Hindawi
Scientific Programming
Volume 2018, Article ID 1435810, 9 pages
https://doi.org/10.1155/2018/1435810

Research Article

A Novel Multimean Particle Swarm Optimization Algorithm for Nonlinear Continuous Optimization: Application to Feed-Forward Neural Network Training

Mehmet Hacibeyoglu and Mohammed H. Ibrahim

Department of Computer Engineering, Necmettin Erbakan University, Konya, Turkey

Correspondence should be addressed to Mehmet Hacibeyoglu; hacibeyoglu@konya.edu.tr

Received 9 February 2018; Accepted 7 June 2018; Published 4 July 2018

Academic Editor: Harald Köstler

Multilayer feed-forward artificial neural networks are one of the most frequently used data mining methods for classification, recognition, and prediction problems. The classification accuracy of a multilayer feed-forward artificial neural networks is proportional to training. A well-trained multilayer feed-forward artificial neural networks can predict the class value of an unseen sample correctly if provided with the optimum weights. Determining the optimum weights is a nonlinear continuous optimization problem that can be solved with metaheuristic algorithms. In this paper, we propose a novel multimean particle swarm optimization algorithm for multilayer feed-forward artificial neural networks training. The proposed multimean particle swarm optimization algorithm searches the solution space more efficiently with multiple swarms and finds better solutions than particle swarm optimization. To evaluate the performance of the proposed multimean particle swarm optimization algorithm, experiments are conducted on ten benchmark datasets from the UCI repository and the obtained results are compared to the results of particle swarm optimization and other previous research in the literature. The analysis of the results demonstrated that the proposed multimean particle swarm optimization algorithm performed well and it can be adopted as a novel algorithm for multilayer feed-forward artificial neural networks training.

1. Introduction

Artificial neural networks (ANNs) are a vital component of artificial intelligence. Machine learning and cognitive sciences depend on ANNs to solve various complex nonlinear mapping relationships [1, 2]. ANNs can provide solutions to problems involving classification, prediction, optimization, and identification in various disciplines [3]. In general, ANNs training is conducted using the backpropagation (BP) algorithm. The BP algorithm determines the weights of ANNs by computing explicit gradients of error such as sum square error (SSE) [4]. However, ANNs trained with the gradient descent-based learning algorithm generally converge slowly and fall into local minima [5]. To get rid of this problem, metaheuristic algorithms can be used for ANNs training. Determining the optimum weights of ANNs is a nonlinear optimization problem and metaheuristic algorithms can solve this problem. For instance, Slowik used an adaptive differential algorithm with multiple trial vectors for ANNs training and increased the efficiency of the data classification when compared to evolutionary algorithms and BP [6]. Mohaghegi et al. trained a radial basis function neural network with BP and particle swarm optimization (PSO) algorithms for identification of a power system. They analyzed the experimental results of these two methods based on convergence speed and robustness. They found that PSO has several advantages in terms of both robustness and finding optimal weights of ANNs [7]. Montana and Davis utilized an improved genetic algorithm (GA) for training feed-forward ANNs. The experimental results showed that the improved GA enhanced the performance of feed-forward ANNs compared to the BP algorithm. In addition, the improved GA added more domain-specific knowledge into ANNs [8]. Malinak and Jaska implemented

evolutionary, gradient, and combined techniques for tuning the weights of ANNs. They partitioned the ANNs into two parts: the output layer versus the other layers and then they trained these two parts with different techniques. The combination of evolution strategies and least mean square algorithm showed promise according to their experimental results [9]. Carvalho and Ldermir applied PSO algorithm for the optimization of ANNs architectures and weights, aiming at better generalization performances through the creation of a compromise between low architectural complexity and low training errors. They used medical benchmarks to evaluate the performance of the proposed method and they compared the experimental results with the results of evolutionary programming and GA. Their proposed method showed better classification error percentage performance than the other algorithms [10]. Apart from PSO, researchers have also applied some swarm intelligence algorithms, such as an ant colony optimization (ACO) algorithm for ANNs training. Krzysztof et al. proposed an ACO variant for continuous optimization and chose the training of feed-forward ANNs for pattern recognition as a test case for the proposed algorithm. In addition, they hybridized the ACO algorithm with some classical gradient techniques such as the BP algorithm. The proposed algorithms were evaluated by applying them to classification problems from medical fields and comparing to the basic GA. The proposed algorithm showed better performance on medical datasets [11]. Liang Hu et al. proposed GA-optimized ANNs (GANNs) to solve a real-life problem multipath ultrasonic gas flowmeter. They decreased the error rate of the multipath ultrasonic flowmeter while detecting the flow rate of the complicated flow field with the GANNs. The GANNs demonstrated better outcomes than ANNs and made the implementation of ANNs faster and easier [12]. Researchers also used bat algorithm (BA) for training ANNs used to solve different real-life problems. The BA depends on the optimal solution in the velocity adjustment [13]. Ahmad et al. increased the training accuracy of a feed-forward multilayer perceptron network (MLP) with cuckoo search (CS). They tested the proposed algorithm on four benchmark classification problems. Furthermore, they compared the obtained results with well-known metaheuristics, such as PSO and guaranteed convergence particle swarm optimization (GCPSO). According to the experimental results CS provided better performance than PSO and GCPSO in all benchmark problems [14]. Bolaji et al. used the fireworks optimization algorithm (FWA) for ANN training and performed the experimental tests with UCI datasets. The experimental results were compared to the results obtained from the krill herd algorithm (KHA) [15], harmony search algorithm (HSA), and GA [16]. According to the experimental results, the FWA algorithm showed better classification performance [17]. Kider et al. used speed-constrained multiobjective particle swarm optimization (SMPSO) and NSGA-II optimization algorithms to determine the optimal input values (head type, feed rate, rotation speed, and chip depth) of force and surface roughness to minimize the output parameters that are applied to ANNs. The experimental results showed that SMPSO algorithm obtained better value than the NSGA-II algorithm [18].

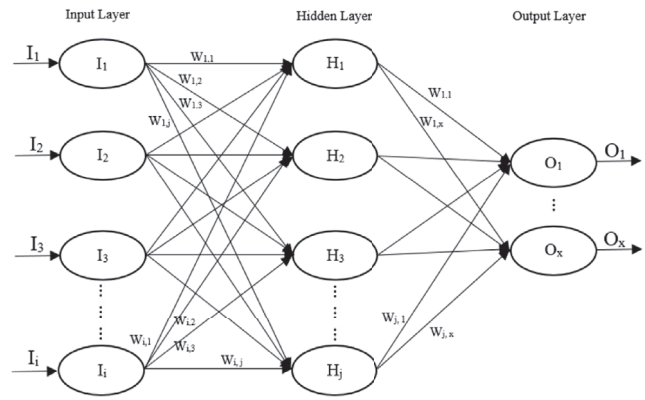

Figure 1: Structure of MLFNNs.

In this study, we propose a multimean particle swarm optimization (MMPSO) algorithm that makes novel use of PSO for solving continuous optimization problems. To assess the performance of the proposed MMPSO algorithm, it is applied on multilayer feed-forward artificial neural networks (MLFNNs) training and compared with PSO and other algorithms used in previous research. Analysis of the experimental results shows that the proposed MMPSO algorithm improved the classification accuracy of MLFNNs and showed a better performance than other algorithms.

The paper is organized as follows: in Section 2, MLFNNs are explained in detail. The used metaheuristic approaches PSO and the proposed MMPSO are clarified in Section 3. The application of metaheuristic algorithms to MLFNNs training is given in Section 4. In Section 5, the computational results are given and the paper is finalized with conclusions and future work.

2. Multilayer Feed-Forward Neural Networks

MLFNNs can be defined as a system by modeling the human brain functions. MLFNNs consist of artificial neural cells linked to each other in various forms and are usually organized in layers. They can be implemented as hardware in electronic circuits or as software in computers. In accordance with the brain information processing method, MLFNN has the ability to store and generalize information after a learning process [19]. Some successful networks can be created with a single layer, although most applications require networks that contain at least three layers: an input layer, hidden layer, and output layer. A network consisting of a single layer can only predict linear functions. MLFNNs remove the conflicting limits of single-layer systems with hidden layers located between the input and output layers [19]. Basically, all MLFNNs have a similar structure to that shown in Figure 1. In this structure, neurons in the input layer are used to get inputs, while neurons in the output layer are used to carry outputs and all neurons in the hidden layers are used to aid system training [20].

When we look at Figure 1, $I_1, I_2, I_3, \ldots, I_i$ represent the nodes in the input layer, $H_1, H_2, H_3, \ldots, H_j$ represent the nodes in the hidden layer, and $O_1, O_2, O_3, \ldots, O_x$ represent

Initialize all particles of the swarm with randomly generated position and velocity
Repeat
  For *each particle in the swarm*
    Calculate the fitness function
    Update the local best position of the particle
    Update the global best position of the swarm
  End for
  For *each particle in the swarm*
    Update the velocity and the position of the particle according to equations (4) and (5)
  End for
Until *(Stopping criteria met)*

Algorithm 1: The pseudocode of the PSO algorithm.

the nodes in the output layer. $W_{1,1}, W_{1,2}, W_{1,3}, \ldots, W_{1,j}$ are defined as weights and show the effect of the information received by a node. It is necessary to first calculate the outputs of the nodes in the hidden layers to calculate the output of the MLFNNs by the addition function (NET) and activation NET function (FNET) which are shown in (1) and (2), respectively [21].

$$NET_j = \sum_{i=1}^{m} I_i W_{ij} \tag{1}$$

$$FNET = \frac{1}{1 + e^{-NET}} \tag{2}$$

where m is the number of nodes connected to node j, I_i is the i_{th} node, and W_{ij} is the weight of the i_{th} node and the j_{th} node. The output value of the activation function is the output value of the node. This value can be either given to the outside world as the output of the MLFNNs or used in a different network again [22, 23]. The weights of the MLFNNs are updated by calculating the errors of the output obtained from the training process of the MLFNNs and the target output. These errors are known as SSE and are calculated according to (3). At the same time, the SSE can be used as an activation function of the metaheuristic algorithm.

$$SSE = \sum_{x=1}^{n} \left(O_x - T_x\right)^2 \tag{3}$$

where n represents the number of samples in a dataset, O_x is the output generated from the x_{th} input, and T_x is the target output of the x_{th} input.

3. Metaheuristic Algorithms

Metaheuristic algorithms are generally used in many areas for solving different problems such as optimization, scheduling, training of ANNs, fuzzy logic systems, and image processing [24]. In this study, we used two metaheuristic algorithms, the original PSO and MMPSO, which is a novel use of PSO for determining the optimum weights of MLFNNs.

3.1. Particle Swarm Optimization. PSO is a population-based metaheuristic optimization technique developed by Dr. Eberhart and Dr. Kennedy in 1995 [25]. It was inspired by the social interactions of birds with each other and with their environment. Potential solutions, which are called particles in the PSO, navigate the problem space by following the best available solutions. In PSO, each particle represents either a bird or a solution. All particles have a fitness value, which is found according to the fitness function, each particle position is updated using the particle's best solution (pbest) and the best solution of all particles (gbest), and these values are stored in the memory by [26, 27]

$$\begin{aligned} v_{id}^{t+1} &= w * v_{id}^{t} + c_1 r_1 \left(pbest_i - x_{id}^{t}\right) \\ &\quad + c_2 r_2 \left(gbest - x_{id}^{t}\right) \end{aligned} \tag{4}$$

$$x_{id}^{t+1} = x_{id}^{t} + v_{id}^{t+1} \tag{5}$$

where t is the number of iterations, w is the inertia weight, i is the index of a particle in a swarm, and d is the dimension of the problem. Each particle has a position and velocity for each dimension: x_{id}^{t} is the position of the ith particle in dimension d, v_{id}^{t} is the previous velocity of the ith particle in dimension d, $pbest_i$ is the best fitness value of the ith particle, $gbest$ is the best fitness value of all particles, c_1 and c_2 are two positive constants that represent the acceleration factor, and r_1 and r_2 are two random functions in the range [0, 1]. In (5), x_{id}^{t} is the old position of the ith particle and x_{id}^{t+1} is the new position of the ith particle in the swarm. The pseudocode of the PSO algorithm is given in Algorithm 1 [28].

3.2. Multimean Particle Swarm Optimization Algorithm. Multiswarm optimization (MSO) is a technique that is used to predict the optimal solution to nonlinear continuous optimization problems. The effectiveness and the productivity of many metaheuristic algorithms worsen as the dimensionality of the problem increases [29]. To overcome this problem, we proposed the MMPSO algorithm for obtaining the optimal solutions in a short time. The MMPSO algorithm is a multipopulation-based metaheuristic optimization algorithm developed from the PSO algorithm. In PSO, the velocity of each particle is updated with the parameters *pbest* and *gbest* according to (4) [26, 27]. The proposed MMPSO algorithm has multiple swarms. The velocity of each particle in each swarm is updated according to (6). In this equation,

```
Initialize all particles of all swarms with randomly generated position and velocity
Repeat
  For each swarm
    For each particle in the swarm
      Calculate the fitness function
      Update the local best of positions
      Update the global best position of the swarm
    End for
  Update the best position of all swarms
  End for
  For each swarm
    For each particle in the swarm
      Update the velocity and the position of the particle according to equations (6) and (5)
    End for
  End for
Until (Stopping criteria met)
```

Algorithm 2: The pseudocode of the proposed MMPSO algorithm.

there are two parameters that are different from (4), which are the mean *pbest* of all particles of that swarm (*mpbest*) and the best solution of all swarms (*gsbest*). This modification brings two advantages to PSO. Firstly, using the *mpbest* reduces the particles from going out of search space and reinforces the local search of each particle. Secondly, each particle takes into account not only the gbest of its own swarm but also the *gsbest* of all swarms, so that MMPSO algorithm gets closer to the optimum solution faster.

$$v_{id}^{t+1} = w * v_{id}^{t} + c_1 r_1 \left(mpbest_i - x_{id}^{t}\right) + c_2 r_2 \left(gbest - x_{id}^{t}\right) + \left(gsbest - x_{id}^{t}\right) \tag{6}$$

where t is the number of iterations, w is inertia weight, i is the index of a particle in a swarm, d is the dimension of the problem, each particle has a position and velocity for each dimension, x_{id}^{t} is the position of the ith particle in dimension d, v_{id}^{t} is the previous velocity of the ith particle in dimension d, $mpbest_i$ is the mean value of the particles in a swarm, $gbest$ is the best fitness value of a swarm, $gsbest$ is the best fitness value of all swarms, c_1 and c_2 are two positive constants representing acceleration factor, and r_1 and r_2 are two random numbers in the range $[0, 1]$. The pseudocode of the proposed MMPSO algorithm is given in Algorithm 2.

4. Application of Metaheuristic Algorithms to MLFNNs Training

Determining the optimal weights of MLFNNs is a nonlinear optimization problem so metaheuristic algorithms can be used for MLFNNs training. An application of metaheuristic algorithms to MLFNNs training is explained step by step in the following text.

Step 1 (preprocessing of the dataset). The normalization process, which is a data preprocessing technique, is applied to the dataset to be classified. Thus, the dataset becomes more regular and suitable for MLFNNs. This normalization process is done using a min–max normalization function, which is shown in [30]

$$x' = \frac{\left(x_i - \min\left(x_i\right)\right)}{\left(\max\left(x_i\right) - \min\left(x_i\right)\right)} \tag{7}$$

Step 2 (organization of the dataset for classification). In this study, the datasets are organized in two different ways for two different experiments. In the first experiment, 5-fold cross validation is used for comparing the proposed MMPSO algorithm to the PSO algorithm. In the second experiment, 80% training and 20% testing are used for comparing the proposed MMPSO algorithm to previous research in the literature.

Step 3 (modeling the structure of the MLFNNs). The numbers of inputs and the number of outputs are determined according to the characteristics of the dataset. The number of inputs is equal to the number of attributes of the dataset. Similarly, the number of outputs is equal to the number of classes of the dataset. The number of hidden layers is set to one for all problems and the number of nodes in the hidden layer is determined with GA, which is reported in a previous study by the authors [31].

Step 4 (determining the optimum weights of the MLFNNs with metaheuristic algorithms). A well-trained MLFNNs should have optimum weights and determining the optimum weights is a nonlinear optimization problem. Metaheuristic algorithms can be used to solve this problem owing to the structure of the metaheuristic algorithm. Generally, metaheuristic algorithms initialize with a random population. The fitness of each individual in the population is calculated according to the SSE of MLFNNs. The goal of the metaheuristic algorithms is to minimize the SSE. Therefore, metaheuristic algorithms search the problem space locally

Table 1: The characteristics of the ten datasets used.

Dataset	Number of Discrete Attributes	Number of Continuous Attributes	Classes	Samples
Lymphography	18	0	4	148
Iris	0	4	3	150
Wine	0	13	3	178
Glass	0	9	6	214
Shuttle-landing	6	0	2	253
Ionosphere	0	33	2	351
Balance-scale	4	0	3	625
Breast cancer	0	9	2	699
Diabetes	0	8	2	768
Thyroid	0	21	3	7200

and globally and update the global best solution. The metaheuristic algorithms run until the stopping criteria, such as the number of iterations or the error rate, are met.

Step 5 (testing the MLFNNs). In order to determine the performance of the MLFNNs training with metaheuristic algorithms, the classification accuracy is calculated according to (8) for each test dataset.

$$\begin{aligned} &\textit{Classification Accuracy } (CA) \\ &= \frac{\textit{Number of correctly classified instances}}{\textit{Number of instances in the test dataset}} \\ &* 100 \end{aligned} \tag{8}$$

5. Experimental Results

The application of the proposed MMPSO algorithm and the PSO algorithm to the MLFNNs is implemented using C# Microsoft Visual Studio Ultimate 2013. All experiments are carried out using a computer with an Intel Core i7 3840QM@2.00 GHz processor with 8 GB of memory with Microsoft Windows 8 operating system. Ten different benchmark datasets from the UCI repository [32] are used to evaluate the performance of three metaheuristics, and the characteristics of these datasets are shown in Table 1.

In general, the structure of MLFNNs is represented by *I-H-O* where *I* is the number of nodes in the input layer, *H* is the number of nodes in the hidden layer, and *O* is the number of nodes in the output layer. The number of weights of the MLFNNs with bias is calculated using (9), which represents the dimension size of the optimization problem at the same time [31].

$$\begin{aligned} &\textit{The number of weights of MLFNNs} \\ &= (I * H) + (H * O) + H + O \end{aligned} \tag{9}$$

Furthermore, to determine the optimum structure of the MLFNNs is an optimization problem and the classification accuracy of the MLFNNs is directly affected by it (Ibrahim, Jihad, and Kamal, 2017). The determined structures of the MLFNNs according to the ten benchmark datasets which are shown in Table 2.

Table 2: The structure of the MLFNNs.

Dataset	I	H	O	Number of Weights
Lymphography	18	15	4	349
Iris	4	5	3	43
Wine	13	10	3	173
Glass	9	12	6	198
Shuttle-landing	6	8	2	74
Ionosphere	33	4	2	146
Balance-scale	4	5	3	43
Breast cancer	9	8	2	98
Diabetes	8	6	2	68
Thyroid	21	12	3	303

For the proposed MMPSO algorithm and the PSO algorithm, the acceleration constants c_1 and c_2 are set to 1.49 [33], the random numbers r_1 and r_2 are generated in the range $[0, 1]$, and the number of particles of a swarm is set to 20. For the proposed MMPSO algorithm, the number of a swarm is set to three. For the initial population of the proposed MMPSO algorithm and the PSO algorithm, the weights of the MLFNNs are generated at random numbers in the range $[-10, 10]$. All these experimental parameters are determined empirically. The sigmoid activation function is used in the hidden layer and the output layer of the MLFNNs for training and testing. The maximum number of iterations is used as the stopping criterion. The classification process is applied with 5-fold cross validation, which estimates the mean of the SSEs obtained on five different testing subsets. The results of the 5-fold cross validation experiment of the proposed MMPSO algorithm and the PSO algorithm are shown in Table 3.

When the results of the 5-fold cross validation experiment in Table 3 are investigated, the MLFNNs trained with the proposed MMPSO algorithm obtained better SSE, training AC, and testing AC results than the MLFNNs trained with the PSO algorithm for all datasets. As a result, these experimental

Table 3: The results of 5-fold cross validation experiment of the proposed MMPSO and PSO.

Dataset	Algorithm	SSE	Training CA (%)	Testing CA (%)
Lymphography	PSO	23.46	89.32	78.67
	MMPSO	**13.87**	**91.49**	**83.67**
Iris	PSO	6.72	91.00	90.67
	MMPSO	**1.35**	**97.33**	**91.67**
Wine	PSO	3.86	92.25	87.22
	MMPSO	**0.39**	**99.58**	**95.56**
Glass	PSO	83.60	66.35	55.84
	MMPSO	**62.36**	**69.21**	**58.14**
Shuttle-landing	PSO	2.07	**99.51**	94.12
	MMPSO	**1.79**	**99.51**	**96.08**
Ionosphere	PSO	24.77	93.07	85.35
	MMPSO	**20.16**	**95.79**	**88.73**
Balance-scale	PSO	58.21	90.20	86.56
	MMPSO	**35.85**	**90.24**	**89.12**
Breast cancer	PSO	73.80	98.38	95.14
	MMPSO	**59.43**	**98.57**	**96.29**
Diabetes	PSO	202.39	76.78	72.08
	MMPSO	**182.54**	**77.55**	**78.05**
Thyroid	PSO	528.55	92.95	92.83
	MMPSO	**373.63**	**93.66**	**94.01**

results show that the proposed MMPSO algorithm achieved good performance in the training process of the MLFNNs in accordance with the PSO algorithm.

Additional advantages of the proposed MMPSO algorithm are that it searches the global space more efficiently and convergences the optimum results more rapidly. To provide these advantages, the proposed MMPSO algorithm must minimize the fitness function SSE more rapidly than the PSO algorithm in the training process. The minimization of SSEs according to the iteration number in the training process is given in Figure 2 for the lymphography, ionosphere, glass and diabetes datasets. For the lymphography, ionosphere, and glass datasets, the proposed MMPSO algorithm better minimized the SSE from the initial iteration to the end iteration. For the diabetes dataset, the proposed MMPSO algorithm searches the global space more efficiently after the 90th iteration. Furthermore, the proposed MMPSO algorithm provides better SSE for the initial iteration in all datasets.

Furthermore, for analyzing the computational complexities of the proposed MMPSO and the PSO algorithms, the CPU running times of each algorithm were measured by Microsoft Process Explorer utility in seconds and are given in Table 4.

As shown in Table 4, the running time of the proposed MMPSO algorithm is shorter than that for PSO for all datasets. In addition, the proposed MMPSO algorithm is suitable for parallel implementation and the runtime of the MMPSO algorithm can be reduced to a much shorter time with parallel programming.

Table 4: CPU running times of the PSO and MMPSO algorithms in seconds.

Dataset	PSO	MMPSO
Lymphography	6,48	**5,16**
Iris	2,16	**1,37**
Wine	4,37	**3,37**
Glass	7,59	**5,17**
Shuttle-landing	4,34	**3,19**
Ionosphere	6,52	**6,26**
Balance-scale	8,03	**6,32**
Breast cancer	14,44	**11,05**
Diabetes	11,29	**9,36**
Thyroid	479,08	**396,35**

Finally, the performance of the proposed MMPSO algorithm is compared with the performance reported in the literature for the HSA [16], KHA, GA [15], and the fireworks algorithm (FWA) [17] which split the data into 80% training and 20% testing, for six datasets. In order to make this comparison under the same conditions, six datasets are split into 80% training and 20% testing for this experiment. The proposed MMPSO algorithm is executed ten times and the best results are selected. The proposed MMPSO algorithm was executed with the same parameters as described in the previous experiment. The results of the 80% training and 20% testing experiment of MMPSO algorithm and the literature reports for six datasets are shown in Table 5.

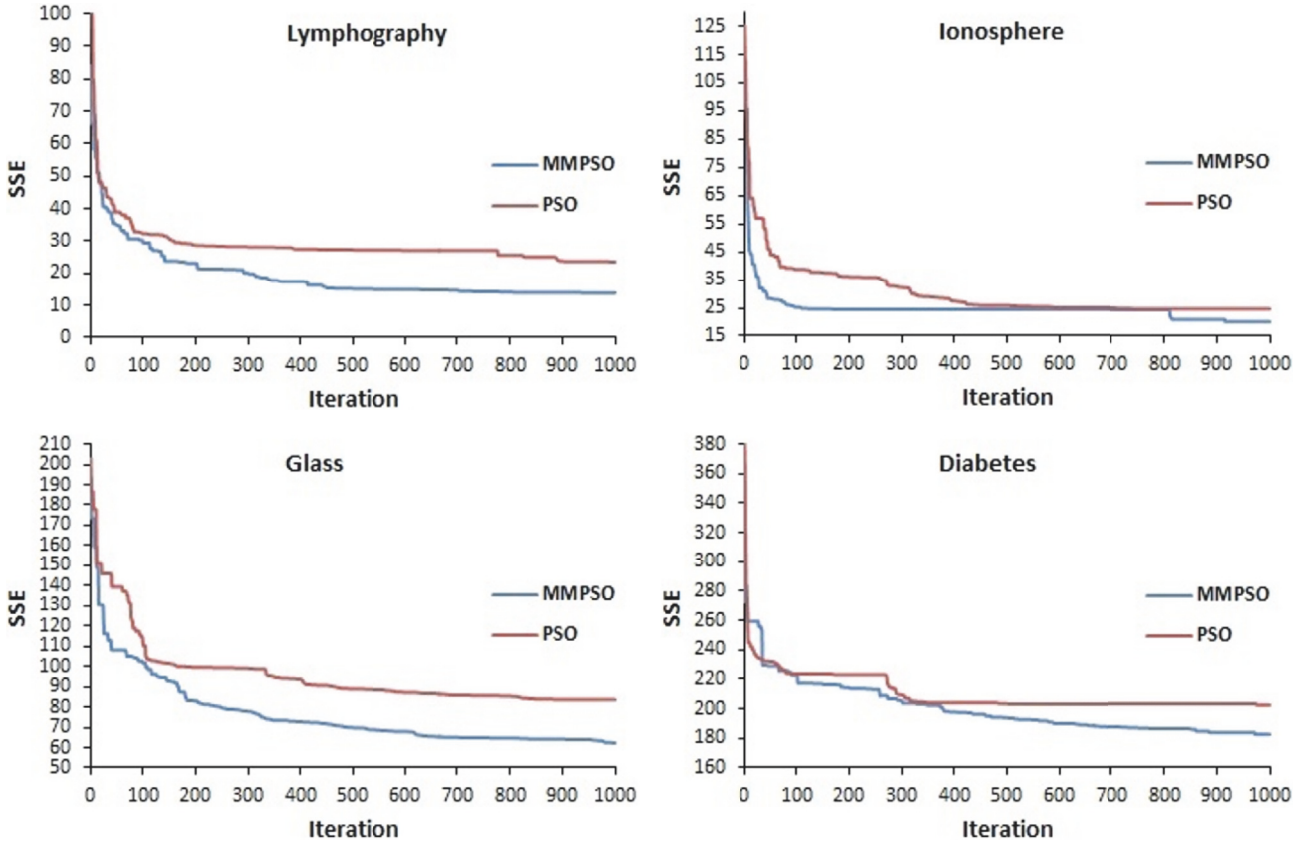

Figure 2: The minimization of the SSE according to the iteration in the training.

When the results of the 80% training and 20% testing experiment in Table 5 are analyzed, it is clear that the proposed MMPSO algorithm yields better results than the other four metaheuristics according to SSE, training CA and testing CA values for the iris, diabetes, and thyroid datasets. Although the proposed MMPSO algorithm obtained better SSE and training CA results than other metaheuristic algorithms, it could not obtain the best testing CA result for the ionosphere and breast cancer datasets. For the glass dataset, the proposed MMPSO algorithm obtained the best result only for the training CA. In summary, when looking at the results of the comparison in Table 5, the proposed MMPSO algorithm performed better classification results than other algorithms in general.

6. Conclusion and Future Work

In this paper, a novel MMPSO algorithm is proposed for MLFNNs training. The proposed MMPSO algorithm based on MSO technique has two advantages according to the PSO algorithm. Firstly, the proposed MMPSO algorithm strengthens the particles to carry out a local search in the search space range. Secondly, the proposed MMPSO algorithm has multiple swarms and takes into account both the best solution of each swarm and the best solution of all swarms and thus it gets closer to the optimum solution. To evaluate the performance of the proposed MMPSO algorithm experiments were conducted on ten benchmark datasets from the UCI repository. According to the experimental results, the proposed MMPSO algorithm yielded better performance than PSO for all datasets. Furthermore, the obtained experimental results were compared with the previous researches in the literature for six datasets. According to this comparison, the proposed MMPSO algorithm showed a competitive advantage over the reported algorithms. In conclusion, the proposed MMPSO algorithm showed good performance and can be adopted as a novel algorithm for MLFNNs training.

For future work, the proposed MMPSO algorithm will be used by intelligent systems to solve complex real-life optimization problems in various fields such as: design, identification, operational development, planning, and schedul ing.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Table 5: The results of the 80% training and 20% testing experiment for six datasets.

Dataset	Algorithm	SSE	Training CA (%)	Testing CA (%)
Iris	MMPSO	**0.31**	**100**	**100**
	FWA	0.52	**100**	**100**
	KHA	21.28	99.59	**100**
	HS	18.00	98.33	96.67
	GA	96.00	90.00	90.00
Glass	MMPSO	47.56	**78.36**	62.79
	FWA	94.33	61.99	60.47
	KHA	**41.21**	58.79	58.14
	HS	355.85	70.12	**72.09**
	GA	544.00	57.89	67.44
Ionosphere	MMPSO	**18.91**	**99.28**	91.54
	FWA	25.28	95.71	90.14
	KHA	31.0	89.00	91.43
	HS	106.4	95.00	**94.37**
	GA	152	93.21	**94.37**
Breast cancer	MMPSO	**47.00**	**99.46**	97.85
	FWA	66.11	93.92	96.43
	KHA	-	-	-
	HS	126.37	-	**100**
	GA	172	-	98.57
Diabetes	MMPSO	**166.21**	**81.27**	**79.87**
	FWA	267.20	65.96	66.88
	KHA	-	-	-
	HS	856	-	77.27
	GA	1108	-	**79.87**
Thyroid	MMPSO	**237.07**	**95.37**	**94.19**
	FWA	749.11	93.21	93.82
	KHA	320.3	94.81	92.90
	HS	3146.4	93.06	92.78
	GA	3416.0	92.58	92.57

Acknowledgments

This paper is supported by BAP Coordination Office of Necmettin Erbakan University.

References

[1] E. Çelik, Y. Uzun, E. Kurt, N. Öztürk, and N. Topaloğlu, "A neural network design for the estimation of nonlinear behavior of a magnetically-excited Piezoelectric Harvester," *Journal of Electronic Materials*, pp. 1–9, 2018.

[2] V. Kecman, *Learning and Soft Computing: Support Vector Machines, Neural Networks, and Fuzzy Logic Models*, MIT press, Cambridge, Mass, USA, 2001.

[3] D. Kriesel, *A brief Introduction on Neural Networks*, 2007.

[4] M. Alhamdoosh and D. Wang, "Fast decorrelated neural network ensembles with random weights," *Information Sciences*, vol. 264, pp. 104–117, 2014.

[5] K. Kapanova, I. Dimov, and J. Sellier, "A genetic approach to automatic neural network architecture optimization," *Neural Computing and Applications*, vol. 29, no. 5, pp. 1481–1492, 2018.

[6] A. Slowik, "Application of an adaptive differential evolution algorithm with multiple trial vectors to artificial neural network training," *IEEE Transactions on Industrial Electronics*, vol. 58, no. 8, pp. 3160–3167, 2011.

[7] S. Mohaghegi, Y. Del Valle, G. K. Venayagamoorthy, and R. G. Harley, "A comparison of PSO and backpropagation for training RBF neural networks for identification of a power system with statcom," in *Proceedings of the 2005 IEEE Swarm Intelligence Symposium, SIS 2005*, pp. 391–394, Pasadena, Calif, USA, June 2005.

[8] D. J. Montana and L. Davis, "Training feedforward neural networks using genetic algorithms," in *Proceedings of the 11th international joint conference on Artificial intelligence*, pp. 762–767, Detroit, Mich, USA, 1989.

[9] P. Malinak and R. Jaksa, "Simultaneous gradient and evolutionary neural network weights adaptation methods," in *Proceedings of the 2007 IEEE Congress on Evolutionary Computation*, Singapore, Singapore, 2007.

[10] M. Carvalho and T. B. Ludermir, "Particle swarm optimization of neural network architectures and weights," in *Proceedings of the 7th International Conference on Hybrid Intelligent Systems*

(HIS '07), pp. 336–339, IEEE, Kaiserlautern, Germany, September 2007.

[11] K. Socha and C. Blum, "An ant colony optimization algorithm for continuous optimization: application to feed-forward neural network training," *Neural Computing and Applications*, vol. 16, no. 3, pp. 235–247, 2007.

[12] L. Hu, L. Qin, K. Mao, W. Chen, and X. Fu, "Optimization of neural network by genetic algorithm for flowrate determination in multipath ultrasonic gas flowmeter," *IEEE Sensors Journal*, vol. 16, no. 5, pp. 1158–1167, 2016.

[13] N. S. Jaddi, S. Abdullah, and A. R. Hamdan, "Optimization of neural network model using modified bat-inspired algorithm," *Applied Soft Computing*, vol. 37, pp. 71–86, 2015.

[14] A. A. Kawam and N. Mansour, "Metaheuristic optimization algorithms for training artificial neural networks," in *Proceedings of the International Journal of Computer and Information Technology*, vol. 1, pp. 156–161, 2012.

[15] P. A. Kowalski, S. Lukasik, and S. Łukasik, "Training neural networks with krill herd algorithm," *Neural Processing Letters*, vol. 44, no. 1, pp. 17-10, 2016.

[16] A. Kattan, R. Abdullah, and R. A. Salam, "Harmony search based supervised training of artificial neural networks," in *Proceedings of the 2010 International Conference on Intelligent Systems, Modelling and Simulation*, Liverpool, UK, 2010.

[17] A. L. a. Bolaji, A. A. Ahmad, and P. B. Shola, "Training of neural network for pattern classification using fireworks algorithm," *International Journal of System Assurance Engineering and Management*, vol. 9, no. 1, pp. 208–215, 2018.

[18] M. S. Kider, H. E. Kocer, I. Asilturk, M. H. Ibrahim, and T. Sag, "Comparative optimization of cutting parameters in turning process," *Journal of Selcuk University Natural and Applied Science*, vol. 4, no. 4, pp. 54–64, 2016.

[19] J. Han, J. Pei, and M. Kamber, *Data Mining: Concepts and Techniques*, Elsevier, 2011.

[20] N. Allahverdi, *Expert Systems: An Application of Artificial Intelligence*, 2002.

[21] V. E. Ismailov, "On the approximation by neural networks with bounded number of neurons in hidden layers," *Journal of Mathematical Analysis and Applications*, vol. 417, no. 2, pp. 963–969, 2014.

[22] H. B. Demuth, M. H. Beale, O. De Jess, and M. T. Hagan, *Neural Network Design*, Martin Hagan, 2014.

[23] S. Haykin, *Neural Networks: A Comprehensive Foundation*, Prentice-Hall, Upper Saddle River, NJ, USA, 2nd edition, 2004.

[24] T. Weise, *Global Optimization Algorithms-Theory and Application*, vol. 2, 2009.

[25] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the ICNN'95 - International Conference on Neural Networks*, Perth, WA, Australia, 1995.

[26] S. Das, A. Abraham, and A. Konar, "Particle swarm optimization and differential evolution algorithms: technical analysis, applications and hybridization perspectives," in *Advances of Computational Intelligence in Industrial Systems*, Studies in Computational Intelligence, pp. 1–38, Springer, Berlin, Germany, 2008.

[27] Y. Shi and R. Eberhart, "Empirical study of particle swarm optimization," in *Proceedings of the IEEE International Congress Evolutionary Computation*, pp. 1945–1950, Washington, Wash, USA, 1999.

[28] R. Poli, J. Kennedy, and T. Blackwell, "Particle swarm optimization," *Swarm Intelligence*, vol. 1, no. 1, pp. 33–57, 2007.

[29] Ş. Gülcü and H. Kodaz, "A novel parallel multi-swarm algorithm based on comprehensive learning particle swarm optimization," *Engineering Applications of Artificial Intelligence*, vol. 45, pp. 33–45, 2015.

[30] C. Saranya and G. Manikandan, "A study on normalization techniques for privacy preserving data mining," *International Journal of Engineering and Technology*, vol. 5, no. 3, pp. 2701–2704, 2013.

[31] V. Tongur, M. Hacibeyoglu, and M. H. Ibrahim, "Determining optimal structure for multilayer feed-forward artificial neural networks using genetic algorithm," in *Proceedings of the 3rd International Conference on Engineering and Natural Science*, Bangkok, Thailand, 2017.

[32] C. Merz, "UCI repository of machine learning databases," 1996, http://www.ics.uci.edu/~mlearn/MLRepository.

[33] L.-P. Zhang, H.-J. Yu, and S.-X. Hu, "Optimal choice of parameters for particle swarm optimization," *Journal of Zhejiang University SCIENCE A*, vol. 6, no. 6, pp. 528–534, 2005.