Optimal Artificial Neural Network using Particle Swarm Optimization

M. N. Alam[1] , *Amin* Sallem[2] , *Pedro* Pereira[3] , *Benhala* Bachir[4] , *Nouri* Masmoudi[5]

[1] NIT Warangal, Warangal, India
[2] LETI-ENIS, University of Sfax, Sfax, Tunisia
[3] NOVA, University of Lisbon, Portugal
[4] FSDM, University SMBA, Fez, Morocco
[5] LETI-ENIS, University of Sfax, Sfax, Tunisia

Abstract. Artificial neuron networks (ANNs) are widely used for data analyticS in broad areas of engineering applications and commercial services. The ANN has one to two hidden layers. In advanced ANN, multiple-layer ANN is used where the network extracts different features until it can recognize what it is looking for through deep learning approaches. Usually, a backpropagation algorithm is used to train the network and fix weights and biases associated with each network neuron. This paper proposes a particle swarm optimization (PSO) based algorithm for training ANN for better performance and accuracy. Two types of ANN models and their training using PSO have been developed. The performance of the developed models has been analyzed on a standard dataset. Also, the effectiveness and suitability of the developed approach have been demonstrated through statistics of the obtained results.

Index Terms. Activation function, hidden layer, input layer, output layer, transfer function, artificial neural networks, particle swarm optimization.

1. INTRODUCTION

Artificial neuron networks (ANNs) are one of the main tools used in machine learning [1]. In most engineering streams, ANN is used for curve fitting, classification and other data analytics. This tool is very effective in analytics, even on a complicated dataset and identifies relationships between input features and targets. Once the network is trained, it can predict output accurately and faster without any difficulty. This tool is widely used for online applications because of its fast response and straightforward concept [2]. In most key engineering fields, ANN has proven well accepted and adopted the tool for analysis, design and operation-related applications.

The ANN replicate human learning processes and brain-inspired systems [3]. Its structure and functions are similar to biological neural networks inside human brains. The flow of information through ANN affects the structure because a neural network changes - or learns, in a sense - based on that input and output [4]. In this way, complex relationships

between inputs and outputs are identified through ANN. Usually, ANN has three layers, namely, the input layer, the hidden layer and the output layer. Hidden and output layers have neurons and are associated with weights and biases. These weights and biases are changed during training when inputs are passed through the input layer to give a targeted output in the ANN. These weights and biases are fixed during the training phase when inputs whose output is known are used. This training is done using different algorithms [5]. The backpropagation algorithm is a traditional method widely used for training ANN. Once training is done, the network gets ready for application.

The backpropagation algorithm is one of the most widely used algorithms for training ANNs [6]–[9]. This technique allows networks to adjust their hidden layers of neurons in situations where the outcome doesn't match what the creator hopes for. Neurons of one layer are connected to the neurons of the other layer through a connection link with a weighting factor. All neurons also have a bias to scale their output. All the training algorithms, including backpropagation, try to find suitable values of weights and biases of the ANN so that the required output can be obtained for a given input. The relationship between the input and output of ANN is very complex. Hence, training ANN properly is a challenging task requiring powerful algorithms. The backpropagation algorithm is not inspired by any biological process of neuron structure of the brain but is very efficient in training the ANN in most field applications. Despite being very efficient, the backpropagation algorithm gets trapped in local optima, and the obtained weights and biases do not always give a proper prediction for the unknown [10]. As a result, it is required to retrain the ANN repeatedly to get a better prediction for unknown inputs. Therefore, there is a scope for developing a more efficient algorithm to train ANN in a much better way so that the prediction of unknown inputs can be improved.

To overcome the shortcomings of ANN training algorithms, this book chapter discusses metaheuristic particle swarm optimization-based optimal ANN [11]–[16]. In this chapter, the training problem of ANN is formulated as an optimization problem and solved using PSO based algorithm. Here, weights and biases of ANN are considered the decision variables and root-mean-square errors are regarded as the objective to be minimized. Also, weight and biases are scaled within [-1.5, 1.5], and inputs and outputs given to the ANN are scaled within the same range. Additionally, four different ANN models and PSO codes for training them in the MATLAB environment [17] have been discussed in detail. The Single and multiple input-output ANN models with single and multiple hidden layers have been considered. With the developed codes, all parameters tuning of PSO along with changing transfer functions of various layers of ANN and the number of neurons can vary. Therefore, the readers can adjust these parameters of ANN and PSO for their problems to get even better results.

The rest of the paper is organized as follows. Section II gives details of ANN structures, their mathematical modelling and problem formulation of ANN. Section III gives details of the ANN-PSO algorithm devised for training ANN using PSO. Results and discussion are provided in Section IV. Finally, the conclusions are presented in Section V.

2. ANN STRUCTURE AND MATHEMATICAL MODELING

The ANN structure and mathematical representation provide a model that can relate highly complicated input and output datasets. Their performances are exceptionally good for very complex data sets, which usually are hard to predict using conventional mathematical modelling [6]–[9].

2.1 Working Principle of ANN

The ANN is a network of neurons connected through weights and biases in a fixed structure similar to human brain neural systems. A typical ANN model is shown in Fig. 1. This ANN model consists of one input layer, a hidden layer, and an output layer. Hidden and output layers have neurons. These neurons are interconnected through weights and have a bias associated with each. The hidden layer's neurons are connected through each input data point of the input data sample. These can be observed in Fig 1.

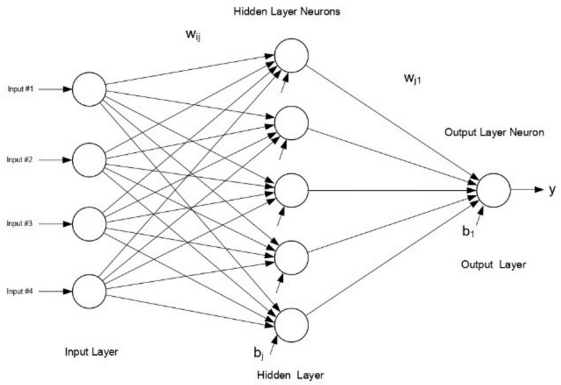

Fig. 1. A typical Structure of Artificial Neural Network.

Fig. 1 shows that four inputs produce one output while passing through ANN, having five neurons in the hidden layer. Mathematically, the output of the above ANN structure can be expressed as:

$$y = f(x_1,x_2,x_3,x_4,\mathrm{W},\mathrm{b}) \quad (1)$$

where x1,x2,x3 and x4 are independent input features or variables, and y is the target or output of the given dataset. Here, matrix W represents the weights associated with the input and hidden layers and with the hidden layer and the output layer neurons of the ANN and matrix b represents biases associated with each neuron of the ANN structure. There is one activation function associated with each neuron. Usually, the same activation function is used in all hidden layer neurons. Also, the output layer neurons have the activation function in all the output neurons but differ from that of the hidden layer activation function. The activation function is called the transfer function. The sigmoid or hyperbolic tangent function is commonly used in the hidden layer neurons, and the linearized transfer function is used in the output layer neurons.

The number of hidden layer neurons is independent of input size, whereas the number of output layer neurons is the same as the number of output variables in the dataset.

The output of a neuron is calculated using the inputs, weights and bias of the network through the transfer function. Fig. 2 shows the input-output relationship of the dataset. Mathematically, the output of jth neuron are expressed follows:

$$z = g\left(b_j + \sum_{i=1}^{n} w_{ij}x_i\right) \quad (2)$$

where g is the activation or transfer function of the neuron and n is the total number of inputs to the neuron. The most commonly used transfer functions are shown in Fig 3.

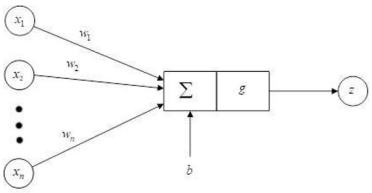

Fig. 2. Activation function.

Each input is multiplied by a weighting factor and added with a bias factor at each hidden neuron to produce output by that hidden layer neuron. The output of each hidden layer neuron is the input to the subsequent layer neurons. In a two-layer ANN structure,

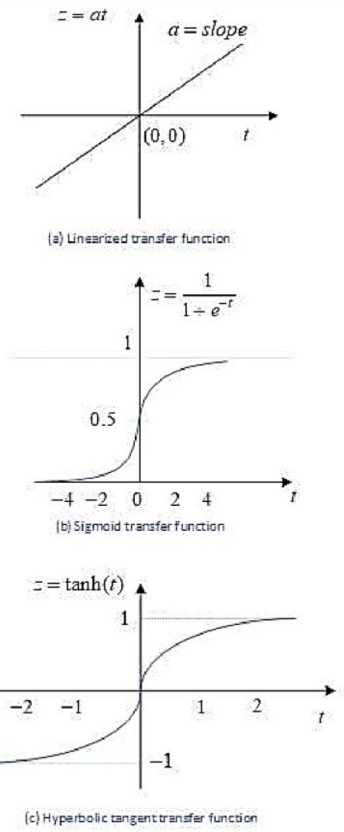

Fig. 3. Commonly used transfer function in ANN: (a) Linearized transfer function, (b) Sigmoid transfer function, and (c) Hyperbolic tangent transfer function.

as shown in Fig. 1, the output of each hidden layer neuron is the input to the output layer neuron. Again, the output of each hidden layer neuron is multiplied by another weighting factor and added with another bias to give the final output of the ANN model. The final output y (of Fig. 1) depends on weights and biases associated with various network neurons. The primary task in the ANN model is to find suitable values of weights and biases of the network for a given dataset. These weights and biases are obtained

through training the structure so that for a known input-output set, the error in the output (which we call target) and the calculated output of the structure should be the least.

2.2 Training ANN Model

Once the ANN structure is formed, the next task is to train the network. The training task is associated with finding the optimum values of ANN parameters, which are weights and biases of the structure. Several techniques are used to find the suitable weights and biases of the ANN. In this work, the PSO algorithm is devised for training ANN.

Here, the training problem is formulated as an optimization problem to minimize the root mean square error of the target and actual output of the ANN subject to keeping the values of weights and biases within an appropriate range. This range can be different for different problems.

Mathematically, the ANN training problem is formulated as follows:

$$OF = \min \sum_{k=1}^{N} \left(target(k) - output(k)\right)^2 \tag{3}$$

Subject to

$$w_{min} \leq w_{ijk} \leq w_{max} \tag{4}$$

$$w_{min} \leq b_{kj} \leq w_{max} \tag{5}$$

where target(k) is the known target value of a given input sample and output(k) is the actual output value of the ANN model for data sample k. Here, this optimization problem aims to find the weights and biases of the ANN to minimise the error mentioned above between the target and output.

2.3 Various ANN Models

In this paper, we discuss two different types of ANN models based on their structure. The two structures are a) a single hidden layer and a single output, and b) Multiple hidden layers and a single output layer. Typically, the output layer is just one; however, hidden layers may be more than one in the ANN model.

- One Hedden Layer with One Output (H1O1)
- Two Hedden Layers with One Output (H2O1)

2.3.1 One Hedden Layer with One Output (H1O1)

The ANN structure with one hidden layer and one output for an n input sample is shown in Fig. 4. This network poses m neurons in the hidden layer and only one output neuron. This ANN structure is given for a relationship with multiple inputs and one output dataset. Most of the problems come under this category.

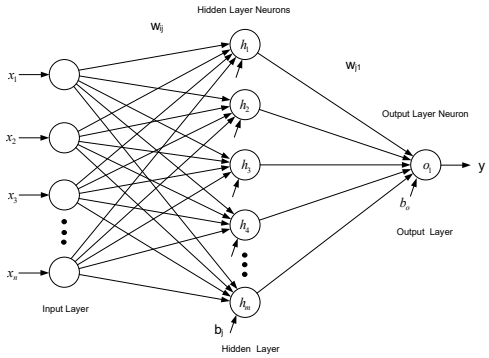

Fig. 4. ANN model of one hidden layer with one output.

Let us assume that the hidden layer's neurons follow a hyperbolic tangent activation function, whereas the only output layer neuron follows a linear activation function. Then, the final output of this ANN model can be expressed as:

$$y = b_o + \sum_{j=1}^{m} \tanh\left(b_j + \sum_{i=1}^{n} w_{ij} x_i\right) w_{j,1} \tag{6}$$

The total number of variables (i.e., weights and biases) of this ANN model can be expressed as:

$$N = n \times m + 2m + 1 \tag{7}$$

Out of N number of variables n × m + m are weighting factors, and the remaining m + 1 are biases.

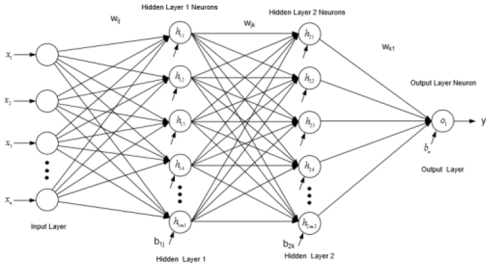

Fig. 5. ANN model of two hidden layers with one output.

2.3.1 Two Hedden Layers with One Output (H2O1):

The ANN structure with two hidden layers and one output for an n input sample is shown in Fig. 5. This network poses m1 and m2 numbers of neurons in the first and second hidden layers, respectively and only one output neuron.

The output of this ANN model considers the same activation functions (hyperbolic tangent for hidden layers and linear for the output layer). The final output of H2O1 is expressed as:

$$y = b_o + \sum_{k=1}^{m_2} \tanh\left(b_{2k} + \sum_{j=1}^{m_1} \tanh\left(b_{1j} + \sum_{i=1}^{n} w_{ij} x_i\right) w_{jk}\right) w_{k1} \tag{8}$$

In this model, the total number of weight variables is n × m1 + m1 × m2 + m2, and the total number of bias variables is m1 + m2 + 1. Therefore, the total number of variables (i.e., weights and biases) can be expressed as:

$$N = n \times m_1 + m_1 \times m_2 + m_1 + 2m_2 + 1 \quad (9)$$

Once the network is formed as per the given input-output dataset, the next task is to train the ANN model to find suitable values of weights and biases so that the network can predict any unknown input of the same data type as has been used in training.

3. DEVELOPED ANN-PSO ALGORITHM

This work has obtained optimum network training through PSO. The details about the PSO can be found in [11]–[15]. The PSO algorithm used in this work is well explained in [16].

Particle Swarm Optimization: Basics

Particle swarm optimization is one of the most popular nature-inspired metaheuristic optimization algorithms developed by James Kennedy and Russell Eberhart in 1995 [11]. Since its development, many variates have also been developed for solving practical issues related to optimization [18]–[20]. Recently, PSO has emerged as a promising algorithm for solving various optimization problems in science and engineering [21]–
[30].

The PSO is inspired by social and cooperative behaviour displayed by various species to fill their needs in the search space. The PSO algorithm uses two mathematical equations, guided by personal experience (Pbest), overall experience (Gbest) and the present movement of the particles in the search space to decide their successive positions. Further, the experiences are accelerated by two factors, c1 and c2, and two uniformly generated random numbers in [0,1]. The present movement is weighted by a factor called inertia weight w varying between [wmin,wmax].

The initial population (swarm) of size N and dimension D is denoted as X = [X1,X2,...,XN]T , where T denotes the transpose operator. Each individual (particle) Xi(i = 1,2,...,N) is given as Xi = [Xi1,Xi2,...,XiD]. Also, the initial velocity of the population is denoted as V = [V1,V2,...,VN]T . Thus, the velocity of each particle Xi(i = 1,2,...,N) is given as Vi = [Vi1,Vi2,...,ViD]. The index i varies from 1 to N, whereas the index j varies from 1 to D. Mathematically, the PSO updating mechanism is expressed using the following two sets of equations:

$$V_{ijk+1} = wV_{ijk} + c_1 r_1 (P_{ijk} - X_{ijk}) + c_2 r_2 (G_{kj} - X_{ijk}) \quad (10)$$

$$X_{ijk+1} = X_{ijk} + V_{ijk+1} \quad (11)$$

where Pijk represents the personal best j-th component of i-th individual, whereas Gkj represents j-th component of the best individual of the population up to iteration k. The inertia factor w varying between [wmin,wmax] are defined as follows:

$$w = w_{max} - (w_{max} - w_{min}) \times k/Maxite \quad (12)$$

where Maxite is the maximum iteration fixed, and k is the current iteration in operation.

B. Particle Swarm Optimization: Algorithm

The steps of the ANN-PSO algorithm are given as Algorithm 1 in this section.

1.1 C. Steps of ANN-PSO Algorithm

The following seven steps are followed in the ANN-PSO algorithm for any given dataset.

1) Collect data
2) Create the network
3) Configure the network
4) Initialize the weights and biases
5) Train the network using PSO
6) Validate the network
7) Use the network

The developed approach is validated using chemical dataset. This is a fitting problem whose complete dataset is given in MATLAB [17]. This dataset has eight independent inputs to produce a single output. There are 498 data vectors in the dataset, which are challenging to include in the paper because of page limitations. A few data are given in Table 1.

Table 1. Dataset for Eight Features and One Target

Sl. No.	Input								Target
	*x*1	*x*2	*x*3	*x*4	*x*5	*x*6	*x*7	*x*8	*y*1
1	157	9596	4714	376	2.58	407	564	510354	514
2	155	9487	5049	381	2.28	411	567	504718	516
3	154	9551	5070	374	2.98	406	563	456972	512
4	154	9637	5087	382	2.57	408	565	512311	516
5	152	9486	5065	380	3.04	408	567	489312	515
6	153	9633	5319	377	2.72	408	565	495420	513
7	151	9637	4987	379	3.07	408	563	474552	512
.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.
498	164	9872	7252	386	2.08	420	589	537719	520

Algorithm 1 ANN-PSO

1: Read problem datasets

2: Set parameters such as N, D, c1, c2, wmin, wmax

3: Initialize population of particles having positions X where each vector of X represents the weight and biases of the network.

4: Initialize velocity of the population V

5: Set iteration count k = 1 and Maxite

6: Calculate fitness of each particle Fik = f(Xki),∀i and identify the best particle b among the population X

7: Select personal and global bests

$$P_{ijk} = X_{ijk}, \forall ij \text{ and } G^k = \mathbf{X}_b^k$$

8: Update inertia weight factor

$$w = w_{max} - (w_{max} - w_{min}) \times k/Maxite$$

9: Update velocity and position of particles

$$V_{ij}^{k+1} = wV_{ij}^{k} + c_1 r_1 (P_{ij}^{k} - X_{ij}^{k}) + c_2 r_2 (G_j^{k} - X_{ij}^{k})$$
$$X_{ij}^{k+1} = X_{ij}^{k} + V_{ij}^{k+1}$$

10: Evaluate fitness of each particle Fik+1 = f(Xki +1),∀i and identify new best particle b_k

11: Update Pbest of population ∀i

12: if $F_{ik+1} \leq F_{ik}$ then $P_{ijk+1} = X_{ijk+1}, \forall j$

13: else $P_{ij}^{k+1} = X_{ij}^{k}, \forall ij$

14: end if

15: Update Gbest of population

16: **if** $F_{b1}^{k+1} \leq F_b^k$ then $G^{k+1} = \mathbf{P}_{b_k j}^{k+1}, \forall j$ and set $b = b_k$

17: else $G^{k+1} = G^k$

18: end if

19: Check stopping criteria

20: if $k \leq Max_k$ then

21: *Goto* step 8

22: else

23: *Goto* step 25

24: end if

25: Print G^k and update ANN weights and biases

4. RESULTS AND DISCUSSION

The effectiveness and suitability of the developed approach-based ANN have been demonstrated by solving the fitting problem using H1O1 and H2O1 approaches. Fig. 6 shows the regression plot of the problem considered using the H1O1 ANN model trained using PSO. Similarly, Fig. 7 shows the regression plot of the problem considered using the H2O1 ANN model trained using PSO.

The R value case of H1O1 is over 0.96, and that for H2H1 is over 0.93. This indicates that the regression is quite good in both cases. Among these two, H1O1 performs better than H2O1. Table II gives the statistical details of the solution.

Table 2. Statistics of Regression Performance

ANN Model	R_2	*MAE*	*RMSE*
H1O1	0.9354	1.4251	1.8263
H2O1	0.8756	2.5341	1.9245

Table 2 shows that the coefficient of determination R2 values corresponding to the H1O1 ANN model are higher than that of H2O1, which indicates that the H1O1 model

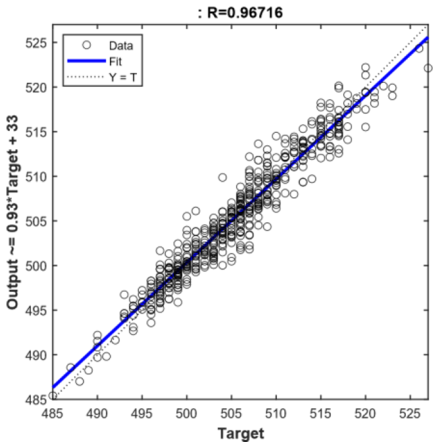

Fig. 6. Regression plot for H1O1 ANN-PSO.

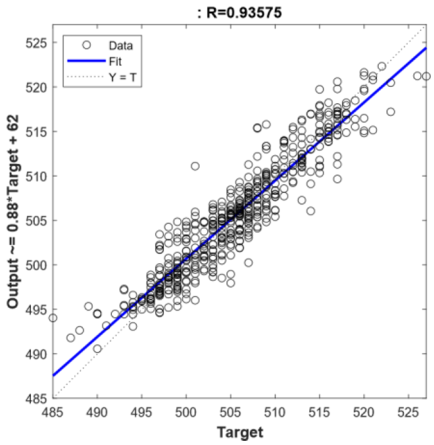

Fig. 7. Regression plot for H2O1 ANN-PSO.

trained using PSO is better. Similarly, the other statistics, such as mean average error (MAE) and root mean square error (RMSE), also confirm that the H1O1 model gives lower errors than the H2O1 model.

For testing the performance of the ANN-PSO for the H1O1 and H2O1 models, 7 data samples (sl numbers - 50, 100, 150, 200, 250, 300, 350, 400, 450) have been selected. The regression plots for H1O1 and H2O1 are shown in Figs. 8 and 9, respectively. Also, the other statistics (R2, MAE, RMSE) are given in Table 3.

Table 3. Statistics of Regression Performance on Testing Data

ANN Model	R_2	*MAE*	*RMSE*
H1O1	0.9456	1.7167	2.0475
H2O1	0.9071	2.4396	2.7194

From the testing results, it can be observed that the R values are relatively better than those of training data. Additionally, the coefficient of determination R2 values are better in the testing data than in training. However, the errors are slightly higher in the case of training data than in the testing data for both H1O1 and H2O1 models.

Hence, for ANN, a single hidden layer-based network is superior when trained using the PSO algorithm. This study will be extended to real engineering problems such as fault location in power networks, load forecasting and data analytics in the future.

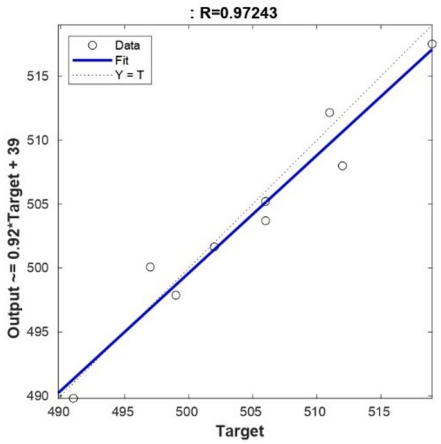

Fig. 8: Regression plot for H1O1 ANN-PSO for testing data.

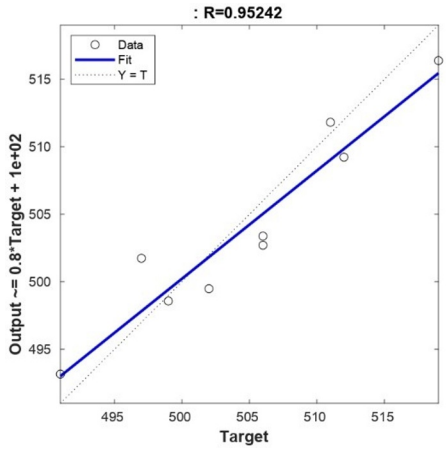

Fig. 9: Regression plot for H2O1 ANN-PSO for testing data.

5. CONCLUSION

In this work, various artificial neural network models have been discussed. The training of these models using the metaheuristic particle swarm optimization algorithm has been discussed. Also, a detailed algorithm for the same has been given, making it easier to program the developed approach. Based on the results obtained, it is concluded that a single hidden layer-based artificial neural network performs better than two hidden layer networks when training is performed using particle swarm optimization. Also, the regression coefficient is 0.967 for the chemical dataset problem given in MATLAB.

REFERENCES

[1] R. King, "Artificial neural networks and computational intelligence," *IEEE Computer Applications in Power*, vol. 11, no. 4, pp. 14–16, 1998.

[2] S. Halpin and R. Burch, "Applicability of neural networks to industrial and commercial power systems: a tutorial overview," *IEEE Transactions on Industry Applications*, vol. 33, no. 5, pp. 1355–1361, 1997.

[3] A. Jain, J. Mao, and K. Mohiuddin, "Artificial neural networks: a tutorial," *Computer*, vol. 29, no. 3, pp. 31–44, 1996.

[4] Q.-J. Zhang, K. Gupta, and V. Devabhaktuni, "Artificial neural networks for rf and microwave design - from theory to practice," *IEEE Transactions on Microwave Theory and Techniques*, vol. 51, no. 4, pp. 1339–1350, 2003.

[5] M. Chen, U. Challita, W. Saad, C. Yin, and M. Debbah, "Artificial neural networks-based machine learning for wireless networks: A tutorial," *IEEE Communications Surveys Tutorials*, vol. 21, no. 4, pp. 3039–3071, 2019.

[6] L. C. Jain and A. M. Fanelli, *Recent advances in artificial neural networks : design and applications / edited by Lakhmi Jain, Anna Maria Fanelli*. CRC Press Boca Raton, 2000.

[7] A. Zayegh and N. A. Bassam, "Neural network principles and applications," in *Digital Systems*, V. Asadpour, Ed. Rijeka: IntechOpen, 2018, ch. 7. [Online]. Available: https://doi.org/10.5772/intechopen.80416

[8] K. Mehrotra, C. Mohan, and S. Ranka, *Elements of Artificial Neural Networks*, ser. A Bradford book. MIT Press, 1997. [Online]. Available: https://books.google.co.in/books?id=6d68Y4WqR4C

[9] M. Zhang and M. Zhang, *Artificial Higher Order Neural Networks for Economics and Business*. USA: IGI Global, 2008.

[10] S. Rukhaiyar, M. N. Alam, and N. K. Samadhiya, "A pso-ann hybrid model for predicting factor of safety of slope," *International Journal of Geotechnical Engineering*, vol. 12, no. 6, pp. 556–566, 2018. [Online]. Available: https://doi.org/10.1080/19386362.2017.1305652

[11] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95 - International Conference on Neural Networks*, vol. 4, 1995, pp. 1942–1948 vol.4.

[12] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, 1995, pp. 39–43.

[13] R. Eberhart and Y. Shi, "Comparing inertia weights and constriction factors in particle swarm optimization," in *Proceedings of the 2000 Congress on Evolutionary Computation. CEC00 (Cat. No.00TH8512)*, vol. 1, 2000, pp. 84–88 vol.1.

[14] M. Clerc and J. Kennedy, "The particle swarm - explosion, stability, and convergence in a multidimensional complex space," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 1, pp. 58–73, 2002.

[15] Y. del Valle, G. K. Venayagamoorthy, S. Mohagheghi, J.-C. Hernandez, and R. G. Harley, "Particle swarm optimization: Basic concepts, variants and applications in power systems," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 2, pp. 171–195, 2008.

[16] M. N. Alam, B. Das, and V. Pant, "A comparative study of metaheuristic optimization approaches for directional overcurrent relays coordination," *Electric Power Systems Research*, vol. 128, pp. 39–52, 2015.

[17] MATLAB, Mathworks Inc., Massachusetts, USA, version R2018a.

[18] J. Liang, A. Qin, P. Suganthan, and S. Baskar, "Comprehensive learning particle swarm optimizer for global optimization of multimodal functions," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 3, pp. 281–295, 2006.

[19] C. Coello, G. Pulido, and M. Lechuga, "Handling multiple objectives with particle swarm optimization," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 256–279, 2004.

[20] J.-H. Seo, C.-H. Im, C.-G. Heo, J.-K. Kim, H.-K. Jung, and C.-G. Lee, "Multimodal function optimization based on particle swarm optimization," *IEEE Transactions on Magnetics*, vol. 42, no. 4, pp. 1095–1098, 2006.

[21] Z.-L. Gaing, "Particle swarm optimization to solving the economic dispatch considering the generator constraints," *IEEE Transactions on Power Systems*, vol. 18, no. 3, pp. 1187–1195, 2003.

[22] J. Lu, D. Ireland, and A. Lewis, "Multi-objective optimization in high frequency electromagnetics—an effective technique for smart mobile terminal antenna (smta) design," *IEEE Transactions on Magnetics*, vol. 45, no. 3, pp. 1072–1075, 2009.

[23] Z.-L. Gaing, "A particle swarm optimization approach for optimum design of pid controller in avr system," *IEEE Transactions on Energy Conversion*, vol. 19, no. 2, pp. 384–391, 2004.

[24] R. Abousleiman and O. Rawashdeh, "Electric vehicle modelling and energy-efficient routing using particle swarm optimisation," *IET Intelligent Transport Systems*, vol. 10, pp. 65–72(7), March 2016. [Online]. Available: https://digital-library.theiet.org/content/journals/10.1049/iet-its.2014.0177

[25] M. N. Alam, A. Mathur, and K. Kumar, "Economic load dispatch using a differential particle swarm optimization," in *2016 IEEE 1st International Conference on Power Electronics, Intelligent Control and Energy Systems (ICPEICES)*, 2016, pp. 1–5.

[26] M. N. Alam, K. Kumar, and A. Mathur, "Economic load dispatch considering valve-point effects using time varying constriction factor based particle swarm optimization," in *2015 IEEE UP Section Conference on Electrical Computer and Electronics (UPCON)*, 2015, pp. 1–6.

[27] M. Hedayati, Z. H. Firouzeh, and H. K. Nekoei, "Hybrid quantum particle swarm optimisation to calculate wideband green's functions for microstrip structures," *IET Microwaves, Antennas & Propagation*, vol. 10, pp. 264–270(6), February 2016. [Online]. Available: https://digitallibrary.theiet.org/content/journals/10.1049/iet-map.2015.0169

[28] X. Xia, H. Song, Y. Zhang, L. Gui, X. Xu, K. Li, and Y. Li, "A particle swarm optimization with adaptive learning weights tuned by a multiple-input multiple-output fuzzy logic controller," *IEEE Transactions on Fuzzy Systems*, pp. 1–15, 2022.

[29] L. Ge, Y. Li, J. Yan, Y. Wang, and N. Zhang, "Short-term load prediction of integrated energy system with wavelet neural network model based on improved particle swarm optimization and chaos optimization algorithm," *Journal of Modern Power Systems and Clean Energy*, pp. 1–12, 2021.

[30] L. Hu, Y. Yang, Z. Tang, Y. He, and X. Luo, "Fcan-mopso: An improved fuzzy-based graph clustering algorithm for complex networks with multi-objective particle swarm optimization," *IEEE Transactions on Fuzzy Systems*, pp. 1–16, 2023.