Information Sciences 269 (2014) 188–209

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Let a biogeography-based optimizer train your Multi-Layer Perceptron

Seyedali Mirjalili [a,*], Seyed Mohammad Mirjalili [b], Andrew Lewis [a]

[a] School of Information and Communication Technology, Griffith University, Nathan, Brisbane, QLD 4111, Australia
[b] Zharfa Pajohesh System (ZPS) Co., Unit 5, NO. 30, West 208 St., Third Sq. Tehranpars, P.O. Box: 1653745696, Tehran, Iran

ARTICLE INFO

Article history:
Received 7 October 2012
Received in revised form 10 October 2013
Accepted 26 January 2014
Available online 3 February 2014

Keywords:
FNN
Neural network
Learning neural network
Biogeography-Based Optimization
BBO
Evolutionary algorithm

ABSTRACT

The Multi-Layer Perceptron (MLP), as one of the most-widely used Neural Networks (NNs), has been applied to many practical problems. The MLP requires training on specific applications, often experiencing problems of entrapment in local minima, convergence speed, and sensitivity to initialization. This paper proposes the use of the recently developed Biogeography-Based Optimization (BBO) algorithm for training MLPs to reduce these problems. In order to investigate the efficiencies of BBO in training MLPs, five classification datasets, as well as six function approximation datasets are employed. The results are compared to five well-known heuristic algorithms, Back Propagation (BP), and Extreme Learning Machine (ELM) in terms of entrapment in local minima, result accuracy, and convergence rate. The results show that training MLPs by using BBO is significantly better than the current heuristic learning algorithms and BP. Moreover, the results show that BBO is able to provide very competitive results in comparison with ELM.

1. Introduction

One of the more significant inventions in the field of soft computing is Neural Networks (NN), inspired by biological neurons in the human brain. The rudimentary concepts of NN were first mathematically modeled by McCulloch and Pitts [33]. The simplicity, low computational cost, and high performance have made this computational tool remarkably popular over the last decade. Among different types of NNs, the Feedforward Neural Network (FNN) [14] is the simplest and most widely-used.

FNNs receive information as inputs on one side and provide outputs from the other side using one-directional connections between the neurons in different layers. There are two types of FNN: Single-Layer Perceptron (SLP) [54] and Multi-Layer Perceptron (MLP) [34,73]. In the SLP there is only a single perceptron that makes it suitable for solving linear problems. However, an MLP has more than one perceptron, established in different layers. This makes it capable of solving non-linear problems.

Generally speaking, the applications of MLPs are categorized as pattern classification [35], data prediction [19], and function approximation [16]. Pattern classification implies classifying data into pre-defined discrete classes [4], whereas prediction refers to the forecasting of future trends according to current and previous data [19]. Finally, function approximation involves the process of modeling relationships between input variables. It has been proven that MLPs with one hidden layer

* Corresponding author. Tel.: +61 434555738.
E-mail addresses: seyedali.mirjalili@griffithuni.edu.au (S. Mirjalili), mohammad.smm@gmail.com (S.M. Mirjalili), a.lewis@griffith.edu.au (A. Lewis).
URL: http://www.alimirjalili.com (S. Mirjalili).

http://dx.doi.org/10.1016/j.ins.2014.01.038
0020-0255/

are able to approximate any continuous or discontinuous functions [10,23]. Regardless of the applications, the distinguishing capability of MLPs is learning [3]. MLPs are equipped with a learning concept that gives them the ability to learn from experience, similar to a human brain. This component is an essential part of all NNs. It may be divided into two types: supervised [53,74] and unsupervised [50,58] learning. For MLPs, most applications use the standard [26] or improved Back-Propagation (BP) [2,9,21,81] algorithms as their learning methods, which belong to the supervised learning family. Back Propagation (BP) is a gradient-based algorithm that has some drawbacks such as slow convergence [13,64,49] and a tendency to get trapped in local minima [17,29,32], making it unreliable for practical applications.

The ultimate goal of the learning process is to find the best combination of connection weights and biases in the NN to achieve the minimum error for training and test samples. However, often the error of MLP stays constantly large for some extended period of time during the learning process, as the learning algorithm leads MLPs to local minima rather than the global minimum. This problem is quite common in gradient-based learning approaches such as BP. The convergence of BP is also highly dependent on the initial values of learning rate and momentum. Unsuitable values for these variables may even result in divergence. There are many studies focused on resolving these problems of BP [28,63,72], but there is no reported significant improvement, and each method has its own side effects. The literature shows that heuristic optimization methods are promising alternatives for gradient-based learning algorithms [8,78] since the stochastic nature of these algorithms allows them to avoid local minima better than gradient-based techniques and optimize challenging problems [44]. Moreover, convergence rates of heuristic methods to the global minimum can be faster than BP, as investigated in [18].

Various heuristic optimization methods have been utilized to train FNNs, such as Particle Swarm Optimization (PSO) [36], Genetic Algorithm (GA) [57], Ant Colony Optimization (ACO) [6], and Evolutionary Strategies (ES) [75]. It has been proven by the well-known No Free Lunch theorem (NFL) that there is no heuristic algorithm best suited to solving all optimization problems [22,77,7,39]. This theory on one hand, and the problems of gradient-based methods on another, motivates many researchers to investigate the effectiveness of different heuristic algorithms in learning MLPs and other different field [1,40,41,43,45–47,55,56]. In this paper the efficiency of the recently proposed heuristic algorithm, Biogeography-Based Optimization (BBO) [60] is investigated in training MLPs. As discussed in Section 3, the BBO algorithm is an Evolutionary Algorithm (EA) that offers specific evolutionary mechanisms to each individual in a population. Generally speaking, the search space of an MLP is changed for different datasets. Therefore, EAs such as BBO and GAs can provide more flexible training procedures compared to gradient-based algorithms. Despite the merits of GAs for training MLPs, the variety of evolutionary operators of BBO for each individual potentially allow BBO to outperform a GA.

The mutation operator of BBO is another motivation for us to design a BBO-based trainer for MLPs. In contrast to Swarm Intelligence (SI) techniques (PSO and ACO for instance), EAs mostly have mutation operators, which enhances their exploitation capability. This potentially allows BO to outperform SI techniques in training MLPs as well. Moreover, different mutation constants for each individual in a population may also help BBO outperform a GA, which usually has a single mutation operator for the whole population. Finally, the intrinsically different adaptive mechanisms of evolutionary operators and mutations for each individual assist BBO to provide diverse exploration and exploitation behaviors when solving different problems.

There is currently little in the literature focusing on the efficiency of BBO in training MLPs. The only related work was by Ovreiu and Simon in 2010 [51]. They trained a neuro-fuzzy network for classifying P wave features for the diagnosis of cardiomyopathy. However, the main focus was the application of a trained neuro-fuzzy network since the data set was a real data set. In this work we employ 11 standard datasets to provide a comprehensive test bed for investigating the abilities of BBO in training MLPs.

The rest of the paper is organized as follows: Section 2 presents a brief introduction to the MLP. Section 3 discusses the principles of Biogeography-Based Optimization. The method of applying BBO as a heuristic training algorithm for MLP is described in Section 4. The experimental results are discussed in Section 5. Finally, Section 6 provides concluding remarks and suggests some directions for future research.

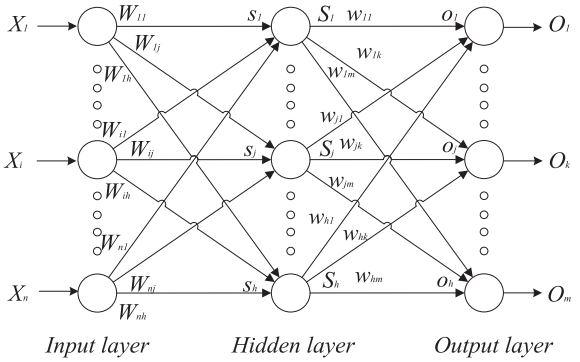


Fig. 1. An MLP with one hidden layer.

2. Multi-Layer Perceptron

Fig. 1 shows an MLP with three layers, where the number of input nodes is n, the number of hidden nodes is h, and the number of output nodes is m. It can be seen that there are one-way connections between the nodes, since the MLP belongs to the FNN family. The output of the MLP is calculated as follows:

The weighted sums of inputs are first calculated by Eq. (2.1).

$$s_j = \sum_{i=1}^{n} (W_{ij}.X_i) - \theta_j, \quad j = 1, 2, \ldots, h \tag{2.1}$$

where n is the number of the input nodes, W_{ij} shows the connection weight from the ith node in the input layer to the jth node in the hidden layer, θ_j is the bias (threshold) of the jth hidden node, and X_i indicates the ith input.

The output of each hidden node is calculated as follows:

$$S_j = sigmoid(s_j) = \frac{1}{(1 + exp(-s_j))}, \quad j = 1, 2, \ldots h \tag{2.2}$$

After calculating the outputs of hidden nodes, the final outputs are defined as follows:

$$o_k = \sum_{j=1}^{h} (W_{jk}.S_j) - \theta'_k, \quad k = 1, 2, \ldots, m \tag{2.3}$$

$$O_k = sigmoid(o_k) = \frac{1}{(1 + exp(-o_k))}, \quad k = 1, 2, \ldots, m \tag{2.4}$$

where w_{jk} is the connection weight from the jth hidden node to the kth output node, and θ'_k is the bias (threshold) of the kth output node.

The most important parts of MLPs are the connection weights and biases. As may be seen in the above equations, the weights and biases define the final values of output. Training an MLP involves finding optimum values for weights and biases in order to achieve desirable outputs from certain given inputs.

3. Biogeography-Based Optimization algorithm

The BBO algorithm was first proposed by Simon in 2008 [60]. The basic idea of this algorithm was inspired by biogeography, which refers to the study of biological organisms in terms of geographical distribution (over time and space). The case studies might include different islands, lands, or even continents over decades, centuries, or millennia. In this field of study, different ecosystems (habitats or territories) are investigated to find the relationships between different species (habitants) in terms of immigration, emigration, and mutation. The evolution of ecosystems to reach a stable situation while considering different kinds of species (such as predator and prey), and the effects of migration and mutation was the main inspiration for the BBO algorithm.

Similarly to other EAs [65–69], BBO employs a number of search agents called habitats. These habitats are analogous to chromosomes in GAs. The BBO algorithm assigns each habitat a vector of habitants (similar to genes in a GA), which represents the variables of problems. In addition, a Habitat Suitability Index (HSI) defines the overall fitness of a habitat. The higher the HSI, the more fit the habitat. The habitats evolve over time based on three main rules as follows [31]:

- Habitants living in habitats with high HSI are more likely to emigrate to habitats with low HSI.
- Habitats with low HSI are more prone to attract new immigrant habitants from those with high HSI.
- Habitats might face random changes in their habitants regardless of their HSI values.

In nature, these concepts bring a balance between different ecosystems. In other words nature tends to improve the overall stability of different geographical regions. The BBO algorithm utilizes these concepts to improve the HSI of all habitats, which results in evolving the initial random solutions for a particular problem.

The BBO algorithm starts with a random set of habitats. Each habitat has n different habitants that correspond to the number of variables of a particular problem. In addition, each habitat has its own immigration, emigration, and mutation rates. This mimics the characteristic of various geographically separated locations in nature.

Emigration (μ_k) and immigration (λ_k) are formulated as functions of the number of habitants as follows:

$$\mu_k = \frac{E \times n}{N} \tag{3.1}$$

$$\lambda_k = I \times \frac{1 - n}{N} \tag{3.2}$$

where n is the current number of habitants, N is the allowed maximum number of habitants, which is increased by HSI (the more suitable the habitat, the higher the number of habitants), E is the maximum emigration rate, and I indicates the maximum immigration rate.

These latter two rates are depicted in Fig. 2. It can be inferred from this figure that a high number of habitants coincides with a high probability of emigration and a low probability of immigration [20].

The third component of BBO, mutation, improves the exploration of BBO and keeps habitats as diverse as possible. This component is defined as follows:

$$m_n = M \times \left(1 - \frac{p_n}{p_{max}}\right) \tag{3.3}$$

where M is an initial value for mutation defined by the user, p_n is the mutation probability of the nth habitat, and $p_{max} = \text{argmax}(p_n)$, $n = 1, 2, \ldots, N$.

The general steps of the BBO algorithm are illustrated in the flowchart of Fig. 2. This figure shows that the BBO algorithm starts with a random set of habitats. After calculating the HSI of each habitat, the emigration, immigration, and mutation rates are updated. The non-elite habitants are migrated and mutated according these rates. A pre-defined number of the best habitats are saved as elites for the next generation. Finally, the BBO algorithm is terminated by the satisfaction of a termination criterion. Note that elitism prevents the best solutions from being corrupted by immigration. To do this, we retain some of the best solutions (habitats) at each iteration. So, the best solutions can be recovered if their HSI is ruined by mutation.

Simon proved that the BBO algorithm is able to outperform some well-known heuristic algorithms such as PSO, GA, ACO, ES, and Probability-based incremental learning (PBIL) [11,52] on fourteen benchmark functions and a real problem [60]. He offered BBO as a competitive algorithm in the field of optimization [61]. In the following sections BBO is first applied to an MLP, and then compared to PSO, GA, ACO, ES, PBIL, BP, and Extreme Learning Machine (ELM) on 11 datasets.

4. BBO for training an MLP

Generally, there are three methods of using a heuristic algorithm for training MLPs. Firstly, heuristic algorithms are utilized for finding a combination of weights and biases that provide the minimum error for an MLP. Secondly, heuristic algorithms are employed to find a proper architecture for an MLP in a particular problem. The last method is to use a heuristic algorithm to tune the parameters of a gradient-based learning algorithm, such as the learning rate and momentum.

In the first method, the architecture does not change during the learning process. The training algorithm is required to find proper values for all connection weights and biases in order to minimize the overall error of the MLP.

In the second approach, the structure of the MLPs varies. In this case, a training algorithm determines the best structure for solving a certain problem. Changing the structure can be accomplished by manipulating the connections between neurons, the number of hidden layers, and the number of hidden nodes in each layer. For example, Yu et al. employed PSO to define the structure of MLP to solve two real problems [79].

Leung et al. used the last method to tune the parameters of an FNN utilizing EAs [30]. There are also some studies that utilized a combination of methods simultaneously. For instance, Mizuta et al. [48] and Leung et al. [30] employed a GA and improved GA to define the structure of an FNN.

In this study, the BBO algorithm is applied to an MLP using the first method. In order to design a BBO trainer for MLPs, the following main phases need to be completed:

1. Representation strategy: the weights and biases should be represented in the proper format (habitats) for BBO.
2. HSI: a fitness function using the error of the MLP should be defined to evaluate habitats.

These phases are explained in detail in the following sections.

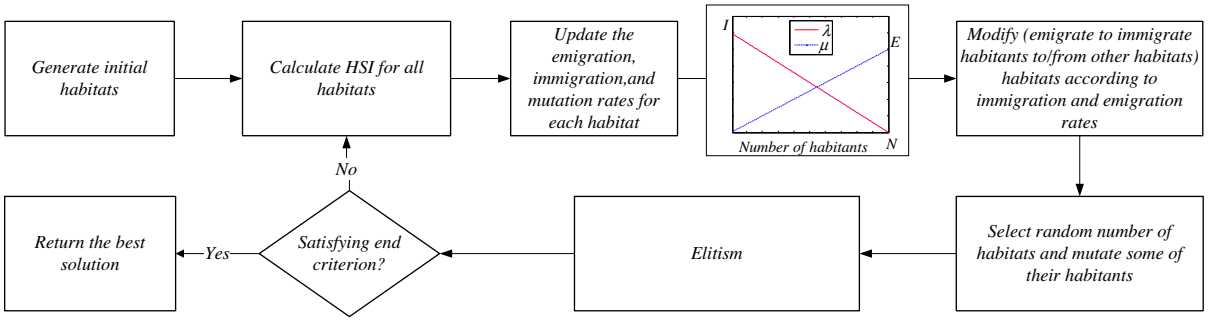


Fig. 2. General steps of the BBO algorithm.

4.1. Representing the MLP training problem for BBO (habitat formation)

There are three methods for representing (encoding) the weights and biases: vector, matrix, and binary [80]. In vector encoding, every agent is encoded as a vector. To train an MLP, each individual represents all the weights and biases. In matrix representation, every individual is encoded as a matrix. For the binary representation, agents are encoded as strings of binary bits. Each of these strategies has its own advantages and disadvantages that can be useful in a particular application [42].

In the first strategy, the encoding phase is easy, but the decoding process (decoding particles' vectors to weights and biases) is complicated. This method is often used for simple NNs. In the second strategy, the decoding stage is easy but the encoding is difficult for NNs with complex structures. This method is very suitable for learning algorithms from generalized NN toolboxes. In the third strategy, we need to represent variables in binary form. In this case, the length of each particle will be increased when the structure becomes more complex, so the process of decoding and encoding also becomes very complicated.

In this study, the vector method is used since we are not dealing with MLPs with complex structure. We do not use any generic toolboxes because the run time is much less for our hand-coded MLPs. As an example of this encoding strategy, the final vector of the MLP presented in Fig. 3 is as follows:

$$habitat = [w_{13}w_{23}w_{14}w_{24}w_{15}w_{25}w_{36}w_{46}w_{56}\theta_1\theta_2\theta_3\theta_4] \tag{4.1}$$

After representing MLPs in the form of habitat vectors, an HSI formulation (fitness function) is required for evaluating each of them.

4.2. Habitat Suitability Index (fitness function)

Generally speaking, the ultimate goal of learning methods is to train an MLP so that it is able to recognize training, validation, and test sets completely. The most important set in the learning phase is the training set. Each training sample should be involved in calculating the overall HSI of each individual. The following HSI function, which is the Mean Square Error (MSE) for all training samples, is utilized in this work:

$$E = \sum_{k=1}^{q} \frac{\sum_{i=1}^{m} (o_i^k - d_i^k)^2}{q} \tag{4.2}$$

where q is the number of training samples, m is the number of outputs, d_i^k is the desired output of the ith input unit when the kth training sample is used, and o_i^k is the actual output of the ith input unit when the kth training sample appears in the input.

As an example, the HSI value for the ith habitat is calculated by (4.3):

$$HSI(Habitat_i) = E(Habitat_i) \tag{4.3}$$

Learning in the MLP can be formulated for BBO algorithms using these two phases. The flowchart of the proposed method is illustrated in Fig. 4. As may be seen in this figure, the proposed method starts by generating a random set of MLPs based on the defined number of habitats. Each MLP corresponds to a habitat, and each weight/bias corresponds to habitants in the habitat. After the initialization step, the MSE for each MLP is calculated by Eq. (4.2). The next step is to update emigration, immigration, and mutation rates by Eqs. (3.1)–(3.3), respectively. Afterwards, the MLPs are combined based on the emigration and immigration rates. Each MLP is then mutated based on its habitat's mutation rate. The last step of the proposed method is elite selection, in which some of the best MLPs are saved in order to prevent them from being corrupted by the evolutionary and mutation operators in the next generation. These steps (from calculating MSE for every MLP to elitism) are iterated until satisfaction of a termination criterion.

The computational complexity of the proposed method depends on the number of training samples in datasets, the structure of the MLP, the number of habitats, the number of generations, the migration mechanism, mutation mechanism, and elitism mechanism. So, the overall computational complexity is as follows:

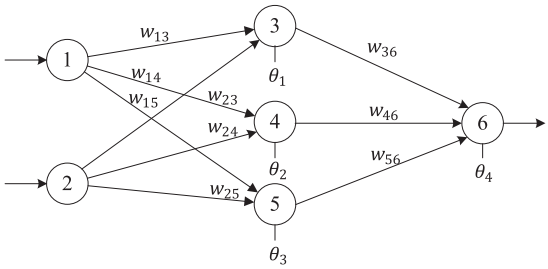


Fig. 3. MLP with the structure of 2-3-1.

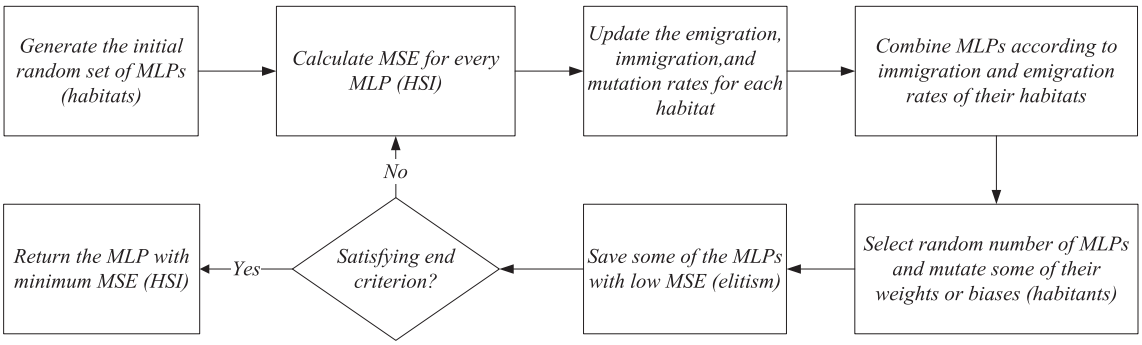


Fig. 4. Flow chart of the proposed method.

$$O(MLP, BBO) = O(g(O(MLP) + O(migration) + O(mutation) + O(elitism))) \tag{4.4}$$

where g is the maximum number of generations.

The computational complexity of an MLP with h hidden nodes, o outputs, and t training samples is equal to $O(t(h+o))$. The computational complexity of migration is of $O(mn^2)$ in our implementation where m is the number of habitants and n is the number of habitats. Note that the migration operator is of $O(mn)$ in the best case. The computational complexity of the mutation operator is of $O(nm)$ in the worst case. Since we utilized quicksort to find the best habitats in the elitism phase, the computational complexity of the elitism is of $O(nlogn)$ in the best case and $O(n^2)$ in the worst case. Therefore, the final computational complexity of the proposed method is as follows:

$$O(MLP, BBO) = O(g(t(h+o) + mn^2 + nm + n^2)) \tag{4.5}$$

where g is the maximum number of generations, t is the number of training samples, h is the number of hidden nodes, o is the number of output nodes, m is the number of habitants, and n is the number of habitats.

To see how the proposed method works, a conceptual picture of the migration between the habitats for learning in an MLP using BBO is visualized in Fig. 5. In this figure, habitat 1 is the fittest (least HSI, which indicates the least MSE for all training samples), followed by habitat 2, habitat 3, and habitat 4. The habitants represent the weights and biases of the MLPs. It can be seen that habitat 1 has the highest emigration, whereas habitat 4 provides the highest immigration. So the fourth habitat accepts many habitants (weights and biases) from other habitats as illustrated in different colors. The green nodes and connections also depict mutation which happen for all the habitats regardless of their HSI values. This example shows how the MLPs evolve using the proposed method.

To see how the proposed BBO-based technique theoretically has the potential to improve the training of MLPs, some observations are:

- Various values of emigration and immigration rates provide different evolutionary mechanism for every habitat, which promotes exploration.
- Higher exploration prevents BBO from easily getting trapped in local optima and provides higher probability of resolving stagnation situations (in case of local optima stagnation).
- The MSEs (HSI) of all MLPs (habitats) are improved over the generations since weights/biases of better MLPs tend to migrate to the worse MLPs. This guarantees the convergence of the proposed method and the improvement of all MLPs during generations.
- Different mutation rates for each habitat assist the proposed method to show diverse exploitation mechanisms.
- Elitism assists the proposed method to save and retrieve some of the best solutions, so they are never lost.

In the following section, a comparative study is conducted to investigate these theoretical claims in practice.

5. Results and discussion

In this section the BBO algorithm is benchmarked on 5 classification and 6 function approximation datasets. The classification datasets are XOR, balloon, iris, breast cancer, and heart, obtained from the University of California at Irvine (UCI) Machine Learning Repository [5]. The function approximation datasets are a one-dimensional sigmoid, one-dimensional cosine with one peak, one-dimensional sine with four peaks, two-dimensional sphere, two-dimensional Griewank, and five-dimensional Rosenbrock functions. The BBO algorithm is compared to PSO, GA, ACO, ES, and PBIL over these benchmark datasets in order to verify its performance. The comparison with BP and ELM are discussed in Sections 5.3 and 5.4, respectively. Note that the source codes of the BBO trainer are available in http://www.alimirjalili.com/Projects.html.

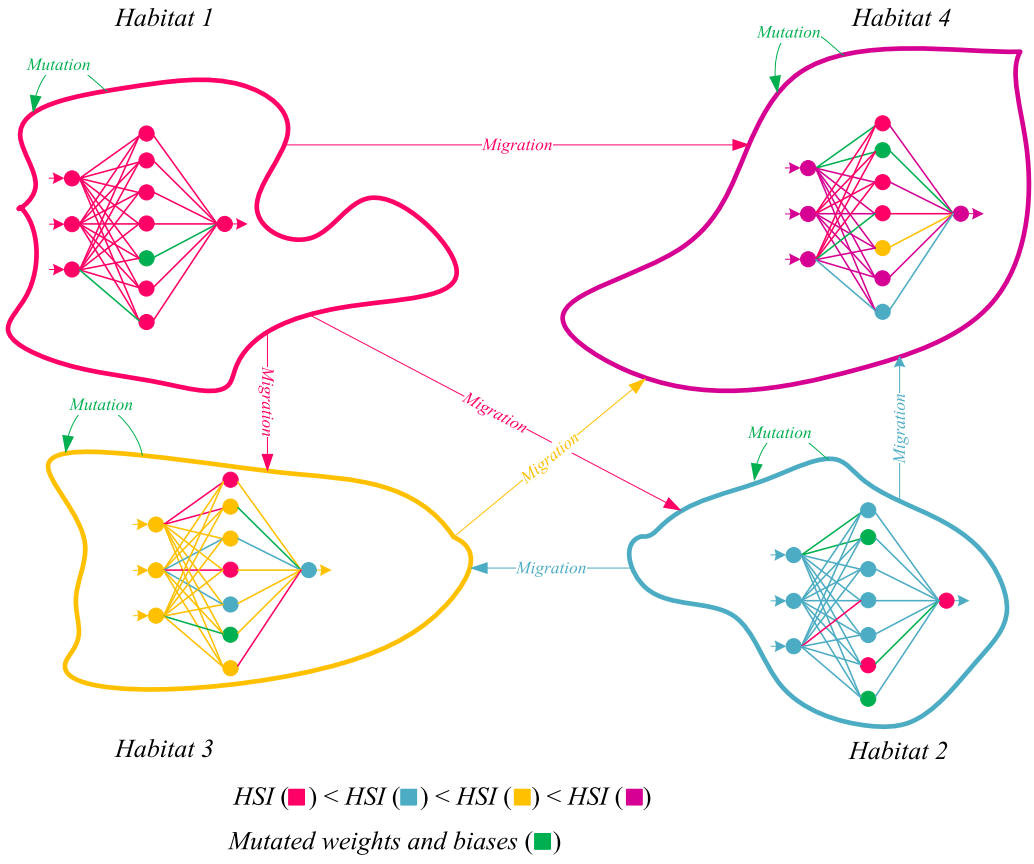


Fig. 5. Conceptual model of migration between the habitats for training MLP.

It is assumed that every habitat is randomly initialized in the range $[-10,10]$. The population size is 50 for the XOR and balloon datasets; and 200 for the iris, breast cancer, heart, and all function approximation datasets. Tables 2 and 8 show how the datasets are divided in terms of training and test sets. It can be seen that the training and test samples are similar for the XOR, balloon, and iris datasets. However, the test sets are different from the training sets in the breast cancer and heart datasets. The breast cancer dataset has 599 training samples and 100 test samples, while the heart dataset has 80 training samples and 187 test samples. It can be seen in Table 8 that the training and test samples for the function approximation datasets are chosen using different step sizes from the domains of the functions. In most of the function approximation datasets the test samples are twice the training samples.

The other assumptions and initial values are presented in Table 1. Fine-tuning of the algorithms is beyond the scope of this paper. Each algorithm was run 10 times and the average (AVE) and standard deviation (STD) are reported in the table of results. These two measures indicate the ability of algorithms to avoid local minima. The lower the value of AVE ± STD, the greater the capability of the algorithm to avoid local minima. AVE indicates the average of MSE over 10 runs, so a lower value for this metric is evidence of an algorithm more successfully avoiding local optima and finding solutions near the global optimum. However, AVE is not a good metric alone because two algorithms can have equal averages, but have different performance in terms of finding the global optimum in each run. Therefore, STD (standard deviation) can help to determine the dispersion of results. The lower the STD, the lower the dispersion of results. So, AVE ± STD may be a good combination to indicate the performance of an algorithm in terms of avoiding local minima. Note that the best results are highlighted in bold type in Tables 3–7 and 9–15.

According to Derrac et al. [12], statistical tests should be conducted to properly evaluate the performance of heuristic algorithms. It is not enough to compare algorithms based on the mean and standard deviation values [15]; a statistical test is necessary to prove that a proposed new algorithm presents a significant improvement over other existing methods for a particular problem [38].

In order to judge whether the results of BBO differ from PSO, GA, ACO, ES, and PBIL in a statistically significant way, a nonparametric statistical test, Wilcoxon's rank-sum test [76], was carried out at 5% significance level. The calculated p-values in the Wilcoxon's rank-sum are given in the results as well. In the tables, N/A indicates "Not Applicable" which means that the corresponding algorithm cannot be compared with itself in the rank-sum test. Conventionally, p-values less than

Table 1
The initial parameters of algorithms.

Algorithm	Parameter	Value
BBO	Habitat modification probability	1
	Immigration probability bounds per gene	[0,1]
	Step size for numerical integration of probabilities	1
	Max immigration (I) and Max emigration (E)	1
	Mutation probability	0.005
	Population size	50 for XOR and Balloon, 200 for the rest
	Maximum number of generations	250
PSO	Topology	Fully connected
	Cognitive constant (C_1)	1
	Social constant (C_2)	1
	Inertia constant (w)	0.3
	Population size	50 for XOR and Balloon, 200 for the rest
	Maximum number of iterations	250
GA	Type	Real coded
	Selection	Roulette wheel
	Crossover	Single point (probability = 1)
	Mutation	Uniform (probability = 0.01)
	Population size	50 for XOR and Balloon, 200 for the rest
	Maximum number of generations	250
ACO	Initial pheromone (τ_0)	1e−06
	Pheromone update constant (Q)	20
	Pheromone constant (q_0)	1
	Global pheromone decay rate (p_g)	0.9
	Local pheromone decay rate (p_l)	0.5
	Pheromone sensitivity (α)	1
	Visibility sensitivity (β)	5
	Population size	50 for XOR and Balloon, 200 for the rest
	Maximum number of iterations	250
ES	λ	10
	σ	1
	Population size	50 for XOR and Balloon, 200 for the rest
	Maximum number of generations	250
PBIL	Learning rate	0.05
	Good population member	1
	Bad population member	0
	Elitism parameter	1
	Mutational probability	0.1
	Population size	50 for XOR and Balloon, 200 for the rest
	Maximum number of generations	250

Table 2
Classification datasets.

Classification datasets	Number of attributes	Number of training samples	Number of test samples	Number of classes
3-bits XOR	3	8	8 as training samples	2
Balloon	4	16	16 as training samples	2
Iris	4	150	150 as training samples	3
Breast cancer	9	599	100	2
Heart	22	80	187	2

0.05 are considered as strong evidence against the null hypothesis. Note that p-values greater than 0.05 are underlined in the tables.

The other comparative measures shown in the results are: classification rates and test errors. We chose the best trained MLP among 10 runs and used it to classify or approximate the test set. To provide a fair comparison, all algorithms were terminated when a maximum number of iterations (250) was reached. Finally, the convergence behavior is also investigated in the results to provide a comprehensive comparison.

It should be noted that min–max normalization was used for those datasets containing data with different ranges. The normalization method was formulated as follows:

Suppose that we are going to map x in the interval of $[a,b]$ to $[c,d]$. The normalization process is done by the following equation:

$$x' = \frac{(x-a)\times(d-c)}{(b-a)} \tag{5.1}$$

Regarding the structure of the MLPs, we used the suggestions in [27,37] for classification datasets, since there is no standard rule for selecting the number of hidden nodes, i.e.:

$$H = 2\times N + 1 \tag{5.2}$$

where N indicates the number of inputs and H is the number of hidden nodes.

Note that we used 15 hidden nodes for function approximation datasets.

In the following sections the simulation results of benchmark datasets are presented and discussed.

5.1. Classification problems

The classification problems are listed in Table 2. It can be seen that these datasets are chosen with various levels of difficulty: XOR is the simplest and the Heart dataset is the most difficult [62]. Note that the larger the number of training samples, the lower the difficulty of the classification problem. However, with a large number of features, the size of the neural network is larger, so that more weights need to be correctly determined. The results of the training algorithms on these datasets are as follows:

5.1.1. XOR classification problem

The N-bit XOR problem is a famous non-linear benchmark problem. The problem is to recognize the number of "1"s in the input vector. The XOR result of the input vector should be returned; if the input vector contains an odd number of "1"s, the output is "1". If the input vector contains an even number of "1"s, the output is "0".

We used an MLP with the structure 3-7-1 to solve this problem. The experimental results for this problem are shown in Table 3. It can be seen that the results of BBO and the GA are better than for the other training algorithms. There is no statistically significant difference between BBO and GA in terms of avoiding local minima, and the classification rate is 100% for both algorithms. However, as shown in Fig. 6, the convergence rate of BBO is much faster than all other algorithms.

5.1.2. Balloon classification problem

This is a simple dataset that is based on different conditions of an experiment in blowing up a balloon. There are 16 instances with 4 attributes such as color, size, act, and age in this dataset. The attributes are in string format. There is one output that indicates whether the balloon is inflated or not. MLPs with the structure 4-9-1 were used for classifying this dataset. The experimental results are shown in Table 4 and Fig. 7.

The MSE results show that BBO has the best performance in terms of avoiding local minima, and this is confirmed to be statistically significantly better than for other algorithms. However, the classification rates are equal in this dataset.

Fig. 7 shows the convergence curves of the algorithms. As may be observed, the fastest convergence rate is that of the BBO algorithm, followed by the GA and ACO.

5.1.3. Iris classification problem

The Iris dataset has 150 samples that can be divided into three classes: Setosa, Versicolor, and Virginica. All samples have four features: sepal length, sepal width, petal length, and petal width. This dataset was solved using MLPs with the structure 4-9-3. The results of training algorithms to solve this problem are presented in Table 5.

According to the results of AVE, STD, and p-values, BBO provides the best ability to avoid local minima for this dataset. Moreover, this algorithm is able to classify 90% of the test samples, more than all the other algorithms. Fig. 8 illustrates the convergence curves for this dataset. As may be seen in this figure, the convergence rate of BBO is dramatically better than the other algorithms. Overall, the results of BBO are quite impressive for this dataset.

Table 3
Experimental results for XOR dataset.

Algorithm	MSE (AVE ± STD)	p-Values	Classification rate (%)
BBO	**3.65e−07 ± 0.000000**	**N/A**	**100.00**
PSO	0.084050 ± 0.035945	6.39e−05	37.50
GA	0.000181 ± 0.000413	0.1153	**100.00**
ACO	0.180328 ± 0.025268	6.39e−05	62.50
ES	0.118739 ± 0.011574	6.39e−05	62.50
PBIL	0.030228 ± 0.039668	6.39e−05	62.50

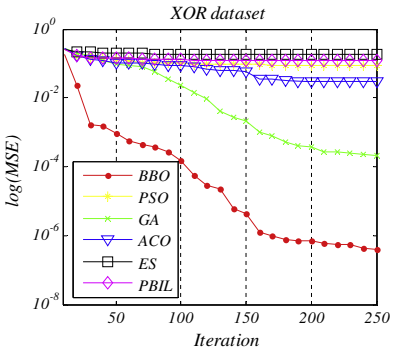


Fig. 6. Convergence curves of algorithms for XOR dataset.

Table 4
Experimental results for balloon dataset.

Algorithm	MSE (AVE ± STD)	*p*-Values	Classification rate (%)
BBO	**8.09e−27 ± 1.51e−42**	**N/A**	**100.00**
PSO	0.000585 ± 0.000749	1.83e−04	**100.00**
GA	5.08e−24 ± 1.06e−23	2.20e−03	**100.00**
ACO	0.004854 ± 0.007760	1.83e−04	**100.00**
ES	0.019055 ± 0.170260	1.83e−04	**100.00**
PBIL	2.49e−05 ± 5.27e−05	1.83e−04	**100.00**

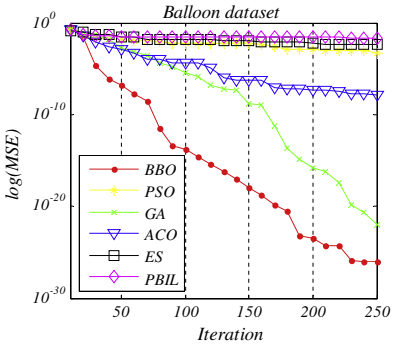


Fig. 7. Convergence curves of algorithms for balloon dataset.

Table 5
Experimental results for iris dataset.

Algorithm	MSE (AVE ± STD)	*p*-Values	Classification rate (%)
BBO	**0.019150 ± 3.66e−18**	**N/A**	**90.00**
PSO	0.228680 ± 0.057235	6.39e−05	37.33
GA	0.089912 ± 0.123638	6.39e−05	89.33
ACO	0.405979 ± 0.053775	6.39e−05	32.66
ES	0.314340 ± 0.052142	6.39e−05	46.66
PBIL	0.116067 ± 0.036355	6.39e−05	86.66

5.1.4. Breast cancer classification problem

This dataset was established by William H. Wolbergby from the University of Wisconsin Hospitals, Madison, and has 699 instances and 9 attributes such as clump thickness, uniformity of cell size, uniformity of cell shape, and marginal adhesion [33].

The output is equal to 2 for benign and 4 for malignant cancers. The architecture of MLPs for solving this dataset was 9-19-1. The experimental results of algorithms for this dataset are presented in Table 6. The averages and standard deviations indicate that BBO has the best ability to avoid local minima for this dataset. However, *p*-values show that the

Table 6
Experimental results for breast cancer dataset.

Algorithm	MSE (AVE ± STD)	*p*-Values	Classification rate (%)
BBO	**0.002807 ± 0.000000**	**N/A**	95.00
PSO	0.034881 ± 0.002472	6.39e−05	11.00
GA	0.003026 ± 0.001500	1.0000	**98.00**
ACO	0.013510 ± 0.002137	6.39e−05	40.00
ES	0.040320 ± 0.002470	6.39e−05	06.00
PBIL	0.032009 ± 0.003065	6.39e−05	07.00

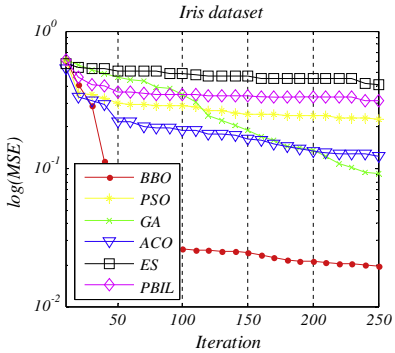


Fig. 8. Convergence curves of algorithms for iris dataset.

results of BBO are not significantly better than the GA. The classification rate of the GA is also better. The results of BBO and GA are very close in this dataset, and they both significantly outperform the other algorithms.

Fig. 9 shows the convergence curves of algorithm for this dataset. It is evident that BBO has the fastest convergence rate, followed by the GA.

5.1.5. Heart classification problem

This dataset was created for diagnosing cardiac Single Proton Computed Tomography (SPECT) images. There are 267 images in this dataset, and 22 features are extracted from them to summarize these images. All the features are in binary

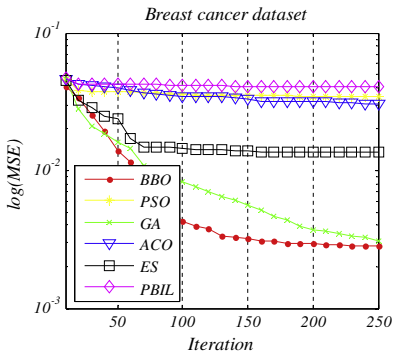


Fig. 9. Convergence curves of algorithms for breast cancer dataset.

Table 7
Experimental results for heart dataset.

Algorithm	MSE (AVE ± STD)	*p*-Values	Classification rate (%)
BBO	**0.065625 ± 1.46e−17**	**N/A**	**76.25**
PSO	0.188568 ± 0.008939	0.0001	68.75
GA	0.093047 ± 0.022460	0.0072	58.75
ACO	0.228430 ± 0.004979	0.0001	00.00
ES	0.192473 ± 0.015174	0.0001	71.25
PBIL	0.154096 ± 0.018204	0.0001	45.00

format and the output indicates whether the situation of a patient is normal or abnormal. We trained MLPs with the structure of 22-45-1 using 80 instances and examined them for the remaining 187 instances.

The experimental results are provided in Table 7. The results of BBO are significantly better than all the other algorithms according to the MSEs and p-values. Moreover, BBO has the highest classification accuracy as well.

The convergence curves in Fig. 10 show that BBO has very fast convergence compared to the other algorithms.

Statistically speaking, BBO has the best results in all of the classification datasets in terms of avoiding local minima, significantly better than the other algorithms in 2 cases. The classification rates attained by BBO are better than the other algorithm in three out of five datasets. According to the convergence curves, BBO tends to have the fastest convergence behavior on all datasets.

5.2. Function approximation problems

The function approximation datasets are presented in Table 8. In this section, MLPs with structures 1-15-1, 2-15-1, and 5-15-1 are trained for approximating the one, two, and five-dimensional functions, respectively. Similarly to the classification datasets, the function approximation datasets are also chosen with a range of difficulty to challenge the training algorithms. The dimension and training samples are increased which increases the level of difficulty for datasets.

5.2.1. Sigmoid function

The sigmoid dataset is in the interval $[-3,3]$ with increments of 0.1, so the number of training data is 61. The number of test samples is 121, lying in the same range. The results of training algorithms for this dataset are presented in Table 9, Figs. 11 and 12. The results for AVE, STD, and p-values indicate that BBO is much better at avoiding local minima than

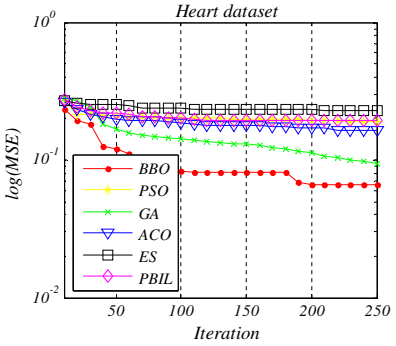


Fig. 10. Convergence curves of algorithms for heart dataset.

Table 8
Function approximation datasets.

Function approximation datasets	Training samples	Test samples
Sigmoid: $y = 1/(1+e^{-x})$	61: x in [−3:0.1:3]	121: x in [−3:0.05:3]
Cosine: $y = (\cos(x\pi/2))^7$	31: x in [1.25:0.05:2.75]	38: x in [1.25:0.04:2.75]
Sine: $y = \sin(2x)$	126: x in $[-2\pi{:}0.1{:}2\pi]$	252: x in $[-2\pi{:}0.05{:}2\pi]$
Sphere: $z = \sum_{i=1}^{2} x_i^2, x = x_1$ and $y = x_2$	21 × 21: x,y in [−2:0.2:2]	41 × 41: x,y in [−2:0.1:2]
Griewank: $z = \frac{1}{4000}\sum_{i=1}^{2} x_i^2 - \Pi_{i=1}^{2}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1, x = x1$ and $y = x_2$	21 × 21: x,y in [−1:0.1:1]	41 × 41: x,y in [−1:.05:1]
Rosenbrock: $z = \sum_{i=1}[x_i^2 - 10\cos(2\pi x_i) + 10], i = 1,2,3,4,5$	1024: $x_1 - x_5$ in [−5:3:5]	7776: $x_1 - x_5$ in [−5:2:5]

Table 9
Experimental results for sigmoid dataset (one dimensional).

Algorithm	MSE (AVE ± STD)	p-Values	Test error
BBO	**1.33e−05 ± 3.57e−21**	N/A	**0.14381**
PSO	0.022989 ± 0.009429	6.39e−05	3.35630
GA	0.001093 ± 0.000916	6.39e−05	0.44969
ACO	0.023532 ± 0.010042	6.39e−05	3.99740
ES	0.075575 ± 0.016410	6.39e−05	8.80150
PBIL	0.004046 ± 2.74e−17	6.34e−05	2.9446

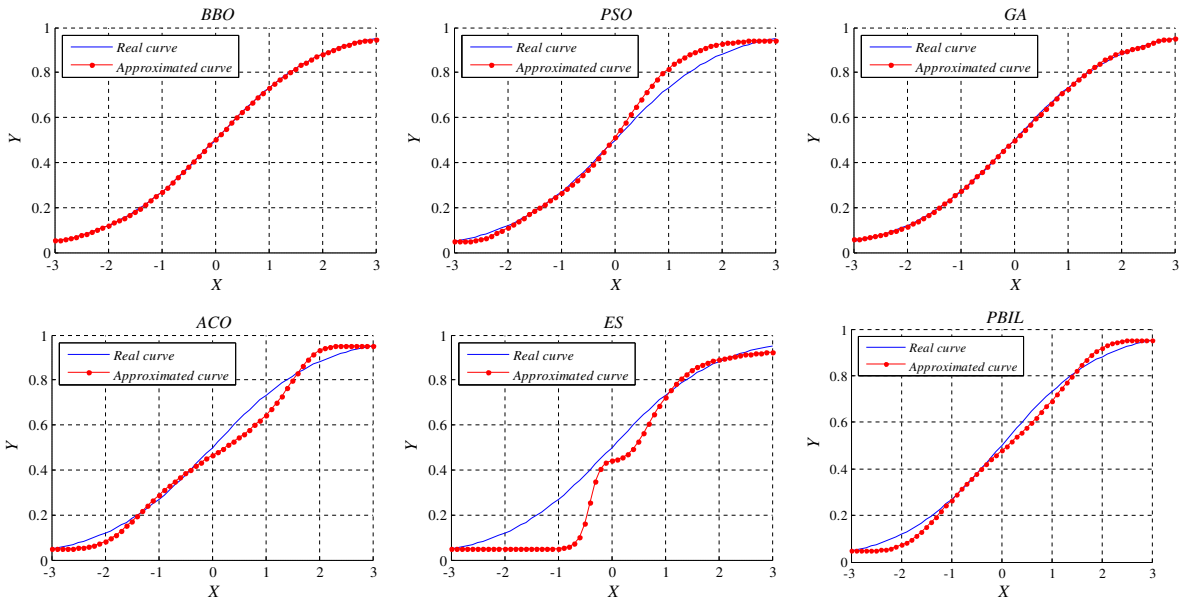


Fig. 11. Approximated curves for sigmoid function.

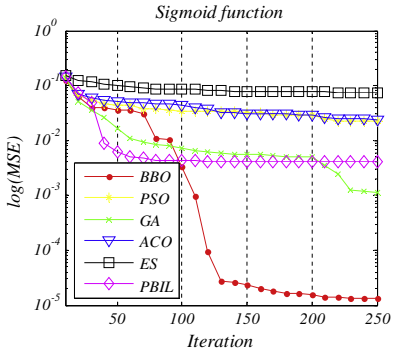


Fig. 12. Convergence curves of algorithms for sigmoid function.

the other algorithms. The test errors in Table 9 and approximated curves in Fig. 11 show that the BBO algorithm has the best approximate accuracy as well. Fig. 12 shows that BBO has the fastest convergence rate.

5.2.2. Cosine function

This dataset has 31 training samples and 38 test samples. The results are shown in Table 10. It can be seen that the GA has the best results for AVE, STD, and test errors. However, the results of BBO are very close to this: the statistical tests show that the differences between the GA and BBO are not statistically significant. Figs. 13 and 14 illustrate the approximated curve and convergence, respectively.

5.2.3. Sine function

This dataset is created by a four-peak sine function in the interval $[-2\pi, 2\pi]$ with a step size of 0.1 for training samples and 0.05 for test samples. For training the algorithms use 126 samples and are tested by 252 samples. The experimental results are provided in Table 11, Figs. 15 and 16. The results show that the BBO algorithm is significantly better than the other algorithms in terms of avoiding local minima. In addition, test errors and Fig. 15 verify the superior accuracy of this algorithm. Finally, the curves of Fig. 16 confirm that BBO provides the fastest convergence speed.

5.2.4. Sphere function

The sphere function dataset consists of 441 training samples and 1681 test samples. The results of this dataset are reported in Table 12. The averages and standard deviations of the MSEs show that BBO is the most reliable algorithm in terms

Table 10
Experimental results for one-peak cosine dataset (one dimensional).

Algorithm	MSE (AVE ± STD)	*p*-Values	Test error
BBO	0.013674 ± 1.83e−18	0.4429	1.4904
PSO	0.058986 ± 0.021041	0.0001	2.0090
GA	**0.010920 ± 0.006316**	**N/A**	**0.7105**
ACO	0.050872 ± 0.010809	0.0001	2.4498
ES	0.086640 ± 0.022208	0.0001	3.1461
PBIL	0.094342 ± 0.018468	0.0001	3.7270

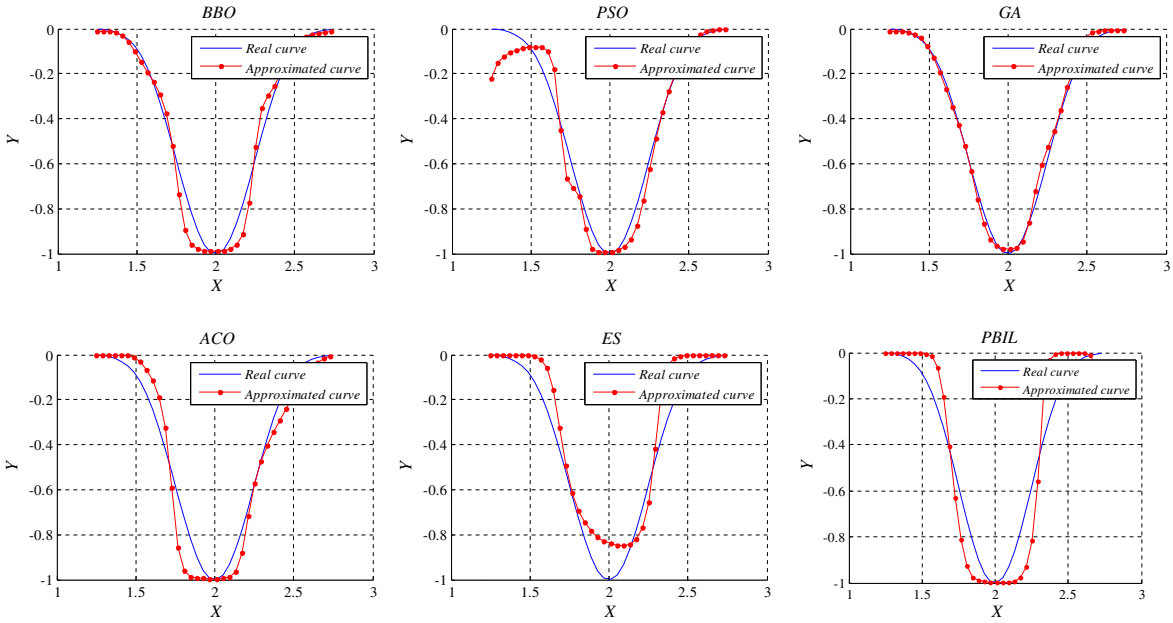


Fig. 13. Approximated curves for cosine function.

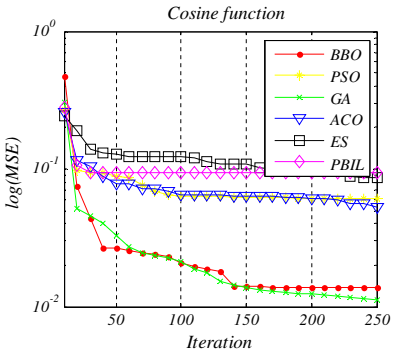


Fig. 14. Convergence curves of algorithms for cosine function.

of avoiding local minima. However, the results for the test error criterion and the best approximation surfaces in Fig. 17 indicate that the GA provides better accuracy. The convergence behaviors of the training algorithm are shown in Fig. 18. The convergence rates are entirely consistent with the previous datasets: BBO has the fastest rate, followed by the GA.

5.2.5. Griewank function

The Griewank dataset includes 441 training instances and 1681 test samples. The results for this dataset are shown in Table 13, Figs. 19 and 20. The results are quite similar to the previous datasets in that the BBO algorithm significantly

Table 11
Experimental results for four-peak sine dataset (one dimensional).

Algorithm	MSE (AVE ± STD)	*p*-Values	Test error
BBO	**0.102710 ± 0.000000**	**N/A**	**64.261**
PSO	0.526530 ± 0.072876	6.39e−05	124.89
GA	0.421070 ± 0.061206	6.39e−05	111.25
ACO	0.529830 ± 0.053305	6.39e−05	117.71
ES	0.706980 ± 0.077409	6.39e−05	142.31
PBIL	0.483340 ± 0.007935	6.39e−05	149.60

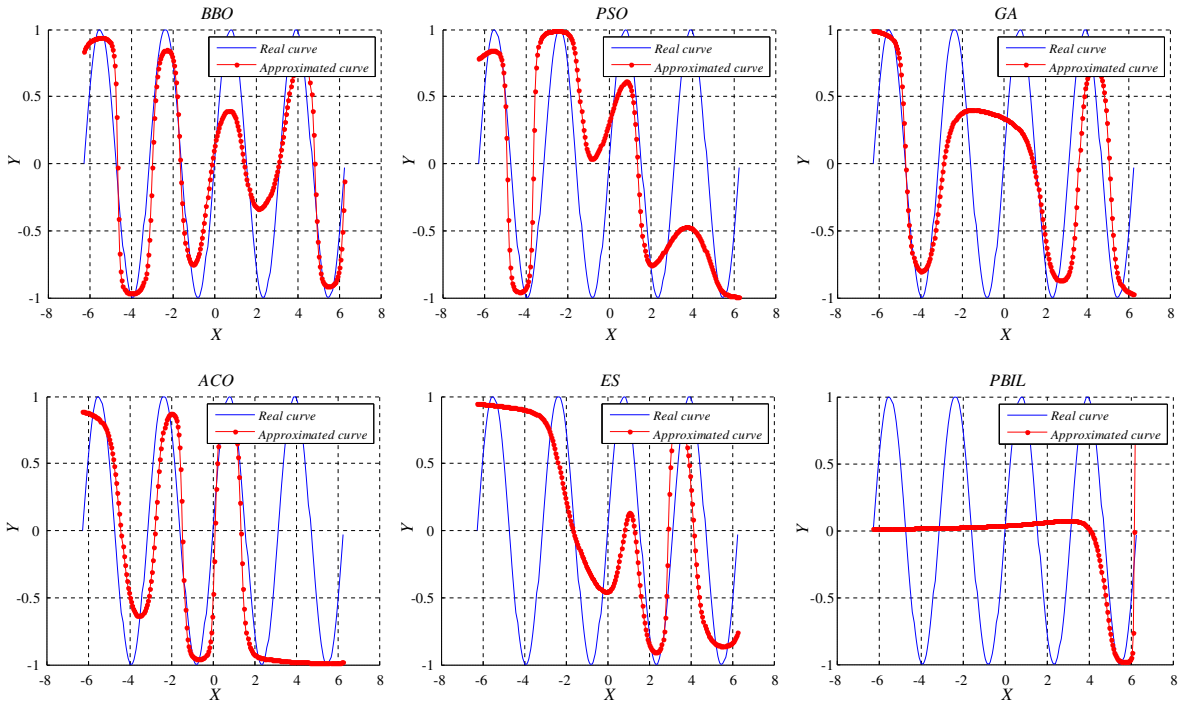


Fig. 15. Approximated curves for sine function.

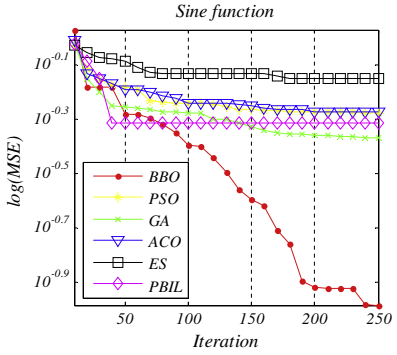


Fig. 16. Convergence curves of algorithms for sine function.

outperforms the other algorithms in terms of avoiding local minima. According to test errors, Figs. 19 and 20, the BBO algorithm has the best approximation accuracy and convergence speed as well.

5.2.6. Rosenbrock function

This dataset includes 1024 training samples and 7776 test samples. The results for this dataset are provided in Table. 14. According to the values of AVE, STD, and *p*-values, the BBO algorithm significantly outperforms others in terms of avoiding

Table 12
Experimental results for sphere dataset (two dimensional).

Algorithm	MSE (AVE ± STD)	*p*-Values	Test error
BBO	**1.7740 ± 2.34e−16**	**N/A**	770.4425
PSO	7.1094 ± 0.8528	6.38e−05	1.2686e+03
GA	2.9276 ± 0.7289	6.38e−05	**452.3744**
ACO	6.5626 ± 0.9543	6.38e−05	1.222e+03
ES	10.530 ± 0.8427	6.38e−05	1.500e+03
PBIL	10.459 ± 0.8529	6.38e−05	1.805e+03

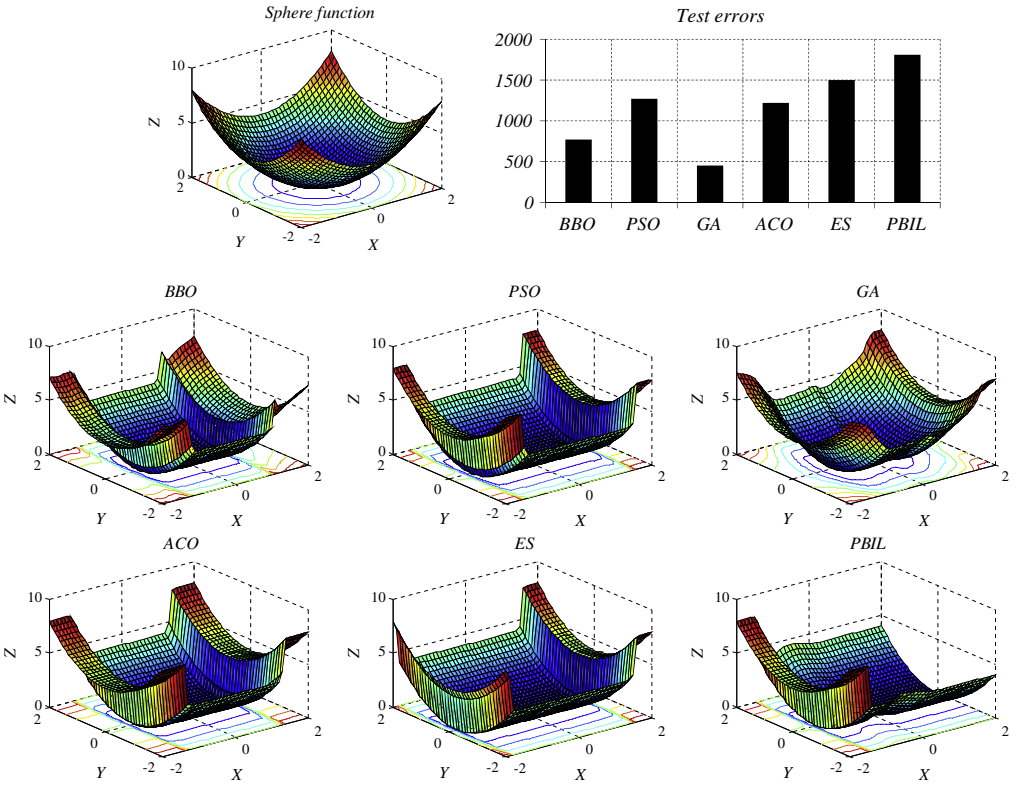


Fig. 17. Test errors and approximated surfaces for sphere function.

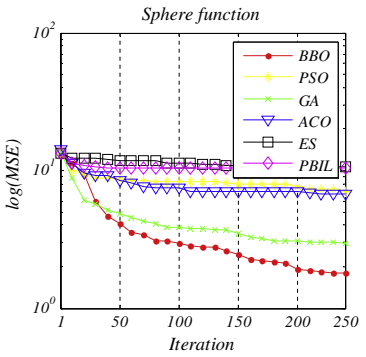


Fig. 18. Convergence curves of algorithms for sphere function.

Table 13
Experimental results for Griewank dataset (two dimensional).

Algorithm	MSE (AVE ± STD)	*p*-Values	Test error
BBO	**2.02560 ± 0.0000**	**N/A**	**5.0200e+03**
PSO	16.9043 ± 1.1482	6.38e−05	7.6328e+03
GA	16.4218 ± 1.3306	6.38e−05	1.0585e+04
ACO	14.9586 ± 1.2976	6.38e−05	6.1790e+03
ES	19.1069 ± 1.4984	6.38e−05	1.1185e+04
PBIL	13.5587 ± 0.7532	6.29e−05	1.2409e+04

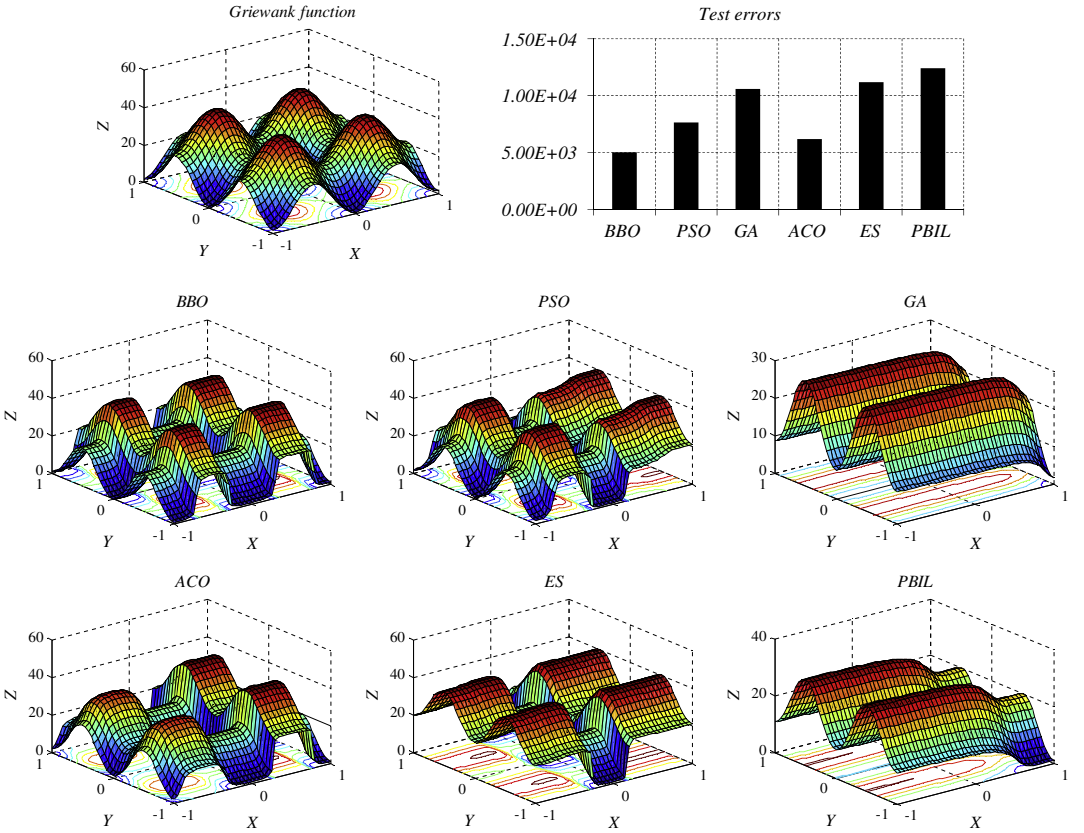


Fig. 19. Test errors and approximated surfaces for Griewank function.

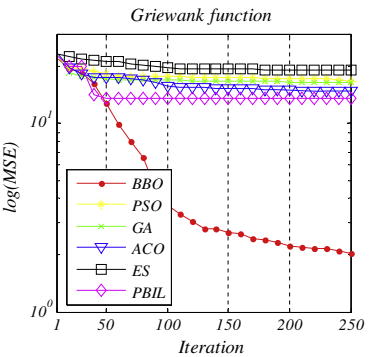


Fig. 20. Convergence curves of algorithms for Griewank function.

local minima for this dataset. The test errors show that BBO has the highest accuracy for approximating the Rosenbrock function. Fig. 21 shows that the BBO algorithm has the fastest convergence speed.

Statistically speaking, the BBO algorithm provides significant ability to avoid local minima and a high convergence rate on four out of six function approximation datasets. Moreover, the approximated functions using BBO are more accurate than the other training algorithms on four out of six function approximation datasets.

Overall, the statistical results of all comparative measures are presented in Fig. 22. It appears that the BBO algorithm has significantly superior performance. The algorithm shows improved ability to avoid local minima in seven of the datasets. The classification rates and test errors of BBO are the best in eight of the datasets. Furthermore, BBO provides the fastest convergence rate in ten out of eleven datasets.

Note that the test errors of all algorithms are not very small for all function approximation datasets because the end criterion for all algorithms is the maximum number of iterations and the number of search agents is fixed throughout the experiments. Of course, increasing the number of iterations and population size would improve the absolute classification rate or test error but it was the comparative performance between algorithms that was of interest. Determining the ability of algorithms in terms of avoiding local minima in the classification or approximation search spaces was the main objective of this work, so we did not focus on finding the best maximum number of iterations and population size.

One of the things that can be inferred from the statistical results of Fig. 22 is the superior performance of BBO and the GA in all search spaces. This is possibly due to the nature of these algorithms which have migration and crossover strategies to

Table 14
Experimental results for Rosenbrock dataset (five-dimensional).

Algorithm	MSE (AVE ± STD)	*p*-Values	Test error
BBO	**104.4251 ± 1.59e−14**	**N/A**	**2.3503e+06**
PSO	215.1669 ± 3.5521	0.007937	6.4955e+09
GA	264.6713 ± 16.3603	0.007937	5.2351e+06
ACO	200.9749 ± 7.0641	0.007937	3.5399e+07
ES	252.8454 ± 8.1445	0.007937	7.3733e+09
PBIL	208.6148 ± 4.6813	0.007937	2.3463e+08

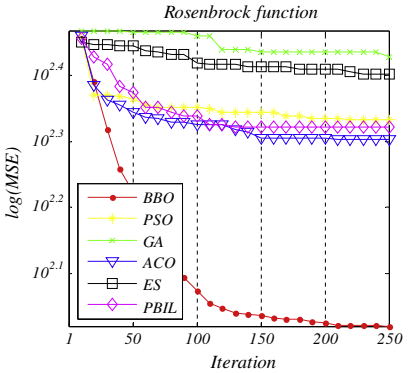
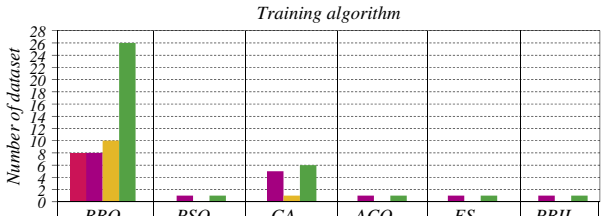


Fig. 21. Convergence curves of algorithms for Rosenbrock function.



	BBO	PSO	GA	ACO	ES	PBIL
Significant AVE ± STD	8	0	0	0	0	0
Classification rate and test error	8	1	5	1	1	1
Convergence rate	10	0	1	0	0	0
Total	26	1	6	1	1	1

Fig. 22. Final statistical results (*each bar represents the number of datasets each algorithm "won" on a variety of measures*).

avoid local minima. These operators cause abrupt changes in the candidate solutions that results in enhancing the exploration ability of BBO and the GA significantly. Since the problem of training MLPs has a very difficult search space and is changed for every dataset, these algorithms performed very well in this work because of their very good exploration. The poor results of ACO and PSO are also interesting, originating from the nature of these problems. ACO and PSO do not have operators that promote sudden changes in the candidate solutions, so they are trapped in local minima more often than BBO and the GA. Moreover, ACO uses a pheromone matrix that enhances reinforcement learning and exploitation, an advantage for combinatorial problems but again leading to a tendency toward local minima entrapment. The PSO algorithms are also highly dependent on the distribution of the initial swarm, and their prime motivators are based on attraction between members of the swarm. If many of the particles become trapped in a wide local optimum, there is little in the algorithm to prevent other particles from also being attracted and becoming trapped in that same local optimum.

Another fact worth mentioning in the results is the poor performance of ES, especially on the function approximation datasets. The ES algorithm mainly relies on sophisticated mutation operators and the selection procedure of the individuals is deterministic compared to the GA. Consequently, the mechanism of selection and mutation mostly emphasizes exploitation rather than exploration which results in very poor outcomes on the datasets. The PBIL algorithm also performed very badly on most of the datasets. PBIL evolves a vector consisting of the entire population rather than each individual itself. This vector is updated based on the fittest individual and mutated randomly. This mechanism appears to provide less exploration than the GA, where the production of new individuals by crossover operators, introducing bulk changes in their structure, enhances exploration. This may be the main reason of PBIL's poor performance.

The reason for superior performance of the BBO algorithm compared to GA in the majority of the datasets (as suggested by the final statistical results of Fig. 22) is due to the various emigration and immigration rates for habitats. In contrast to the GA that has a general reproduction rate for all individuals, the BBO algorithm assigns two rates (emigration and immigration) to each habitat, which results in it providing different evolutionary behaviors and eventually greater exploration. As the results of Tables 3–7 and 9–14 show, the significant superiority of the BBO algorithm is increased as the complexity of the data sets increases. This is again due to diverse evolutionary mechanisms (greater exploration) of the BBO algorithm that assists this algorithm to outperform the GA algorithm with a unified mechanism for reproduction. The superior performance of the BBO algorithm compared to PSO and ACO is due to the migration mechanism of this algorithm. The PSO and ACO algorithms are not equipped with an evolutionary operator, so they do not provide sudden changes in the candidate solutions, as discussed above. The BBO algorithm, however, promotes diverse abrupt change mechanisms for candidate solutions, making this algorithm able to outperform PSO and ACO in almost all of the datasets.

The superiority of BBO compared to other evolutionary algorithms employed in this paper (EA and PBIL) can be reasoned from both the exploration and exploitation mechanisms. The selection process of the EA algorithm is deterministic, and the main characteristic is sophisticated mutation operators. These two facts promote exploitation. The BBO algorithm, however, promotes exploration and exploitation simultaneously. On one hand, the various migration rates assist BBO with greater exploration. On the other hand, the different mutation rates for each habitat allow BBO to use different exploitation behaviors for every habitat. These are the main reasons for the results shown in Fig. 22 which shows BBO can outperform EA not only in terms of local minima avoidance but also convergence rate. Comparing the results of BBO and PBIL, we can come to the same conclusion. As discussed above, the PBIL evolves a vector consisting of the entire population rather than each individual itself. This vector is updated based on the fittest individual and mutated randomly. This mechanism appears to provide less exploration than BBO, where the production of new habitats by emigration and immigration operators enhances exploration. This is the main reason of BBO's significantly better performance.

To sum up, it can be stated that exploration is very important in the problem of training MLPs. There is a need to have more random and abrupt search steps (emphasizing exploration) to avoid local minima for solving complex data sets using MLPs. This study shows that the operators of BBO (migration) are highly suitable for addressing this issue.

To further investigate the efficiency of the BBO algorithm in training FNNs, a comparative study between this algorithm and the BP algorithm on all the datasets is provided in the following section.

5.3. Comparative study with the BP algorithm

The BP algorithm is a gradient-based algorithm that uses gradient information for minimizing the error function. We chose this algorithm for comparison because it is totally based on mathematical concepts, quite different from the other, bio-inspired algorithms. This can benchmark the performance of the proposed approach on a contrasting test bed. We solved the datasets using the BP algorithm 10 times and provide the statistical results in Table 15, as well as those of BBO. Note that the learning rate, momentum, and maximum number of iterations are set to 0.01, 0.001, and 250 respectively. It can be seen that the results of the BBO algorithm for AVE and STD are better than BP in most of the datasets. This shows the superior performance of BBO in terms of improved avoidance of local minima. The classification rates and test errors also show that BBO is better than BP in most of the datasets. However, the BP algorithm provided very competitive results for simple datasets. This shows that the performance of BP is degraded as the complexity of the search space is increased (due to the problem of entrapment in local minima).

The reasons for the better performance of BBO compared to BP and other heuristic algorithms are as follows:

Table 15
Comparison results between BBO and BP.

Dataset	Algorithm	MSE (AVE ± STD)	*p*-Values	Classification rate/test error
XOR	BBO	**3.65e−07 ± 0.000000**	6.3864e−005	**100.00%**
	BP	1.17e−03 ± 4.12e−04		**100.00%**
Balloon	BBO	**8.09e−27 ± 1.51e−42**	6.3864e−005	**100.00%**
	BP	3.50e−06 ± 1.44e−06		**100.00%**
Iris	BBO	0.019150 ± 3.66e−18	0.0014	**90.00%**
	BP	**0.016960 ± 0.00170**		78.66%
Breast cancer	BBO	**0.002807 ± 0.000000**	6.3864e−005	**95.00%**
	BP	0.072640 ± 0.005612		94.00%
Heart	BBO	0.065625 ± 1.46e−17	0.0049	**76.25%**
	BP	**0.053650 ± 8.78e−03**		72.50%
Sigmoid	BBO	**1.33e−05 ± 3.57e−21**	6.3864e−005	**0.14381**
	BP	3.70e−04 ± 1.26e−04		1.3894
Cosine	BBO	0.013674 ± 1.83e−18	1.8267e−004	**1.4904**
	BP	**0.007932 ± 0.003799**		2.5663
Sine	BBO	0.102710 ± 0.000000	6.3864e−005	**64.261**
	BP	**0.020270 ± 0.00745**		23.3423
Sphere	BBO	**1.774000 ± 2.34e−16**	1.8267e−004	**770.4425**
	BP	1.544630 ± 0.396995		1.3618e+03
Griewank	BBO	**2.025600 ± 0.0000**	6.3864e−005	**5.0200e+03**
	BP	3.325595 ± 6.1857		9.8313e+003

- Varying values of emigration and immigration rates provide diverse information exchange behavior between habitats, and consequently improve exploration.
- Over the course of generations, the HSI of all habitats are improved since habitants living in high-HSI habitats tend to migrate to the low-HSI habitats. This guarantees the convergence of BBO.
- Migration operators emphasize exploration and consequently prevent BBO from easily getting trapped in local minima.
- Different mutation rates keep habitats as diverse as possible.
- Elitism assists BBO to save and retrieve some of the best solutions, so they are never lost.

In the following section the BBO algorithm is compared to a very effective method called ELM.

5.4. Comparative study with Extreme Learning Machines (ELM)

ELM was first proposed by Huang et al. in 2006 [24]. ELM can be considered as an MLP with one hidden layer without any need to tune the hidden layer [70]. In ELM the hidden nodes are independent of each other and training samples [71]. It has been demonstrated that this technique is very efficient for classification problems [25]. We again chose this method with the purpose of benchmarking the proposed technique. We obtained the results from Huang et al. [24] and Silva et al. [59] in order to compare them with the BBO algorithm. The available results for three datasets that match our experiment are: Iris, Heart, and Cancer. Note that training and testing sample sizes of these three datasets are quite similar to those of this work.

As the results of these studies show, ELM is able to classify the Iris dataset with an accuracy of 95.60%. The BBO algorithm reached 90% accuracy for the Iris dataset. It should be noted that BBO achieved 90% accuracy with 9 hidden nodes, whereas ELM shows 95.60% accuracy with 15 hidden nodes. ELM also classified the Cancer and Heart datasets with accuracies of 96.09% and 80.48% respectively. Comparing these results with those of BBO (Tables 6 and 7) shows BBO is capable of providing very competitive results compared to ELM. For ELM, the average MSE for the Cancer dataset is 0.06101 with standard deviation equal to 0.00014, which is much worse than that of BBO as shown in Table 6. This shows that BBO can show better local minima avoidance.

According to this comparative study and discussion, it appears that BBO is highly suitable for training MLPs. This algorithm successfully reduces the problem of entrapment in local minima in training MLPs, with very fast convergence rates. These improvements are accompanied by high classification rates and low test errors as well.

6. Conclusion

In this paper, an MLP was trained by the recently proposed BBO algorithm. Eleven benchmark datasets (balloon, iris, breast cancer, heart, sigmoid, cosine with one peak, sine with four peaks, sphere, Griewank, and Rosenbrock) were employed to investigate the effectiveness of BBO in training MLPs. The results were statistically compared with PSO, GA, ACO, ES, PBIL, BP, and ELM to verify the performance. The results demonstrate that BBO is significantly better able to avoid local minima

compared to PSO, GA, ACO, ES, and PBIL. Moreover, the superior performance of BBO for training MLPs in terms of accuracy of results and convergence rate can clearly be seen from the results.

In future studies, it would be interesting to investigate the efficiency of BBO in training other types of NNs such as recurrent, Kohonen, or Radial basis function (RBF) networks. In addition, employing BBO to define the structure of MLPs is worth investigating.

Acknowledgments

The authors would like to thank Professor Dan Simon for his comments and recommendations throughout this project. The first author also gratefully thanks Hossein Agha Moradian Sardroudi for his support during this project.

References

[1] V. Abedifar, M. Eshghi, S. Mirjalili, S.M. Mirjalili, An optimized virtual network mapping using PSO in cloud computing, in: Electrical Engineering (ICEE), 2013 21st Iranian Conference on, 14-16 May, 2013, pp.1–6, http://dx.doi.org/10.1109/IranianCEE.2013.6599723.
[2] H. Adeli, S. Hung, An adaptive conjugate gradient learning algorithm for efficient training of neural networks, Appl. Math. Comput. 62 (1994) 81–102.
[3] P. Auer, H. Burgsteiner, W. Maass, A learning rule for very simple universal approximators consisting of a single layer of perceptrons, Neural Networks 21 (2008) 786–795.
[4] M. Barakat, D. Lefebvre, M. Khalil, F. Druaux, O. Mustapha, Parameter selection algorithm with self adaptive growing neural network classifier for diagnosis issues, Int. J. Mach. Learn. Cybern. 4 (3) (2013) 217–233.
[5] C. Blake, C.J. Merz, 1998. UCI Repository of Machine Learning Databases. <http://www.ics.uci.edu/~mlearn/MLRepository.html> (accessed).
[6] C. Blum, K. Socha, Training feed-forward neural networks with ant colony optimization: an application to pattern classification, in: IEEE, 2005, 6 pp.
[7] I. Boussaid, J. Lepagnot, P. Siarry, A survey on optimization metaheuristics, Inform. Sci. 237 (2013) 82–117. ISSN 0020-0255, http://dx.doi.org/10.1016/j.ins.2013.02.041 (10.07.13).
[8] J. Branke, Evolutionary Algorithms for Neural Network Design and Training, Citeseer, 1995.
[9] C. Charalambous, Conjugate gradient algorithm for efficient training of artificial neural networks, in: IET, 1992, pp. 301–310.
[10] B.C. Csáji, Approximation with Artificial Neural Networks, Faculty of Sciences, Etvs Lornd University, Hungary, 2001.
[11] D. Dasgupta, Z. Michalewicz, Evolutionary Algorithms in Engineering Applications, Springer, 2001.
[12] J. Derrac, S. García, D. Molina, F. Herrera, A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms, Swarm Evol. Comput. (2011).
[13] S.E. Fahlman, An Empirical Study of Learning Speed in Back-propagation Networks, Technical report, 1988, <http://repository.cmu.edu/cgi/viewcontent.cgi?article=2799&context=compsci>.
[14] T.L. Fine, Feedforward Neural Network Methodology, Springer Verlag, 1999.
[15] S. García, D. Molina, M. Lozano, F. Herrera, A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the CEC'2005 special session on real parameter optimization, J. Heuristics 15 (2009) 617–644.
[16] M.W. Gardner, S.R. Dorling, Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences, Atmos. Environ. 32 (1998) 2627–2636.
[17] M. Gori, A. Tesi, On the problem of local minima in backpropagation, IEEE Trans. Pattern Anal. Mach. Intell. 14 (1992) 76–86.
[18] V.G. Gudise, G.K. Venayagamoorthy, Comparison of particle swarm optimization and backpropagation as training algorithms for neural networks, in: IEEE, 2003, pp. 110–117.
[19] Z.X. Guo, W.K. Wong, M. Li, Sparsely connected neural network-based time series forecasting, Inform. Sci. 193 (2012) 54–71. ISSN 0020-0255, http://dx.doi.org/10.1016/j.ins.2012.01.011 (15.06.12).
[20] W. Guo, L. Wang, Q. Wu, An analysis of the migration rates for biogeography-based optimization, Inform. Sci. 254 (2014) 111–140. ISSN 0020-0255, http://dx.doi.org/10.1016/j.ins.2013.07.018 (01.01.14).
[21] M.T. Hagan, M.B. Menhaj, Training feedforward networks with the Marquardt algorithm, IEEE Trans. Neural Networks 5 (1994) 989–993.
[22] Y.C. Ho, D.L. Pepyne, Simple explanation of the no-free-lunch theorem and its implications, J. Optim. Theory Appl. 115 (2002) 549–570.
[23] K. Hornik, M. Stinchcombe, H. White, Multilayer feedforward networks are universal approximators, Neural Networks 2 (1989) 359–366.
[24] G.-B. Huang, Q.-Y. Zhu, C.-K. Siew, Extreme learning machine: theory and applications, Neurocomputing 70 (2006) 489–501.
[25] G.-B. Huang, D.H. Wang, Y. Lan, Extreme learning machines: a survey, Int. J. Mach. Learn. Cybern. 2 (2) (2011) 107–122.
[26] D.R. Hush, B.G. Horne, Progress in supervised neural networks, IEEE Signal Process. Mag. 10 (1993) 8–39.
[27] A.S. Ismail Wdaa, Differential Evolution for Neural Network Learning Enhancement, Master, Universiti Teknologi Malaysia (UTM), 2005.
[28] R.A. Jacobs, Increased rates of convergence through learning rate adaptation, Neural Networks 1 (1988) 295–307.
[29] Y. Lee, S.H. Oh, M.W. Kim, An analysis of premature saturation in back propagation learning, Neural Networks 6 (1993) 719–728.
[30] F.H.F. Leung, H. Lam, S. Ling, P.K.S. Tam, Tuning of the structure and parameters of a neural network using an improved genetic algorithm, IEEE Trans. Neural Networks 14 (2003) 79–88.
[31] H. Ma, D. Simon, M. Fei, Z. Xie, Variations of biogeography-based optimization and Markov analysis, Inform. Sci. 220 (2013) 492–506. ISSN 0020-0255, http://dx.doi.org/10.1016/j.ins.2012.07.007 (20.01.13).
[32] G. Magoulas, M. Vrahatis, G. Androulakis, On the alleviation of the problem of local minima in back-propagation, Nonlinear Anal. 30 (1997) 4545–4550.
[33] O.L. Mangasarian, W.H. Wolberg, Cancer Diagnosis via Linear Programming, University of Wisconsin-Madison, Computer Sciences Department, 1990.
[34] W.S. McCulloch, W. Pitts, A logical calculus of the ideas immanent in nervous activity, Bull. Math. Biol. 5 (1943) 115–133.
[35] P. Patricia Melin, D. Sanchez, O. Castillo, Genetic optimization of modular neural networks with fuzzy response integration for human recognition, Inform. Sci. 197 (2012) 1–19. ISSN 0020-0255, http://dx.doi.org/10.1016/j.ins.2012.02.027 (15.08.12).
[36] R. Mendes, P. Cortez, M. Rocha, J. Neves, Particle swarms for feedforward neural network training, in: IJCNN'02. Proceedings of the 2002 International Joint Conference on Neural Networks, 2002, IEEE, pp. 1895–1899.
[37] S. Mirjalili, Hybrid Particle Swarm Optimization and Gravitational Search Algorithm for Multilayer Perceptron Learning. Master, Universiti Teknologi Malaysia (UTM), 2011.
[38] S. Mirjalili, A. Lewis, S-shaped versus V-shaped transfer functions for binary particle swarm optimization, Swarm Evol. Comput. 9 (2013) 1–14 (ISSN 2210-6502, http://dx.doi.org/10.1016/j.swevo.2012.09.002 (April 2013)).
[39] S. Mirjalili, S.Z.M. Hashim, A new hybrid PSOGSA algorithm for function optimization, in: 2010 International Conference on Computer and Information Application (ICCIA), 3–5 December 2010, pp. 374, 377.
[40] S. Mirjalili, S.M. Mirjalili, A. Lewis, Grey wolf optimizer, Adv. Eng. Softw. 69 (2014) 46–61, http://dx.doi.org/10.1016/j.advengsoft.2013.12.007.
[41] S. Mirjalili, S.M. Mirjalili, X.-S. Yang, Binary bat algorithm, Neural Comput. Appl. (2013), in press, http://dx.doi.org/10.1007/s00521-013-1525-5.
[42] S. Mirjalili, S.Z. Mohd Hashim, H. Moradian Sardroudi, Training feedforward neural networks using hybrid particle swarm optimization and gravitational search algorithm, Appl. Math. Comput. 218 (22) (2012) 11125–11137.
[43] S.M. Mirjalili, K. Abedi, S. Mirjalili, Light property and optical buffer performance enhancement using particle swarm optimization in oblique Ring-Shape-Hole photonic crystal waveguide, in: Photonics Global Conference (PGC), 2012, pp. 1–4.

[44] S.M. Mirjalili, K. Abedi, S. Mirjalili, Optical buffer performance enhancement using particle swarm optimization in ring-shape-hole photonic crystal waveguide, Optik – Int. J. Light Electron Opt. 124 (23) (2013) 5989–5993. ISSN 0030-4026, http://dx.doi.org/10.1016/j.ijleo.2013.04.114.
[45] S.M. Mirjalili, S. Mirjalili, Oval-Shaped-Hole photonic crystal waveguide design by MoMIR framework, Photonics Technol. Lett. IEEE (2014), in press, http://dx.doi.org/10.1109/LPT.2014.2302478.
[46] S.M. Mirjalili, S. Mirjalili, A. Lewis, A novel multi-objective optimization framework for designing photonic crystal waveguides, Photonics Technol. Lett. IEEE 26 (2014) 146–149, http://dx.doi.org/10.1109/LPT.2013.2290318.
[47] S.M. Mirjalili, S. Mirjalili, A. Lewis, K. Abedi, A tri-objective particle swarm optimizer for designing line defect photonic crystal waveguides, Photon Nanostruct. Fundam. Appl. (2013), http://dx.doi.org/10.1016/j.photonics.2013.11.001.
[48] S. Mizuta, T. Sato, D. Lao, M. Ikeda, T. Shimizu, Structure design of neural networks using genetic algorithms, Complex Syst. 13 (2001) 161–176.
[49] S. Ng, C. Cheung, S. Leung, A. Luk, Fast convergence for backpropagation network with magnified gradient function, in: IEEE, vol. 3, 2003, pp. 1903–1908.
[50] E. Oja, Unsupervised learning in neural computation, Theoret. Comput. Sci. 287 (2002) 187–207.
[51] M. Ovreiu, D. Simon, Biogeography-based optimization of neuro-fuzzy system parameters for diagnosis of cardiac disease, in: Genetic and Evolutionary Computation Conference, 2010, pp. 1235–1242.
[52] I.C. Parmee, Evolutionary and Adaptive Computing in Engineering Design, Springer Verlag, 2001.
[53] R.D. Reed, R.J. Marks, Neural Smithing: Supervised Learning in Feedforward Artificial Neural Networks, MIT Press, 1998.
[54] F. Rosenblatt, The Perceptron, A Perceiving and Recognizing Automaton Project Para, Cornell Aeronautical Laboratory, 1957.
[55] S. Saremi, S.M. Mirjalili, S. Mirjalili, Unit cell topology optimization of line defect photonic crystal waveguide, Procedia Technol. 12 (2014) 174–179. ISSN 2212-0173, http://dx.doi.org/10.1016/j.protcy.2013.12.472.
[56] S. Saremi, S.M. Mirjalili, S. Mirjalili, Chaotic krill herd optimization algorithm, Procedia Technol. 12 (2014) 180–185. ISSN 2212-0173, http://dx.doi.org/10.1016/j.protcy.2013.12.473.
[57] U. Seiffert, Multiple Layer Perceptron Training using Genetic Algorithms, Citeseer, 2001, pp. 159–164.
[58] T.J. Sejnowski, Unsupervised Learning: Foundations of Neural Computation, The MIT Press, 1999.
[59] D. Silva, L. Pacifico, T. Ludermir, An evolutionary extreme learning machine based on group search optimization, in: 2011 IEEE Congress on Evolutionary Computation (CEC), 2011, pp. 574–580.
[60] D. Simon, Biogeography-based optimization, IEEE Trans. Evol. Comput. 12 (2008) 702–713.
[61] D. Simon, R. Rarick, M. Ergezer, D. Du, Analytical and numerical comparisons of biogeography-based optimization and genetic algorithms, Inform. Sci. 181 (7) (2011) 1224–1248. ISSN 0020-0255, http://dx.doi.org/10.1016/j.ins.2010.12.006 (01.04.11).
[62] UCI Machine Learning Repository. <http://archive.ics.uci.edu/ml/datasets.html> (accessed 2012).
[63] A. van Ooyen, B. Nienhuis, Improving the convergence of the back-propagation algorithm, Neural Networks 5 (1992) 465–471.
[64] T.P. Vogl, J. Mangis, A. Rigler, W. Zink, D. Alkon, Accelerating the convergence of the back-propagation method, Biol. Cybern. 59 (1988) 257–263.
[65] G. Wang, L. Guo, H. Wang, H. Duan, L. Liu, J. Li, Incorporating mutation scheme into krill herd algorithm for global numerical optimization, Neural Comput. Appl. (2012), http://dx.doi.org/10.1007/s00521-012-1304-8.
[66] G.-G. Wang, A.H. Gandomi, A.H. Alavi, A chaotic particle-swarm krill herd algorithm for global numerical optimization, Kybernetes 42 (6) (2013) 962–978.
[67] G.-G. Wang, A.H. Gandomi, A.H. Alavi, An effective krill herd algorithm with migration operator in biogeography-based optimization, Appl. Math. Model. (2013), in press, http://dx.doi.org/10.1016/j.apm.2013.10.052.
[68] G.-G. Wang, A.H. Gandomi, A.H. Alavi, Stud krill herd algorithm, Neurocomputing 128 (2014) 363–370, http://dx.doi.org/10.1016/j.neucom.2013.08.031.
[69] G.-G. Wang, A.H. Gandomi, A.H. Alavi, G.-S. Hao, Hybrid krill herd algorithm with differential evolution for global numerical optimization, Neural Comput. Appl. (2013), http://dx.doi.org/10.1007/s00521-013-1485-9.
[70] X.Z. Wang, A. Chen, H. Feng, Upper integral network with extreme learning mechanism, Neurocomputing 74 (16) (2011) 2520–2525.
[71] X.Z. Wang, Q.Y. Shao, M. Qing, J.H. Zhai, Architecture selection for networks trained with extreme learning machine using localized generalization error model, Neurocomputing 102 (2013) 3–9.
[72] M.K. Weir, A method for self-determination of adaptive learning rates in back propagation, Neural Networks 4 (1991) 371–379.
[73] P. Werbos, Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences, Ph.D. thesis, Harvard University, 1974.
[74] P.J. Werbos, Neurocontrol and supervised learning: An overview and evaluation, Handbook Intell. Control 65 (1992) 89.
[75] W. Wienholt, Minimizing the system error in feedforward neural networks with evolution strategy, in: S. Gielen, B. Kappen (Eds.), ICANN '93, Springer, London, 1993, pp. 490–493.
[76] F. Wilcoxon, Individual comparisons by ranking methods, Biometrics Bull. 1 (1945) 80–83.
[77] D.H. Wolpert, W.G. Acready, No free lunch theorems for optimization, IEEE Trans. Evol. Comput. 1 (1997) 67–82.
[78] X. Yao, Evolutionary artificial neural networks, Int. J. Neural Syst. 4 (1993) 203–222.
[79] J. Yu, S. Wang, L. Xi, Evolving artificial neural networks using an improved PSO and DPSO, Neurocomputing 71 (2008) 1054–1060.
[80] J.R. Zhang, J. Zhang, T.M. Lok, M.R. Lyu, A hybrid particle swarm optimization – back-propagation algorithm for feedforward neural network training, Appl. Math. Comput. 185 (2007) 1026–1037.
[81] N. Zhang, An online gradient method with momentum for two-layer feedforward neural networks, Appl. Math. Comput. 212 (2009) 488–498.