Chapter 7
Evolutionary Multi-layer Perceptron

7.1 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 modelled by McCulloch and Pitts [1]. 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) [2] 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) [3] and Multi-Layer Perceptron (MLP) [4]. 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 [5], data prediction [6], and function approximation [7]. Pattern classification implies classifying data into pre-defined discrete classes [8], whereas prediction refers to the forecasting of future trends according to current and previous data [6]. Finally, function approximation involves the process of modelling relationships between input variables. It has been proven that MLPs with one hidden layer are able to approximate any continuous or discontinuous functions [9, 10]. Regardless of the applications, the distinguishing capability of MLPs is learning [1]. 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 and unsupervised learning. For MLPs, most applications use the standard [11] or improved Back-Propagation (BP) [12] algorithms as their learning methods, which belong to the supervised learning fam-

Part of this chapter has been reprinted from Seyedali Mirjalili, Seyed Mohamed Mirjalili and Andrew Lewis article: Let a biogeography-based optimizer train your multilayer perceptron, Information Science, Volume 269, pp. 188–209, 2014

S. Mirjalili, *Evolutionary Algorithms and Neural Networks*, Studies in Computational Intelligence 780, https://doi.org/10.1007/978-3-319-93025-1_7

ily. Back Propagation (BP) is a gradient-based algorithm that has some drawbacks such as slow convergence [13] and a tendency to get trapped in local minima [14], 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 (e.g. [15]), but there is no reported significant improvement, and each method has its own side effects. The literature shows that heuristic optimisation methods are promising alternatives for gradient-based learning algorithms [16] since the stochastic nature of these algorithms allows them to avoid local minima better than gradient-based techniques and optimise challenging problems. Moreover, convergence rates of heuristic methods to the global minimum can be faster than BP, as investigated in [17].

7.2 Multi-layer Perceptrons

In MLP formulation, it is normally assumed that there are n number of input neurons, the number of hidden nodes is h, and the number of output nodes is m. 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. 7.1.

$$s_j = \sum_{i=1}^{n} \left(W_{ij}X_i\right) - \theta_j, \qquad j = 1, 2, \ldots, h \tag{7.1}$$

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:

$$S_j = sigmoid(s)j) = \frac{1}{1 + e^{-s_j}}, \qquad j = 1, 2, \ldots, m \tag{7.2}$$

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

$$o_k = \sum_{j=1}^{h} \left(w_{jk}S_j\right) - \theta'_k, \qquad k = 1, 2, \ldots, m \tag{7.3}$$

$$O_k = sigmoid(o_k) = \frac{1}{1 + e^{-o_k}}, \qquad k = 1, 2, \ldots, m \tag{7.4}$$

where w_{jk} is the connection weight from the jth hidden node to the kth output node, and $'_k$ 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.

7.3 Evolutionary Multi-layer Percenptron

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 minimise 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 [18].

Leung et al. used the last method to tune the parameters of an FNN utilizing EAs [19]. There are also some studies that utilized a combination of methods simultaneously. For instance, Mizuta et al. [20] and Leung et al. [19] employed a GA and improved GA to define the structure of an FNN.

In the next section, a variety of EAs are employed to train several MLPs for function approximation.

7.4 Results

In this section the EA algorithms are benchmarked on six function approximation datasets. 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 algorithms employed are BBO, PSO, GA, ACO, ES, and PBIL over these benchmark datasets. The comparison of the best WA with BP and ELM are also given.

Table 7.1 Function-approximation datasets

Function-approximation datasets	Training samples	Test samples
Sigmoid: $y = \frac{1}{1+e^{-x}}$	61: x in [-3:0.1:3]	121: x in [-3:0.05:3]
Cosine: $y = cos\left(\frac{x\pi}{2}\right)^7$	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 [-2:0.1:2]	252: x in [-2:0.05:2]
Sphere: $z = \sum_{i=1}^{2}, x = x_1, y = x_2$	21*21: x, y in [-2:0.2:2]	41*41: x, y in [-2:0.1:2]
Griewank: $z = \frac{1}{4000}\sum_{i=1}^{2} x_i^2 cos\left(\frac{x_i}{\sqrt{i}} + 1, x = x_1, y = x_2\right)$	21*21: x, y in [-1:0.1:1]	41*41: x, y in [-1:0.0.5:1]
Rosenbrock: $z = \sum_{i=1}^{5}\left[x_i^2 - 10cos(2\pi x_i) + 10\right]$	1024: $x_1 - x_5$ in [-5:3:5]	10247776: $x_1 - x_5$ in [-5:2:5]

It is assumed that the weights and biases are randomly initialized in the range $[-10, 10]$. The population size is 200 for all function approximation datasets. Table 7.1 show hows the datasets are divided in terms of training and test sets. It can be seen 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 7.2. 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 $\pm$ 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$\pm$STD may be a good combination to indicate the performance of an algorithm in terms of avoiding local minima.

According to Derrac et al. [21], 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; a statistical test is necessary to prove that a proposed new algorithm presents a significant improvement over other existing methods for a particular problem [22].

In order to judge whether the results of the best EA are better in a statistically significant way, a nonparametric statistical test, Wilcoxon's rank-sum test [23], 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 0.05 are considered as strong evidence against the null hypothesis. Note that p-values greater than 0.05 are underlined in the tables.

Table 7.2 Initial values for the controlling parameters of EAs

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	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_t)	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

The other comparative measures shown in the results are: classification rates and test errors. The best trained MLP among 10 runs is chosen to 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 behaviour 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:

$$x' = \frac{(x-a) \times (d-c)}{(b-a)} + c \tag{7.5}$$

Regarding the structure of the MLPs, 15 hidden nodes are used for function approximation datasets. In the following sections the simulation results of benchmark datasets are presented and discussed [24].

7.4.1 *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 7.3, Figs. 7.1 and 7.2. The results for AVE, STD, and p-values indicate that BBO is much better at avoiding local minima than the other algorithms. The test errors in Table 7.3 and approximated curves in Fig. 7.1 show that the BBO algorithm has the best approximate accuracy as well. Figure 7.2 shows that BBO has the fastest convergence rate.

7.4.2 *Cosine Function*

This dataset has 31 training samples and 38 test samples. The results are shown in Table 7.4. 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. Figures 7.3 and 7.4 illustrate the approximated curve and convergence, respectively.

Table 7.3 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	3.3563
GA	0.001093 ± 0.000916	6.39E-05	0.44969
ACO	0.023532 ± 0.010042	6.39E-05	3.9974
ES	0.075575 ± 0.016410	6.39E-05	8.8015
PBIL	0.004046 ± 2.74e-17	6.34E-05	2.9446

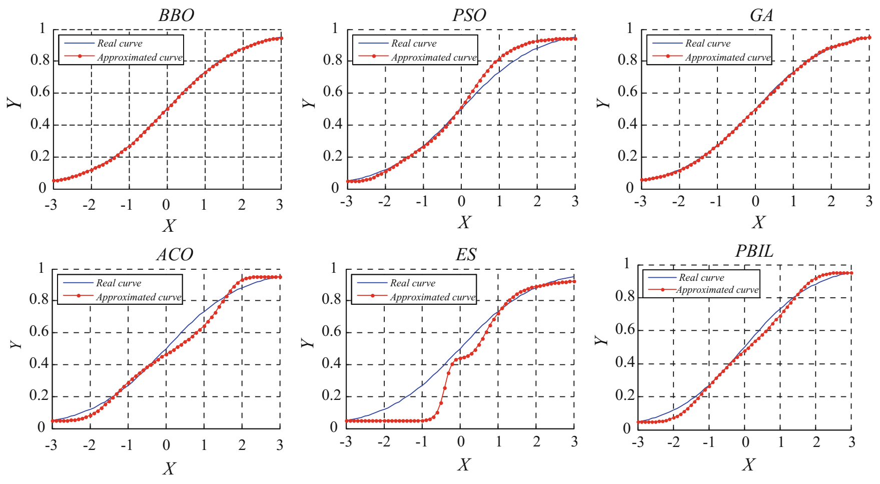


Fig. 7.1 Approximated curves for sigmoid function

Fig. 7.2 Convergence curves of algorithms for sigmoid function

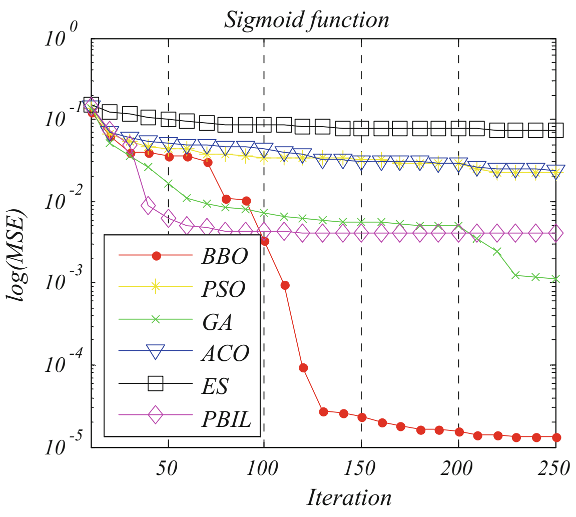


Table 7.4 Experimental results for one-peak cosine dataset (one dimensional)

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	2.009
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	3.727

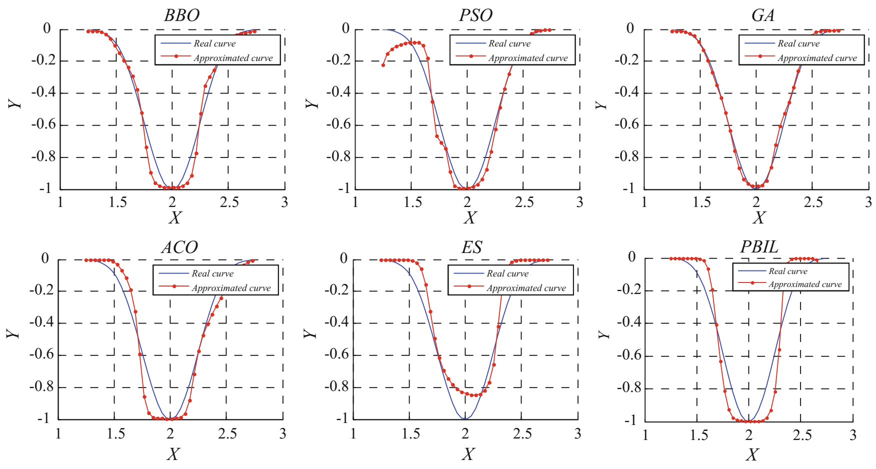


Fig. 7.3 Approximated curves for cosine function

7.4.3 Sine Function

This dataset is created by a four-peak sine function in the interval $[-2, 2]$ 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 7.5, Figs. 7.5 and 7.6. 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. 7.5 verify the superior accuracy of this algorithm. Finally, the curves of Fig. 7.6 confirm that BBO provides the fastest convergence speed.

7.4.4 Sphere Function

The sphere function dataset consists of 441 training samples and 1681 test samples. The results of this dataset are reported in Table 7.6. The averages and standard deviations of the MSEs show that BBO is the most reliable algorithm in terms of

Table 7.5 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	149.6

avoiding local minima. However, the results for the test error criterion and the best approximation surfaces in Fig. 7.7 indicate that the GA provides better accuracy. The convergence behaviours of the training algorithm are shown in Fig. 7.8. The convergence rates are entirely consistent with the previous datasets: BBO has the fastest rate, followed by the GA.

Fig. 7.4 Convergence curves of algorithms for cosine function

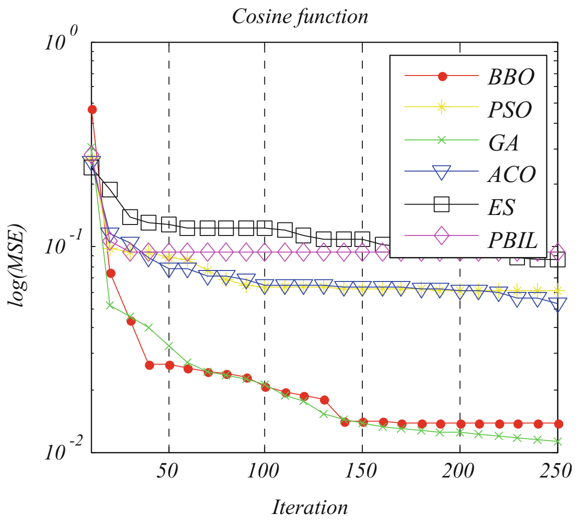


Table 7.6 Experimental results for sphere dataset (two dimensional)

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	1.27E+03
GA	2.9276 ± 0.7289	6.38E-05	452.3744
ACO	6.5626 ± 0.9543	6.38E-05	1.22E+03
ES	10.530 ± 0.8427	6.38E-05	1.50E+03
PBIL	10.459 ± 0.8529	6.38E-05	1.81E+03

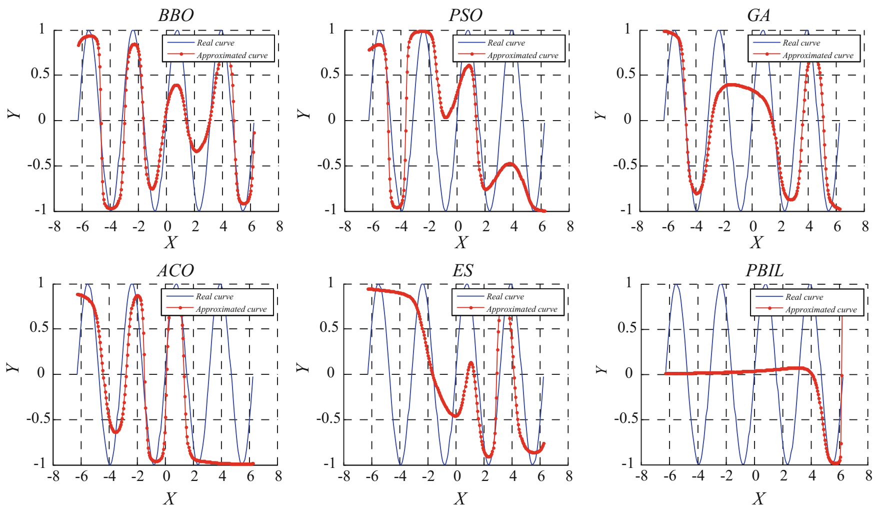
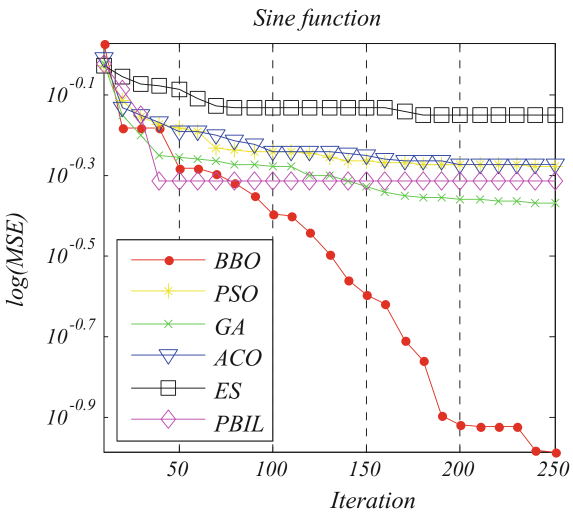


Fig. 7.5 Approximated curves for sin function

Fig. 7.6 Convergence curves of algorithms for sin function



7.4.5 Griewank Function

The Griewank dataset includes 441 training instances and 1681 test samples. The results for this dataset are shown in Table 7.7, Figs. 7.9 and 7.10. The results are quite similar to the previous datasets in that the BBO algorithm significantly outperforms the other algorithms in terms of avoiding local minima. According to test errors,

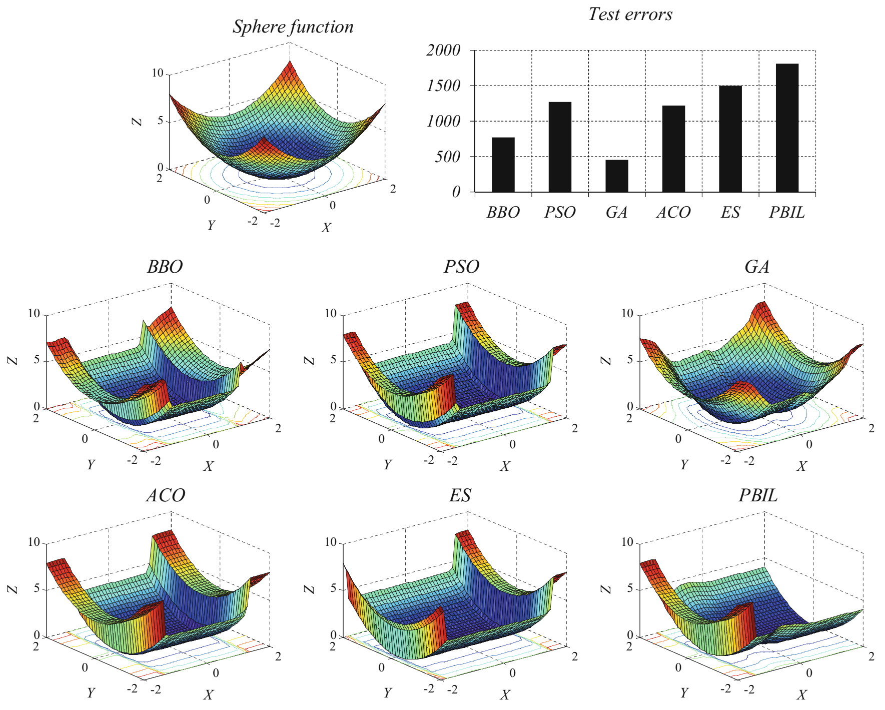


Fig. 7.7 Approximated curves for the sphere function

Fig. 7.8 Convergence curves of algorithms for the sphere function

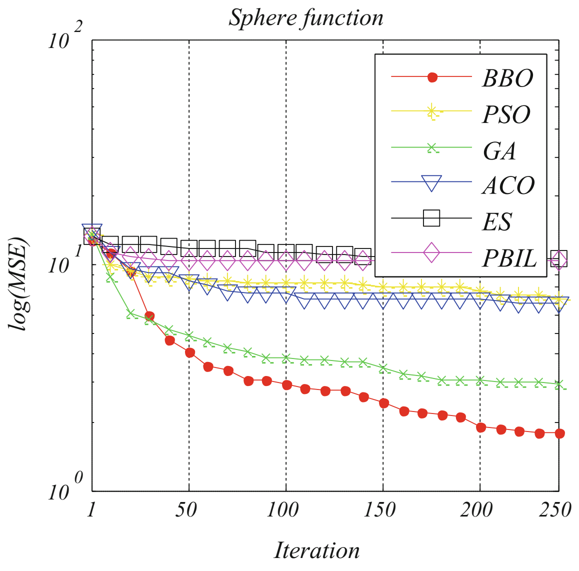


Table 7.7 Experimental results for Griewank dataset (two dimensional)

Algorithm	MSE (AVE ± STD)	P-values	Test error
BBO	2.02560 ± 0.0000	N/A	5.02E+03
PSO	16.9043 ± 1.1482	6.38E-05	7.63E+03
GA	16.4218 ± 1.3306	6.38E-05	1.06E+04
ACO	14.9586 ± 1.2976	6.38E-05	6.18E+03
ES	19.1069 ± 1.4984	6.38E-05	1.12E+04
PBIL	13.5587 ± 0.7532	6.29E-05	1.24E+04

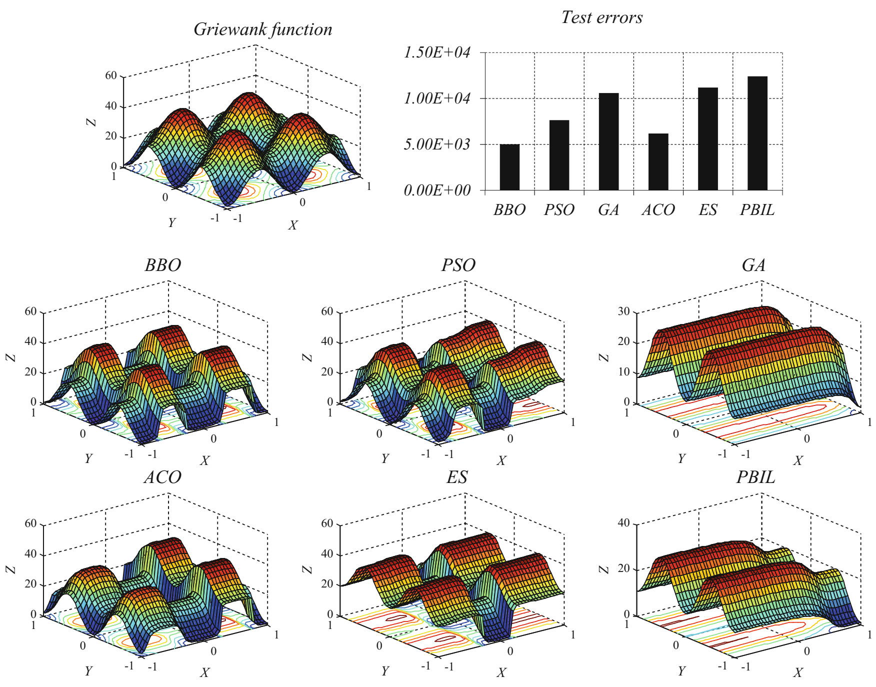


Fig. 7.9 Approximated curves for the Griewank function

Figs. 7.9 and 7.10, the BBO algorithm has the best approximation accuracy and convergence speed as well.

7.4.6 Rosenbrock Function

This dataset includes 1024 training samples and 7776 test samples. The results for this dataset are provided in Table. 7.8. According to the values of AVE, STD, and p-

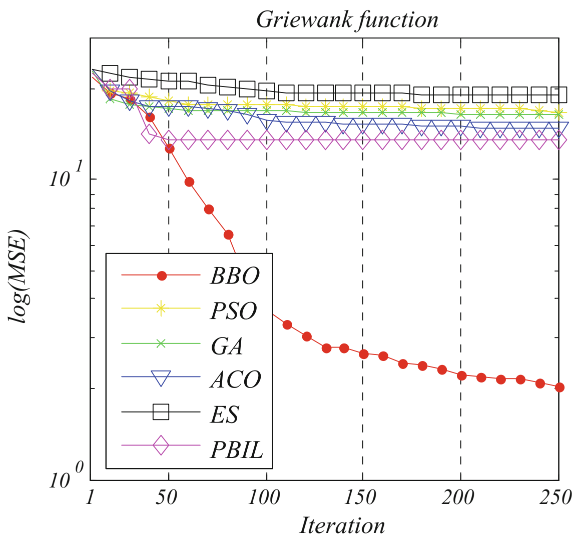


Fig. 7.10 Convergence curves of algorithms for the Griewank function

Table 7.8 Experimental results for Rosenbrock dataset (five-dimensional)

Algorithm	MSE (AVE ± STD)	P-values	Test error
BBO	104.4251 ± 1.59e-14	N/A	2.35E+06
PSO	215.1669 ± 3.5521	0.007937	6.50E+09
GA	264.6713 ± 16.3603	0.007937	5.24E+06
ACO	200.9749 ± 7.0641	0.007937	3.54E+07
ES	252.8454 ± 8.1445	0.007937	7.37E+09
PBIL	208.6148 ± 4.6813	0.007937	2.35E+08

values, the BBO algorithm significantly outperforms others in terms of avoiding local minima for this dataset. The test errors show that BBO has the highest accuracy for approximating the Rosenbrock function. Figure 7.11 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. The algorithm shows improved ability to avoid local minima.

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

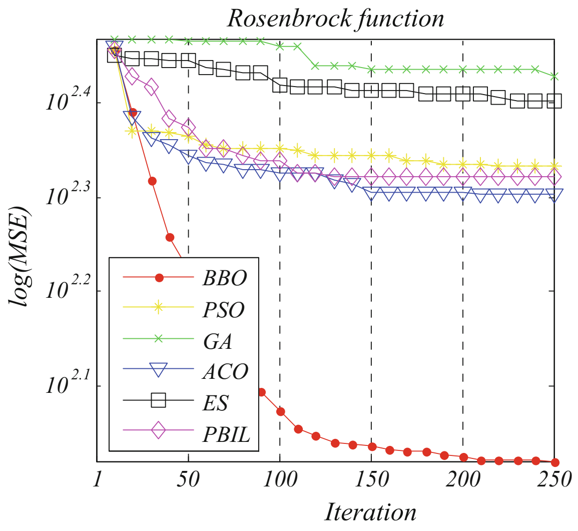


Fig. 7.11 Convergence curves of algorithms for the Rosenbrock function

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 results 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 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 PBILs poor performance.

The reason for superior performance of the BBO algorithm compared to GA in the majority of the datasets 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 behaviours and eventually greater exploration. As the results of Table 7.3–7.8 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 behaviours for every habitat. The results show that 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 BBOs 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.

7.4.7 Comparison with BP

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

The BP algorithm is a gradient-based algorithm that uses gradient information for minimising 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 7.9, 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:

- Varying values of emigration and immigration rates provide diverse information exchange behaviour 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.

Table 7.9 Comparison results between BBO and BP

Dataset	Algorithm	MSE (AVE ± STD)	P-values	Test error
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

- Elitism assists BBO to save and retrieve some of the best solutions, so they are neverlost. In the following section the BBO algorithm is compared to a very effective method called ELM.

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.

7.5 Conclusion

In this chapter, an MLP was trained by six evolutionary algorithms. Six benchmark datasets (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.

References

1. McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, *5*(4), 115–133.
2. Fahlman, S. E. (1988). An empirical study of learning speed in back-propagation networks.
3. Raudys, Š. (1998). Evolution and generalization of a single neurone: I. Single-layer perceptron as seven statistical classifiers. *Neural Networks*, *11*(2), 283–296.
4. Amendolia, S. R., Cossu, G., Ganadu, M. L., Golosio, B., Masala, G. L., & Mura, G. M. (2003). A comparative study of k-nearest neighbour, support vector machine and multi-layer perceptron for thalassemia screening. *Chemometrics and Intelligent Laboratory Systems*, *69*(1–2), 13–20.
5. Melin, P., Snchez, D., & Castillo, O. (2012). Genetic optimization of modular neural networks with fuzzy response integration for human recognition. *Information Sciences*, *197*, 1–19.
6. Guo, Z. X., Wong, W. K., & Li, M. (2012). Sparsely connected neural network-based time series forecasting. *Information Sciences*, *193*, 54–71.
7. Gardner, M. W., & Dorling, S. R. (1998). Artificial neural networks (the multilayer perceptron) a review of applications in the atmospheric sciences. *Atmospheric Environment*, *32*(14–15), 2627–2636.
8. Barakat, M., Lefebvre, D., Khalil, M., Druaux, F., & Mustapha, O. (2013). Parameter selection algorithm with self adaptive growing neural network classifier for diagnosis issues. *International Journal of Machine Learning and Cybernetics*, *4*(3), 217–233.
9. Csji, B. C. (2001). Approximation with artificial neural networks. *Faculty of Sciences, Etvs Lornd University, Hungary*, *24*, 48.
10. Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, *2*(5), 359–366.

11. Hush, D. R., & Horne, B. G. (1993). Progress in supervised neural networks. *IEEE Signal Processing Magazine*, *10*(1), 8–39.
12. Hagan, M. T., & Menhaj, M. B. (1994). Training feedforward networks with the Marquardt algorithm. *IEEE Transactions on Neural Networks*, *5*(6), 989–993.
13. Ng, S. C., Cheung, C. C., Leung, S. H., & Luk, A. (2003). Fast convergence for backpropagation network with magnified gradient function. In *2003 Proceedings of the international joint conference on neural networks* (Vol. 3, pp. 1903-1908). IEEE.
14. Lee, Y., Oh, S. H., & Kim, M. W. (1993). An analysis of premature saturation in back propagation learning. *Neural Networks*, *6*(5), 719–728.
15. Weir, M. K. (1991). A method for self-determination of adaptive learning rates in back propagation. *Neural Networks*, *4*(3), 371–379.
16. Yao, X. (1993). Evolutionary artificial neural networks. *International Journal of Neural Systems*, *4*(03), 203–222.
17. Gudise, V. G., & Venayagamoorthy, G. K. (2003). Comparison of particle swarm optimization and backpropagation as training algorithms for neural networks. In *Proceedings of the 2003 swarm intelligence symposium, SIS'03* (pp. 110–117). IEEE.
18. Yu, J., Wang, S., & Xi, L. (2008). Evolving artificial neural networks using an improved PSO and DPSO. *Neurocomputing*, *71*(4–6), 1054–1060.
19. Leung, F. H. F., Lam, H. K., Ling, S. H., & Tam, P. K. S. (2003). Tuning of the structure and parameters of a neural network using an improved genetic algorithm. *IEEE Transactions on Neural networks*, *14*(1), 79–88.
20. Mizuta, S., Sato, T., Lao, D., Ikeda, M., & Shimizu, T. (2001). Structure design of neural networks using genetic algorithms. *Complex Systems*, *13*(2), 161–176.
21. Derrac, J., Garca, S., Molina, D., & Herrera, F. (2011). A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation*, *1*(1), 3–18.
22. Mirjalili, S., & Lewis, A. (2013). S-shaped versus V-shaped transfer functions for binary particle swarm optimization. *Swarm and Evolutionary Computation*, *9*, 1–14.
23. Wilcoxon, F. (1945). Individual comparisons by ranking methods. *Biometrics Bulletin*, *1*(6), 80–83.
24. Mirjalili, S., Mirjalili, S. M., & Lewis, A. (2014). Let a biogeography-based optimizer train your multi-layer perceptron. *Information Sciences*, *269*, 188–209.