


Training radial basis function networks using biogeography-based optimizer

Ibrahim Aljarah¹  · Hossam Faris¹ · Seyedali Mirjalili² · Nailah Al-Madi³

Received: 20 April 2016 / Accepted: 12 August 2016
© The Natural Computing Applications Forum 2016

Abstract Training artificial neural networks is considered as one of the most challenging machine learning problems. This is mainly due to the presence of a large number of solutions and changes in the search space for different datasets. Conventional training techniques mostly suffer from local optima stagnation and degraded convergence, which make them impractical for datasets with many features. The literature shows that stochastic population-based optimization techniques suit this problem better and are reliably alternative because of high local optima avoidance and flexibility. For the first time, this work proposes a new learning mechanism for radial basis function networks based on biogeography-based optimizer as one of the most well-regarded optimizers in the literature. To prove the efficacy of the proposed methodology, it is employed to solve 12 well-known datasets and compared to 11 current training algorithms including gradient-based and stochastic approaches. The paper considers changing the number of

neurons and investigating the performance of algorithms on radial basis function networks with different number of parameters as well. A statistical test is also conducted to judge about the significance of the results. The results show that the biogeography-based optimizer trainer is able to substantially outperform the current training algorithms on all datasets in terms of classification accuracy, speed of convergence, and entrapment in local optima. In addition, the comparison of trainers on radial basis function networks with different neurons size reveal that the biogeography-based optimizer trainer is able to train radial basis function networks with different number of structural parameters effectively.

Keywords Optimization · Biogeography-based optimizer · BBO · Radial basis function network · Training neural network · Evolutionary algorithm

✉ 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

¹ Business Information Technology Department, 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 The King Hussein Faculty of Computing Sciences, Princess Sumaya University for Technology, Amman, Jordan

1 Introduction

Radial basis function (RBF) networks are one of the most popular and applied type of neural networks. RBF networks are universal approximators and considered as special form of multilayer feedforward neural networks that contain only one hidden layer with Gaussian based activation functions. RBF networks were first introduced by Broomhead et al. [8] with a strong foundation in the conventional approximation theory [13, 16].

The advantages of RBF networks compared to other neural networks include the high generalization capability, its simple and compact structure (i.e; only three layers), easier parameters adjustment, very good noise tolerance and the high learning speed [26, 60]. Due to these advantages, RBF networks have been a common alternative to

MLP networks [16]. Moreover, RBF networks have been successfully applied to many applications like: systems identification [2, 36], process faults classification [31], nonlinear control [30] and time series forecasting [11, 18].

Like other neural networks, RBF networks have two major components: the structure and the training method. The training method has a significance influence on the performance of the network. In literature, researchers proposed and investigated a wide variety of learning schemes for RBF network.

Castao et al. [9] classified the training methods of RBF networks into two categories: quick learning and full learning. The quick learning methods are more popular where the learning process can be performed in two independent stages: In the first stage, the structure of the network (i.e; the centers and widths of the network) is identified usually by an unsupervised learning algorithm like K-Means algorithm, while in the second stage, the connection weights between the hidden and output layers are tuned using least mean squares (LMS), gradient-based methods and variations of the backpropagation algorithm [42]. The drawback of using an unsupervised technique to locate the centers is that it depends only on the input features and it does not consider the distribution of the label class [50]. Moreover, using the common K-Means algorithm does not necessarily guarantee that the centers are best located [33]. On the other side, the main issue with the gradient methods is that its highly probable that the search process will be trapped in a local minima. Moreover, Vakil-Baghmisheh and Pave reported in [48] that applying customized version of the backpropagation algorithm to RBF networks could suffer from some drawbacks like the slow convergence and over-training which consequently affects the generalization ability of the model. Alternatively, the full learning methods optimize the RBF parameters simultaneously as a supervised task.

Nature-inspired Metaheuristic algorithms have been widely investigated in evolving and training RBF networks. These algorithms are based on stochastic search algorithm that are simulated by natural systems and phenomena. Most of Nature-inspired metaheuristics are population based and rely on randomness as an essential principle of their process. The advantage of these search methods is their flexibility, self-adaptation, conceptual simplicity and ability of searching for a global optima rather than a local one [17]. Nature-inspired metaheuristic were deployed in different ways in training RBF networks. Some were applied for finding one parameter of the network like the centers [28], others optimized all the parameters [43], while others investigated optimizing all the parameters along with the structure of the network. Such algorithms applied to RBF networks training include: Genetic Algorithms (GA) [6, 15, 21], particle swarm

optimization (PSO) [25, 37, 40, 47], Ant Colony Optimization (ACO) [12, 45], Differential Evolution (DE) [4, 14, 58], Firefly algorithm (FFA) [24], Cuckoo search (CS) [3, 10], Honey Bee Mating Optimization (HBMO) [23], Artificial Bee Colony [29] and BAT Algorithm [46].

According to the No Free Lunch theorem (NFL), there is no heuristic algorithm that certainly performs better than all other algorithms in all optimization problems [7, 22, 49]. Motivated by this reason, in this work, we propose a novel RBF training algorithm based on the recent biogeography-based optimizer (BBO) for optimizing the parameters (centers, widths and weights) of RBF network, simultaneously. BBO is an evolutionary algorithm, which was developed by Simon [44]. BBO was inspired by the studies related to the geographical distribution of biological organisms in terms of time and space. Recently, BBO optimizer has been applied in training neuro-fuzzy networks [38] and feedforward MLPs [35] and showed high modeling capability. However, according to our knowledge, there is no previous work investigating the efficiency of the well-regarded BBO algorithm in any type of RBF network training.

In order to evaluate the efficiency and effectiveness of the new BBO trainer, the proposed trainer is applied on twelve popular benchmark datasets, which are selected from the UCI machine learning repository.¹ The results of the proposed BBO trainer are compared with those obtained with other eleven algorithms. Six algorithms out of the eleven are classical evolutionary algorithms, which are the GA, PSO, DE, Evolutionary Strategy (ES), ACO and the population-based incremental learning (PBIL). While four algorithms are recent nature-inspired algorithms which are the FFA, CS, ABC and BAT Algorithm. The eleventh algorithm is actually a hybrid two stages training algorithm based on K-Means and gradient decent optimization.

This paper is organized as follows: in Sect. 2 a description of the RBF network, and its classical two-phases learning approach is given. In Sect. 3 the BBO algorithm is explained. Section 4 describes in detail the developed BBO-based approach for training RBF network. Experimental results are outlined in Sect. 5. Finally, the finding and remarks of this work, and future works are concluded in Sect. 6.

2 Radial based function neural networks

RBF neural network is a special type of fully connected feedforward networks that consists of only three layers: input, hidden and output layers. The number of neurons in

¹ <http://archive.ics.uci.edu/ml/>.

the input layer depends on the number of dimensions of the input vector, whereas output layer neurons depend on the number of class labels in the data. The number of neurons in the hidden layer determines the topology of the network which also determines the decision boundary between data clusters. Each hidden neuron has an RBF activation function that calculates the similarity between the input and a stored prototype in that neuron. Having more prototypes results in a more complex decision boundary, which means higher accuracy. However, it results of more computations to evaluate the network.

Figure 1 shows the structure of RBF network in comparison with the Multilayer Perceptron network. Inspecting this figure, it may be seen that the arrows between the input layer and the hidden neurons in the RBF network represent the Euclidean distance between the input vector and the prototypes stored in the hidden neurons. On the other side, in MLP, arrows between the input layer and output layer represent weights. Moreover, in RBF networks, activation functions in the hidden nodes are Gaussian basis functions, while in MLP the sigmoidal functions are typically used.

RBF ANN process works as follows, first the input data enter the network through the input layer. After that, each neuron in the RBF layer (hidden layer) calculates the similarity between the input data and the prototype stored inside it, using the nonlinear Gaussian function shown in Eq. 1.

$$\phi(\|x - c_j\|) = \exp - \left(\frac{\|x - c_j\|^2}{2\sigma_j^2} \right) \tag{1}$$

where $\|x - c_j\|$ is Euclidean norm.

The output of the RBF is calculated using weighted average method by the following equation:

$$y_i = \sum_{j=1}^n \omega_{ji} \phi_j(x) \tag{2}$$

where ω_{ji} represents the i th weight between the hidden layer and output layer, and n represents the number of hidden nodes.

The output of the RBF neuron is closer to 1 whenever the similarity between the input and the prototype is high, and close to zero otherwise. The output layer neurons takes the weighted sum of every RBF neuron output in order to decide the class label. Which means that every RBF neuron contributes in the labeling decision, higher similarity has larger contribution.

2.1 Classical radial basis function network training

Classical RBF process depends mainly on three points, the prototypes inside each RBF neurons and how to be chosen perfectly, the beta value in the similarity equation, and the weights between hidden layer and output layer (which affects the last decision). Choosing the prototypes can be done using many approaches such as choosing random data points from the data, or using K-Means clustering approach and use the clusters centers as prototypes or any other approach you may choose. Using K-Means clustering is the most used approach in the literature as it helps in smartly choosing small number of RBF neurons (K neurons), where each neuron represents a cluster in the data. Moreover, having only K neurons does not affect the complexity of the RBF network nor the accuracy of the classification decision.

Beta coefficient in the RBF activation function controls the width of the bell curve and should be determined in a manner that optimizes the fit between the activation function and the data. When K-Means is used to choose the

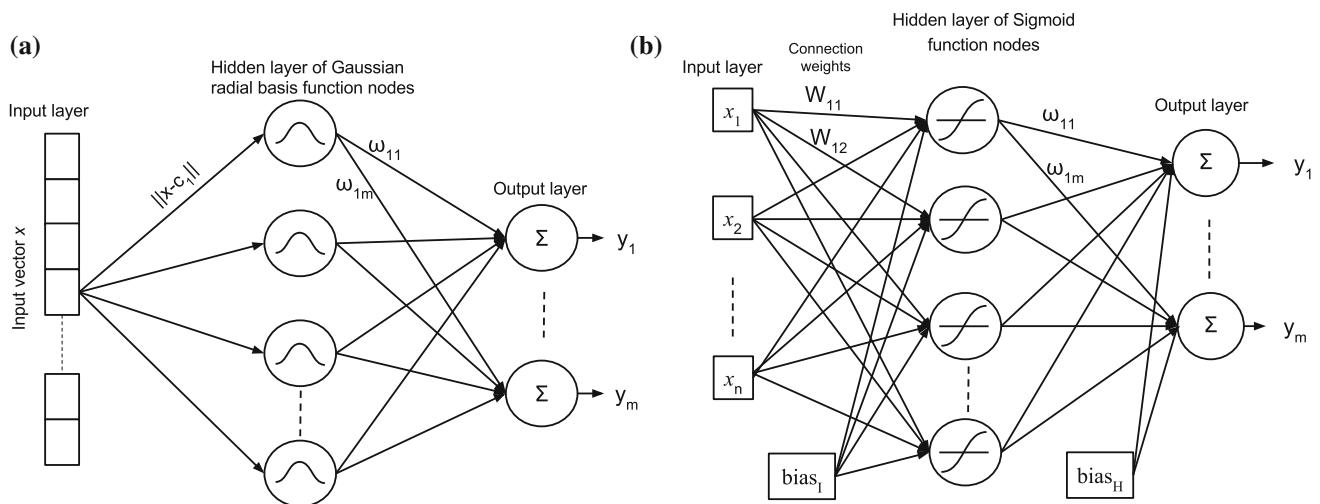


Fig. 1 Representation of ANN networks. a RBF Networks. b MLP Networks

RBF neurons prototypes, then Beta can be set using the following equation:

$$\text{Beta} = \frac{1}{2 \times \sigma^2} \quad (3)$$

where σ equals to the average distance between all points in the cluster and the cluster center.

Training the output weights is the third important parameter to set for RBF to work perfectly. Training these weights can be done using the Gradient decent which is an optimization technique that takes the outputs of the RBF neurons as input and optimize the weights according to them. Gradient descent must be run separately for each output node. The following subsections describe more details about K-Means and gradient decent methods that are selected in this work as a classical approach for optimizing the connection weights.

2.1.1 K-Means

K-Means is considered one of the most efficient clustering algorithms that used in many applications in the literature. K-Means clustering has many advantages, such as simplicity to implement and has good performance with large dataset. K-Means is a partitioning clustering algorithm, where the objective is to maximize the similarity between the members in each cluster and minimize the similarity between the members in different clusters. The main idea of the K-Means clustering is to define k centers, one center for each cluster. The data points are assigned to the proper cluster based on the minimum distances to all cluster centers. After that, cluster centers should be modified in an iterative way by calculating the mean of cluster's members to achieve the best clustering quality (The squared error function). This process is continued until centers do not change any more.

2.1.2 Gradient decent

Gradient decent (GD) is considered an optimization algorithm uses the first-order derivative calculation to find a local minimum of a function. The algorithm applies consecutive steps to find the gradient of the objective function at the current point. The output of a RBF network can be represented as shown in Eqs. 4, and 5, while the error function E is given in Eq. 7, where $\hat{y}_{i,k}$ is the response value of the i th output unit, and $y_{i,k}$ is the actual response. The Gradient decent algorithms can be used to find the solution matrix W as shown in Eq. 7, where η is a small decreasing value called the learning rate.

$$\hat{y} = (y_1, y_2, \dots, y_m) = \begin{bmatrix} \omega_{11} & \omega_{11} & \dots & \omega_{1m} \\ \omega_{21} & \omega_{21} & \dots & \omega_{2m} \\ \dots & \dots & \dots & \dots \\ \omega_{l1} & \omega_{l1} & \dots & \omega_{lm} \end{bmatrix} \begin{bmatrix} \phi_1(x) \\ \phi_2(x) \\ \dots \\ \phi_m(x) \end{bmatrix} \quad (4)$$

$$O = W \cdot H \quad (5)$$

$$E = \frac{1}{2} \sum_{k=1}^M \sum_{i=1}^L (y_{i,k} - \hat{y}_{i,k})^2 \quad (6)$$

$$\omega_{ij} = \omega_{ij} - \eta \frac{\partial E}{\partial \omega} \quad (7)$$

In this work, the conjugate gradient (CG) is used to optimize the weights in the standard RBF network. CG is a special type of gradient descent with regularization that used to compute search directions. The CG uses a line search with quadratic, and cubic polynomial approximations. The stopping criteria that used in CG is the Wolfe–Powell, and CG guesses the initial step sizes using slope ratio method.

3 Biogeography-based optimization optimizer

Evolutionary Algorithms (EAs) belong to the class of stochastic population-based algorithms. As the name implies, such techniques approximate the global optima for optimization problems using stochastic operators. The optimization process first starts with a set of random solutions as candidate solutions for a given problem. This set is then evolved using different mechanism defined by the algorithm to find a better approximation of the global optimum. This framework is common between all EAs despite different mechanisms to evolve the solutions.

One of the most recent and well-regarded EAs proposed in the literature is the biogeography-based optimization (BBO) algorithm [44]. This algorithm mimics evolutionary phenomena in the field of biogeography to solve optimization problem. The main inspiration of BBO is based on the fact that nature balances between the prey and predator using migration in the same habitat and different habitats.

In BBO, each solution represents a habitat and each variable in the solution indicates a habitant (prey or predator). The objective function is called Habitat Suitability Index (HSI), which obviously shows how suitable a habitat is. The following rules should be considered to simulate the evolvement of habitats and habitants in nature:

1. Habitants in any habitat face mutation regardless of their HSI.
2. Habitants in a habitat with better HSI are more likely to migrate to habitats with worse HSI.
3. Habitants in a habitat with worse HSI are more likely not to migrate.
4. Immigration is always from better habitat to worse habitat.
5. Each habitat has a rate of immigration and emigration, which define the rate of immigration to or from other habitats.

The immigration between habitats are simulated by exchanging the variables of solutions. In BBO algorithm, each habitat has different emigration and immigration rates to simulate habitats with different characteristics in nature. Obviously, with constant migration rates between habitats, the BBO is not able to balance exploration and exploitation. Therefore, this algorithm has been equipped with the following adaptive immigration and emigration rates:

$$\mu_k = \frac{E \times n}{N} \tag{8}$$

$$\lambda_k = I \frac{1 - n}{N} \tag{9}$$

where n is the habitants number, N is the maximum habitants allowed, E is the maximum allowed emigration rate, and I is the maximum immigration rate. The mutation rate has also been required to change adaptively as follows:

$$m_n = M \left(1 - \frac{P_n}{P_{\max}} \right) \tag{10}$$

where M is the initial value, P_n is the mutation probability, and P_{\max} shows the maximum probability.

The significant number of works in the literature proves that the BBO algorithm is able to solve optimization problems. This is due to the high exploration of this algorithm, which originate from the migration mechanism between the habitats. The migration mechanisms abruptly changes the solution, which assist the BBO to avoid local solutions and determine an accurate approximation of the global optima for challenging problems effectively. This

motivated our attempts to propose a trainer based on BBO to train RBFN for the first time in the literature.

4 Biogeography-based optimization for training radial basis function networks

In contrast to the classical approach where RBF network are trained in two independent phases, our proposed BBO-based approach searches for all RBF network parameters simultaneously. The parameters are the centers, widths and connection weights including the bias terms. In the proposed training algorithm, each habitat is encoded to represent these parameters as shown in Fig. 2 where C_i is the center of the hidden neuron i , σ_i is the width of that neuron and ω_{ij} is weight connecting between neuron i and output unit j . Habitats are implemented as real vectors with a length D which can be calculated as follows: suppose that n is number of hidden neurons, I is the number of features in the dataset and m is the number of output units then D can be calculated as given in Eq. 11.

$$D = (n \times I) + n + (n \times m) + m \tag{11}$$

In order to evaluate the fitness value (HSI) of the habitats (candidate RBF networks), the mean squared error (MSE) is calculated over all training samples for each habitat. MSE can be given as in Eq. 12 where y is the actual output, \hat{y} is the estimated output and k is the total number of instances in the training dataset.

$$MSE = \frac{1}{k} \sum_{i=1}^k (y - \hat{y})^2 \tag{12}$$

Based on the encoding scheme and the fitness evaluation described above, the BBO algorithm is designed to train the RBF networks as described in the flowchart in Fig. 3. This figure shows that the BBO first creates a set of random candidate solutions, which includes RBF networks with random connection weights and biases. This algorithm then repeatedly calculates the MSE for all the RBF networks when classifying the training data. The MSE shows which “random” RBF is

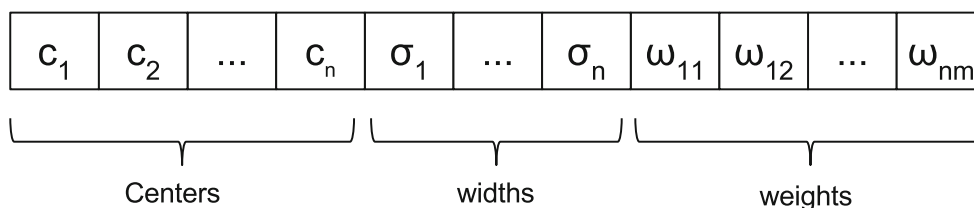


Fig. 2 Representation of BBO individuals structure

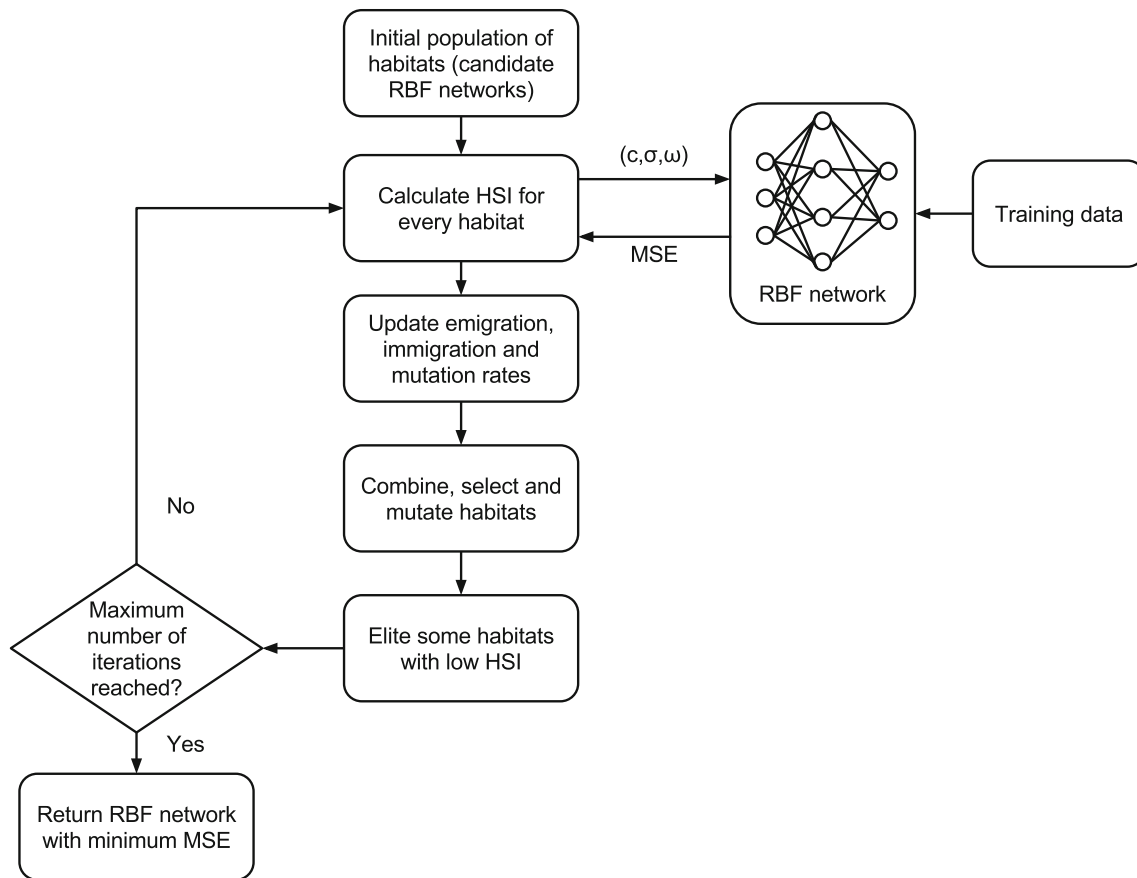


Fig. 3 RBF networks using BBO trainer flowchart

better. Based on the rules discussed above, the BBO algorithm creates a set of new RBF networks considering the best RBF networks found so far. The process of calculating MSEs and improving the RBFs continues until the satisfaction of the end criterion, which could be a threshold or maximum iterations. It should be noted that the average MSE is calculated when classifying all training samples in the dataset for each RBF network in the proposed BBO-based trainer. Therefore, the computational complexity is of $O(ntd)$ where n is the number of random RBF networks, t indicates the maximum iterations, and d shows the number of training samples in the dataset.

5 Experiments and results

In this section, the BBO training algorithm is evaluated on twelve datasets to verify the power of BBO for RBF neural network training. Furthermore, a comprehensive comparison of the BBO with other ten well-known metaheuristic

algorithms is conducted. The metaheuristic algorithms that are used in this experiment are: GA, PSO, ACO, ES, PBIL, DE, Firefly, Cuckoo search, ABC and BAT Algorithm which are the most common metaheuristic-based trainers for RBF network in the literature. In addition, the BBO trainer is compared with the RBFclassic (Gradient-based) technique, which is considered the common method for training the RBF neural network.

5.1 Experimental setup

The MATLAB R2010b is used to implement the proposed BBO trainer and other algorithms. All datasets are divided using 66, 34 % for training and testing, respectively. 10 different runs are executed for all experiments, and 250 iterations in each run. Moreover, the population size is fixed to 50 individuals for all algorithms. The parameter settings for each algorithm are shown in Table 1.

In CS, besides the population size, the discovery rate p_x is the only parameter needs to be tuned. p_x is set to 0.25 since it was stated in [55] that this value is sufficient for

Table 1 The metaheuristic algorithms with initial parameters

Algorithm	Parameter	Value
GA	Crossover probability	0.9
	Mutation probability	0.1
	Selection mechanism	Stochastic Universal Sampling
PSO	Acceleration constants	[2.1, 2.1]
	Inertia weights	[0.9, 0.6]
DE	Crossover probability	0.9
	Differential weight	0.5
BBO	Habitat modification probability	1.0
	Immigration probability	[0, 1]
	Step size	1.0
	Maximum immigration	1.0
	Migration rates	1.0
ACO	Mutation probability	0.05
	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_l)	0.5
	Pheromone sensitivity (α)	1
ES	Visibility sensitivity (β)	5
	λ	10
PBIL	σ	1
	Learning rate	0.05
	Good population member	1
	Bad population member	0
	Elitism parameter	1
FireFly	Mutational probability	0.1
	Alpha	0.2
	Beta	1
Cuckoo	Gamma	1
	Discovery rate P_x	0.25
ABC	Acceleration Coefficient Upper Bound	1
BAT	Loudness	0.5
	Pulse rate	0.5
	Frequency minimum	0
	Frequency maximum	1

most optimization problems. For Firefly, Beta is set to 1 as it was reported in [56] that parametric studies suggest setting the value of Beta to 1 can be used for most applications while gamma can be set to $1/\sqrt{L}$ where L is a scaling factor and if the scaling variations are not significant, then we can set gamma = $O(1)$. Alpha is roughly tuned and set to 0.2. Same values were used and applied in previous studies as in [52, 53].

For PSO, acceleration constants are typically set to ≈ 2 [1, 57]. We use also linear decreasing strategy to update the

Inertia in the interval [0.9,0.6]. It was found by experiments in the literature that this strategy improves the efficiency and performance of PSO achieving excellent results [5, 51].

For GA, the crossover probability is usually set to a much high rate, while the mutation probability is set to a much low probability [19]. With a rough tuning, the crossover and mutation probabilities are set to 0.9 and 0.1, respectively. For DE, the DE/rand/1/bin variant is applied with the crossover probability and differential weight equal to 0.9 and 0.5 as applied and recommended in [34, 59].

For ACO, ES and PBIL, all parameters are set as used and applied in [35, 44]. And for ABC and BAT, the defaults parameters are used [27, 54].

For BBO, we used the same parameters as in [35, 44] habitat modification probability is set to 1, immigration probability bounds per gene = [0, 1], step size is set to 1, maximum immigration and migration rates for each island is set to 1, while the mutation probability is set 0.05 as in [35].

However, it is worth mentioning that finding the best parameters of these algorithms is considered as another optimization problem by itself and it is known as meta-optimization. Therefore, fine tuning the optimization algorithms is out of scope if this work [39]. All dataset are normalized to the interval of [0, 1].

An RBF network with large number of neurons in the hidden layer may achieve good results based on the training data; however, this could lead to a bad generalization [41]. In our experiments, we assess the performance of the proposed training algorithm based on different number of neurons in the hidden layer: 4, 6, 8, 10 respectively.

In our experiments, we used five different measurements to evaluate the developed RBF network models. The measurements are Accuracy, Specificity, Sensitivity, Complexity and MSE. MSE was given previously in Eq. 12, while the rest are calculated using the following Eqs. 13, 14, 15 and 16, respectively. Accuracy, Specificity, Sensitivity and MSE assess the prediction accuracy, while the complexity equation reflects the network structure complexity based on the number of neurons.

$$\text{Accuracy} = \frac{\text{Number of correctly classified instances}}{\text{Total number of instances}} \quad (13)$$

$$\text{Specificity} = \frac{\text{Number of predicted instances of negative class}}{\text{Number of actual negative instances}} \quad (14)$$

$$\text{Sensitivity} = \frac{\text{Number of predicted instances of positive class}}{\text{Number of actual positive instances}} \quad (15)$$

$$\text{Complexity} = \frac{1}{2} \sum_{i=1}^{|w|} (w_i)^2 \quad (16)$$

where $|w| = 2 * (n + 1)$, n is the number of neurons.

5.2 Datasets description

The proposed BBO trainer is evaluated using twelve known real datasets, which are selected from UCI Repository [32]. All datasets contains two classes. Table 2 describes these datasets in terms of number of features, training samples, testing samples and the accuracy of the baseline classifier for each dataset. The baseline classifier is the Zero Rule classifier or ZeroR for short. ZeroR is the simplest classifier which relies only on the output class by simply predicting the majority class.

5.3 Results

The proposed BBO trainer is evaluated by comparing its results with standard RBFclassic and other ten meta-heuristic (GA, PSO, ACO, ES, PBIL, DE, FF, Cuckoo, ABC and BAT) trainers using the Accuracy, Sensitivity, Specificity, MSE, and Complexity evaluation measures.

Table 3 shows the the results in terms of the average accuracy (AVE) and standard deviation (STD), as well as the best accuracy result of the proposed BBO and other algorithms on Blood dataset. The table reports the results with different number of neurons in the hidden layer. The best accuracy results are highlighted in bold. According to the results of AVE, STD, and best results using 4 neurons, BBO is able to classify 77.1 % of the test samples, which is more than PSO, ACO, ES, PBIL, DE and ABC results and

Table 2 Summary of the classification datasets

Dataset	#Features	#Training samples	#Testing samples	Accuracy of baseline classifier
Blood	4	493	255	0.7647
Breast	8	461	238	0.6597
Hepatitis	10	102	53	0.8113
Diabetes	8	506	262	0.6336
Vertebral	6	204	106	0.7075
Diagnosis I	6	79	41	0.5366
Diagnosis II	6	79	41	0.5366
Parkinson	22	128	67	0.7612
Liver	6	227	118	0.5763
Sonar	60	137	71	0.5352
German	24	660	340	0.6706
Australian	14	455	235	0.5702

Table 3 The average accuracy, and standard deviation results of the Blood dataset using different algorithms

Algorithm	The number of neurons of hidden layer							
	4		6		8		10	
	AVE \pm STD	Best	AVE \pm STD	Best	AVE \pm STD	Best	AVE \pm STD	Best
BBO	0.7710 \pm 0.0046	0.7765	0.7729 \pm 0.0039	0.7765	0.7745 \pm 0.0028	0.7765	0.7733 \pm 0.0036	0.7765
GA	0.7718 \pm 0.0045	0.7765	0.7714 \pm 0.0045	0.7765	0.7722 \pm 0.0057	0.7765	0.7718 \pm 0.0052	0.7765
PSO	0.7671 \pm 0.0033	0.7725	0.7690 \pm 0.0039	0.7765	0.7663 \pm 0.0027	0.7725	0.7671 \pm 0.0053	0.7765
ACO	0.7031 \pm 0.1515	0.7686	0.6929 \pm 0.1305	0.7686	0.7290 \pm 0.0523	0.7765	0.7639 \pm 0.0045	0.7686
ES	0.7643 \pm 0.0050	0.7725	0.7549 \pm 0.0332	0.7765	0.7455 \pm 0.0445	0.7686	0.7408 \pm 0.0554	0.7725
PBIL	0.7671 \pm 0.0096	0.7804	0.7663 \pm 0.0059	0.7765	0.7616 \pm 0.0118	0.7725	0.7667 \pm 0.0081	0.7765
DE	0.7655 \pm 0.0017	0.7686	0.7624 \pm 0.0083	0.7725	0.7553 \pm 0.0252	0.7725	0.7663 \pm 0.0033	0.7725
FireFly	0.7722 \pm 0.0043	0.7804	0.7694 \pm 0.0045	0.7765	0.7686 \pm 0.0041	0.7765	0.7694 \pm 0.0055	0.7765
Cuckoo	0.7718 \pm 0.0041	0.7765	0.7702 \pm 0.0046	0.7765	0.7714 \pm 0.0042	0.7765	0.7686 \pm 0.0037	0.7725
RBFclassic	0.7765 \pm 0.0000	0.7765	0.7686 \pm 0.0000	0.7686	0.7686 \pm 0.0000	0.7686	0.8118 \pm 0.0000	0.8118
ABC	0.7639 \pm 0.0086	0.7725	0.7659 \pm 0.0026	0.7725	0.7663 \pm 0.0027	0.7725	0.7663 \pm 0.0038	0.7725
BAT	0.7729 \pm 0.0034	0.7765	0.7725 \pm 0.0037	0.7765	0.7718 \pm 0.0045	0.7765	0.7757 \pm 0.0052	0.7804

Table 4 The average accuracy and standard deviation results of the Breast dataset using different algorithms

Algorithm	The number of neurons of hidden layer							
	4		6		8		10	
	AVE \pm STD	Best	AVE \pm STD	Best	AVE \pm STD	Best	AVE \pm STD	Best
BBO	0.9655 \pm 0.0119	0.9790	0.9761 \pm 0.0053	0.9832	0.9697 \pm 0.0108	0.9832	0.9786 \pm 0.0037	0.9832
GA	0.9647 \pm 0.0168	0.9790	0.9639 \pm 0.0143	0.9790	0.9689 \pm 0.0121	0.9832	0.9664 \pm 0.0097	0.9790
PSO	0.9294 \pm 0.0510	0.9790	0.9294 \pm 0.0438	0.9622	0.9315 \pm 0.0272	0.9622	0.9118 \pm 0.0333	0.9538
ACO	0.6765 \pm 0.1691	0.8151	0.6303 \pm 0.2806	0.9034	0.6013 \pm 0.2291	0.9622	0.6853 \pm 0.2100	0.9328
ES	0.7025 \pm 0.1942	0.8866	0.5887 \pm 0.2510	0.9244	0.6605 \pm 0.2319	0.9412	0.6441 \pm 0.1497	0.8445
PBIL	0.9311 \pm 0.0361	0.9706	0.9105 \pm 0.0914	0.9706	0.8874 \pm 0.0475	0.9496	0.8319 \pm 0.1982	0.9706
DE	0.7895 \pm 0.1094	0.9412	0.6929 \pm 0.2318	0.8908	0.8626 \pm 0.0917	0.9706	0.8546 \pm 0.0736	0.9664
FireFly	0.9567 \pm 0.0086	0.9664	0.9563 \pm 0.0147	0.9832	0.9462 \pm 0.0239	0.9706	0.9660 \pm 0.0090	0.9790
Cuckoo	0.9639 \pm 0.0077	0.9790	0.9567 \pm 0.0116	0.9748	0.9643 \pm 0.0103	0.9790	0.9525 \pm 0.0131	0.9706
RBFclassic	0.9580 \pm 0.0000	0.9580	0.9622 \pm 0.0000	0.9622	0.9538 \pm 0.0000	0.9538	0.9454 \pm 0.0000	0.9454
ABC	0.9613 \pm 0.0090	0.9706	0.9546 \pm 0.0106	0.9748	0.9248 \pm 0.0443	0.9748	0.9311 \pm 0.0329	0.9790
BAT	0.9550 \pm 0.0480	0.9832	0.9685 \pm 0.0112	0.9790	0.9744 \pm 0.0046	0.9790	0.9693 \pm 0.0136	0.9790

with slight difference with GA, FireFly, Cuckoo, BAT and RBFclassic. Furthermore, BBO outperforms all other methods using 6 neurons and 8 neurons with accuracy rates 77.29 and 77.45 %, respectively. In addition, the accuracy results of RBFclassic, BAT and BBO using 10 neurons are very close for this dataset, and the three algorithms outperform the other methods.

The accuracy results of BBO and other optimizers for Breast cancer dataset are presented in Table 4. According to the results of AVE, STD, and best results, BBO outperforms all other methods using 4, 6 and 10 neurons.

Moreover, the BBO able to classify 96.55, 97.61, 96.97, and 97.86 % of the test samples using 4, 6, 8, and 10 neurons, respectively.

Table 5 presents the accuracy results of the Hepatitis dataset. The results of BBO are significantly better than all the other algorithms using 6 neurons. Moreover, BBO has better results compared with most algorithms using 4, 8, and 10 neurons as well.

The accuracy results of BBO and other training algorithms on Diabetes and Vertebral datasets are presented in Tables 6, and 7, respectively. According to the results of

Table 5 The average accuracy and standard deviation results of the Hepatitis dataset using different algorithms

Algorithm	The number of neurons of hidden layer							
	4		6		8		10	
	AVE \pm STD	Best	AVE \pm STD	Best	AVE \pm STD	Best	AVE \pm STD	Best
BBO	0.8377 \pm 0.0221	0.8868	0.8472 \pm 0.0165	0.8679	0.8283 \pm 0.0188	0.8679	0.8302 \pm 0.0126	0.8491
GA	0.8453 \pm 0.0232	0.8868	0.8321 \pm 0.0165	0.8491	0.8453 \pm 0.0318	0.9057	0.8472 \pm 0.0259	0.9057
PSO	0.8358 \pm 0.0252	0.8868	0.8245 \pm 0.0296	0.8868	0.8321 \pm 0.0273	0.8868	0.8245 \pm 0.0268	0.8491
ACO	0.5472 \pm 0.2871	0.8302	0.6642 \pm 0.2200	0.9057	0.6585 \pm 0.1972	0.8491	0.5528 \pm 0.2655	0.8302
ES	0.7472 \pm 0.1249	0.8113	0.5830 \pm 0.2892	0.8113	0.6887 \pm 0.2292	0.8491	0.7057 \pm 0.1845	0.8679
PBIL	0.8283 \pm 0.0259	0.8679	0.8264 \pm 0.0214	0.8679	0.8321 \pm 0.0259	0.8679	0.8189 \pm 0.0419	0.9245
DE	0.8000 \pm 0.0497	0.8302	0.7038 \pm 0.1797	0.8491	0.6755 \pm 0.2328	0.8302	0.7887 \pm 0.0716	0.8491
FireFly	0.8547 \pm 0.0252	0.9057	0.8321 \pm 0.0165	0.8491	0.8340 \pm 0.0292	0.8679	0.8340 \pm 0.0149	0.8491
Cuckoo	0.8623 \pm 0.0282	0.8868	0.8396 \pm 0.0160	0.8679	0.8472 \pm 0.0243	0.8868	0.8245 \pm 0.0296	0.8679
RBFclassic	0.8302 \pm 0.0000	0.8302	0.8302 \pm 0.0000	0.8302	0.8113 \pm 0.0000	0.8113	0.7736 \pm 0.0000	0.7736
ABC	0.8340 \pm 0.0306	0.8868	0.8358 \pm 0.0252	0.8679	0.8321 \pm 0.0273	0.8868	0.8377 \pm 0.0347	0.8868
BAT	0.8509 \pm 0.0165	0.8868	0.8453 \pm 0.0149	0.8679	0.8321 \pm 0.0208	0.8679	0.8264 \pm 0.0195	0.8679

Table 6 The average accuracy and standard deviation results of the Diabetes dataset using different algorithms

Algorithm	The number of neurons of hidden layer							
	4		6		8		10	
	AVE \pm STD	Best	AVE \pm STD	Best	AVE \pm STD	Best	AVE \pm STD	Best
BBO	0.6950 \pm 0.0237	0.7443	0.6996 \pm 0.0268	0.7481	0.6996 \pm 0.0181	0.7214	0.7160 \pm 0.0190	0.7481
GA	0.6702 \pm 0.0223	0.6947	0.6863 \pm 0.0239	0.7176	0.6916 \pm 0.0275	0.7405	0.7000 \pm 0.0206	0.7290
PSO	0.6599 \pm 0.0360	0.7061	0.6645 \pm 0.0310	0.7176	0.6752 \pm 0.0291	0.7023	0.6569 \pm 0.0257	0.7023
ACO	0.6008 \pm 0.0855	0.6527	0.5767 \pm 0.0921	0.6489	0.5687 \pm 0.1170	0.6489	0.5668 \pm 0.1248	0.6527
ES	0.5794 \pm 0.1263	0.6794	0.6244 \pm 0.0510	0.7176	0.5679 \pm 0.0880	0.6489	0.6130 \pm 0.0452	0.6603
PBIL	0.6676 \pm 0.0242	0.7176	0.6630 \pm 0.0363	0.7557	0.6626 \pm 0.0425	0.7557	0.6401 \pm 0.0282	0.6794
DE	0.6378 \pm 0.0120	0.6527	0.6176 \pm 0.0642	0.7023	0.6263 \pm 0.0686	0.7099	0.6126 \pm 0.0743	0.6679
FireFly	0.6771 \pm 0.0421	0.7595	0.6931 \pm 0.0374	0.7634	0.6863 \pm 0.0365	0.7557	0.6763 \pm 0.0318	0.7290
Cuckoo	0.6817 \pm 0.0375	0.7710	0.6863 \pm 0.0223	0.7405	0.6954 \pm 0.0267	0.7443	0.6672 \pm 0.0275	0.7099
RBFclassic	0.7405 \pm 0.0000	0.7405	0.7672 \pm 0.0000	0.7672	0.7748 \pm 0.0000	0.7748	0.7481 \pm 0.0000	0.7481
ABC	0.6985 \pm 0.0250	0.7405	0.6989 \pm 0.0219	0.7481	0.6985 \pm 0.0305	0.7519	0.6782 \pm 0.0282	0.7214
BAT	0.7034 \pm 0.0228	0.7366	0.7118 \pm 0.0176	0.7405	0.7069 \pm 0.0239	0.7443	0.7088 \pm 0.0254	0.7634

AVE, STD, and best results using 4, 6, 8, and 10 neurons, BBO outperforms most of other methods except RBFclassic which has better accuracy. These results support the merits of the proposed BBO algorithm in training RBF networks.

The accuracy results of BBO and other algorithms on Diagnosis I and Diagnosis II datasets are presented in Tables 8, and 9, respectively. According to the results of AVE, STD, and best results, the results of BBO on the Diagnosis I are significantly better than the other algorithms using 4, 6, and 8 neurons. Furthermore, the BBO

results on the Diagnosis II are very comparable with other optimizers using different neurons.

Tables 10, and 11 show the accuracy results of BBO and other algorithms on Parkinsons and Liver datasets, respectively. The accuracy results of BBO on the Parkinson, and Liver datasets outperform all other algorithms using different number of neurons.

The accuracy results of BBO and other training algorithms on Sonar and German datasets are presented in Tables 12, and 13, respectively. According to both dataset results using 4, 6, 8, and 10 neurons, BBO comes second

Table 7 The average accuracy and standard deviation results of the Vertebral dataset using different algorithms

Algorithm	The number of neurons of hidden layer							
	4		6		8		10	
	AVE \pm STD	Best	AVE \pm STD	Best	AVE \pm STD	Best	AVE \pm STD	Best
BBO	0.7632 \pm 0.0340	0.8019	0.7708 \pm 0.0298	0.8208	0.7840 \pm 0.0201	0.8208	0.7792 \pm 0.0260	0.8302
GA	0.7604 \pm 0.0236	0.8019	0.7519 \pm 0.0281	0.8019	0.7623 \pm 0.0217	0.8113	0.7708 \pm 0.0236	0.8113
PSO	0.7406 \pm 0.0425	0.8302	0.7104 \pm 0.0256	0.7453	0.7255 \pm 0.0525	0.8019	0.7283 \pm 0.0444	0.8113
ACO	0.6189 \pm 0.1727	0.7075	0.6453 \pm 0.1328	0.7547	0.6217 \pm 0.1547	0.7830	0.6094 \pm 0.1458	0.7075
ES	0.7208 \pm 0.0549	0.8302	0.6698 \pm 0.0712	0.7736	0.6481 \pm 0.1059	0.7358	0.6613 \pm 0.0805	0.7264
PBIL	0.7368 \pm 0.0421	0.8396	0.7406 \pm 0.0339	0.7925	0.7113 \pm 0.0408	0.7925	0.7000 \pm 0.0307	0.7453
DE	0.6943 \pm 0.0214	0.7264	0.7019 \pm 0.0244	0.7453	0.6698 \pm 0.0868	0.7358	0.6623 \pm 0.1048	0.7547
FireFly	0.7443 \pm 0.0365	0.7925	0.7519 \pm 0.0263	0.7925	0.7519 \pm 0.0248	0.7925	0.7575 \pm 0.0336	0.8113
Cuckoo	0.7557 \pm 0.0296	0.8019	0.7500 \pm 0.0223	0.7736	0.7462 \pm 0.0296	0.8113	0.7566 \pm 0.0481	0.8396
RBFclassic	0.8208 \pm 0.0000	0.8208	0.8302 \pm 0.0000	0.8302	0.8302 \pm 0.0000	0.8302	0.8302 \pm 0.0000	0.8302
ABC	0.7208 \pm 0.0271	0.7642	0.7264 \pm 0.0537	0.8302	0.7047 \pm 0.0199	0.7358	0.7019 \pm 0.0214	0.7264
BAT	0.7594 \pm 0.0463	0.8208	0.7632 \pm 0.0245	0.8113	0.7623 \pm 0.0217	0.8113	0.7783 \pm 0.0253	0.8113

Table 8 The average accuracy and standard deviation results of the Diagnosis I dataset using different algorithms

Algorithm	The number of neurons of hidden layer							
	4		6		8		10	
	AVE \pm STD	Best	AVE \pm STD	Best	AVE \pm STD	Best	AVE \pm STD	Best
BBO	1.0000 \pm 0.0000	1.0000	1.0000 \pm 0.0000	1.0000	1.0000 \pm 0.0000	1.0000	0.9780 \pm 0.0694	1.0000
GA	0.9951 \pm 0.0154	1.0000	1.0000 \pm 0.0000	1.0000	1.0000 \pm 0.0000	1.0000	0.9707 \pm 0.0926	1.0000
PSO	0.8634 \pm 0.0899	0.9512	0.8171 \pm 0.1357	1.0000	0.7610 \pm 0.1114	0.9756	0.8439 \pm 0.0672	0.9512
ACO	0.5293 \pm 0.1436	0.7561	0.5732 \pm 0.1869	0.9268	0.5244 \pm 0.2179	0.9756	0.5195 \pm 0.1048	0.7805
ES	0.5244 \pm 0.1068	0.7561	0.5634 \pm 0.0533	0.6341	0.6024 \pm 0.1602	0.8780	0.5634 \pm 0.0817	0.6829
PBIL	0.8146 \pm 0.1455	1.0000	0.8220 \pm 0.1260	1.0000	0.7073 \pm 0.1988	0.9268	0.8000 \pm 0.1739	1.0000
DE	0.6341 \pm 0.1751	0.8780	0.7024 \pm 0.1476	0.8780	0.5951 \pm 0.1771	1.0000	0.6317 \pm 0.2177	0.9024
FireFly	0.9805 \pm 0.0427	1.0000	0.9659 \pm 0.0504	1.0000	0.9463 \pm 0.0561	1.0000	0.9683 \pm 0.0431	1.0000
Cuckoo	0.9707 \pm 0.0926	1.0000	0.9805 \pm 0.0427	1.0000	0.9878 \pm 0.0263	1.0000	0.9683 \pm 0.0920	1.0000
RBFclassic	0.8780 \pm 0.0000	0.8780	0.8780 \pm 0.0000	0.8780	1.0000 \pm 0.0000	1.0000	0.9756 \pm 0.0000	0.9756
ABC	0.9244 \pm 0.0825	1.0000	0.7659 \pm 0.1207	0.9756	0.7951 \pm 0.0997	0.9268	0.8683 \pm 0.0997	1.0000
BAT	0.9878 \pm 0.0386	1.0000	0.9780 \pm 0.0694	1.0000	1.0000 \pm 0.0000	1.0000	1.0000 \pm 0.0000	1.0000

after the RBFclassic method and it outperforms all other metaheuristic methods.

The accuracy results of BBO and other optimizers for Australian dataset are presented in Table 14. According to the results of using 4, 6, 8, and 10 neurons, BBO has superior classification accuracy results compared with other methods. Moreover, the BBO was able to classify 85.32, 84.98, 84.51, and 85.11 % of the test samples using 4, 6, 8, and 10 neurons, respectively.

To give a better insight on the classification performance regarding each class label, the specificity and sensitivity

are measured and listed in Tables 15, 16, 17 and 18 for RBF networks with 2, 6, 8 and 10 neurons in the hidden layer respectively. According to these tables, it can be noticed that RBF networks optimized by BBO have higher and more balanced specificity and sensitivity than most of the other optimizers in the following datasets: Breast cancer, Hepatitis, Vertebral, Diagnosis I, Diagnosis II, Parkinsons, Liver, Sonar and Australian.

To summarize the above results, we can note that BBO outperform most of other algorithms, which supports the merits of the proposed BBO algorithm in

Table 9 The average accuracy and standard deviation results of the Diagnosis II dataset using different algorithms

Algorithm	The number of neurons of hidden layer							
	4		6		8		10	
	AVE \pm STD	Best	AVE \pm STD	Best	AVE \pm STD	Best	AVE \pm STD	Best
BBO	1.0000 \pm 0.0000	1.0000	0.9829 \pm 0.0364	1.0000	0.9951 \pm 0.0154	1.0000	0.9927 \pm 0.0231	1.0000
GA	0.9976 \pm 0.0077	1.0000	1.0000 \pm 0.0000	1.0000	1.0000 \pm 0.0000	1.0000	1.0000 \pm 0.0000	1.0000
PSO	0.8220 \pm 0.0948	1.0000	0.8390 \pm 0.1302	1.0000	0.8854 \pm 0.0830	1.0000	0.8537 \pm 0.1222	1.0000
ACO	0.5732 \pm 0.1086	0.7073	0.5488 \pm 0.1317	0.8537	0.5195 \pm 0.0976	0.7317	0.5683 \pm 0.2136	0.9268
ES	0.4780 \pm 0.1036	0.6098	0.5902 \pm 0.1412	0.8049	0.6220 \pm 0.2173	1.0000	0.6561 \pm 0.1005	0.8049
PBIL	0.8512 \pm 0.1779	1.0000	0.8073 \pm 0.1169	1.0000	0.7512 \pm 0.1957	1.0000	0.8098 \pm 0.0967	1.0000
DE	0.6122 \pm 0.1303	0.8049	0.6268 \pm 0.1965	1.0000	0.6317 \pm 0.1652	0.8537	0.6707 \pm 0.1157	0.8293
FireFly	0.9707 \pm 0.0524	1.0000	0.9585 \pm 0.0502	1.0000	0.9512 \pm 0.0660	1.0000	0.9512 \pm 0.0586	1.0000
Cuckoo	1.0000 \pm 0.0000	1.0000	0.9756 \pm 0.0325	1.0000	0.9659 \pm 0.0611	1.0000	0.9463 \pm 0.0648	1.0000
RBFclassic	1.0000 \pm 0.0000	1.0000	1.0000 \pm 0.0000	1.0000	1.0000 \pm 0.0000	1.0000	1.0000 \pm 0.0000	1.0000
ABC	0.9537 \pm 0.0775	1.0000	0.9293 \pm 0.0848	1.0000	0.8951 \pm 0.0927	1.0000	0.8756 \pm 0.0841	1.0000
BAT	0.9927 \pm 0.0165	1.0000	0.9634 \pm 0.0773	1.0000	0.9927 \pm 0.0231	1.0000	1.0000 \pm 0.0000	1.0000

Table 10 The average accuracy and standard deviation results of the Parkinson dataset using different algorithms

Algorithm	The number of neurons of hidden layer							
	4		6		8		10	
	AVE \pm STD	Best	AVE \pm STD	Best	AVE \pm STD	Best	AVE \pm STD	Best
BBO	0.8358 \pm 0.0416	0.8657	0.8493 \pm 0.0164	0.8657	0.8522 \pm 0.0110	0.8657	0.8642 \pm 0.0192	0.8955
GA	0.8194 \pm 0.0458	0.8955	0.8194 \pm 0.0431	0.8657	0.8373 \pm 0.0192	0.8657	0.8478 \pm 0.0231	0.8806
PSO	0.7716 \pm 0.0483	0.8657	0.7806 \pm 0.0559	0.8657	0.7746 \pm 0.0495	0.8657	0.7478 \pm 0.1213	0.8657
ACO	0.6090 \pm 0.2162	0.8060	0.6522 \pm 0.2121	0.8209	0.5866 \pm 0.2497	0.8507	0.5821 \pm 0.2338	0.7910
ES	0.5672 \pm 0.2055	0.7761	0.5104 \pm 0.2386	0.7612	0.5090 \pm 0.2293	0.7761	0.5343 \pm 0.2351	0.7761
PBIL	0.7776 \pm 0.0164	0.8060	0.7716 \pm 0.0173	0.8060	0.7612 \pm 0.0000	0.7612	0.7612 \pm 0.0000	0.7612
DE	0.5478 \pm 0.2429	0.7761	0.5552 \pm 0.2491	0.7612	0.7403 \pm 0.0672	0.7910	0.5463 \pm 0.2372	0.7761
FireFly	0.7910 \pm 0.0244	0.8358	0.7881 \pm 0.0271	0.8209	0.7955 \pm 0.0338	0.8507	0.7955 \pm 0.0467	0.8806
Cuckoo	0.8239 \pm 0.0297	0.8507	0.8134 \pm 0.0354	0.8507	0.8030 \pm 0.0252	0.8358	0.7896 \pm 0.0277	0.8209
RBFclassic	0.7910 \pm 0.0000	0.7910	0.8209 \pm 0.0000	0.8209	0.8358 \pm 0.0000	0.8358	0.8507 \pm 0.0000	0.8507
ABC	0.7955 \pm 0.0330	0.8358	0.7985 \pm 0.0292	0.8358	0.7791 \pm 0.0321	0.8507	0.7642 \pm 0.0609	0.8358
BAT	0.8254 \pm 0.0273	0.8657	0.8463 \pm 0.0299	0.8955	0.8149 \pm 0.0399	0.8806	0.8343 \pm 0.0192	0.8507

training MLPs. Moreover, and to support this summary, Friedman statistical test is calculated to check the significance of the accuracy results. Friedman test is accomplished by ranking the different trainers (BBO, GA, PSO, ACO, ES, PBIL, DE, FireFly, Cuckoo, ABC, BAT and RBFclassic) based on the average accuracy values for each dataset using different neurons. Table 19 shows the average ranks for each technique in using Friedman test for 4, 6, 8, and 10 neurons. The Friedman

test in the Table 19 shows that significant difference exists between the 12 trainers (lower is better). In terms of F-test ranking, BBO outperforms other trainers for all number of neurons that used.

Figure 4 shows the complexity of trained RBF network and its corresponding MSE on different datasets for each number of neurons in the hidden layer. As shown in all sub-figures for all datasets, BBO has the best results, which has relatively the smallest MSE comparing with all other

Table 11 The average accuracy and standard deviation results of the Liver dataset using different algorithms

Algorithm	The number of neurons of hidden layer							
	4		6		8		10	
	AVE \pm STD	Best	AVE \pm STD	Best	AVE \pm STD	Best	AVE \pm STD	Best
BBO	0.6288 \pm 0.0298	0.6780	0.6500 \pm 0.0256	0.6864	0.6568 \pm 0.0266	0.6949	0.6915 \pm 0.0436	0.7542
GA	0.6059 \pm 0.0358	0.6864	0.6068 \pm 0.0260	0.6610	0.6466 \pm 0.0277	0.6949	0.6288 \pm 0.0379	0.6864
PSO	0.5797 \pm 0.0256	0.6271	0.5805 \pm 0.0517	0.7034	0.5898 \pm 0.0260	0.6271	0.5746 \pm 0.0450	0.6356
ACO	0.5136 \pm 0.0601	0.5763	0.5068 \pm 0.0675	0.5763	0.5203 \pm 0.0655	0.5932	0.5220 \pm 0.0634	0.5847
ES	0.5508 \pm 0.0504	0.5763	0.5051 \pm 0.0663	0.5763	0.5339 \pm 0.0744	0.5847	0.5339 \pm 0.0594	0.6102
PBIL	0.5627 \pm 0.0582	0.6356	0.5720 \pm 0.0286	0.6271	0.5593 \pm 0.0325	0.5932	0.5542 \pm 0.0272	0.5847
DE	0.5610 \pm 0.0232	0.5932	0.5525 \pm 0.0527	0.6186	0.5432 \pm 0.0612	0.6102	0.5220 \pm 0.0638	0.6356
FireFly	0.6169 \pm 0.0236	0.6525	0.6051 \pm 0.0398	0.6864	0.5856 \pm 0.0220	0.6271	0.6110 \pm 0.0431	0.6780
Cuckoo	0.6229 \pm 0.0302	0.6695	0.6102 \pm 0.0496	0.6864	0.6178 \pm 0.0380	0.6780	0.5983 \pm 0.0335	0.6356
RBFclassic	0.5763 \pm 0.0000	0.5763	0.5932 \pm 0.0000	0.5932	0.6017 \pm 0.0000	0.6017	0.6102 \pm 0.0000	0.6102
ABC	0.5712 \pm 0.0223	0.6186	0.5653 \pm 0.0400	0.6102	0.5720 \pm 0.0358	0.6186	0.5534 \pm 0.0383	0.5932
BAT	0.6449 \pm 0.0287	0.6949	0.6254 \pm 0.0312	0.6695	0.6568 \pm 0.0166	0.6864	0.6483 \pm 0.0332	0.6949

Table 12 The average accuracy and standard deviation results of the Sonar dataset using different algorithms

Algorithm	The number of neurons of hidden layer							
	4		6		8		10	
	AVE \pm STD	Best	AVE \pm STD	Best	AVE \pm STD	Best	AVE \pm STD	Best
BBO	0.7549 \pm 0.0365	0.8028	0.7296 \pm 0.0608	0.8169	0.7197 \pm 0.0698	0.8028	0.7380 \pm 0.0431	0.7887
GA	0.6507 \pm 0.1148	0.7746	0.5718 \pm 0.1016	0.7183	0.5254 \pm 0.1208	0.7746	0.6183 \pm 0.1166	0.7324
PSO	0.5352 \pm 0.0594	0.6197	0.4944 \pm 0.0438	0.5775	0.4887 \pm 0.0953	0.6761	0.5000 \pm 0.0761	0.6197
ACO	0.5028 \pm 0.0567	0.6338	0.5014 \pm 0.0451	0.5915	0.5296 \pm 0.0853	0.6901	0.5113 \pm 0.0358	0.5352
ES	0.4930 \pm 0.0398	0.5493	0.5099 \pm 0.0759	0.7042	0.5014 \pm 0.0498	0.5915	0.4915 \pm 0.0490	0.5775
PBIL	0.4648 \pm 0.0000	0.4648	0.4648 \pm 0.0000	0.4648	0.4648 \pm 0.0000	0.4648	0.4648 \pm 0.0000	0.4648
DE	0.5085 \pm 0.0348	0.5352	0.5239 \pm 0.0459	0.5915	0.4944 \pm 0.0462	0.5493	0.5000 \pm 0.0840	0.6479
FireFly	0.5408 \pm 0.0876	0.7042	0.5254 \pm 0.0903	0.6338	0.5155 \pm 0.1143	0.7183	0.5056 \pm 0.0672	0.6338
Cuckoo	0.5423 \pm 0.0772	0.7042	0.5423 \pm 0.0851	0.6479	0.5549 \pm 0.1012	0.7183	0.5268 \pm 0.0930	0.6901
RBFclassic	0.8451 \pm 0.0000	0.8451	0.7746 \pm 0.0000	0.7746	0.7746 \pm 0.0000	0.7746	0.8028 \pm 0.0000	0.8028
ABC	0.5183 \pm 0.0555	0.6197	0.5225 \pm 0.0764	0.7183	0.5465 \pm 0.0875	0.6761	0.4775 \pm 0.0776	0.6197
BAT	0.5577 \pm 0.1152	0.6761	0.5634 \pm 0.0888	0.6761	0.6014 \pm 0.1255	0.7324	0.5901 \pm 0.1162	0.8028

algorithms except the RBF classic. BBO outperforms all algorithms in terms of complexity, which has the smallest complexity values. Moreover, the RBFclassic has the smallest MSE values, but the complexity values are the largest, which outputs complex structure of the RBF network with very low smoothness. The complexity results show the merits of the proposed BBO algorithm in achieving very smooth RBF networks.

Convergence graphs for all algorithms are shown in the Figs. 5, 6, 7, and 8 using 4, 6, 8, and 10 neurons, respectively. The convergence curves show the MSE averages of 10 independent runs over 250 iterations. All sub-figures show that BBO is the fastest algorithm in convergence for all datasets. Furthermore, most of other algorithms like DE, ACO, ES, and PBIL have some drawbacks such as trapping at local minima with

Table 13 The average accuracy and standard deviation results of the German dataset using different algorithms

Algorithm	The number of neurons of hidden layer							
	4		6		8		10	
	AVE \pm STD	Best	AVE \pm STD	Best	AVE \pm STD	Best	AVE \pm STD	Best
BBO	0.7091 \pm 0.0268	0.7382	0.7094 \pm 0.0193	0.7382	0.7138 \pm 0.0150	0.7441	0.7191 \pm 0.0098	0.7353
GA	0.6756 \pm 0.0059	0.6882	0.6809 \pm 0.0222	0.7147	0.6779 \pm 0.0191	0.7029	0.6794 \pm 0.0211	0.7059
PSO	0.5638 \pm 0.1095	0.6765	0.5532 \pm 0.0947	0.6765	0.5556 \pm 0.1034	0.6765	0.5247 \pm 0.1112	0.6706
ACO	0.4609 \pm 0.1400	0.6706	0.5200 \pm 0.1604	0.6706	0.5591 \pm 0.1452	0.7000	0.4679 \pm 0.1363	0.6588
ES	0.5685 \pm 0.1381	0.6706	0.4747 \pm 0.1662	0.6706	0.5156 \pm 0.1552	0.6706	0.5615 \pm 0.1166	0.6706
PBIL	0.6703 \pm 0.0009	0.6706	0.6709 \pm 0.0009	0.6735	0.6706 \pm 0.0014	0.6735	0.6703 \pm 0.0009	0.6706
DE	0.4406 \pm 0.1436	0.6647	0.4824 \pm 0.1635	0.6706	0.5265 \pm 0.1435	0.6706	0.4862 \pm 0.1395	0.6706
FireFly	0.5703 \pm 0.1100	0.6706	0.5691 \pm 0.1279	0.6912	0.6012 \pm 0.1159	0.6765	0.6129 \pm 0.0971	0.6882
Cuckoo	0.6682 \pm 0.0048	0.6735	0.6579 \pm 0.0168	0.6706	0.6429 \pm 0.0435	0.6794	0.6471 \pm 0.0392	0.6765
RBFclassic	0.7235 \pm 0.0000	0.7235	0.7235 \pm 0.0000	0.7235	0.7265 \pm 0.0000	0.7265	0.7147 \pm 0.0000	0.7147
ABC	0.6712 \pm 0.0041	0.6824	0.6615 \pm 0.0117	0.6735	0.6571 \pm 0.0243	0.6765	0.6550 \pm 0.0313	0.6971
BAT	0.6674 \pm 0.0170	0.6941	0.6685 \pm 0.0086	0.6824	0.6609 \pm 0.0414	0.7059	0.6726 \pm 0.0113	0.6912

Table 14 The average accuracy and standard deviation results of the Australian dataset using different algorithms

Algorithm	The number of neurons of hidden layer							
	4		6		8		10	
	AVE \pm STD	Best	AVE \pm STD	Best	AVE \pm STD	Best	AVE \pm STD	Best
BBO	0.8532 \pm 0.0177	0.8894	0.8498 \pm 0.0164	0.8638	0.8451 \pm 0.0095	0.8638	0.8511 \pm 0.0083	0.8638
GA	0.8413 \pm 0.0149	0.8596	0.8255 \pm 0.0175	0.8426	0.8362 \pm 0.0234	0.8681	0.8455 \pm 0.0117	0.8638
PSO	0.7157 \pm 0.0666	0.8043	0.6630 \pm 0.0533	0.7617	0.6209 \pm 0.0847	0.7319	0.6272 \pm 0.1146	0.7872
ACO	0.5098 \pm 0.0998	0.6596	0.5885 \pm 0.1160	0.8383	0.4962 \pm 0.1538	0.7489	0.5289 \pm 0.0935	0.6766
ES	0.5409 \pm 0.0903	0.7064	0.5604 \pm 0.1344	0.7362	0.5911 \pm 0.0804	0.7362	0.5745 \pm 0.1425	0.7319
PBIL	0.6885 \pm 0.1087	0.8340	0.6949 \pm 0.1498	0.8298	0.7085 \pm 0.0926	0.7745	0.6009 \pm 0.1197	0.7660
DE	0.6064 \pm 0.0887	0.7106	0.5221 \pm 0.1139	0.6809	0.5740 \pm 0.0737	0.7064	0.6038 \pm 0.1160	0.7404
FireFly	0.7574 \pm 0.0721	0.8255	0.7694 \pm 0.0606	0.8170	0.7400 \pm 0.0507	0.8128	0.7421 \pm 0.0784	0.8468
Cuckoo	0.7843 \pm 0.0440	0.8298	0.7945 \pm 0.0366	0.8340	0.7872 \pm 0.0160	0.8128	0.7851 \pm 0.0272	0.8085
RBFclassic	0.8511 \pm 0.0000	0.8511	0.8468 \pm 0.0000	0.8468	0.8426 \pm 0.0000	0.8426	0.8426 \pm 0.0000	0.8426
ABC	0.8064 \pm 0.0235	0.8340	0.7574 \pm 0.0569	0.8426	0.7489 \pm 0.0530	0.8340	0.7694 \pm 0.0547	0.8426
BAT	0.8281 \pm 0.0405	0.8723	0.8340 \pm 0.0261	0.8553	0.8362 \pm 0.0200	0.8553	0.8255 \pm 0.0379	0.8596

slow convergence rate. Based on the convergence results, BBO has the superior ability to avoid the local optima.

In summary, the algorithms employed in this work can be classified into four groups: random, evolutionary, swarm-based, and gradient-based.

The results show that evolutionary trainers (including BBO) outperform the other four groups. This is due to the

superior local optima avoidance of these algorithms. Evolutionary algorithms mostly have cross-over operators that combine the search agents to create new population(s). Such operators abruptly change the individuals in the population, which results in emphasizing exploration of the search space and local optima avoidance. The gradient-based technique has the least local optima avoidance capability, which resulted in showing the worse

Table 15 The average sensitivity and specificity results of all datasets using different algorithms with (4 Neurons)

Algorithm	BBO		GA		PSO		ACO		ES		PBIL	
	Sen.	Spec.	Sen.	Spec.	Sen.	Spec.	Sen.	Spec.	Sen.	Spec.	Sen.	Spec.
Blood	1.0000	0.0267	0.9969	0.0400	0.9995	0.0117	0.8764	0.1400	0.9908	0.0283	0.9887	0.0467
Breast	0.9771	0.9432	0.9771	0.9407	0.9573	0.8753	0.7924	0.4519	0.8408	0.4346	0.9885	0.8198
Diabetes	0.3583	0.8898	0.2115	0.9355	0.2635	0.8892	0.2271	0.8169	0.2719	0.7572	0.1677	0.9566
Hepatitis	0.5500	0.9047	0.5300	0.9186	0.2700	0.9674	0.2900	0.6070	0.2700	0.8581	0.2700	0.9581
Vertebral	0.9027	0.4258	0.8933	0.4387	0.9120	0.3258	0.7587	0.2806	0.8893	0.3129	0.9133	0.3097
Diagnosis I	1.0000	1.0000	1.0000	0.9909	0.8684	0.8591	0.5526	0.5091	0.8632	0.2318	0.9684	0.6818
Diagnosis II	1.0000	1.0000	1.0000	0.9947	0.9136	0.7158	0.6273	0.5105	0.6000	0.3368	0.9091	0.7842
Parkinsons	0.9961	0.3250	0.9941	0.2625	0.9451	0.2188	0.6667	0.4250	0.5843	0.5125	0.9882	0.1063
Liver	0.1900	0.9515	0.0980	0.9794	0.1900	0.8662	0.3740	0.6162	0.1820	0.8221	0.2880	0.7647
Sonar	0.6447	0.8818	0.5053	0.8182	0.5895	0.4727	0.4342	0.5818	0.3184	0.6939	0.0000	1.0000
German	0.9513	0.2161	0.9820	0.0518	0.6811	0.3250	0.3833	0.6188	0.6925	0.3161	0.9996	0.0000
Australian	0.8426	0.8612	0.8485	0.8358	0.6614	0.7567	0.3535	0.6276	0.3901	0.6545	0.3317	0.9575
Algorithm	DE		FireFly		Cuckoo		RBFclassic		ABC		BAT	
	Sen.	Spec.	Sen.	Spec.	Sen.	Spec.	Sen.	Spec.	Sen.	Spec.	Sen.	Spec.
Blood	0.9995	0.0050	0.9995	0.0333	1.0000	0.0300	0.9897	0.0833	0.9938	0.0167	0.9964	0.0467
Breast	0.9191	0.5383	0.9605	0.9494	0.9745	0.9432	0.9809	0.9136	0.9643	0.9556	0.9752	0.9160
Diabetes	0.0531	0.9759	0.2823	0.9054	0.3271	0.8867	0.5625	0.8434	0.3729	0.8867	0.4094	0.8735
Hepatitis	0.0600	0.9721	0.5300	0.9302	0.5000	0.9465	0.7000	0.8605	0.4100	0.9326	0.6000	0.9093
Vertebral	0.9387	0.1032	0.8587	0.4677	0.8920	0.4258	0.8667	0.7097	0.8427	0.4258	0.8720	0.4871
Diagnosis I	0.6842	0.5909	1.0000	0.9636	1.0000	0.9455	1.0000	0.7727	0.9526	0.9000	1.0000	0.9773
Diagnosis II	0.6273	0.5947	1.0000	0.9368	1.0000	1.0000	1.0000	1.0000	0.9909	0.9105	1.0000	0.9842
Parkinsons	0.5843	0.4313	0.9824	0.1812	0.9804	0.3250	0.9412	0.3125	0.9667	0.2500	0.9863	0.3125
Liver	0.1540	0.8603	0.2260	0.9044	0.2160	0.9221	0.5600	0.5882	0.2260	0.8250	0.3140	0.8882
Sonar	0.6553	0.3394	0.6237	0.4455	0.3105	0.8091	0.7895	0.9091	0.3816	0.6758	0.4158	0.7212
German	0.3224	0.6813	0.6618	0.3839	0.9864	0.0205	0.8947	0.3750	0.9982	0.0054	0.9807	0.0295
Australian	0.4208	0.7463	0.6881	0.8097	0.7396	0.8179	0.9010	0.8134	0.8069	0.8060	0.8069	0.8440

performance on the test cases. The swarm-based algorithms perform better than the gradient-based algorithm because of the higher local optima avoidance. The high local optima avoidance of swarm-based algorithms mostly originate from the population-based nature of these algorithms. However, such algorithms have less intrinsic exploration ability compared to evolutionary algorithms because there are less sudden changes in the search agents.

The results also prove that evolutionary algorithms employed in this work show a better result accuracy and faster convergence rate in average. This shows that high random changes in the search agents of such algorithms do not negatively impact the result accuracy and convergence curve. This originates from the fact that evolutionary algorithms reduce randomness and favor gradual changes

with a mechanism called mutation. The mutation operator causes small perturbations and consequently local search around individuals in the population. In other words, the effects of mutation in the overall population are much less than cross-over operators. This operator assists evolutionary algorithms to improve the accuracy of solutions proportional to the number of iterations. Also, the convergence rate is accelerated toward the global optimum by the mutation operator.

Among the swarm-based techniques employed in this work, BAT algorithm outperforms Cuckoo, Firefly algorithm, PSO, ABC and ACO. BAT algorithm is equipped with frequency tuning principle which gives solutions that closed the ideal solutions. Furthermore, Cuckoo algorithm gives good results and very close to BAT algorithm. The

Table 16 The average sensitivity and specificity results of all datasets using different algorithms with (6 Neurons)

Algorithm	BBO		GA		PSO		ACO		ES		PBIL	
	Sen.	Spec.	Sen.	Spec.	Sen.	Spec.	Sen.	Spec.	Sen.	Spec.	Sen.	Spec.
Blood	0.9969	0.0450	1.0000	0.0283	0.9969	0.0283	0.8574	0.1583	0.9626	0.0800	0.9872	0.0483
Breast	0.9796	0.9691	0.9822	0.9284	0.9662	0.8580	0.6904	0.5136	0.6987	0.3753	0.9879	0.7605
Diabetes	0.3719	0.8892	0.3187	0.8988	0.2083	0.9283	0.2729	0.7524	0.1427	0.9030	0.1552	0.9566
Hepatitis	0.5900	0.9070	0.4900	0.9116	0.1300	0.9860	0.4900	0.7047	0.2500	0.6605	0.1600	0.9814
Vertebral	0.8787	0.5097	0.8880	0.4226	0.8893	0.2774	0.8253	0.2097	0.7413	0.4968	0.9253	0.2935
Diagnosis I	1.0000	1.0000	1.0000	1.0000	0.9053	0.7409	0.7263	0.4409	0.6211	0.5136	0.9789	0.6864
Diagnosis II	1.0000	0.9632	1.0000	1.0000	0.9182	0.7474	0.6182	0.4684	0.7136	0.4474	0.8727	0.7316
Parkinsons	0.9941	0.3875	1.0000	0.2437	0.9490	0.2437	0.7745	0.2625	0.5569	0.3625	0.9980	0.0500
Liver	0.3040	0.9044	0.1460	0.9456	0.3140	0.7765	0.4800	0.5265	0.3900	0.5897	0.2220	0.8294
Sonar	0.6579	0.8121	0.3447	0.8333	0.3289	0.6848	0.3974	0.6212	0.3605	0.6818	0.0000	1.0000
German	0.9500	0.2196	0.9311	0.1714	0.6478	0.3607	0.5561	0.4464	0.4360	0.5536	0.9996	0.0018
Australian	0.8505	0.8493	0.8178	0.8313	0.4941	0.7903	0.4238	0.7127	0.3149	0.7455	0.6970	0.6933
Algorithm	DE		FireFly		Cuckoo		RBFclassic		ABC		BAT	
	Sen.	Spec.	Sen.	Spec.	Sen.	Spec.	Sen.	Spec.	Sen.	Spec.	Sen.	Spec.
Blood	0.9908	0.0200	0.9985	0.0250	0.9933	0.0450	0.9795	0.0833	1.0000	0.0050	0.9985	0.0383
Breast	0.7987	0.4877	0.9656	0.9383	0.9707	0.9296	0.9809	0.9259	0.9484	0.9667	0.9777	0.9506
Diabetes	0.2667	0.8205	0.3469	0.8934	0.3052	0.9066	0.5729	0.8795	0.3271	0.9139	0.4260	0.8771
Hepatitis	0.2300	0.8140	0.4500	0.9209	0.5500	0.9070	0.7000	0.8605	0.3900	0.9395	0.6700	0.8860
Vertebral	0.9267	0.1581	0.8760	0.4516	0.9187	0.3419	0.8933	0.6774	0.8720	0.3742	0.8720	0.5000
Diagnosis I	0.8579	0.5682	0.9789	0.9545	1.0000	0.9636	1.0000	0.7727	0.9526	0.6045	0.9947	0.9636
Diagnosis II	0.8273	0.3947	1.0000	0.9105	1.0000	0.9474	1.0000	1.0000	0.9955	0.8526	1.0000	0.9211
Parkinsons	0.5882	0.4500	0.9706	0.2062	0.9882	0.2562	1.0000	0.2500	0.9569	0.2938	0.9922	0.3812
Liver	0.2160	0.8000	0.4400	0.7265	0.2420	0.8809	0.4600	0.6912	0.1440	0.8750	0.2760	0.8824
Sonar	0.6079	0.4273	0.6000	0.4394	0.3447	0.7697	0.7368	0.8182	0.4395	0.6182	0.4026	0.7485
German	0.4526	0.5429	0.6684	0.3670	0.9522	0.0589	0.8904	0.3839	0.9596	0.0545	0.9689	0.0571
Australian	0.4228	0.5970	0.7554	0.7799	0.7950	0.7940	0.8812	0.8209	0.7604	0.7552	0.8495	0.8224

Cuckoo algorithm has been equipped with a lévy flight which abruptly changes the search agents of this algorithm. Similarly to evolutionary algorithms, this causes extensive exploration of the search space and local optima avoidance significantly. However, other swarm-based algorithm have less operators to promote sudden changes. The ACO algorithm utilizes a pheromone matrix which mostly boosts exploitation and makes this algorithm more suitable for combinatorial problems. The performance of the PSO and ABC algorithms also largely depends on the distribution of initial population. These algorithms can easily be trapped in local solution if there is no good distribution in the initial population. The Firefly algorithm also does not random walk or lévy flight, which leads this algorithm to have

tendency toward local solutions and less able to avoid them.

In contrary, most of the evolutionary algorithms performed well on the test cases and suppressed swarm-based techniques. Among them, ES and DE showed the poorest performance. In ES, the selection of individual is deterministic which reduces randomness level and exploration of this algorithm. The main operators in this algorithm are mutations which favor exploitation and convergence. These are the main facts that contributed to the failure of this algorithm in solving the datasets. The same statements can be made for DE, but this algorithm has stochastic selection and more crossover operators, which assist it for showing a better performance compared to ES. The

Table 17 The average sensitivity and specificity results of all datasets using different algorithms with (8 Neurons)

Algorithm	BBO		GA		PSO		ACO		ES		PBIL	
	Sen.	Spec.	Sen.	Spec.	Sen.	Spec.	Sen.	Spec.	Sen.	Spec.	Sen.	Spec.
Blood	0.9979	0.0483	0.9969	0.0417	0.9995	0.0083	0.9185	0.1133	0.9672	0.0250	0.9851	0.0350
Breast	0.9790	0.9519	0.9720	0.9630	0.9656	0.8654	0.7261	0.3593	0.7191	0.5469	0.9943	0.6802
Diabetes	0.4354	0.8524	0.3063	0.9145	0.3469	0.8651	0.3052	0.7211	0.1927	0.7849	0.1833	0.9398
Hepatitis	0.6300	0.8744	0.5600	0.9116	0.2100	0.9767	0.4300	0.7116	0.2400	0.7930	0.3300	0.9488
Vertebral	0.8973	0.5097	0.8920	0.4484	0.8773	0.3581	0.7453	0.3226	0.8000	0.2806	0.9000	0.2548
Diagnosis I	1.0000	1.0000	1.0000	1.0000	0.9474	0.6000	0.5000	0.5455	0.5947	0.6091	0.8158	0.6136
Diagnosis II	1.0000	0.9895	1.0000	1.0000	0.9591	0.8000	0.5545	0.4789	0.7773	0.4421	0.8364	0.6526
Parkinsons	0.9922	0.4063	0.9922	0.3438	0.9255	0.2938	0.6549	0.3688	0.5373	0.4188	1.0000	0.0000
Liver	0.3500	0.8824	0.2700	0.9235	0.1880	0.8853	0.3360	0.6559	0.3280	0.6853	0.1580	0.8544
Sonar	0.6816	0.7636	0.1237	0.9879	0.3763	0.6182	0.5947	0.4545	0.3684	0.6545	0.0000	1.0000
German	0.9518	0.2295	0.9662	0.0911	0.5877	0.4902	0.6596	0.3545	0.5592	0.4268	0.9996	0.0009
Australian	0.8634	0.8313	0.8218	0.8470	0.5356	0.6851	0.6307	0.3948	0.5673	0.6090	0.6099	0.7828
Algorithm	DE		FireFly		Cuckoo		RBFclassic		ABC		BAT	
	Sen.	Spec.	Sen.	Spec.	Sen.	Spec.	Sen.	Spec.	Sen.	Spec.	Sen.	Spec.
Blood	0.9754	0.0400	0.9990	0.0200	0.9985	0.0333	0.9744	0.1000	1.0000	0.0067	0.9974	0.0383
Breast	0.9191	0.7531	0.9732	0.8938	0.9688	0.9556	0.9745	0.9136	0.9675	0.8420	0.9822	0.9593
Diabetes	0.1438	0.9054	0.3104	0.9036	0.3740	0.8813	0.5833	0.8855	0.3542	0.8976	0.4031	0.8825
Hepatitis	0.2300	0.7791	0.3800	0.9395	0.3900	0.9535	0.7000	0.8372	0.3100	0.9535	0.6100	0.8837
Vertebral	0.8760	0.1710	0.8667	0.4742	0.8827	0.4161	0.8933	0.6774	0.8293	0.4032	0.8813	0.4742
Diagnosis I	0.6053	0.5864	0.9579	0.9364	1.0000	0.9773	1.0000	1.0000	0.8895	0.7136	1.0000	1.0000
Diagnosis II	0.6591	0.6000	0.9955	0.9000	1.0000	0.9263	1.0000	1.0000	0.9318	0.8526	1.0000	0.9842
Parkinsons	0.8725	0.3187	0.9922	0.1688	0.9549	0.3187	0.9412	0.5000	0.9118	0.3563	0.9902	0.2562
Liver	0.2020	0.7941	0.3020	0.7941	0.3420	0.8206	0.4600	0.7059	0.1820	0.8588	0.3800	0.8603
Sonar	0.6132	0.3576	0.5789	0.4424	0.3947	0.7394	0.7632	0.7879	0.4395	0.6697	0.4684	0.7545
German	0.5662	0.4455	0.7925	0.2116	0.8939	0.1321	0.8904	0.3929	0.9513	0.0580	0.9351	0.1027
Australian	0.5356	0.6030	0.6287	0.8239	0.7802	0.7925	0.8713	0.8209	0.6733	0.8060	0.8545	0.8224

performance of the PBIL is better than ES and DE but worse than GA and BBO. This is because PBIL performs crossover on the entire population combined in a vector, which cause better exploration and local optima avoidance compared to ES and DE. However, each individual faces less sudden random changes in comparison with GA and BBO.

BBO outperformed GA because the random changes in the individuals are much higher in this algorithm. The GA algorithm assigns a similar reproduction rate to all the individuals in the population, which causes the same crossover rate over the course of generations. In contrary, the BBO algorithm assigns each individual a unique emigration and immigration rates. This results in different reproduction rates for each individual and consequently

promotion of the exploration and local optima avoidance. Needless to day, this is the main reason of the significant superiority of the BBO-based trainer compared to all trainers employed on all datasets in this work.

The results and discussion of this section show that the BBO algorithm is able to effectively alleviate the drawbacks of the current algorithms when training RBF networks in terms of local optima entrapment, result accuracy, and convergence rate.

5.4 Comparisons with traditional classifiers in the literature

In this section, we compare the results of the optimized RFB network using BBO with other five popular classifiers

Table 18 The average sensitivity and specificity results of all datasets using different algorithms with (10 Neurons)

Algorithm	BBO		GA		PSO		ACO		ES		PBIL	
	Sen.	Spec.	Sen.	Spec.	Sen.	Spec.	Sen.	Spec.	Sen.	Spec.	Sen.	Spec.
Blood	0.9974	0.0450	0.9995	0.0317	0.9985	0.0150	0.9959	0.0100	0.9513	0.0567	0.9692	0.1083
Breast	0.9809	0.9741	0.9822	0.9358	0.9554	0.8272	0.7076	0.6420	0.8561	0.2333	0.8898	0.7198
Diabetes	0.4625	0.8627	0.3469	0.9042	0.2375	0.8994	0.2104	0.7729	0.1646	0.8723	0.1812	0.9054
Hepatitis	0.6300	0.8767	0.6300	0.8977	0.3300	0.9395	0.4100	0.5860	0.2100	0.8209	0.1400	0.9767
Vertebral	0.8880	0.5161	0.8933	0.4742	0.8907	0.3355	0.8000	0.1484	0.8347	0.2419	0.8413	0.3581
Diagnosis I	1.0000	0.9591	1.0000	0.9455	0.9000	0.7955	0.5526	0.4909	0.5474	0.5773	0.9789	0.6455
Diagnosis II	1.0000	0.9842	1.0000	1.0000	0.9682	0.7211	0.6727	0.4474	0.6500	0.6632	0.9818	0.6105
Parkinsons	0.9961	0.4437	0.9902	0.3937	0.9098	0.2313	0.6392	0.4000	0.5627	0.4437	1.0000	0.0000
Liver	0.4320	0.8824	0.2420	0.9132	0.1980	0.8515	0.3800	0.6265	0.2720	0.7265	0.2540	0.7750
Sonar	0.6395	0.8515	0.4316	0.8333	0.6526	0.3242	0.6658	0.3333	0.3474	0.6576	0.0000	1.0000
German	0.9268	0.2964	0.9263	0.1768	0.5851	0.4018	0.3789	0.6491	0.6912	0.2973	0.9996	0.0000
Australian	0.8881	0.8231	0.8218	0.8634	0.6980	0.5739	0.6168	0.4627	0.3238	0.7634	0.5257	0.6575
Algorithm	DE		FireFly		Cuckoo		RBFclassic		ABC		BAT	
	Sen.	Spec.	Sen.	Spec.	Sen.	Spec.	Sen.	Spec.	Sen.	Spec.	Sen.	Spec.
Blood	1.0000	0.0067	0.9944	0.0383	0.9964	0.0283	0.9590	0.3333	0.9964	0.0183	0.9974	0.0550
Breast	0.9497	0.6704	0.9752	0.9481	0.9764	0.9062	0.9554	0.9259	0.9414	0.9111	0.9732	0.9617
Diabetes	0.1281	0.8928	0.3083	0.8892	0.3042	0.8771	0.5417	0.8675	0.3292	0.8801	0.4229	0.8741
Hepatitis	0.2600	0.9116	0.3300	0.9512	0.4300	0.9163	0.6000	0.8140	0.3000	0.9628	0.6100	0.8767
Vertebral	0.8213	0.2774	0.9013	0.4097	0.8987	0.4129	0.8933	0.6774	0.8853	0.2581	0.8760	0.5419
Diagnosis I	0.6263	0.6364	0.9947	0.9455	1.0000	0.9409	1.0000	0.9545	0.8842	0.8545	1.0000	1.0000
Diagnosis II	0.8000	0.5211	1.0000	0.8947	0.9682	0.9211	1.0000	1.0000	0.9773	0.7579	1.0000	1.0000
Parkinsons	0.5490	0.5375	0.9510	0.3000	0.9941	0.1375	0.9608	0.5000	0.9373	0.2125	0.9843	0.3563
Liver	0.2740	0.7044	0.2800	0.8544	0.2540	0.8515	0.4600	0.7206	0.1900	0.8206	0.3860	0.8412
Sonar	0.5632	0.4273	0.4711	0.5455	0.4737	0.5879	0.7632	0.8485	0.3368	0.6394	0.4184	0.7879
German	0.4899	0.4786	0.8110	0.2098	0.9184	0.0946	0.8684	0.4018	0.9259	0.1036	0.9772	0.0527
Australian	0.5208	0.6664	0.7931	0.7037	0.7267	0.8291	0.8515	0.8358	0.7396	0.7918	0.8525	0.8052

Table 19 The Average ranking results obtained by each algorithm in the Friedman test using all datasets

Algorithm	The number of neurons of hidden layer			
	4	6	8	10
BBO	2.4167	1.6250	2.4167	2.0000
GA	3.7083	3.6250	3.2500	3.0000
PSO	7.9583	8.3333	7.9167	7.9167
ACO	11.3333	11.0000	11.0833	11.1250
ES	10.8750	11.2500	10.8333	10.8333
PBIL	8.3750	8.1667	8.3758	8.7500
DE	10.2500	10.1667	10.3333	9.9583
FireFly	5.3750	5.7500	6.2083	5.3750
Cuckoo	4.0417	4.7083	4.5833	5.9167
RBFclassic	3.5417	3.3750	3.1250	3.0833
ABC	6.4583	7.0000	6.6667	7.0417
BAT	3.6667	3.0000	3.2083	3.0000

from the literature: Naive Bayes (NB), decision trees algorithm C4.5 (J48), Random Forests (RF), Support Vector Machines (SVM), and the Zero-R Rule Classifier (ZeroR) which is the base classifier. We used the Java-based open source data mining framework Weka as an implementation [20].

Table 20 shows the average accuracy results of NB, J48, RF, SVM, Zero-R, and BBO. It can be seen that optimized RFB network achieves very competitive results and performs reasonably. The BBO results for Breast, Liver and Australian datasets are higher and significantly better than all other classifiers. Moreover, the results of the BBO for Parkinson, and Blood datasets are very close to the other classifiers. Examining these results, we can notice that the BBO-RBF classifier has achieved better results than the base classifier in all datasets and better than NB and J48 in 7 and 6 datasets, respectively. Moreover, comparing BBO-

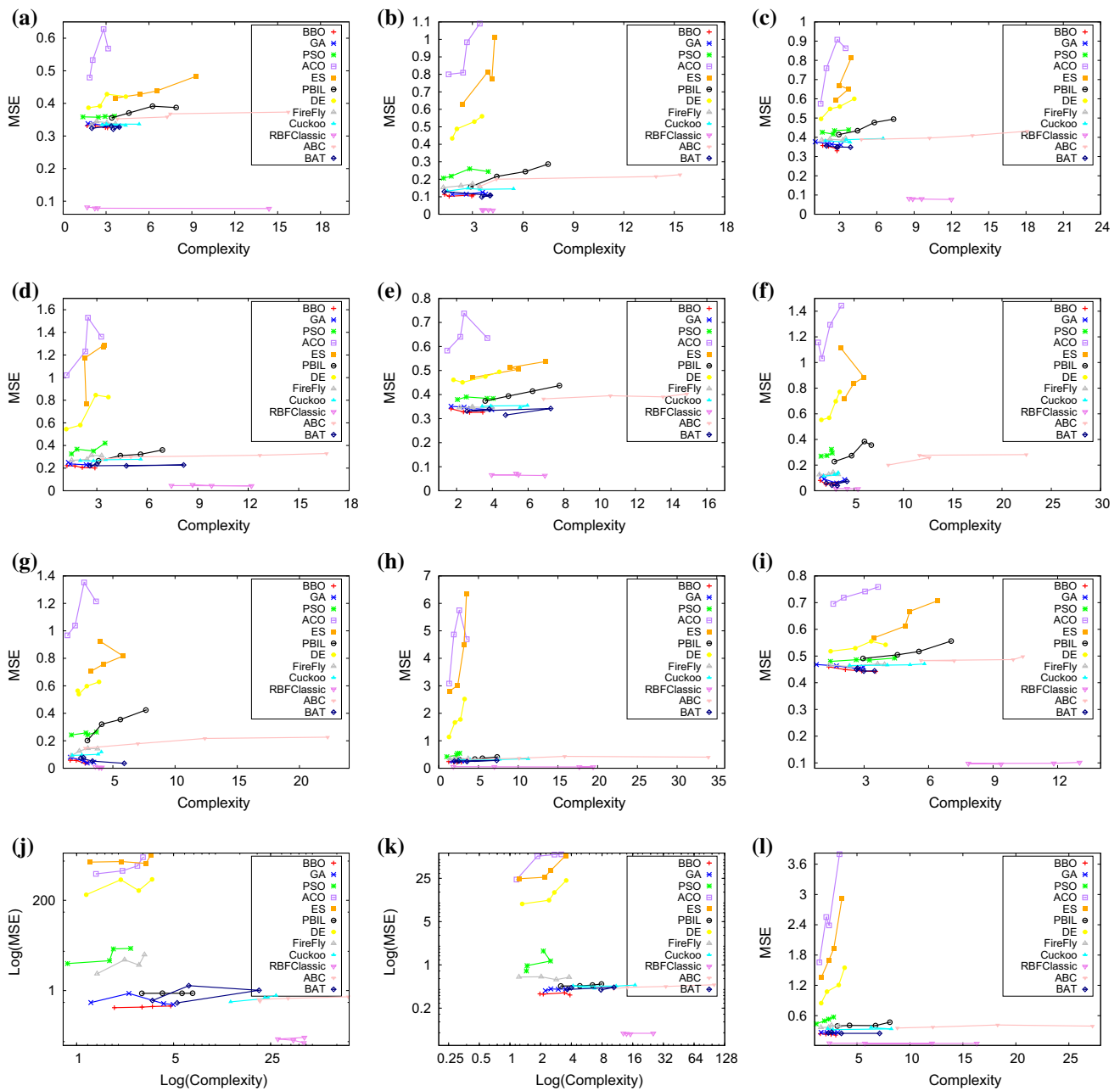


Fig. 4 The MSE versus complexity of the classification of different datasets (with 4, 6, 8, and 10 Neurons). Results for **a** Blood, **b** Breast, **c** Diabetes, **d** Hepatitis, **e** Vertebral, **f** Diagnosis I, **g** Diagnosis II, **h** Parkinson, **i** Liver, **j** Sonar, **k** German, and **l** Australian datasets, respectively

RBF network with very powerful classifiers like RF and SVM, we can see that it stays competitive with better accuracy results in 5 datasets.

As a summary, the obtained results by the proposed BBO support the merits of the proposed BBO algorithm in training RBF networks and solving data classification problems.

6 Conclusion

This paper proposed the use of the well-regarded BBO algorithm for training RBF networks to alleviate the drawbacks of conventional and new training algorithms: local optima entrapment, low result accuracy, and slow convergence speed. After proposing the method of training

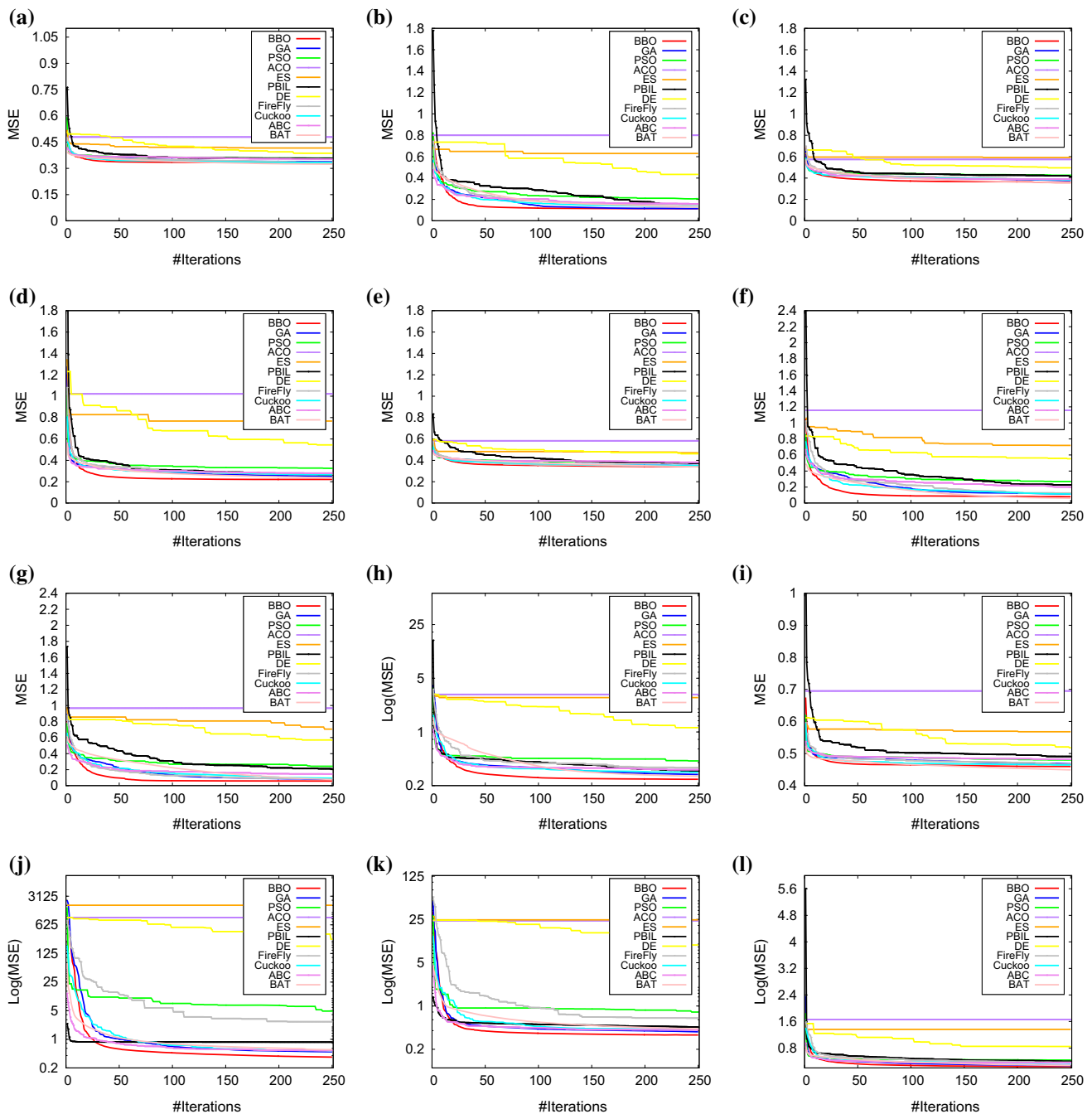


Fig. 5 MSE Convergence curves with (4 Neurons). Convergence curves for **a** Blood, **b** Breast, **c** Diabetes, **d** Hepatitis, **e** Vertebral, **f** Diagnosis I, **g** Diagnosis II, **h** Parkinson, **i** Liver, **j** Sonar, **k** German, and **l** Australian datasets, respectively

using BBO, it was employed to solve 12 well-known datasets and compared to 11 training methods in the literature including gradient-based, evolutionary, and swarm-based algorithms. The algorithms were compared by statistical test on RBF networks with different number of neurons to confidently confirm the performance of the proposed trainer. The results evidently demonstrated that

the BBO algorithm is able to outperform the current techniques on the majority of datasets substantially. According to the results, finding, analysis, and discussion of this paper, the following conclusions can be drawn:

1. BBO shows a fast convergence speed and high result accuracy.

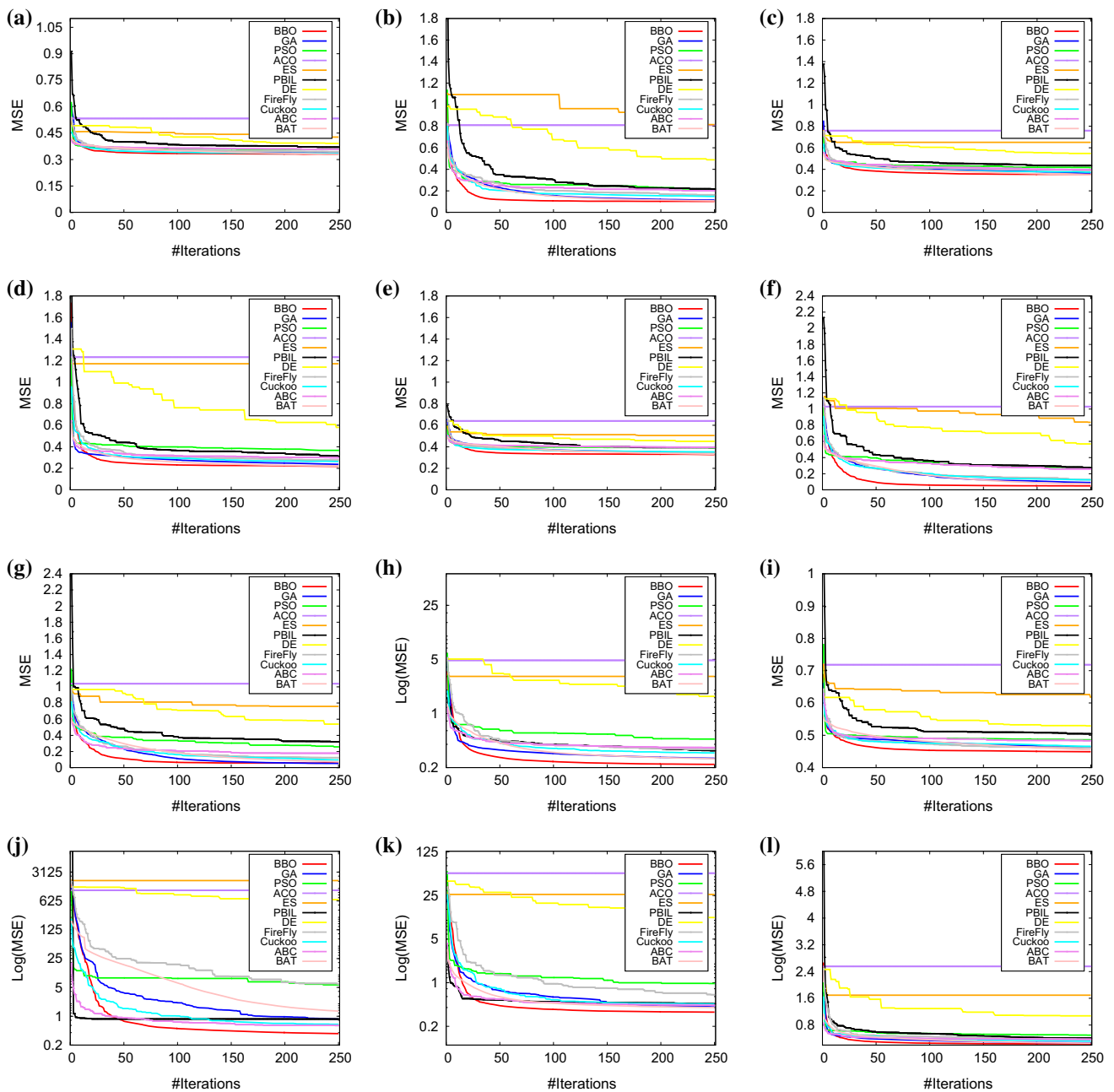


Fig. 6 MSE Convergence curves with (6 Neurons). Convergence curves for **a** Blood, **b** Breast, **c** Diabetes, **d** Hepatitis, **e** Vertebral, **f** Diagnosis I, **g** Diagnosis II, **h** Parkinson, **i** Liver, **j** Sonar, **k** German, and **l** Australian datasets, respectively

2. BBO can avoid local optima in the search space of the training RBF networks problem.
3. BBO is able to train RBF networks effectively to classify different datasets with a diverse number of features and training samples.
4. BBO is able to train RBF networks with different number of neurons.

For future works, this research is planned to be extended in two main lines. First, the proposed BBO-RBF network could be investigated for other data mining

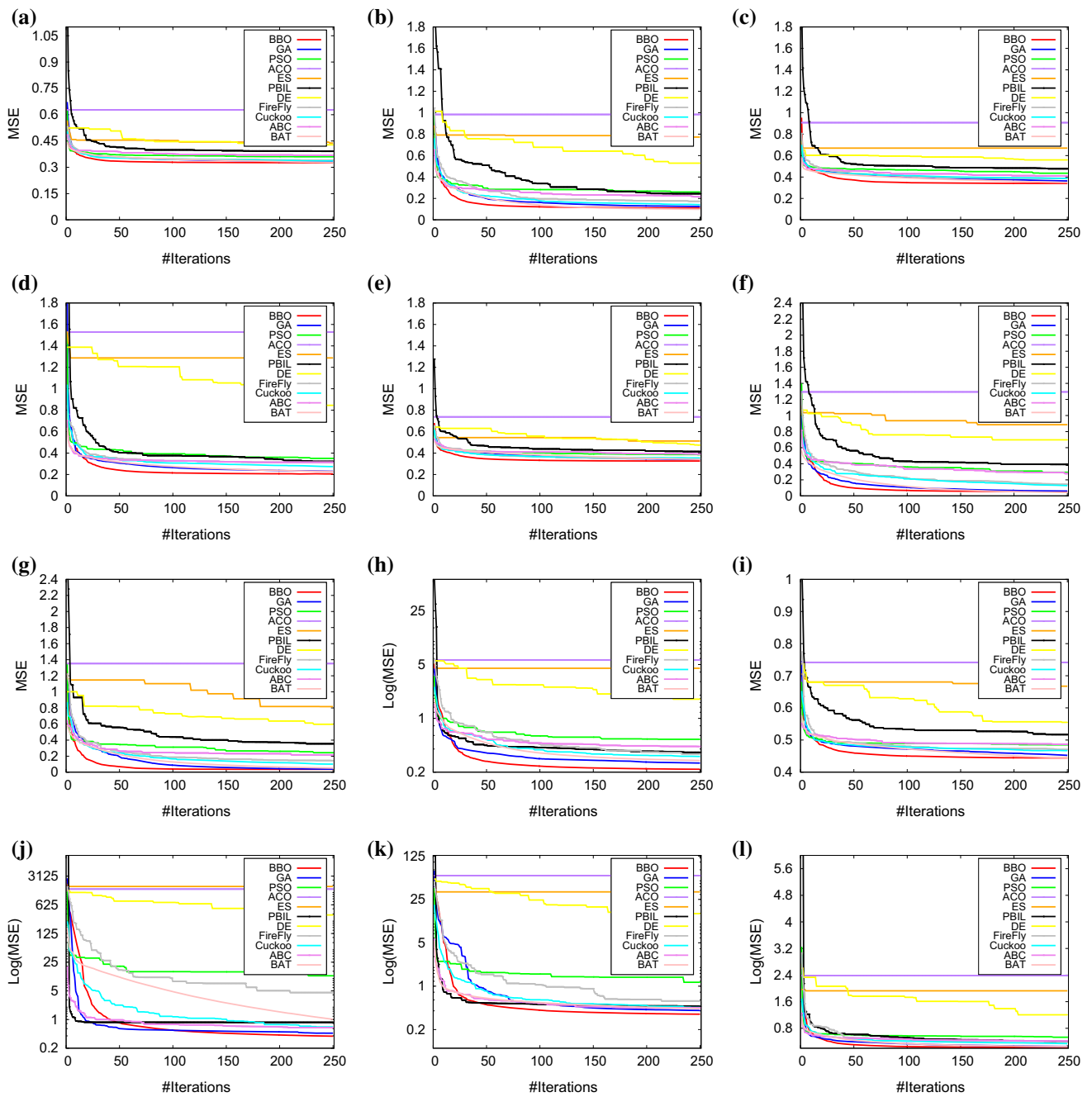


Fig. 7 MSE Convergence curves with (8 Neurons). Convergence curves for **a** Blood, **b** Breast, **c** Diabetes, **d** Hepatitis, **e** Vertebral, **f** Diagnosis I, **g** Diagnosis II, **h** Parkinson, **i** Liver, **j** Sonar, **k** German, and **l** Australian datasets, respectively

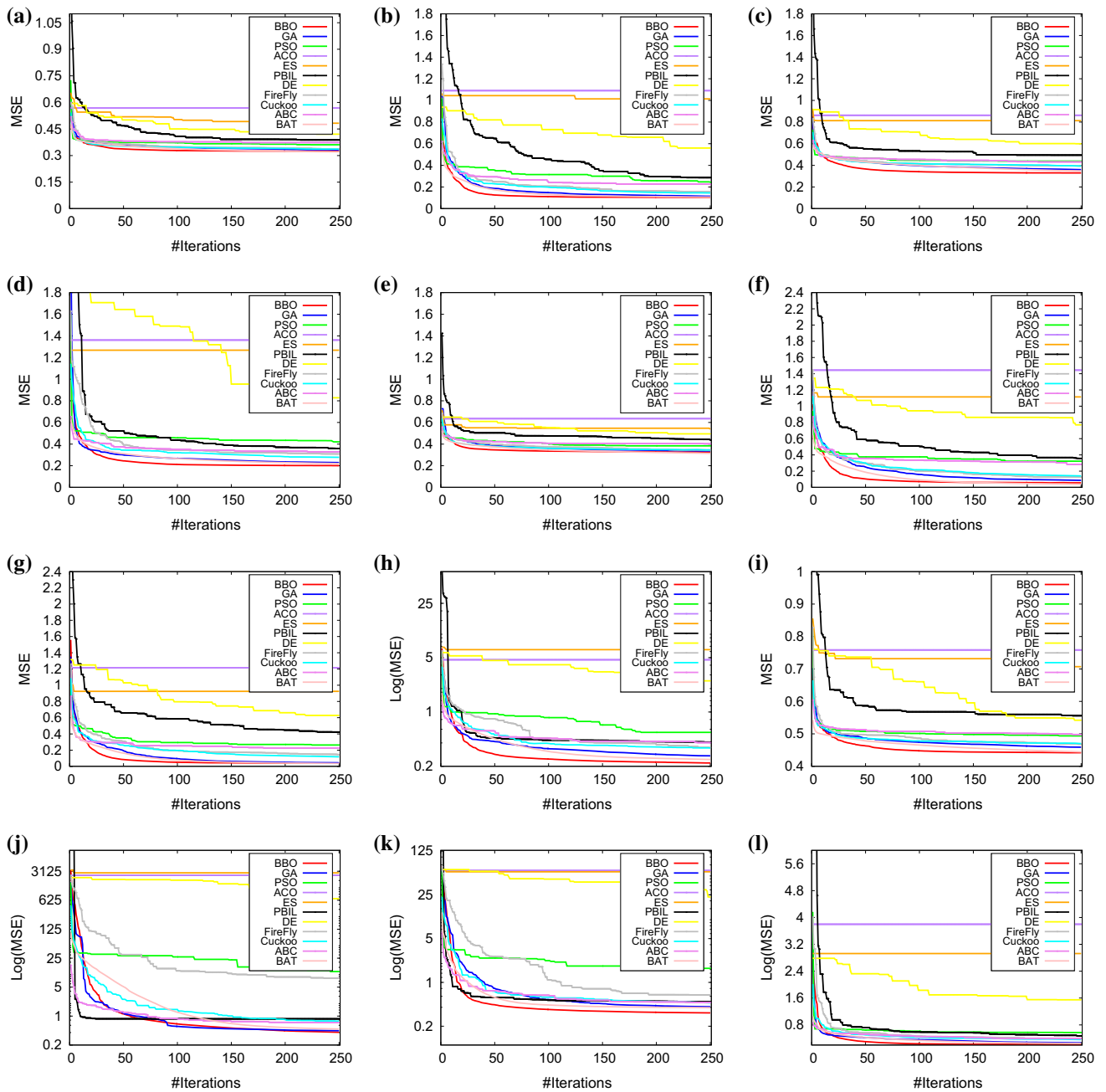


Fig. 8 MSE convergence curves with (10 Neurons). Convergence curves for **a** Blood, **b** Breast, **c** Diabetes, **d** Hepatitis, **e** Vertebral, **f** Diagnosis I, **g** Diagnosis II, **h** Parkinson, **i** Liver, **j** Sonar, **k** German, and **l** Australian datasets, respectively

Table 20 The average accuracy results of BBO-RBF network compared to popular classifiers in the literature

Algorithm	NB	J48	RF	SVM	ZeroR	BBO
Blood	0.74126	0.7804	0.7568	0.7647	0.7647	0.7745
Breast	0.9622	0.9416	0.9622	0.9748	0.6597	0.9786
Hepatitis	0.8679	0.8868	0.9434	0.8868	0.8113	0.8472
Diabetes	0.7214	0.7673	0.7672	0.7558	0.6336	0.7160
Vertebral	0.7736	0.7925	0.8208	0.8396	0.7075	0.7840
Diagnosis I	1.0000	1.0000	1.0000	1.0000	0.4634	1.0000
Diagnosis II	1.0000	1.0000	1.0000	1.0000	0.5366	1.0000
Parkinson	0.7463	0.8507	0.8657	0.8507	0.7612	0.8642
Liver	0.4831	0.6610	0.6610	0.5763	0.5763	0.6915
Sonar	0.6901	0.6901	0.7887	0.7747	0.4648	0.7380
German	0.7618	0.7177	0.7118	0.7529	0.6706	0.7191
Australian	0.736	0.8170	0.8511	0.8426	0.5702	0.8532

tasks like regression and time series prediction. Second, it is planned to study the efficiency of optimizing the structure of the RBF network along with its widths, centers and weights, simultaneously. As it is expected that the complexity of the search space will increase, the evaluation will consider complexity and executing time in addition to the prediction accuracy of the optimized models.

Compliance with ethical standards

Conflict of interest The authors declare that there is no conflict of interest regarding the publication of this paper.

References

- Aljarah I, Ludwig SA (2012) Parallel particle swarm optimization clustering algorithm based on mapreduce methodology. In: Nature and biologically inspired computing (NaBIC), 2012 fourth world congress on, pp 104–111. doi:[10.1109/NaBIC.2012.6402247](https://doi.org/10.1109/NaBIC.2012.6402247)
- de Almeida Rego JB, de Medeiros Martins A, de Costa B (2014) Deterministic system identification using RBF networks. *Math Probl Eng* 2014:1–10
- Ayala H, Vicente H, Coelho LDS (2014) Multiobjective cuckoo search applied toradial basis function neural networks training for system identification. In: World congress, vol 19, pp 2539–2544
- Bajer D, Zorić B, Martinović G (2015) Automatic design of radial basis function networks through enhanced differential evolution. In: Hybrid artificial intelligent systems, Springer, pp 244–256
- Bansal J, Singh P, Saraswat M, Verma A, Jadon SS, Abraham A (2011) Inertia weight strategies in particle swarm optimization. In: Nature and biologically inspired computing (NaBIC), 2011 third world congress on, IEEE, pp 633–640
- Billings SA, Zheng GL (1995) Radial basis function network configuration using genetic algorithms. *Neural Netw* 8(6):877–890
- BoussaiD I, Lepagnot J, Siarry P (2013) A survey on optimization metaheuristics. *Inform Sci* 237:82–117
- Broomhead D, Lowe D (1988) Multivariable functional interpolation and adaptive networks. *Complex Syst* 2:321–355
- Castao A, Hervas-Martinez C, Gutiérrez PA, Fernandez-Navarro F, Garcia MM (2009) Classification by evolutionary generalized radial basis functions. In: Intelligent systems design and applications, 2009. ISDA'09. Ninth international conference on IEEE, pp 203–208
- Chaowanawatee K, Heednacram A (2012) Implementation of cuckoo search in RBF neural network for flood forecasting. In: Proceedings of the 2012 fourth international conference on computational intelligence, communication systems and networks, IEEE Computer Society, Washington, DC, USA, CIC-SYN '12, pp 22–26. doi:[10.1109/CICSyN.2012.15](https://doi.org/10.1109/CICSyN.2012.15)
- Chng E, Chen S, Mulgrew B (1996) Gradient radial basis function networks for nonlinear and nonstationary time series prediction. *Neural Netw IEEE Trans* 7(1):190–194
- Chun-tao M, Xiao-xia L, Li-yong Z (2007) Radial basis function neural network based on ant colony optimization. In: Computational intelligence and security workshops, 2007. International Conference on CISW 2007, pp 59–62. doi:[10.1109/CISW.2007.4425446](https://doi.org/10.1109/CISW.2007.4425446)
- Cruz DPF, Maia RD, da Silva LA, de Castro LN (2016) BeeRBF: a bee-inspired data clustering approach to design RBF neural network classifiers. *Neurocomputing* 172:427–437. doi:[10.1016/j.neucom.2015.03.106](https://doi.org/10.1016/j.neucom.2015.03.106)
- Dash CSK, Behera AK, Pandia MK, Dehuri S (2013) Neural networks training based on differential evolution in radial basis function networks for classification of web logs. In: Distributed computing and internet technology, Springer, pp 183–194
- Ding S, Xu L, Su C, Jin F (2012) An optimizing method of RBF neural network based on genetic algorithm. *Neural Comput Appl* 21(2):333–336
- Du KL, Swamy MN (2006) Neural networks in a softcomputing framework. Springer, NewYork
- Fogel DB (1997) The advantages of evolutionary computation. In: BCEC, Citeseer, pp 1–11
- Gan M, Peng H, ping Dong X (2012) A hybrid algorithm to optimize RBF network architecture and parameters for nonlinear time series prediction. *Appl Math Model* 36(7):2911–2919. doi:[10.1016/j.apm.2011.09.066](https://doi.org/10.1016/j.apm.2011.09.066), <http://www.sciencedirect.com/science/article/pii/S0307904X11006251>
- Goldberg DE et al (1989) Genetic algorithms in search optimization and machine learning, vol 412. Addison-wesley Reading, Menlo Park
- Hall M, Frank E, Holmes G, Pfahringer B, Reutemann P, Witten IH (2009) The weka data mining software: an update. *ACM SIGKDD Explor Newsl* 11(1):10–18
- Harpham C, Dawson CW, Brown MR (2004) A review of genetic algorithms applied to training radial basis function networks. *Neural Comput Appl* 13(3):193–201
- Ho YC, Pepyne DL (2002) Simple explanation of the no free lunch theorem of optimization. *Cybern Syst Anal* 38(2):292–298
- Hornig M (2013) Training radial basis function network using the honey bee mating optimization. *Comput Model New Technol* 17(3):43–49
- Hornig MH, Lee YX, Lee MC, Liou RJ (2012) Firefly metaheuristic algorithm for training the radial basis function network for data classification and disease diagnosis. In: Parpinnelli R, Lopes HS (eds) Theory and New Applications of Swarm Intelligence, InTech, Rijeka, pp 1–19
- Huang CM, Wang FL (2007) An RBF network with OLS and EPSO algorithms for real-time power dispatch. *Power Syst IEEE Trans* 22(1):96–104
- Hunter D, Yu H, Pukish MS III, Kolbusz J, Wilamowski BM (2012) Selection of proper neural network sizes and architectures a comparative study. *IEEE Trans Ind Inform* 8(2):228–240

27. Karaboga D, Basturk B (2007) A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm. *J Glob Optim* 39(3):459–471
28. Kuncheva LI (1997) Initializing of an RBF network by a genetic algorithm. *Neurocomputing* 14(3):273–288
29. Kurban T, Beşdok E (2009) A comparison of rbf neural network training algorithms for inertial sensor based terrain classification. *Sensors* 9(8):6312–6329
30. Lee MJ, Choi YK (2004) An adaptive neurocontroller using RBFN for robot manipulators. *Ind Electron IEEE Trans* 51(3):711–717
31. Leonard J, Kramer MA et al (1991) Radial basis function networks for classifying process faults. *Control Syst IEEE* 11(3):31–38
32. Lichman M (2013) UCI machine learning repository. <http://archive.ics.uci.edu/ml>
33. Mak MW, Cho KW (1998) Genetic evolution of radial basis function centers for pattern classification. In: *Neural networks proceedings, 1998. IEEE world congress on computational intelligence. The 1998 IEEE International Joint Conference on IEEE*, vol 1, pp 669–673
34. Mezura-Montes E, Velázquez-Reyes J, Coello Coello CA (2006) A comparative study of differential evolution variants for global optimization. In: *Proceedings of the 8th annual conference on Genetic and evolutionary computation, ACM*, pp 485–492
35. Mirjalili S, Mirjalili SM, Lewis A (2014) Let a biogeography-based optimizer train your multi-layer perceptron. *Inf Sci* 269:188–209
36. Mohaghegi S, Valle YD, Venayagamoorthy GK, Harley RG (2005) A comparison of PSO and backpropagation for training RBF neural networks for identification of a power system with STATCOM. In: *Swarm intelligence symposium, 2005. SIS 2005. Proceedings 2005 IEEE, IEEE*, pp 381–384
37. Noman S, Shamsuddin SM, Hassanien AE (2009) Hybrid learning enhancement of RBF network with particle swarm optimization. In: *Foundations of computational, intelligence Vol 1, Springer*, pp 381–397
38. Ovreiu M, Simon D (2010) Biogeography-based optimization of neuro-fuzzy system parameters for diagnosis of cardiac disease. In: *Proceedings of the 12th annual conference on Genetic and evolutionary computation, ACM*, pp 1235–1242
39. Pedersen MEH, Chipperfield AJ (2008) Tuning differential evolution for artificial neural networks. HL0803 Hvas Laboratories
40. Qasem SN, Shamsuddin SM (2011) Radial basis function network based on time variant multi-objective particle swarm optimization for medical diseases diagnosis. *Appl Soft Comput* 11(1):1427–1438
41. Qasem SN, Shamsuddin SM, Zain AM (2012) Multi-objective hybrid evolutionary algorithms for radial basis function neural network design. *Knowl Based Syst* 27:475–497
42. Schwenker F, Kestler HA, Palm G (2001) Three learning phases for radial-basis-function networks. *Neural Netw* 14(4):439–458
43. Sheta AF, Jong KD (2001) Time-series forecasting using ga-tuned radial basis functions. *Inf Sci* 133(34):221 – 228, doi:10.1016/S0020-0255(01)00086-X, <http://www.sciencedirect.com/science/article/pii/S002002550100086X>, evolutionary Algorithms
44. Simon D (2008) Biogeography-based optimization. *Evol Comput IEEE Trans* 12(6):702–713
45. Sun X, Liu D, Li A (2011) The use of RBF based on ant colony algorithm and fisher ratio for eddy current nondestructive detecting system. In: *Progress In electromagnetics research symposium proceedings*, pp 633–636
46. Talal R (2014) Comparative study between the (ba) algorithm and (pso) algorithm to train (rbf) network at data classification. *Int J Comput Appl* 92(5):16–22
47. Tsekouras GE, Tsimikas J (2013) On training RBF neural networks using inputoutput fuzzy clustering and particle swarm optimization. *Fuzzy Sets Syst* 221:65–89. doi:10.1016/j.fss.2012.10.004, <http://www.sciencedirect.com/science/article/pii/S016511412004460>, theme: Clustering
48. Vakil-Baghmisheh MT, Pavešić N (2004) Training RBF networks with selective backpropagation. *Neurocomputing* 62:39–64. doi:10.1016/j.neucom.2003.11.011, <http://www.sciencedirect.com/science/article/pii/S0925231203005411>
49. Wolpert DH, Macready WG (1997) No free lunch theorems for optimization. *Evol Comput IEEE Trans* 1(1):67–82
50. Wu Y, Wang H, Zhang B, Du (2012) Using radial basis function networks for function approximation and classification. *ISRN Appl Math* 2012:1–34
51. Xin J, Chen G, Hai Y (2009) A particle swarm optimizer with multi-stage linearly-decreasing inertia weight. In: *Computational sciences and optimization, 2009. CSO 2009. International joint conference on IEEE*, vol 1, pp 505–508
52. Yang XS (2009) Firefly algorithms for multimodal optimization. In: *Stochastic algorithms: foundations and applications, Springer*, pp 169–178
53. Yang XS (2010) Firefly algorithm, levy flights and global optimization. In: *Research and development in intelligent systems XXVI, Springer*, pp 209–218
54. Yang XS (2010) A new metaheuristic bat-inspired algorithm. In: *Nature inspired cooperative strategies for optimization (NICSO 2010), Springer*, pp 65–74
55. Yang XS, Deb S (2009) Cuckoo search via lévy flights. In: *Nature and biologically inspired computing, 2009. NaBIC 2009. World congress on IEEE*, pp 210–214
56. Yang XS, He X (2013) Firefly algorithm: recent advances and applications. *Int J Swarm Intell* 1(1):36–50
57. Yang XS, Deb S, Fong S (2011) Accelerated particle swarm optimization and support vector machine for business optimization and applications. In: *Networked digital technologies, Springer*, pp 53–66
58. Yu B, He X (2006) Training radial basis function networks with differential evolution. In: *Granular computing, 2006 IEEE international conference on IEEE*, pp 369–372
59. Zhang J, Sanderson AC (2009) Jade: adaptive differential evolution with optional external archive. *Evol Comput IEEE Trans* 13(5):945–958
60. Zhao ZQ, Huang DS (2007) A mended hybrid learning algorithm for radial basis function neural networks to improve generalization capability. *Appl Math Model* 31(7):1271–1281