627]	*2.02E−01*	2.04E−01 ± 4.23E−03[1.95E−01]
PSO	0.6788 ± 0.0449[0.7881]	1.95E−02	2.16E−01 ± 3.20E−03[2.07E−01]
ACO	0.5819 ± 0.0354[0.6441]	4.01E−11	2.38E−01 ± 5.16E−03[2.28E−01]
ES	0.6596 ± 0.0481[0.7458]	3.25E−04	2.20E−01 ± 3.48E−03[2.14E−01]
ABC	0.6695 ± 0.0508[0.7881]	4.32E−03	2.14E−01 ± 4.29E−03[2.06E−01]

The best results are marked in bold

Table 5 Classification rate, p-values, and MSE results for diagnosis I dataset

Algorithm	Classification rate (AVE ± STD)[Best]	p-values	MSE (AVE ± STD)[Best]
BSA	**1.0** ± 0.0[1.0]	N/A	7.57E−04 ± 1.78E−03[3.19E−06]
DE	0.9341 ± 0.0690[1.0]	1.69E−14	4.11E−02 ± 1.63E−02[1.32E−02]
GA	**1.0** ± 0.0[1.0]	N/A	2.05E−06 ± 3.04E−06[1.23E−07]
PSO	**1.0** ± 0.0[1.0]	N/A	3.91E−03 ± 2.35E−03[2.68E−04]
ACO	0.8626 ± 0.1112[1.0]	5.06E−11	8.09E−02 ± 2.81E−02[1.07E−02]
ES	**1.0** ± 0.0[1.0]	N/A	3.75E−03 ± 2.70E−03[3.71E−04]
ABC	**1.0** ± 0.0[1.0]	N/A	0.0 ± 0.0+0.0[0.0]

The best results are marked in bold

Table 6 Classification rate, p-values, and MSE results for diagnosis II dataset

Algorithm	Classification rate (AVE ± STD)[Best]	p-values	MSE (AVE ± STD)[Best]
BSA	**1.0** ± 0.0[1.0]	N/A	1.76E−04 ± 1.85E−04[8.43E−06]
DE	0.9593 ± 0.0552[1.0]	1.69E−14	2.70E−02 ± 1.79E−02[2.17E−03]
GA	**1.0** ± 0.0[1.0]	N/A	1.33E−07 ± 2.70E−07[1.86E−08]
PSO	**1.0** ± 0.0[1.0]	N/A	1.13E−03 ± 7.27E−04[1.45E−05]
ACO	0.8789 ± 0.0940[1.0]	1.87E−10	7.39E−02 ± 2.84E−02[1.79E−02]
ES	**1.0** ± 0.0[1.0]	N/A	9.13E−04 ± 6.44E−04[1.45E−04]
ABC	**1.0** ± 0.0[1.0]	N/A	0.0 ± 0.0[0.0]

The best results are marked in bold

Table 7 Classification rate, p-values, and MSE results for PlanningRelax dataset

Algorithm	Classification rate (AVE ± STD)[Best]	p-values	MSE (AVE ± STD)[Best]
BSA	**0.6511** ± 0.0177[0.6935]	N/A	1.77E−01 ± 3.42E−03[1.68E−01]
DE	0.6355 ± 0.0253[0.6774]	8.38E−13	1.90E−01 ± 3.68E−03[1.82E−01]
GA	0.6301 ± 0.0236[0.6613]	3.44E−04	1.70E−01 ± 5.02E−03[1.56E−01]
PSO	0.6435 ± 0.0226[0.6774]	*2.53E−01*	1.81E−01 ± 1.20E−03[1.78E−01]
ACO	0.6317 ± 0.0258[0.6774]	1.37E−03	1.95E−01 ± 3.86E−03[1.89E−01]
ES	0.6183 ± 0.0350[0.6774]	2.88E−05	1.89E−01 ± 4.38E−03[1.79E−01]
ABC	0.6237 ± 0.0337[0.6613]	4.29E−04	1.74E−01 ± 2.44E−03[1.68E−01]

The best results are marked in bold

Table 8 Classification rate, p-values, and MSE results for diabetes dataset

Algorithm	Classification rate (AVE ± STD)	p-values	MSE (AVE ± STD)[Best]
BSA	**0.7525** ± 0.0172[0.7824]	N/A	1.54E−01 ± 2.76E−03[1.49E−01]
DE	0.7051 ± 0.0314[0.7786]	1.17E−12	1.82E−01 ± 6.56E−03[1.71E−01]
GA	0.7517 ± 0.0121[0.7824]	*7.72E−01*	1.52E−01 ± 2.56E−03[1.46E−01]
PSO	0.7310 ± 0.0248[0.7824]	4.20E−04	1.64E−01 ± 3.05E−03[1.55E−01]
ACO	0.6819 ± 0.0375[0.7634]	2.34E−09	1.92E−01 ± 9.78E−03[1.64E−01]
ES	0.7257 ± 0.0261[0.8015]	8.81E−06	1.75E−01 ± 4.30E−03[1.60E−01]
ABC	0.7482 ± 0.0198[0.8053]	*3.17E−01*	1.64E−01 ± 3.39E−03[1.56E−01]

The best results are marked in bold

Table 9 Classification rate, p-values, and MSE results for Haberman dataset

Algorithm	Classification rate (AVE ± STD)[Best]	p-values	MSE (AVE ± STD)[Best]
BSA	**0.7305** ± 0.0104[0.7524]	N/A	1.63E−01 ± 2.52E−03[1.60E−01]
DE	0.7222 ± 0.0094[0.7429]	7.69E−13	1.77E−01 ± 3.20E−03[1.70E−01]
GA	0.7248 ± 0.0088[0.7429]	3.06E−02	1.62E−01 ± 1.77E−03[1.58E−01]
PSO	0.7283 ± 0.0078[0.7524]	*4.45E−01*	1.67E−01 ± 1.49E−03[1.64E−01]
ACO	0.7238 ± 0.0087[0.7333]	2.64E−02	1.70E−01 ± 1.08E−03[1.68E−01]
ES	0.7298 ± 0.0113[0.7524]	*9.32E−01*	1.73E−01 ± 1.67E−03[1.68E−01]
ABC	0.7241 ± 0.0155[0.7524]	*1.05E−01*	1.65E−01 ± 2.29E−03[1.57E−01]

The best results are marked in bold

Table 10 Classification rate, p-values, and MSE results for Habitit dataset

Algorithm	Classification rate (AVE ± STD)[Best]	p-values	MSE (AVE ± STD)[Best]
BSA	0.8535 ± 0.0312[0.9057]	*1.17E−01*	8.28E−02 ± 8.58E−03[6.82E−02]
DE	0.8535 ± 0.0246[0.9057]	9.11E−13	1.19E−01 ± 5.61E−03[1.07E−01]
GA	**0.8660** ± 0.0245[0.9245]	N/A	7.09E−02 ± 5.41E−03[6.06E−02]
PSO	0.8484 ± 0.0264[0.9057]	1.05E−02	9.79E−02 ± 3.05E−03[9.08E−02]
ACO	0.8465 ± 0.0300[0.9057]	1.20E−02	1.30E−01 ± 9.87E−03[1.07E−01]
ES	0.8440 ± 0.0357[0.8868]	1.77E−02	1.03E−01 ± 4.10E−03[9.50E−02]
ABC	0.8346 ± 0.0327[0.8868]	1.58E−04	9.65E−02 ± 3.04E−03[8.94E−02]

The best results are marked in bold

Table 11 Classification rate, p-values, and MSE results for heart dataset

Algorithm	Classification rate (AVE ± STD)[Best]	p-values	MSE (AVE ± STD)[Best]
BSA	**0.8362** ± 0.0245[0.8804]	N/A	1.08E−01 ± 5.79E−03[9.86E−02]
DE	0.7819 ± 0.0382[0.8587]	1.08E−12	1.52E−01 ± 1.29E−02[1.32E−01]
GA	0.8232 ± 0.0180[0.8587]	1.04E−02	8.60E−02 ± 6.12E−03[7.63E−02]
PSO	0.8188 ± 0.0264[0.8696]	9.37E−03	1.21E−01 ± 3.52E−03[1.11E−01]
ACO	0.7819 ± 0.0164[0.8261]	1.81E−09	1.63E−01 ± 7.34E−03[1.47E−01]
ES	0.8069 ± 0.0270[0.8587]	4.11E−05	1.37E−01 ± 7.81E−03[1.15E−01]
ABC	0.8185 ± 0.0313[0.8587]	2.73E−02	1.21E−01 ± 5.06E−03[1.09E−01]

The best results are marked in bold

Figure 8 shows the convergence curves of BSA, DE, GA, PSO, ACO, ES, and ABC based on averages of the MSE for all function approximation datasets. These convergence curves qualitatively show that BSA provides the fastest convergence rate on all function approximation datasets.

Table 12 Classification rate, p-values, and MSE results for Phoneme dataset

Algorithm	Classification rate (AVE ± STD)[Best]	p-values	MSE (AVE ± STD)[Best]
BSA	0.7678 ± 0.0078[0.7818]	1.60E−06	1.54E−01 ± 2.05E−03[1.49E−01]
DE	0.7514 ± 0.0203[0.7933]	1.21E−12	1.66E−01 ± 4.83E−03[1.57E−01]
GA	0.7722 ± 0.0049[0.7824]	1.23E−05	1.52E−01 ± 1.49E−03[1.48E−01]
PSO	0.7609 ± 0.0106[0.7835]	5.98E−08	1.57E−01 ± 1.00E−03[1.55E−01]
ACO	0.7391 ± 0.0225[0.7840]	2.01E−09	1.67E−01 ± 3.83E−03[1.63E−01]
ES	0.7637 ± 0.0125[0.7829]	6.68E−07	1.57E−01 ± 2.29E−03[1.52E−01]
ABC	**0.7816** ± 0.0106[0.8009]	N/A	1.51E−01 ± 2.77E−03[1.44E−01]

The best results are marked in bold

Table 13 Classification rate, p-values, and MSE results for SAheart dataset

Algorithm	Classification rate (AVE ± STD)[Best]	p-values	MSE (AVE ± STD)[Best]
BSA	**0.7302** ± 0.0159[0.7595]	N/A	1.73E−01 ± 2.92E−03[1.65E−01]
DE	0.7034 ± 0.0340[0.7658]	1.14E−12	1.96E−01 ± 6.14E−03[1.79E−01]
GA	0.7181 ± 0.0176[0.7595]	9.20E−03	1.69E−01 ± 2.30E−03[1.65E−01]
PSO	0.7266 ± 0.0277[0.7785]	*6.56E−01*	1.80E−01 ± 2.03E−03[1.76E−01]
ACO	0.6863 ± 0.0471[0.7595]	7.09E−05	2.06E−01 ± 8.37E−03[1.89E−01]
ES	0.7148 ± 0.0358[0.7595]	*1.20E−01*	1.94E−01 ± 3.84E−03[1.86E−01]
ABC	0.7116 ± 0.0212[0.7658]	3.23E−04	1.82E−01 ± 2.32E−03[1.76E−01]

The best results are marked in bold

Table 14 Classification rate, p-values, and MSE results for Spectf dataset

Algorithm	Classification rate (AVE ± STD)[Best]	p-values	MSE (AVE ± STD)[Best]
BSA	**0.7850** ± 0.0201[0.8242]	N/A	1.19E−01 ± 6.16E−03[1.06E−01]
DE	0.7714 ± 0.0179[0.8022]	1.05E−12	1.53E−01 ± 6.05E−03[1.38E−01]
GA	0.7736 ± 0.0221[0.8132]	3.03E−02	9.63E−02 ± 6.94E−03[8.23E−02]
PSO	0.7641 ± 0.0284[0.8022]	1.31E−03	1.30E−01 ± 4.46E−03[1.18E−01]
ACO	0.7648 ± 0.0259[0.8022]	7.04E−04	1.59E−01 ± 8.73E−03[1.35E−01]
ES	0.7689 ± 0.0164[0.7912]	9.80E−04	1.39E−01 ± 4.53E−03[1.31E−01]
ABC	0.7780 ± 0.0183[0.8352]	3.44E−02	1.29E−01 ± 3.84E−03[1.20E−01]

The best results are marked in bold

Table 15 Classification rate, p-values, and MSE results for vertebral dataset

Algorithm	Classification rate (AVE ± STD)[Best]	p-values	MSE (AVE ± STD)[Best]
BSA	**0.8679** ± 0.0238[0.9057]	N/A	1.10E−01 ± 6.85E−03[9.81E−02]
DE	0.7752 ± 0.0401[0.8679]	1.10E−12	1.54E−01 ± 1.19E−02[1.34E−01]
GA	0.8579 ± 0.0105[0.8868]	2.51E−03	9.83E−02 ± 1.13E−03[9.59E−02]
PSO	0.8503 ± 0.0335[0.9057]	1.56E−02	1.20E−01 ± 4.13E−03[1.13E−01]
ACO	0.7358 ± 0.0412[0.8491]	4.59E−11	1.70E−01 ± 1.43E−02[1.40E−01]
ES	0.8321 ± 0.0298[0.8774]	5.41E−06	1.18E−01 ± 3.53E−03[1.06E−01]
ABC	0.8437 ± 0.0327[0.9057]	5.69E−04	1.06E−01 ± 5.21E−03[9.66E−02]

The best results are marked in bold

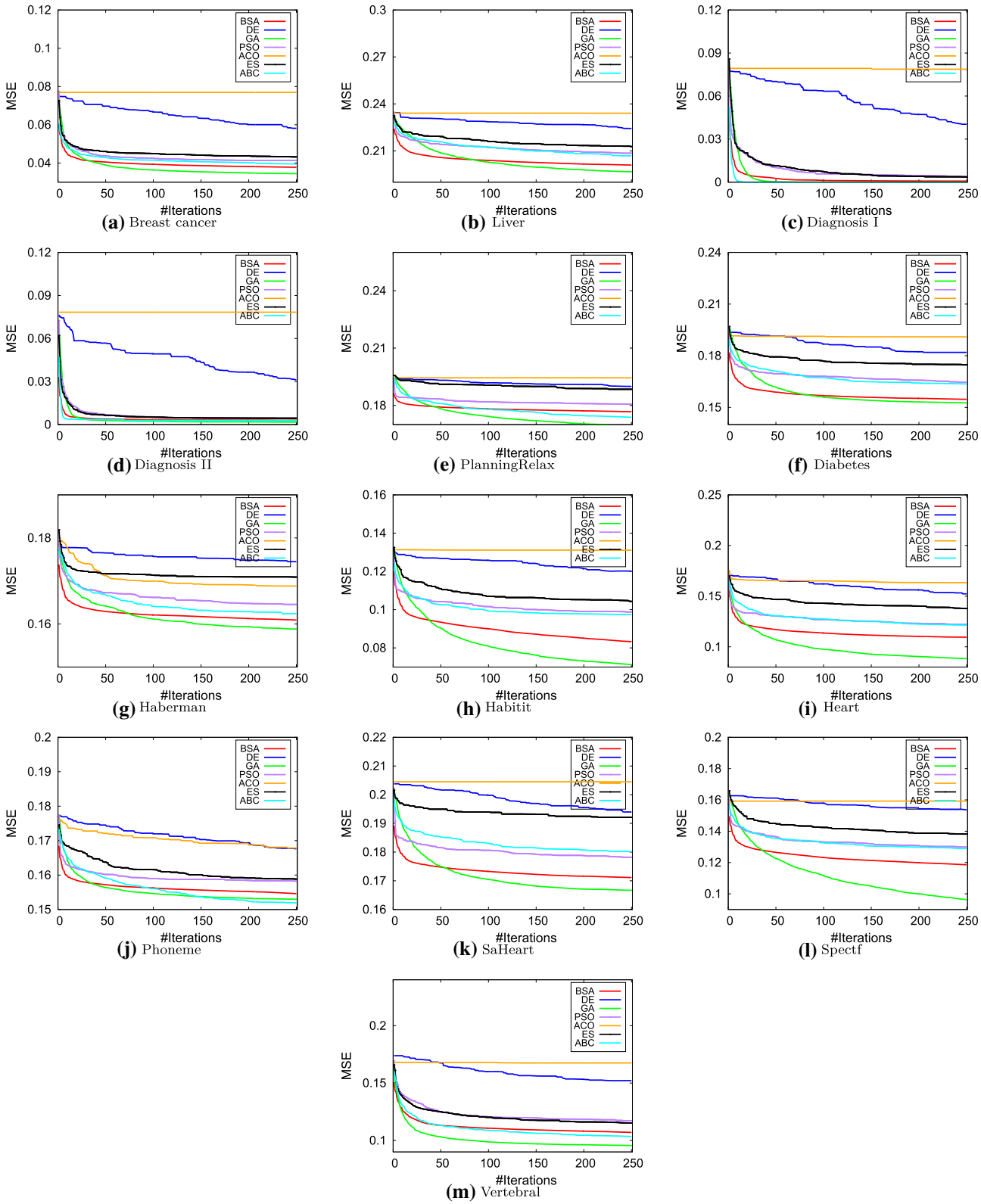


Fig. 4 Convergence curves for all datasets

Table 16 Classification rates, p-values, and MSEs, for BSA, BP, and LM, learners on all datasets

Dataset/optimizer	BSA	BP	LM
Breast			
C. Rate (AVE ± STD)[Best]	**0.9703** ± 0.0054[0.9790]	0.8576 ± 0.0977[0.9748]	0.9458 ± 0.0164[0.9706]
p-values	N/A	1.01E−12	5.48E−09
MSE (AVE ± STD)	3.17E−02 ± 2.25E−03	1.22E−01 ± 7.49E−02	1.11E−02 ± 1.38E−02
Liver			
C. Rate (AVE ± STD)[Best]	**0.7028** ± 0.0340[0.6441]	0.5395 ± 0.0531[0.6441]	0.6559[0.7712] ± 0.0515
p-values	N/A	1.17E−12	2.19E−04
MSE (AVE ± STD)	2.08E−01 ± 3.96E−03	2.80E−01 ± 6.40E−02	1.18E−01 ± 8.84E−02
Diagnosis I			
C. Rate (AVE ± STD)[Best]	**1.0000** ± 0.0000[1.0000]	0.7390 ± 0.1572[1.0000]	0.9951 ± 0.0267[1.0000]
p-values	N/A	1.69E−14	3.34E−01
MSE (AVE ± STD)	7.57E−04 ± 1.78E−03	1.72E−01 ± 1.14E−01	6.33E−03 ± 3.47E−02
Diagnosis II			
C. Rate (AVE ± STD)[Best]	**1.0000** ± 0.0000[1.0000]	0.7756 ± 0.1323[1.0000]	0.9837 ± 0.0619[1.0000]
p-values	N/A	1.69E−14	1.61E−01
MSE (AVE ± STD)	1.76E−04	1.76E−01 ± 1.08E−01	1.31E−02 ± 5.20E−02
PlanningRelax			
C. Rate (AVE ± STD)[Best]	**0.6511** ± 0.0177 [0.6935]	0.6048 ± 0.0784 [0.6613]	0.5478 ± 0.0636 [0.6452]
p-values	N/A	8.38E−13	4.42E−10
MSE (AVE ± STD)	1.77E−01 ± 3.42E−03	2.41E−01 ± 1.17E−01	3.89E−02 ± 5.52E−02
Diabetes			
C. Rate (AVE ± STD)[Best]	**0.7525** ± 0.0172 [0.7824]	0.6391 ± 0.0361 [0.7214]	0.7270 ± 0.0377 [0.7901]
p-values	N/A	1.17E−12	8.96E−04
MSE (AVE ± STD)	1.54E−01 ± 2.76E−03	2.37E−01 ± 3.36E−02	1.35E−01 ± 7.59E−02
Haberman			
C. Rate (AVE ± STD)[Best]	**0.7305** ± 0.0104 [0.7524]	0.6876 ± 0.0862 [0.7524]	0.6883 ± 0.0388 [0.7524]
p-values	N/A	7.69E−13	9.98E−08
MSE (AVE ± STD)	1.63E−01 ± 2.52E−03	2.20E−01 ± 6.55E−02	1.37E−01 ± 4.69E−02
Habitit			
C. Rate (AVE ± STD)[Best]	**0.8535** ± 0.0312 [0.9057]	0.8063 ± 0.0652 [0.8868]	0.8252 ± 0.0467 [0.9057]
p-values	N/A	1.08E−12	2.01E−02
MSE (AVE ± STD)	8.28E−02 ± 8.58E−03	1.72E−01 ± 4.47E−02	2.29E−02 ± 5.62E−02
Heart			
C. Rate (AVE ± STD)[Best]	**0.8362** ± 0.0245 [0.8804]	0.6826 ± 0.0898 [0.8043]	0.7572 ± 0.0406 [0.8152]
p-values	N/A	1.08E−12	8.40E−10
MSE (AVE ± STD)	1.08E−01 ± 5.79E−03	2.23E−01 ± 8.14E−02	2.87E−02 ± 6.26E−02
Phoneme			
C. Rate (AVE ± STD)[Best]	0.7678 ± 0.0078 [0.7818]	0.6999 ± 0.0334 [0.7519]	**0.8308** ± 0.0165 [0.8624]
p-values	8.08E−11	1.20E−12	N/A
MSE (AVE ± STD)	1.54E−01 ± 2.05E−03	2.20E−01 ± 3.17E−02	1.12E−017.94E−03
Saheart			
C. Rate (AVE ± STD)[Best]	**0.7302** ± 0.0159 [0.7595]	0.6411 ± 0.0737 [0.7405]	0.6793 ± 0.0298 [0.7532]
p-values	N/A	1.14E−12	1.76E−08
MSE (AVE ± STD)	1.73E−012.92E−03	2.53E−018.02E−02	1.09E−014.60E−02
Spectf			
C. Rate (AVE ± STD)[Best]	**0.7850** ± 0.0201 [0.8242]	0.7670 ± 0.0256 [0.8022]	0.7560 ± 0.0259 [0.8022]
p-values	N/A	1.05E−12	4.97E−05
MSE (AVE ± STD)	1.19E−016.16E−03	1.68E−012.26E−02	1.80E−025.17E−02

Table 16 (continued)

Dataset/optimizer	BSA	BP	LM
Vertebral			
C. Rate (AVE ± STD)[Best]	**0.8679** ± 0.0238 [0.9057]	0.6513 ± 0.1011 [0.7925]	0.8028 ± 0.0550 [0.8585]
p-values	N/A	1.10E−12	1.12E−08
MSE (AVE ± STD)	1.10E−016.85E−03	2.58E−018.91E−02	9.41E−021.16E−01

The best results are marked in bold

Table 17 Function approximation datasets

Functions	Training samples	Testing samples	Dim	MLP structure
Sigmoid: $y = \frac{1}{1+e^{-x}}$	$61 : x\,in\,[-3.0 : 0.1 : 3.0]$	$121 : x\,in\,[-3.0 : 0.05 : 3.0]$	1	1–3–1
Sine: $y = sin(2x)$	$126 : x\,in\,[-2\pi : 0.1 : 2\pi]$	$252 : x\,in\,[-2\pi : 0.05 : 2\pi]$	1	1–3–1
Sphere: $z = \frac{1}{2}\sum_{i=1}^{2}(x_i)^2; x = x_1, y = x_2$	$21 \times 21 : x, y\,in\,[-2 : 0.2 : 2]$	$41 \times 41 : x, y\,in\,[-2 : 0.1 : 2]$	2	2–5–1

Table 18 MSE and test error (MAE) results for Sigmoid dataset

Algorithm	MSE (AVE ± STD)	p-values	Test Error (AVE ± STD)
BSA	**6.41E−05 ± 6.41E−05**	N/A	**0.691553 ± 0.357952**
GA	0.000358 ± 0.000192	6.12E−10	1.859383 ± 0.542019
DE	0.002191 ± 0.001415	3.02E−11	4.517302 ± 1.371452
ES	0.002767 ± 0.001140	3.02E−11	5.045192 ± 1.061819
PSO	0.000498 ± 0.000197	6.07E−11	2.196162 ± 0.451043
ACO	0.000441 ± 7.11E−05	2.96E−11	2.184762 ± 0.231517
ABC	0.000635 ± 0.000321	6.70E−11	2.386619 ± 0.588302

The best results are marked in bold

Finally, the BP, and LM results for different function approximation datasets in terms of Test error, MSE, and p-values are reported in Table 21. The results show that the BSA results are competitive compared with BP and LM learners.

As an overall summary for the classification and function approximation datasets, Table 22 shows the results of all comparative measures; namely, the classification rate/test error, and p-values. The table values represent the number of datasets each algorithm won/losses/ties on a variety of measures. It appears that the BSA algorithm is superior on 11 datasets out of 16 in terms of classification rate or test error. Moreover, in term of significant measure based on p-values, the results shows that the BSA are better on 10 datasets out of 16. It appears that the BSA algorithm ranked first 21 out of 32 times. We also used Friedman test to rank the different algorithms applied on 13 classification and 3 function approximation datasets. The results of Friedman test in the last row of Table 22 show that the BSA obtains the best rank. This confirms the ability of the BSA algorithm to evolve the MLP network. All of these results provide a solid evidence to support the BSA algorithm for training MLPs.

6.3 Tennessee Eastman chemical process reactor (TE) problem

The TE chemical process was first presented by [22] as an academic research process. The process was a simulation of an actual system at the Tennessee Eastman Company, USA. It is considered as a large-scale nonlinear, open-loop unstable system with both fast and slow variable dynamics [43]. This makes it a challenge process for both system identification and control. A simplified diagram of the process is shown in Fig. 9. The TE chemical plant consists of five major operations: a two phase reactor, a product condenser, a vapor/liquid separator, a recycle compressor, and a product stripper. The nonlinear dynamics of the plant are mainly due to the chemical reactions within the reactor. In 1995, N. L. Ricker provided a TE archive of software simulation for the process. This archive was updated in 2005 [74]. The process is still interesting although it was

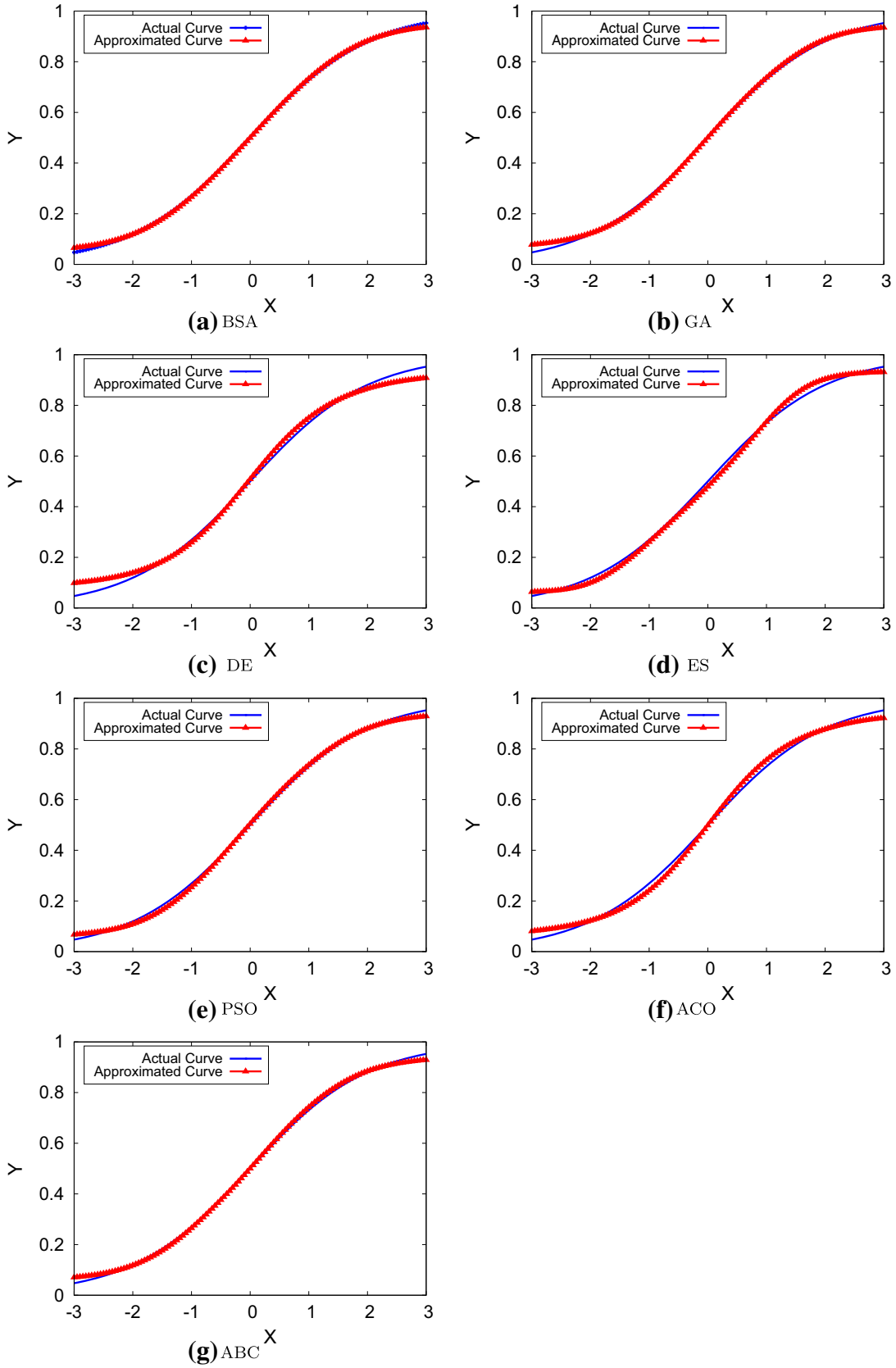
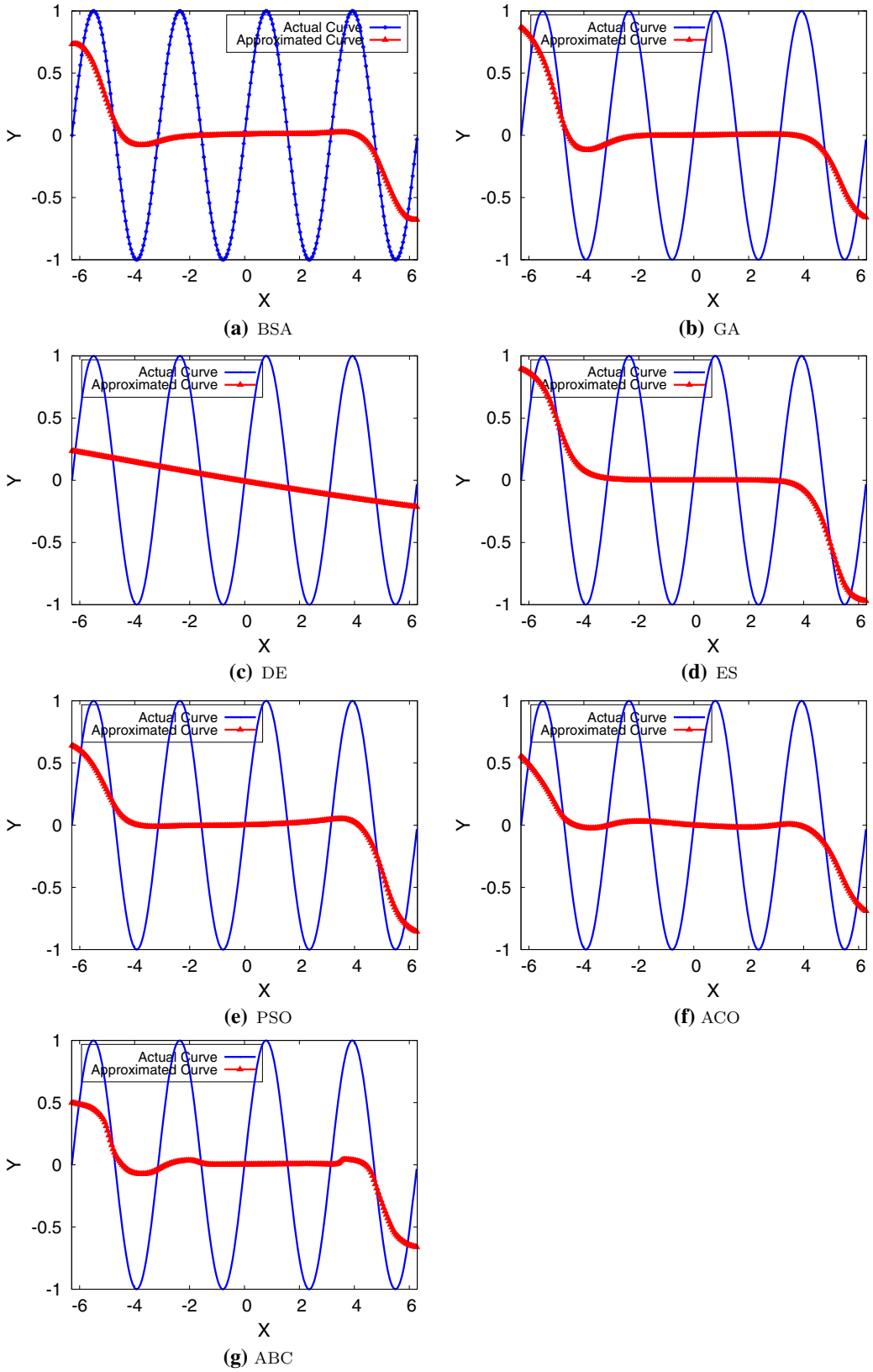


Fig. 5 Approximated curves versus actual curves for Sigmoid function



(a) BSA **(b)** GA **(c)** DE **(d)** ES **(e)** PSO **(f)** ACO **(g)** ABC

Fig. 6 Approximated curves versus actual curves for sine function

Table 19 MSE and test error (MAE) results for Sine dataset

Algorithm	MSE (AVE ± STD)	p-values	Test Error (AVE ± STD)
BSA	0.426660 ± 0.022908	*0.61*	**142.8721 ± 5.914724**
GA	**0.426332 ± 0.018264**	N/A	143.2926 ± 4.845956
DE	0.482856 ± 0.002272	3.02E−11	155.2637 ± 0.794425
ES	0.450017 ± 0.005971	3.47E−10	145.6933 ± 1.405172
PSO	0.445715 ± 0.010231	2.20E−07	145.6614 ± 2.850955
ACO	0.453036 ± 0.010317	2.67E−09	148.6939 ± 3.249246
ABC	0.445424 ± 0.015359	2.00E−06	146.0510 ± 4.223806

The best results are marked in bold

Table 20 MSE and test error (MAE) results for Sphere dataset

Algorithm	MSE (AVE ± STD)	p-values	Test Error (AVE ± STD)
BSA	**0.129917 ± 0.024166**	N/A	**12.1338 ± 1.061407**
GA	0.148478 ± 0.028735	0.0112	12.2941 ± 1.180819
DE	3.506696 ± 0.902888	3.02E−11	63.3252 ± 9.598388
ES	0.697140 ± 0.254678	3.02E−11	26.5494 ± 5.106035
PSO	0.294729 ± 0.063154	3.69E−11	17.1874 ± 2.095087
ACO	2.434236 ± 0.906094	3.02E−11	48.8703 ± 11.375460
ABC	0.506430 ± 0.154559	3.02E−11	22.3203 ± 3.733071

The best results are marked in bold

presented almost three decades ago. In [14] Ricker provided a revision of his original TE process model presented in [43].

Tennessee Eastman chemical process reactor was explored in a number of publications [2, 73]. The reactor process is decomposed of four subsystems: reactor level, reactor pressure, reactor cooling water temperature, and reactor temperature subsystems. The TE chemical reactor process, given in Fig. 10, was simulated for control purposes in [72, 77]. The use of BP neural network for modeling the TE chemical process was proposed in 1990. Bhat and McAvoy [16] were among the first whom used ANNs for modeling nonlinear chemical processes. They show that two layer BP-ANN is equivalent to the procedure of impulse response convolution modeling of linear systems.

However, the standard BP learning algorithm suffers from many drawbacks such as slow convergence, lack of robustness, and inefficiency [10, 50]. To address the slow convergence rate problem of the BP learning algorithm, many researchers proposed the use of conjugate gradient method to provide a faster convergence as given in [46, 51, 67]. In this section, we explore the use of BSA algorithm to learn the MLP to solve TE problem. Table 23 shows the summary of the reactor sub-problems with the number of the training and testing samples, and MLP structures that are used in this experiment.

The experimental results of modeling the TE reactor for Level, Pressure, Cooling, and Temperature sub-problems using BSA-MLP optimizer and other meta-heuristics optimizes algorithms are reported in Tables 24, 25, 26, and 27, respectively.

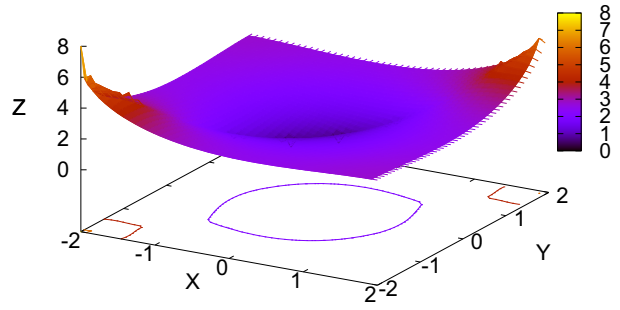
Inspecting the results presented in Tables 24, 25, 26, and 27, it can be noted that the average VAF, MSE, and MAE results show that BSA outperforms other meta-heuristics in Level, Pressure, and Cooling reactors, and it provides very competitive results in the Temperature reactor. In addition, the p-values of the statistical tests show that the differences between BSA results and the majority of other meta-heuristics are statistically significant. These results prove that BSA is very effective in learning the MLPs.

Finally, the BP, and LM results for different TE reactors sub-problems in terms of VAF, MSE, MAE, and p-values are reported in Table 28. The results show that the BSA learner has the best results compared to BP and LM learners in all TE sub-problems.

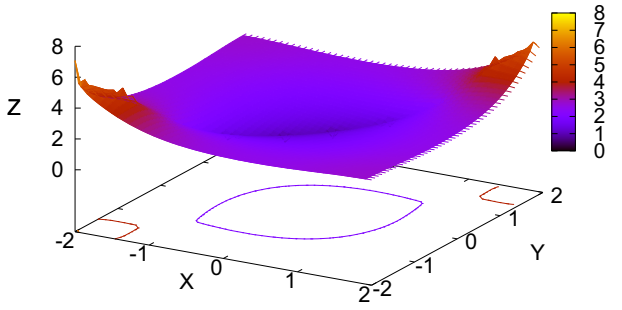
The performance of the BSA-MLP in tracking the actual TE process through the testing stage of the Level, Pressure, Cooling, and Temperature reactors is shown in Fig. 11. The approximation curves in Fig. 11 verify that the ability of BSA algorithm to represent the behavior of the TE process.

Figure 12 shows the convergence curves of BSA, DE, GA, PSO, ACO, ES, and ABC based on averages of the MSE for TE reactors sub-problems. These convergence curves prove that BSA achieves the fastest convergence for the all sub-problems.

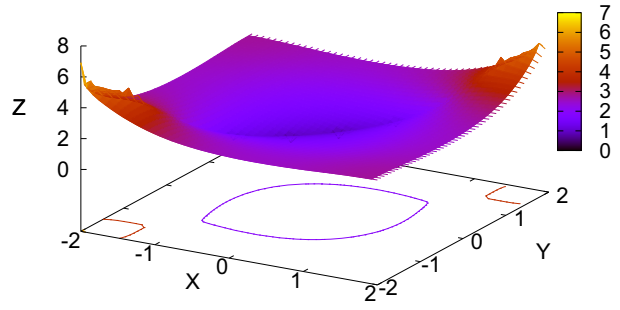
To sum up, the results and discussions of this paper showed that the BSA algorithm is able to efficiently train



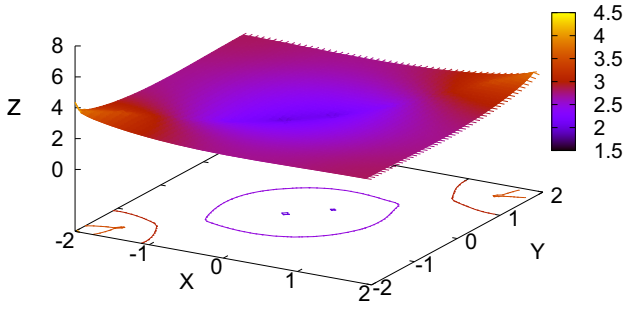
(a) Actual



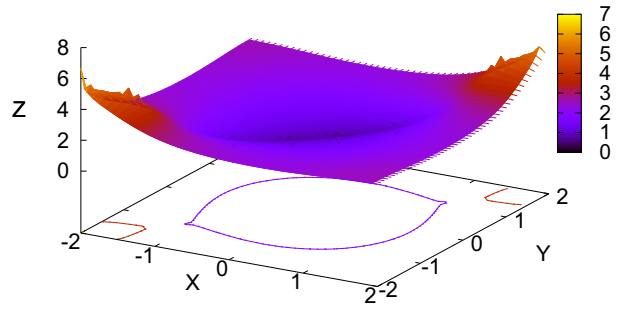
(b) BSA



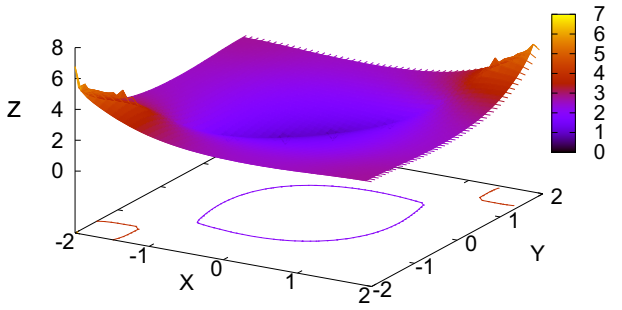
(c) GA



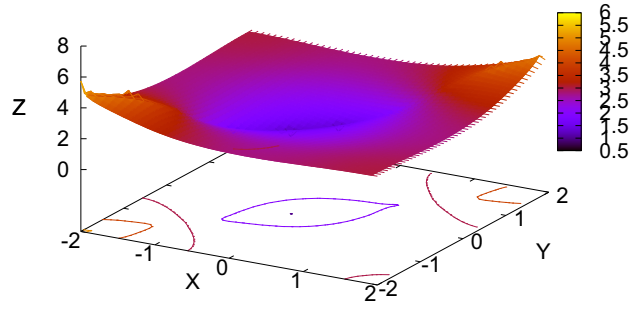
(d) DE



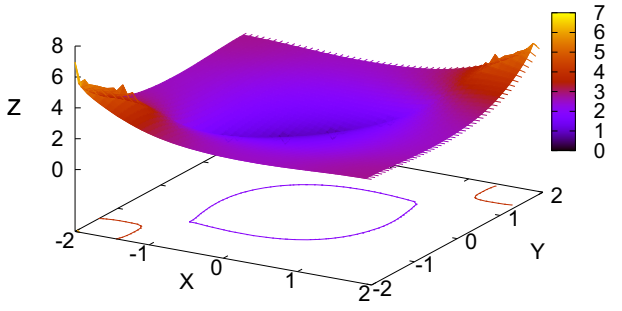
(e) ES



(f) PSO



(g) ACO



(h) ABC

Fig. 7 Approximated curves versus actual curve for sphere function

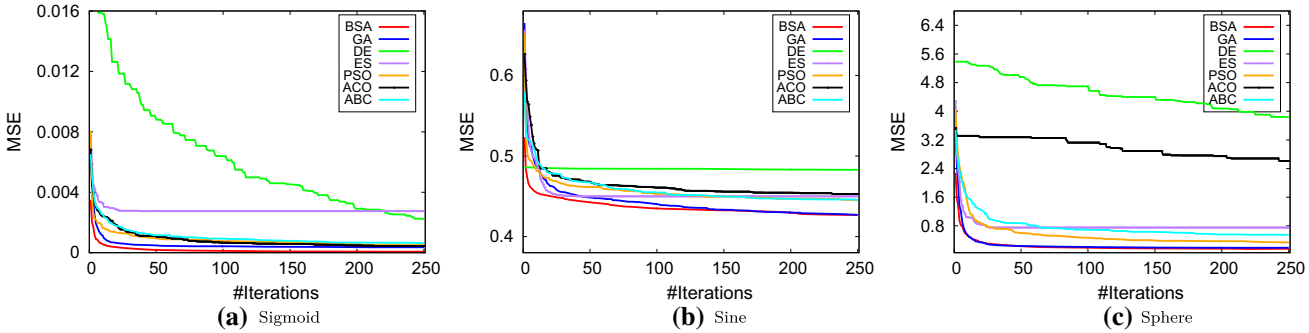


Fig. 8 Convergence curves for sigmod, sine, and sphere functions

Table 21 Test error, p-values, and MSE, for BSA, BP, and LM on Sigmoid, Sine, and Sphere function approximation datasets

Dataset/optimizer	BSA	LM	BP
Sigmoid			
Test error (AVE ± STD)	0.6916 ± 0.3580	**0.1854** ± 0.2018	29.6807 ± 16.8143
p-values	4.57E−09	N/A	3.02E−11
MSE (AVE ± STD)	6.41E−05 ± 6.41E−05	8.33E−06 ± 2.06E−05	1.10E−01 ± 1.05E−01
Sine			
Test error (AVE ± STD)	142.8721 ± 5.9147	**111.5707** ± 8.8675	156.6969 ± 5.2270
p-values	2.98E−11	N/A	2.98E−11
MSE (AVE ± STD)	4.27E−01 ± 2.29E−02	3.13E−01 ± 3.28E−02	4.86E−01 ± 2.84E−02
Sphere			
Test error (AVE ± STD)	**12.1338** ± 1.0614	15.0053 ± 25.6014	15.7599 ± 8.9057
p-values	N/A	9.79E−05	5.61E−05
MSE (AVE ± STD)	12.99E−02 ± 2.42E−02	1.38E+00 ± 3.16E+00	4.14E−01 ± 1.26E+00

The best results are marked in bold

Table 22 Final statistical results on a variety of measures

	BSA			DE			GA			PSO			ACO			ES			ABC			LM			BP		
	W	L	T	W	L	T	W	L	T	W	L	T	W	L	T	W	L	T	W	L	T	W	L	T	W	L	T
Class. rate or Test Err.	11	3	2	0	16	0	1	13	2	1	13	2	0	16	0	1	13	2	1	14	2	3	13	0	0	16	0
Significant (p-values)	10	4	2	0	16	0	2	12	2	1	13	2	0	16	0	1	13	2	1	13	2	3	13	0	0	16	0
Total	**21**	**7**	**4**	0	32	0	3	25	4	2	26	4	0	32	0	2	26	4	2	27	4	6	26	0	0	32	0
Ranking (Friedman)	**1.7308**			5.9231			2.8462			3.4615			7.0385			4.6923			3.9231			6.6923			8.6923		

The best results are marked in bold

The table values represent the Number of datasets each algorithm won/losses/ties on a variety of measures

MLP to classify different datasets. The superiority of the results compared to BP and LM algorithms is due to the gradient-free mechanism of BSA. The search space of training MLP changes for every dataset and consequently a training algorithm deals with a different search space when changing a dataset. Real datasets and all the benchmark datasets employed in this work are very challenging and their search spaces have a massive number of local solutions. Therefore, an algorithm should be able to avoid local solutions to eventually determine the global optimum. BP and LM are gradient-based algorithms which intrinsically suffer from local optima stagnation. This was the main reason of their poor performance on the benchmark datasets. On the other hand, BSA is a stochastic algorithm and

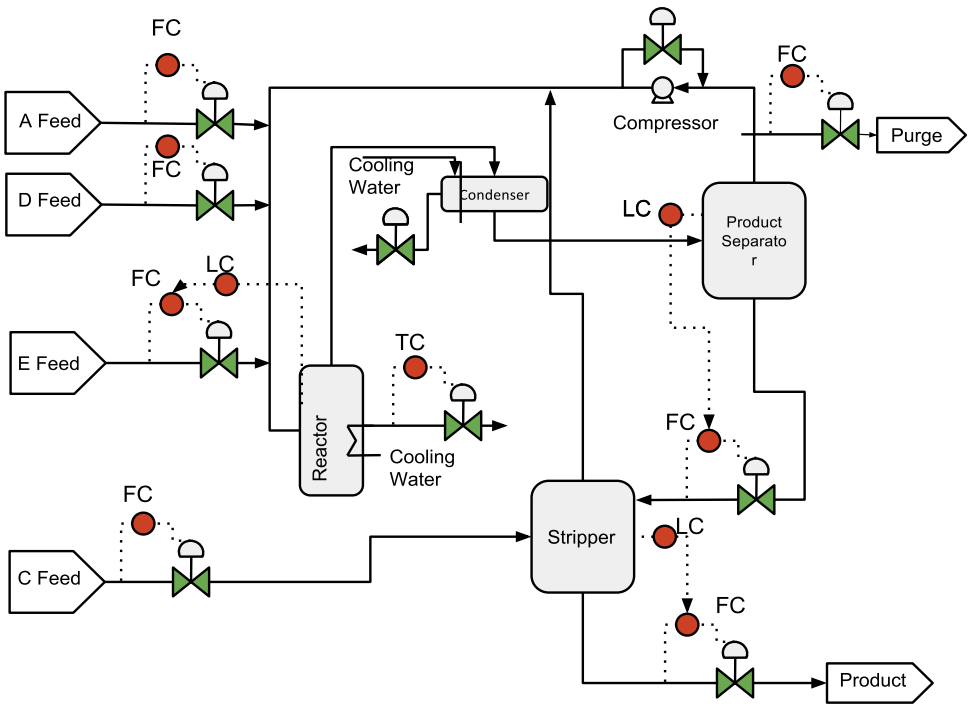


Fig. 9 A simplified diagram of the Tennessee Eastman challenge process [43]

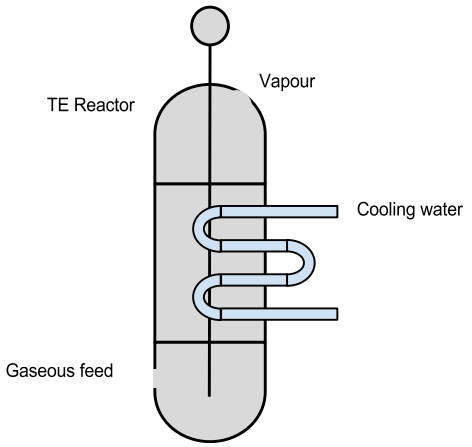


Fig. 10 Description of the reactor system [12]

benefits from a substantially better local optima avoidance. The results proved that BSA is significantly better than BP and LM, which is due to randomness in this algorithm and less probability of local optima avoidance. The discrepancy results of BSA and BP/LM when solving the real case study evidently showed the importance of a stochastic trainer when solving real problems.

BSA also outperformed the current stochastic algorithms on the majority of datasets. BSA were more efficient than PSO since this algorithm divides the particles to different groups with diverse behaviors. In the basic version of PSO, all the particles are considered the same and perform the search in a similar manner. The particles in BSA, however, show different search patters with more randomness which results in a better local optima avoidance. The convergence curves on the benchmark functions showed that this might negatively impacts the convergence rate, yet it is essential when training MLPs due to the large number of local solutions. The ACO algorithm showed worst results in many of the datasets. This algorithm evolves a matrix of metronomes and suits best for combinatorial problems. This is why the BSA algorithm managed to outperform this algorithm on all the datasets.

GA, ES, and DA are all evolutionary algorithms which intrinsically own a higher exploration and local optima avoidance compared to swarm intelligence techniques. However, the results of this study showed that BSA, as a

Table 23 Summary of the TE chemical reactor sub-problems

Problem no.	Problem name	#attributes	#train samples	#test samples	MLP structure
1	Level	4	150	150	4–9–1
2	Pressure	4	150	150	4–9–1
3	Cooling temperature	4	150	150	4-9–1
4	Temperature	4	150	150	4–9–1

Table 24 VAF, MSE, p-values, and MAE results for reactor level sub-problem

Algorithm	VAF (AVE ± STD)	MSE (AVE ± STD)	p-values	MAE (AVE ± STD)
BSA	**57.3666** ± 1.1140	**0.4321** ± 0.0119	N/A	**0.5224** ± 0.0062
GA	57.1058 ± 1.5292	0.4407 ± 0.0195	*0.3075*	0.5258 ± 0.0095
DE	34.4535 ± 14.0966	0.7673 ± 0.1417	1.83E−04	0.6947 ± 0.0700
ES	22.9420 ± 37.7420	0.8253 ± 0.4316	1.83E−04	0.7304 ± 0.1775
PSO	49.0539 ± 3.9203	0.5260 ± 0.0464	5.83E−04	0.5786 ± 0.0243
ACO	1.7336 ± 11.1505	1.0819 ± 0.1564	1.83E−04	0.8326 ± 0.0708
ABC	44.0884 ± 11.8402	0.5998 ± 0.1187	1.83E−04	0.6218 ± 0.0542

The best results are marked in bold

Table 25 VAF, MSE, p-values, and MAE results for reactor pressure sub-problem

Algorithm	VAF (AVE ± STD)	MSE (AVE ± STD)	p-values	MAE (AVE ± STD)
BSA	**41.2088** ± 1.5064	**0.0411** ± 0.0012	N/A	**0.1623** ± 0.0022
GA	40.1118 ± 1.1170	0.0415 ± 0.0007	*0.0890*	0.1632 ± 0.0018
DE	14.2230 ± 14.8915	0.0610 ± 0.0108	1.83E−04	0.1978 ± 0.0185
ES	5.9268 ± 12.9509	0.0670 ± 0.0127	1.81E−04	0.2022 ± 0.0213
PSO	32.8378 ± 3.4979	0.0471 ± 0.0024	2.46E−04	0.1737 ± 0.0055
ACO	6.35690 ± 11.1727	0.0692 ± 0.0077	1.83E−04	0.2112 ± 0.0143
ABC	19.3026 ± 12.9417	0.0570 ± 0.0086	1.83E−04	0.1853 ± 0.0127

The best results are marked in bold

Table 26 VAF, MSE, p-values, and MAE results for reactor cooling sub-problem

Algorithm	VAF (AVE ± STD)	MSE (AVE ± STD)	p-values	MAE (AVE ± STD)
BSA	**73.2451** ± 1.3723	**0.001069** ± 0.0001	N/A	**0.0220** ± 0.0006
GA	71.7688 ± 1.3724	0.001143 ± 0.0001	0.0211	0.0226 ± 0.0006
DE	70.0409 ± 5.9681	0.001244 ± 0.0003	*0.1859*	0.0246 ± 0.0036
ES	71.5651 ± 4.6791	0.001184 ± 0.0001	*0.0640*	0.0242 ± 0.0027
PSO	73.0226 ± 1.4820	0.001078 ± 0.0001	*0.6232*	0.0224 ± 0.0012
ACO	69.6228 ± 2.3113	0.001194 ± 0.0002	0.0376	0.024965 ± 0.0020
ABC	71.8410 ± 3.7949	0.001149 ± 0.0002	*0.5708*	0.0231 ± 0.0018

The best results are marked in bold

Table 27 VAF, MSE, p-values, and MAE results for reactor temperature sub-problem

Algorithm	VAF (AVE ± STD)	MSE (AVE ± STD)	p-values	MAE (AVE ± STD)
BSA	98.5775 ± 0.5889	0.003160 ± 0.0012	*0.6776*	0.0440 ± 0.0086
GA	**98.6946** ± 0.3904	**0.002852** ± 0.0008	N/A	**0.0432** ± 0.0069
DE	89.1314 ± 4.2085	0.027535 ± 0.0097	1.83E−04	0.1346 ± 0.0309
ES	89.5911 ± 10.5617	0.025540 ± 0.0231	1.83E−04	0.1290 ± 0.0417
PSO	96.3750 ± 1.3878	0.008312 ± 0.0030	1.83E−04	0.0736 ± 0.0152
ACO	82.3085 ± 3.4616	0.052792 ± 0.0084	1.83E−04	0.1827 ± 0.0246
ABC	93.1904 ± 1.4173	0.016919 ± 0.0034	1.83E−04	0.1054 ± 0.0103

The best results are marked in bold

swarm intelligence technique, provide very competitive results and tend to be superior. This is due to the involvement of all individuals in defining the center of swarm and their impacts on the overall movement. This mechanism encourages particles not to converge to the local optima and show better exploration compared to other swarm intelligence techniques. This assists BSA to compete with evolutionary algorithms very well in terms of exploration of the search space and avoiding local solutions.

Table 28 AF, p-values, MSE, and MAE for BSA, BP, and LM, learners on all TE reactor sub-problems

Sub-problem/ optimizer	BSA	BP	LM
Level			
VAF (AVE ± STD)	**57.3666** ± 1.1140	52.7786 ± 4.8022	− 18.19654 ± 37.2888
MSE (AVE ± STD)	**0.4321** ± 0.0119	0.4819 ± 0.0576	1.8705 ± 1.9272
p-values	N/A	0.0113	1.83E−04
MAE (AVE ± STD)	**0.5224** ± 0.0062	0.5536 ± 0.0304	1.0047 ± 0.5345
Pressure			
VAF (AVE ± STD)	**41.2088** ± 1.5064	-85.5850 ± 70.6489	-42.7350 ± 32.1506
MSE (AVE ± STD)	**0.0411** ± 0.0012	0.1695 ± 0.1064	0.1001 ± 0.0226
p-values	N/A	7.69E−04	1.83E−04
MAE (AVE ± STD)	**0.1623** ± 0.0022	0.3252 ± 0.1184	0.2326 ± 0.0258
Cooling			
VAF (AVE ± STD)	**73.2451** ± 1.3723	-45.5397 ± 57.1028	65.2363 ± 15.5368
MSE (AVE ± STD)	**0.0011** ± 0.0001	0.0062 ± 0.0016	0.0014 ± 0.0007
p-values	N/A	1.83E−04	1.83E−04
MAE (AVE ± STD)	**0.0220** ± 0.0006	0.0644 ± 0.0110	0.0240 ± 0.0033
Temperature			
VAF (AVE ± STD)	**98.5775** ± 0.5889	82.7270 ± 12.8165	70.0980 ± 61.4790
MSE (AVE ± STD)	**0.0031** ± 0.0012	0.0378 ± 0.0279	0.1396 ± 0.3068
p-values	N/A	1.83E−04	0.0028
MAE (AVE ± STD)	**0.0440** ± 0.0086	0.1510 ± 0.0497	0.1643 ± 0.2964

The best results are marked in bold

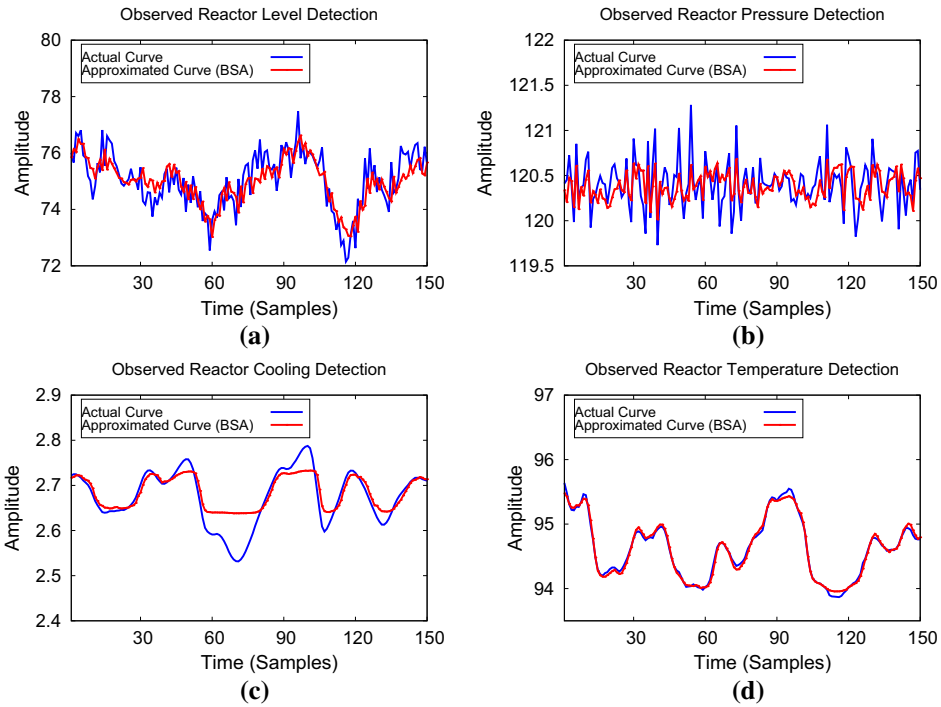


Fig. 11 Approximated curves versus actual curve for all reactor sub-problems

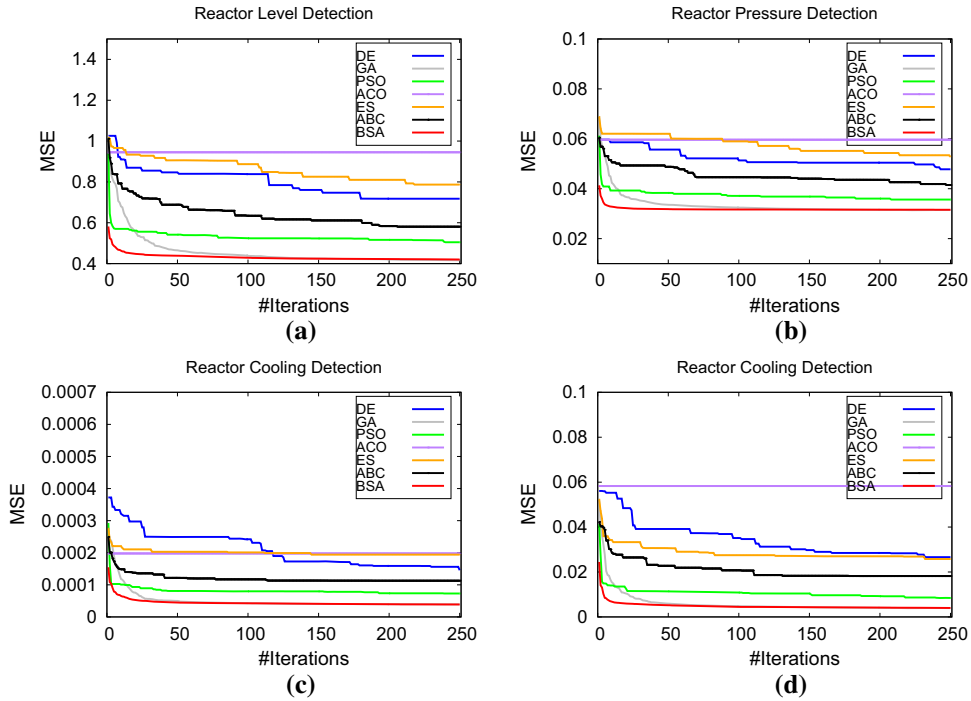


Fig. 12 Convergence curves for all reactor sub-problems

Another finding of this work was the consistent performance of BSA on classification, function approximation, and real datasets. The performance of BSA did not get degraded remarkably in any of the dataset which shows the robustness of this algorithm. The three sets of problems employed in this work have different natures and tested the performance of BSA from different perspective. The results showed that BSA is very flexible to solve a diverse set of problem. This is due to the fact that BSA considers training of MLP as a black box. It just tunes the variables of this problem and observes its outputs to improve the performance. This is also the reason of superiority of all stochastic algorithms in the work on all datasets compared to BP and LM. After this comprehensive study, we assert that BSA has merits to be considered as a training algorithms for MLP and other ANNs. Due to the stochastic nature of this algorithm and the so-called NFL theorem, however, it does solve the problem of training MLP and might show poor performance on a particular set of problems.

7 Conclusion

This paper proposed a new evolutionary MLP based on the recent BSA algorithm. The main motivation for selecting BSA as a trainer was its structure, in which particles are divided to different groups and benefit from a very high local optima avoidance compared to similar algorithms. We first formulated the problem of training MLP and then proposed a BSA-based trainer for optimizing this problem. To test the performance of this new trainer, three main phases of experiments were conducted. In the first phase, 13 well-regarded and challenging classification datasets were employed. In the second phase, three function approximation datasets were created and used to test the proposed algorithm. In the last phase, a real dataset for a reactor system was solved to prove the applicability of the BSA trainer. For results verification, nine algorithms including seven stochastic and two classical training algorithm were employed. The obtained results were compared quantitatively and qualitatively. For quantitative results, a set of performance indicators was selected: MSE, Test error, classification error, and Wilcoxons ranksum test. For the qualitative results, the convergence and shape of the approximated functions were visualized in the paper. The results proved that the BSA is able to train MLP for classifying a wide range of datasets with different characteristics. BSA showed very competitive results in all performance indicators, although the convergence speed was not very fast in some of the case studies. The superiority of the results of BSA can be explained by the high local optima avoidance which was proved to be beneficial when classifying challenging datasets. The results of the real case study testified the performance of the proposed BSA trainer in practice. As per the results, discussions, findings, and analyses of this work, we offer the BSA trainer as a very reliable and robust alternative to the current training algorithms for MLP to be applied to different datasets.

The promising results of BSA in training the single-hidden layer network give a strong motivation to extend this work and investigate its efficiency in training deeper neural networks with more hidden layers. It is interesting also design the BSA trainer in way to include other important parameters of the MLP network such as the number of hidden neurons and the structure of the network. In order to develop more efficient implementation of the proposed trainer, it is planned also to implement this trainer as part of EvoloPy which is nature-inspired optimization framework in Python developed by the authors.

Compliance with ethical standards

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

References

1. Adwan, O., Faris, H., Jaradat, K., Harfoushi, O., Ghatasheh, N.: Predicting customer churn in telecom industry using multilayer preceptron neural networks: modeling and analysis. Life Sci. J. **11**(3), 75–81 (2014)
2. Al-Hiary, H., Sheta, A., Ayesh, A.: Identification of a chemical process reactor using soft computing techniques. In: Proceedings of the 2008 International Conference on Fuzzy Systems (FUZZ2008) within the 2008 IEEE World Congress on Computational Intelligence (WCCI2008), Hong Kong, 1–6 June, pp. 845–653 (2008)
3. Al-Shayea, Q.K.: Artificial neural networks in medical diagnosis. Int. J. Comput. Sci. Issues **8**(2), 150–154 (2011)
4. 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, BDCAT '17, pp. 131–138, New York, NY (2017). ACM
5. Aljarah, I., Ludwig, S.A.: A mapreduce based glowworm swarm optimization approach for multimodal functions. In: 2013 IEEE Symposium on Swarm Intelligence (SIS), pp. 22–31. IEEE (2013)
6. Aljarah, I., Ludwig, S.A.: Towards a scalable intrusion detection system based on parallel pso clustering using MapReduce. In: Proceedings of the 15th Annual Conference Companion on Genetic and Evolutionary Computation, pp. 169–170. ACM (2013)
7. Aljarah, I., Ludwig, S.A.: A scalable mapreduce-enabled glowworm swarm optimization approach for high dimensional multimodal functions. Int. J. Swarm Intell. Res. (IJSIR) **7**(1), 32–54 (2016)
8. Aljarah, I., Faris, H., Mirjalili, S.: Optimizing connection weights in neural networks using the whale optimization algorithm. Soft Comput. **22**(1), 1–15 (2018)
9. Aljarah, I., Faris, H., Mirjalili, S., Al-Madi, N.: Training radial basis function networks using biogeography-based optimizer. Neural Comput. Appl. **29**(7), 529–553 (2018)
10. Amaldi, E., Mayoraz, E., de Werra, D.: A review of combinatorial problems arising in feedforward neural network design. Discret. Appl. Math. **52**(2), 111–138 (1994)
11. Arifovic, J., Gencay, R.: Using genetic algorithms to select architecture of a feedforward artificial neural network. Physica A **289**(3), 574–594 (2001)
12. Barton, I.P., Martinsen, S.W.: Equation-oriented simulator training. In Proceedings of the American Control Conference, Albuquerque, New Mexico, pp. 2960–2965 (1997)
13. Basheer, I.A., Hajmeer, M.: Artificial neural networks: fundamentals, computing, design, and application. J. Microbiol. Methods **43**(1), 3–31 (2000)
14. Bathelt, A., Ricker, N.L., Jelali, M.: Revision of the Tennessee Eastman process model. IFAC-PapersOnLine **48**(8), 309–314 (2015)
15. Bebis, G., Georgiopoulos, M.: Feed-forward neural networks. IEEE Potentials **13**(4), 27–31 (1994)
16. Bhat, N., McAvoy, T.J.: Use of neural nets for dynamic modeling and control of chemical process systems. Comput. Chem. Eng. **14**, 573–582 (1990)
17. Bornholdt, S., Graudenz, D.: General asymmetric neural networks and structure design by genetic algorithms. Neural Netw. **5**(2), 327–334 (1992)
18. BoussaïD, I., Lepagnot, J., Siarry, P.: A survey on optimization metaheuristics. Inf. Sci. **237**, 82–117 (2013)
19. Brajevic, I., Tuba, M.: Training feed-forward neural networks using firefly algorithm. In: Proceedings of the 12th International Conference on Artificial Intelligence, Knowledge Engineering and Data Bases (AIKED'13), pp. 156–161 (2013)
20. Buscema, M.: Back propagation neural networks. Subst. Use Misuse **33**(2), 233–270 (1998)
21. Chen, C.L.P.: A rapid supervised learning neural network for function interpolation and approximation. IEEE Trans. Neural Netw. **7**(5), 1220–1230 (1996)
22. Downs, J.J., Vogel, E.F.: A plant-wide industrial process control problem. Comput. Chem. Eng. **17**(3), 245–255 (1993)
23. Engelbrecht, A.P.: Supervised learning neural networks. Computational Intelligence: An Introduction, 2nd edn., pp. 27-54. Wiley, Singapore (2007)
24. Faris, H., Alkasassbeh, M., Rodan, A.: Artificial neural networks for surface ozone prediction: models and analysis. Pol. J. Environ. Stud. **23**(2), 341–348 (2014)
25. Faris, H., Aljarah, I., et al.: 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)
26. 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)
27. Faris, H., Aljarah, I., Mirjalili, S.: Training feedforward neural networks using multi-verse optimizer for binary classification problems. Applied Intelligence, pp. 1–11 (2016)
28. Faris, H., Aljarah, I., Mirjalili, S.: Evolving radial basis function networks using moth–flame optimizer. In: Handbook of Neural Computation, pp. 537–550. Elsevier (2017)
29. 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)
30. Galić, E., Höhfeld, M.: Improving the generalization performance of multi-layer-perceptrons with population-based incremental learning. In: International Conference on Parallel Problem Solving from Nature, pp. 740–750. Springer (1996)
31. Garro, B.A., Vázquez, R.A.: Designing artificial neural networks using particle swarm optimization algorithms. Comput. Intell. Neurosci. https://doi.org/10.1155/2015/369298 (2015)
32. Goerick, C., Rodemann, T.: Evolution strategies: an alternative to gradient-based learning. In: Proceedings of the International

Conference on Engineering Applications of Neural Networks, vol. 1, pp. 479–482 (1996)
33. Goldberg, D.E. et al.: Genetic Algorithms in Search Optimization and Machine Learning, vol. 412. Addison-Wesley, Reading (1989)
34. Golfinopoulos, E., Tourville, J.A., Guenther, F.H.: The integration of large-scale neural network modeling and functional brain imaging in speech motor control. Neuroimage **52**(3), 862–874 (2010)
35. Gupta, J.N.D., Sexton, R.S.: Comparing backpropagation with a genetic algorithm for neural network training. Omega **27**(6), 679–684 (1999)
36. Gupta, M.M., Jin, L., Homma, N.: Radial basis function neural networks. In: Static and Dynamic Neural Networks: From Fundamentals to Advanced Theory, pp. 223–252 (2003)
37. Hansel, D., Sompolinsky, H.: Learning from examples in a single-layer neural network. EPL Europhys. Lett. **11**(7), 687 (1990)
38. Heidari, A.A., Faris, H., Aljarah, I., Mirjalili, S.: An efficient hybrid multilayer perceptron neural network with grasshopper optimization. Soft Comput. https://doi.org/10.1007/s00500-018-3424-2(2018)
39. Ho, Y.-C., Pepyne, D.L.: Simple explanation of the no-free-lunch theorem and its implications. J. Optim. Theory Appl. **115**(3), 549–570 (2002)
40. Hush, D.R., Horne, B.G.: Progress in supervised neural networks. IEEE Signal Process. Mag. **10**(1), 8–39 (1993)
41. Hwang, Y.-S., Bang, S.-Y.: An efficient method to construct a radial basis function neural network classifier. Neural Netw. **10**(8), 1495–1503 (1997)
42. Ilonen, J., Kamarainen, J.-K., Lampinen, J.: Differential evolution training algorithm for feed-forward neural networks. Neural Process. Lett. **17**(1), 93–105 (2003)
43. Juricek, B.C., Seborg, D.E., Larimore, W.E.: Identification of the tennessee eastman challenge process with subspace methods. Control Eng. Pract. **9**(12), 1337–1351 (2001)
44. Kaelbling, L.P., Littman, M.L., Moore, A.W.: Reinforcement learning: a survey. J. Artif. Intell. Res. **4**, 237–285 (1996)
45. Karaboga, D., Akay, B., Ozturk, C.: Artificial bee colony (abc) optimization algorithm for training feed-forward neural networks. In: International Conference on Modeling Decisions for Artificial Intelligence, pp. 318–329. Springer (2007)
46. Karim, M.N., Rivera, S.L.: Artificial neural networks in bioprocess state estimation. Adv. Biochem. Eng. Biotechnol. **46**, 1–31 (1992)
47. Khan, K., Sahai, A.: A comparison of ba, ga, pso, bp and lm for training feed forward neural networks in e-learning context. Int. J. Intell. Syst. Appl. **4**(7), 23 (2012)
48. Kowalski, P.A., Łukasik, S.: Training neural networks with krill herd algorithm. Neural Process. Lett. **44**, 5–17 (2015)
49. Kruse, R., Borgelt, C., Klawonn, F., Moewes, C., Steinbrecher, M., Held, P.: Multi-layer perceptrons. In: Computational Intelligence, pp. 47–81. Springer (2013)
50. Larochelle, H., Bengio, Y., Louradour, J., Lamblin, P.: Exploring strategies for training deep neural networks. J. Mach. Learn. Res. **10**, 1–40 (2009)
51. Leonard, J., Kramer, M.A.: Improvement of the Backpropagation algorithm for training neural networks. Comput. Chem. Eng. **14**, 337–343 (1990)
52. Leshno, M., Lin, V.Y., Pinkus, A., Schocken, S.: Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. Neural Netw. **6**(6), 861–867 (1993)
53. Leung, F.H.-F., Lam, H.-K., Ling, S.-H., Tam, P.K.-S.: Tuning of the structure and parameters of a neural network using an improved genetic algorithm. IEEE Trans. Neural Netw. **14**(1), 79–88 (2003)
54. Lichman, M.: UCI Machine Learning Repository. University of California, School of Information and Computer Science, Irvine (2013)
55. Lippmann, R.P.: Pattern classification using neural networks. IEEE Commun. Mag. **27**(11), 47–50 (1989)
56. Lo, S.-C.B., Chan, H.-P., Lin, J.-S., Li, H., Freedman, M.T., Mun, S.K.: Artificial convolution neural network for medical image pattern recognition. Neural Netw. **8**(7), 1201–1214 (1995)
57. Mavrovouniotis, M., Yang, S.: Training neural networks with ant colony optimization algorithms for pattern classification. Soft Comput. **19**(6), 1511–1522 (2015)
58. Meissner, M., Schmuker, M., Schneider, G.: Optimized particle swarm optimization (OPSO) and its application to artificial neural network training. BMC Bioinform. **7**(1), 125 (2006)
59. Melin, P., Castillo, O.: Unsupervised learning neural networks. In: Hybrid Intelligent Systems for Pattern Recognition Using Soft Computing, pp. 85–107. Springer (2005)
60. Meng, X.-B., Gao, X.Z., Lu, L., Liu, Y., Zhang, H.: A new bio-inspired optimisation algorithm: bird swarm algorithm. J. Exp. Theor. Artif. Intell. https://doi.org/10.1080/0952813X.2015.1042530(2015)
61. Merkl, D., Rauber, A.: Document classification with unsupervised artificial neural networks. In: Soft Computing in Information Retrieval, pp. 102–121. Springer (2000)
62. Mezura-Montes, E., Velázquez-Reyes, J., Coello Coello, C.A.: A comparative study of differential evolution variants for global optimization. In: Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation, pp. 485–492. ACM (2006)
63. Mirjalili, S.: How effective is the grey wolf optimizer in training multi-layer perceptrons. Appl. Intell. **43**(1), 150–161 (2015)
64. Mirjalili, S., Mirjalili, S.M., Lewis, A.: Let a biogeography-based optimizer train your multi-layer perceptron. Inf. Sci. **269**, 188–209 (2014)
65. Mitchell, T.M: Artificial neural networks. Machine Learning, pp. 81–127 (1997)
66. Montana, D.J., Davis, L.: Training feedforward neural networks using genetic algorithms. IJCAI **89**, 762–767 (1989)
67. Nahas, E.P., Henson, M.A., Seborg, D.E.: Nonlinear internal model control strategy for neural network models. Comput. Chem. Eng. **16**, 1039–1057 (1992)
68. Nawi, N.M., Khan, A., Rehman, M.Z., Tutut H., Mustafa, M.D.: Comparing performances of cuckoo search based neural networks. In: Recent Advances on Soft Computing and Data Mining, pp. 163–172. Springer (2014)
69. Parisi, R., Di Claudio, E.D., Lucarelli, G., Orlandi, G.: Car plate recognition by neural networks and image processing. In: Proceedings of the 1998 IEEE International Symposium on Circuits and Systems, 1998. ISCAS'98, vol. 3, pp. 195–198. IEEE (1998)
70. Pascanu, R., Mikolov, T., Bengio, Y.: On the difficulty of training recurrent neural networks. ICML **3**(28), 1310–1318 (2013)
71. Principe, J.C., Fancourt, C.L.: Artificial neural networks. In: Pardalos, P.M., Romejin, H.E. (eds.) Handbook of Global Optimization, vol. 2, pp. 363–386. Kluwer, Dordrecht (2013)
72. Ricker, N.L.: Nonlinear model predictive control of the tennessee eastman challenge process. Comput. Chem. Eng. **19**(9), 961–981 (1995)
73. Ricker, N.L.: Nonlinear modeling and state estimation of the tennessee eastman challenge process. Comput. Chem. Eng. **19**(9), 983–1005 (1995)
74. Ricker, N.L.: Tennessee Eastman challenge archive (2005)
75. Sanger, T.D.: Optimal unsupervised learning in a single-layer linear feedforward neural network. Neural Netw. **2**(6), 459–473 (1989)

76. Seiffert, U.: Multiple layer perceptron training using genetic algorithms. In: ESANN, pp. 159–164. Citeseer (2001)
77. Sheta, A., Al-Hiary, Heba, Braik, Malik: Identification and model predictive controller design of the Tennessee Eastman chemical process using ann. In: Proceedings of the 2009 International Conference on Artificial Intelligence (ICAI'09), July 13–16, USA, vol. 1, pp. 25–31 (2009)
78. Sibi, P., Allwyn Jones, S., Siddarth, P.: Analysis of different activation functions using back propagation neural networks. J. Theor. Appl. Inf. Technol. **47**(3), 1264–1268 (2013)
79. Simon, D.: Biogeography-based optimization. IEEE Trans. Evol. Comput. **12**(6), 702–713 (2008)
80. Sivagaminathan, R.K., Ramakrishnan, S.: A hybrid approach for feature subset selection using neural networks and ant colony optimization. Expert Syst. Appl. **33**(1), 49–60 (2007)
81. Slowik, A., Bialko, M.: Training of artificial neural networks using differential evolution algorithm. In: 2008 Conference on Human System Interactions, pp. 60–65. IEEE (2008)
82. 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)
83. Stanley, K.O.: Efficient reinforcement learning through evolving neural network topologies. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2002). Citeseer (2002)
84. Subudhi, B., Jena, D.: Differential evolution and Levenberg Marquardt trained neural network scheme for nonlinear system identification. Neural Process. Lett. **27**(3), 285–296 (2008)
85. Suykens, J.A.K., Vandewalle, J.P.L., de Moor, B.L.: Artificial Neural Networks for Modelling and Control of Non-linear Systems. Springer, Berlin (2012)
86. 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)
87. van den Bergh, F., Engelbrecht, A.P., Engelbrecht, A.P.: Cooperative learning in neural networks using particle swarm optimizers. In: South African Computer Journal. Citeseer (2000)
88. Wdaa, A.S.I.: Differential evolution for neural networks learning enhancement. PhD thesis, Universiti Teknologi Malaysia (2008)
89. Whitley, D., Starkweather, T., Bogart, C.: Genetic algorithms and neural networks: optimizing connections and connectivity. Parallel Comput. **14**(3), 347–361 (1990)
90. Wienholt, W.: Minimizing the system error in feedforward neural networks with evolution strategy. In: ICANN'93, pp. 490–493. Springer (1993)
91. Yamany, W., Fawzy, M., Tharwat, A., Hassanien, A.E.: Moth-flame optimization for training multi-layer perceptrons. In: 2015 11th International Computer Engineering Conference (ICENCO), pp. 267–272. IEEE (2015)
92. Yang, C.C., Prasher, S.O., Landry, J.A., DiTommaso, A.: Application of artificial neural networks in image recognition and classification of crop and weeds. Can. Agric. Eng. **42**(3), 147–152 (2000)
93. Yang, Z., Hoseinzadeh, M., Andrews, A., Mayers, C., Evans, D.T., Bolt, R.T., Bhimani, J., Mi, N., Swanson, S.: Autotiering: automatic data placement manager in multi-tier all-flash datacenter. In: 2017 IEEE 36th International on Performance Computing and Communications Conference (IPCCC), pp. 1–8. IEEE (2017)
94. Yang, Z., Jia, D., Ioannidis, S., Mi, N., Sheng, B.: Intermediate data caching optimization for multi-stage and parallel big data frameworks. arXiv:1804.10563 (2018)
95. Yao, X.: A review of evolutionary artificial neural networks. Int. J. Intell. Syst. **8**(4), 539–567 (1993)
96. Yegnanarayana, B.: Artificial neural networks. PHI Learning Pvt. Ltd., New Delhi (2009)
97. Zhang, G.P.: Neural networks for classification: a survey. IEEE Trans. Syst. Man Cybern. C **30**(4), 451–462 (2000)
98. Zhang, N.: An online gradient method with momentum for two-layer feedforward neural networks. Appl. Math. Comput. **212**(2), 488–498 (2009)
99. Zhang, C., Shao, H., Li, Y.: Particle swarm optimisation for evolving artificial neural network. In: 2000 IEEE International Conference on Systems, Man, and Cybernetics, vol. 4, pp. 2487–2490. IEEE (2000)
100. Zhang, J., Sanderson, A.C.: Jade: adaptive differential evolution with optional external archive. IEEE Trans. Evol. Comput. **13**(5), 945–958 (2009)
101. Zhang, J.-R., Zhang, J., Lok, T.-M., Lyu, M.R.: A hybrid particle swarm optimization–back-propagation algorithm for feedforward neural network training. Appl. Math. Comput. **185**(2), 1026–1037 (2007)

Ibrahim Aljarah is an associate professor of BIG Data Mining and Computational Intelligence at the University of Jordan - Department of Information Technology, Jordan. He obtained his bachelor degree in Computer Science from Yarmouk University - Jordan, 2003. Dr. Aljarah also obtained his master degree in computer science and information systems from the Jordan University of Science and Technology - Jordan in 2006. He also obtained his Ph.D. In computer Science from the North Dakota State University (NDSU), USA, in May 2014. He organized and participated in many conferences in the field of data mining, machine learning, and Big data such as NTIT, CSIT, IEEE NABIC, CASON, and BIGDATA Congress. Furthermore, he contributed in many projects in USA such as Vehicle Class Detection System (VCDS), Pavement Analysis Via Vehicle Electronic Telemetry (PAVVET), and Farm Cloud Storage System (CSS) projects. He has published more than 45 papers in refereed international conferences and journals. His research focuses on data mining, Machine Learning, Big Data, MapReduce, Hadoop, Swarm intelligence, Evolutionary Computation, Social Network Analysis (SNA), and large scale distributed algorithms.

Hossam Faris is an associate professor at Information Technology department/King Abdullah II School for Information Technology/ The University of Jordan (Jordan). Hossam Faris received his B.A., M.Sc. degrees (with excellent rates) in Computer Science from Yarmouk University and Al-Balqa' Applied University in 2004 and 2008 respectively in Jordan. Since then, he has been awarded a full-time competition-based Ph.D. scholarship from the Italian Ministry of Education and Research to peruse his Ph.D. degrees in e-Business at University of Salento, Italy, where he obtained his Ph.D. degree in 2011. In 2016, he worked as a

Postdoctoral researcher with GeNeura team at the Information and Communication Technologies Research Center (CITIC), University of Granada (Spain). His research interests include: Applied Computational Intelligence, Evolutionary Computation, Knowledge Systems, Data mining, Semantic Web and Ontologies.

Seyedali Mirjalili is a lecturer in Griffith College, Griffith University. He received his B.Sc. degree in Computer Engineering (software) from Yazd University, M.Sc. degree in Computer Science from Universiti Teknologi Malaysia (UTM), and Ph.D. in Computer Science from Griffith University. He was an active member of Soft Computing Research Group (SCRG) at UTM and Institute for Integrated and Intelligent Systems (IIIS) at Griffith University. His research interests include Robust Optimisation, Engineering Optimisation, Multi-objective Optimisation, Swarm Intelligence, Evolutionary Algorithms, and Artificial Neural Networks. He is working on the application of multi-objective and robust meta-heuristic optimisation techniques in Computational Fluid Dynamic (CFD) problems as well. Dr. Mirjalili is internationally recognised for his advances in Swarm Intelligence (SI) and optimisation, including the first set of SI techniques from a synthetic intelligence standpoint ± a radical departure from how natural systems are typically understood ± and a systematic design framework to reliably benchmark, evaluate, and propose computationally cheap robust optimisation algorithms. He has published over 50 journal articles, many in high-impact journals. Dr Mirjalili has over 2000 citations in total with an H-index of 21 and G-index of 47. From Google Scholar metrics, he is globally the 5th most cited researcher in Engineering Optimisation and the 9th most cited in Robust Optimisation.

Nailah Al-Madi received her Ph.D. degree in Computer Science from North Dakota State University, USA, in 2014. She earned her M.Sc. degree in Computer Science from Jordan University of Science and Technology, Jordan, in 2009. She received her B.Sc. degree in Computer Information Systems from Al al-Bayt University, Jordan, in 2005. She is currently working as an Assistant Professor in Princess Sumaya University for Technology, Jordan. Her research interests include: Optimization and Evolutionary Computation, Data Mining, Big Data, MapReduce and Hadoop Framework, Robotics, and Wireless Sensor Networks.

Alaa Sheta is a Professor in the Department of Computing Sciences, Texas A&M University-Corpus Christi, TX, USA where he has been a faculty member since 2016. Alaa completed his Ph.D. at George Mason University, USA and his undergraduate studies at Cairo University, Egypt. His research interests lie in the area of machine learning and image processing ranging from theory to design to implementation. He has collaborated actively with researchers in several other disciplines, particularly manufacture process modeling, biochemistry, software reliability and many others. Alaa published more than 120 journal and conference papers. He is a co-editor of the book entitled," Business Intelligence and Performance Management - Theory, Systems and Industrial Applications" by Springer Verlag, UK 2013. He received number of awards including the Best Poster Award from the SGAI International Conference on Artificial Intelligence, Cambridge, UK, 2011 for his publication on Quality Management of Manufacturing Processes and the Best Citation Prize based Google Citation Index, Taif University, 2013. Dr. successfully graduated about two dozen of graduate students. Dr. Sheta has served as a chair, co-chair and technical committee on roughly thirty conference and workshop program committees.

Majdi Mafarja received his B.Sc. in Software Engineering and M.Sc. in Computer Information Systems from Philadelphia University and The Arab Academy for Banking and Financial Sciences, Jordan in 2005 and 2007 respectively. He did his Ph.D. in Computer Science at National University of Malaysia (UKM). He was a member in Datamining and Optimization Research Group (DMO). Now he is an assistant professor at the Department of Computer Science at Birzeit University. His research interests include Evolutionary Computation, Meta-heuristics and Data mining.