American Scientific Research Journal for Engineering, Technology, and Sciences (ASRJETS)

ISSN (Print) 2313-4410, ISSN (Online) 2313-4402

© Global Society of Scientific Research and Researchers

http://asrjetsjournal.org/

Grass Root Algorithm Optimize Neural Networks for Classification Problem

Prof. Dr. Hanan A.R. Akkar[a], Firas R. Mahdi[b*]

[a,b]Electrical Engineering Department, University Of Technology, Baghdad, Iraq

[a]Email: dr_hananuot@yahoo.com

[b]Email: firasrasool1980@gmail.com

Abstract

Artificial neural networks are computational models that trying to emulate the structure and functions of biological human networks. They have been extensively used in many applications include science, business, engineering, and data mining. Learning of an artificial neural network means how to adapt the weights of the network interconnections using suitable adaption algorithm. The training algorithms that is used to modify the weights of the network are considered the most important portion that influences the artificial networks performance. In the past few decade, many meta-heuristic algorithms have been used to optimize networks synaptic weights, in order to achieve better performance. This paper proposes a general network training method based on population-based algorithms, proposes a novel meta-heuristic algorithm that is inspired by the general grass plants root system to optimize the weights of the proposed artificial network to classify real data four classes XOR and Iris data comparing the obtained results of the proposed algorithm with other familiar evolutionary meta-heuristic algorithms.

Keywords: Artificial neural networks; Classification; Grass root algorithm; Meta-heuristic techniques; Optimization; Population-based algorithms.

1. Introduction

Artificial Neural Networks (ANNs) are information processing systems that have common characteristics with human biological networks.

----

* Corresponding author.

They have been widely used in many life applications such as medicine [1], financial [2], data mining [3], and industrials [4]. The most significant concern related to this concept is how to train the network to optimize a solution with the most trained weights. Learning of ANN is the process of adjusting the interconnection weights of the neurons. One of the most familiar training algorithms is the Error Back Propagation algorithm (EBP), this algorithm has been widely used for the network training purpose. However, it appears to be suffering from several problems such as easily trapped into local minima, and its low convergence speed [5]. Many types of research have been tried to improve the performance of the EBP, while other have just left this concept and used other meta-heuristic algorithms instead of EBP in the training phase. Therefore, many meta-heuristic algorithms have been developed and applied to raise the performance of the training process such as: Genetic Algorithm (GA) [6], Differential Evolution Algorithm (DEA) [7], Simulated Annealing (SA) [8], Artificial Bee Colony (ABC) [9], and Particle Swarm Optimization (PSO) [10]. This paper aims to propose a general method for optimizing Neural Networks (NNs) weights connections using population-based Grass Root Algorithm (GRA), that is proposed by the author, to classify four classes XOR problem, and 3 classes Iris real data sets comparing the results with other familiar population-based algorithms used for the same purpose.

2. Materials and Methods

This paper uses a novel GRA to optimize an ANN weights interconnections to classify a real data four classes XOR problem and three classes Iris data set, comparing the obtained results with other population-based algorithms. The materials and methods used for that purpose are illustrated in the following sub-sections.

2.1. Multilayer Perceptron

Multilayer Perceptron networks (MLPs) are popular feed-forward supervised ANNs. They consist of the input layer, an output layer, and one or multiple hidden layers. Usually, One hidden layer is sufficient to solve almost all types of problems. Two or more processing hidden layers rarely improve the network efficiency, also they may lead for trapping into a local minima solution. However, all hidden and output neurons are actually considered as the processing elements of the network. Each network layer is usually fully connected to the next layer, and it consists of multiple neurons, their number varies according to the problem the network used to solve. Each processing element neuron consists of multiplayer, adder, and activation function. Usually, each processing layer has the same activation function for all of its elements, neurons may have a linear identity activation function or nonlinear activation function such as hyperbolic tangent, logistic and Gaussian. One of the most important problems associated with NNs learning process is the overfitting, which usually occurs due to using too many neurons in the hidden layer, or when the ANN has too much information and the amount of input data patterns is not enough to justify all the neurons weights in the hidden layers at the required time. Another problem occurs when the training data set is satisfactory, but the amount of training time rise to a point that it is impossible to sufficiently learn the ANN. Therefore, some compromise must be reached between too many and few neurons in the hidden layers, another compromising method represented by regularization the network by modifying the performance function, which in most cases chosen to be the Mean Square Error (MSE) of the network. Once the number of neurons and hidden layers selected, the network's weights must be optimized to minimize the error made by the network. This is the role played by the training algorithms [11].

2.2. Metaheuristic Algorithms

Meta-heuristics are techniques that allow optimizing large and complicated problems by delivering satisfactory and feasible solutions in an acceptable processing time. These techniques have no guarantee to find the global optimal solution. Unlike exact methods, they have the ability to solve large and complex problems. They have been used in many applications such as machine learning, data mining, system modeling, and robotics. Each meta-heuristic algorithm must have a tradeoff between, two contradictory features; diversification which is the exploration of the search space and intensification that represents the exploitation of the best solutions found so far. In intensification, the promising regions are explored more locally in the hope of finding better solutions. Meta-heuristic algorithms could be classified as population or single based search algorithm, these two families have complementary characteristics: single solution based meta-heuristics have the power to intensify the search in local regions only while population-based meta-heuristics allows better diversification in the whole search space [12]. Population-based meta-heuristic algorithms depend on initial random population, each individual in this population is a candidate solution. A new population of solutions is created, based on evolution or adapting blackboard algorithm. The evolution based reproduction of population using multiple of operators such as; crossover, and mutation. Therefore, a new population is generated from different attributes of the current population. Blackboard-based population generation depends on shared memory constructed from the previous population. The selection of the best solution will depend on the fitness value of the applied objective function after a set of rules that govern the population and redirect it towards an optimum or near optimum solution [13].

2.3. Grass Root Algorithm

Optimization is a mathematical technique that finds the best solution to a constrained problem, while an optimization problem is how to find variables that minimize or maximize an objective function, these variables must satisfy optimization function constraints. Many optimization problems have more than one local solution, therefore it is important to choose a good optimization method that has a good tradeoff between global and local search mechanisms without being trapped into a local minima solution. Grass Root Algorithm (GRA) is an optimization meta-heuristic population-based algorithm, it is essentially inspired from the grass plants reproduction, development, and theirs fibrous root system. To understand the proposed algorithm searching mechanism it's important to give a brief review for the reproduction and development of the grass plant. Grasses are generally propagated through two ways; firstly by modified subterranean stems of the plant that is usually found underground, sending out roots and shoots from its nodes that's called rhizomes, and secondly through a stems that grow just below the surface. Both reproduction ways will simultaneously develop secondary roots to replace the vanished primary roots. Hair roots are generated from secondary roots [14]. These roots (primary, secondary, and hair roots) are usually used for global and local searching of water resources and minerals. However, in GRA the global search will be performed at each iteration, while the local search performed when the global search is in stack condition or it does not lead to more improvement in the objective function. GRA has two different mechanisms for global search; survived grasses and the best obtained grass modified and reproduced. On the other hand, local search will have another two searching methods; regenerated secondary roots and secondary hair roots.

2.3.1. GRA model governs rules

Grass plants with rhizomes and stolons execute a global and local search to find better resources by reproduce new grass plants and modifying their own fibrous root system. Both the generated new grasses and their roots are developed almost randomly, but when a root arrives at a place with more resources, the corresponding plant generates more secondary roots, and root's hair, which made the plant growth faster than other. If a grass plant trapped into a local minima point it will generate more new grasses by stolons initiated from the best-obtained grass and from other grasses survived from the initial population process, these stolons are in general longer than rhizomes, which help the plant reaching farther places and escaping from local minima position.

2.3.2. GRA Mathematical Model

Just like other meta-heuristic optimization algorithms, GRA starts with an initial random uniformly distributed population ***swarm(****pop,ndim****)*** in the search space domain of the problem, each row vector of ***swam*** initial matrix represents a grass initiated by seeding process. The number of initial grass plants is considered equal to the population size (*pop*). When iteration (*iter*) starts a new population ***NewSw****(pop,ndim)* will be generated, this new population will consist of; the fittest obtained grass represented by ***Gbest*** = min(f (***swarm***)) ∈ R^{ndim}, $f : R^{ndim} \rightarrow R$, where f is the mean square error (MSE) function, *ndim* is the problem dimension, ***swarm*** and ***NeSw*** are bounded population in the range of ***lb(****pop,ndim****)* ≤ *swarm*, *NewSw* ≤ *ub*(***pop,ndim*)*,*** where ***ub, lb*** $\in R^{ndim}$ are two matrices indicating the lower and upper bounds of population variables. The second element of the new population ***NewSw*** is a number (*GN*) of grasses deviated from ***Gbest*** by stolons (***Grass$_{Branches}$***), these stolons usually deviated from the original grass with step size usually less than maximum ***ub***, the last element of the ***NewSw*** are a new grasses equal to (*pop – GN-1*) deviated randomly from the survived best initial grasses (***Survived$_{deviated}$***).

$$GN = \left\| \left(\frac{avg(\boldsymbol{mse})}{avg(\boldsymbol{mse}) + min(\boldsymbol{mse})} \right) * \left(\frac{pop}{2} \right) \right\| \tag{1}$$

Where *GN* is the number of the newly generated grass branches (stolons) from ***Gbest***, and **mse** is the mean square error value of all grass population. From (1) we notice that the maximum generated new grass branches will be equal to (*pop*/2) and reached when the minimum of **mse** is too small. Each new branch grass deviated from ***Gbest*** according to (2), while the survived deviated grasses will be represented by:

$$\boldsymbol{Grass_{Branches}} = ones(GN,1) * \boldsymbol{Gbest} + 2 * max(\boldsymbol{ub}) * (rand(GN,1) - 0.5) * \boldsymbol{Gbest} \tag{2}$$

$$\boldsymbol{Survived_{deviated}} = \boldsymbol{Grass_{Survived}} + 2 * max(\boldsymbol{ub}) * (rand(pop-GN-1,1) - 0.5) * \boldsymbol{ub} \tag{3}$$

where ones(c,1) represents one's column vector with c rows, and rand(c,1) is a random column vector with c rows their elements greater than 0 and less than 1. ***Grass$_{survived}$*** is the (*pop-GN-1*) highest **mse** initial population. The new population (***NewSw***) will be represented by (4) and (5).

$$\boldsymbol{NewSw} = [\ \boldsymbol{Gbest^{T}},\ \boldsymbol{Grass_{Branches}}^{T},\ \boldsymbol{Survived_{deviated}}^{T}\ ]^{T} \tag{4}$$

Which is the same as:

$$NewSw = [\, Gbest\,;\ Grass_{Branches}\,;\ Survived_{deviated}\,] \qquad (5)$$

The new reproduced population (***NewSw***) will be checked to find the grass with minimum *mse* and bounded it to be within the limits of ***ub*** and ***lb***, If the fittest new grass is best than the old one then save the fittest new grass as the best solution, otherwise calculate the absolute rate of decrease in **mse** between the best obtained so far minimum **mse** ($best_{minmse}$) shown in (6) and the current iteration minimum **mse** (**minmse**) shown in (7), if the rate is less or equal a predefined tolerance value (*tol*) as in (8), than increase a global stack ($stack_g$) counter by one, when *stackg* reached its maximum predefined value then move to the next local search mechanism.

$$minmse = min_{i=1,\ldots,pop}\,(mse) \qquad (6)$$

$$best_{minmse} = min_{\,j=1,\ldots,iter}\,(minmse) \qquad (7)$$

$$\left| \frac{minmse - best_{minmse}}{minmse} \le tol \right| \qquad (8)$$

The local search mechanism consists of two individual loops; secondary roots loop, and hair roots loop. We have considered that the hair roots are equal to the dimension of the objective function problem (*ndim*) so that each secondary root generated by the ***Gbest*** represents a local candidate solution. The secondary roots will be represented by a random number, these secondary roots will have a number of hair roots equal to *ndim*, each single hair root will modify its location according to (9) for a repeated loops equal to secondary roots number (*S*).

$$m\_gbest(1,i)_{i=1,..,ndim}^{\;k=1.,.,S} = avg(Gbest) + Gbest(1,i)_{i=1,..,ndim} + C_2 * (rand-0.5) \qquad (9)$$

$$C = [C_1\,,\,C_2\,,\,C_3\,,\,C_{5\,,\ldots,}\,C_9\,,\,C_{10}] \qquad (10)$$

$$C_c = C(1 + (\|rand\ *\ 10\|)\,) \qquad (11)$$

Where ***mgbest*** is the locally modified ***Gbest*** element by element, *S* is the number of secondary generated roots where ($0 \le S \le ndim$), and ***C*** is the searching step size vector equation (10), and *Cc* will be a random element of ***C*** chose according to the percentage repetition of **C** elements, as shown in (11). If the evaluated ***mgbest*** min(**mse**) is less than $best_{minmse}$ then save ***Gbest*** as ***mgbest***, otherwise calculate the absolute rate of decrease in mean square error as in (7), if the rate is less than *tol* then increase local stack counter ($stack_l$) by one, if $stack_l$ reached its maximum predefined value then break hair root loop and begin new secondary root loop, after each completed iteration check if the stopping condition (*GlobalError*) is satisfied then stop iteration, otherwise go to next iteration until reached the *maxit* then stop. The pseudo code of GRA are as follow:

2.3.3. GRA for NN Weight Optimization

GRA is a population based meta-heuristic algorithm, in which each ***Swarm*** contains a *pop* number of candidate solutions each with *ndim* dimensions, for NN case we have considered the weights vector of the supervised network is the problem that needs to be optimize in order to get the minimum mean square error between the network output and its predefined target. Suppose the input is **input**(*N*, *n*), **target**(*N*, *m*) is the target teacher. Therefore, *N* is the total input patterns, *n* is the number of input neurons, and *m* is the output neurons number. suppose *l* is the hidden neurons number, then the NN dimension with bias connections *ndim* will be represented by:

$$ndim = (n * l + l) + (l * m + m) \qquad (12)$$

Table 3

```
Initialize: maxit, pop, ndim, lb, ub, GlobalError, tol, $stack_g$, $stack_l$, $\mathbf{Branch_{Grass}}$, C, Gbest, minmse, $best_{minmse}$ , GN, F.
Initialize random grass population (Swarm). where: $\mathbf{Swarm} \in R^{ndim}$
Bound the initial population $\mathbf{lb} \leq \mathbf{Swarm} \leq \mathbf{ub}$ where: $\mathbf{ub}, \mathbf{lb} \in R^{ndim}$.
For i=1 : pop // check for best fitness particle in the initial population.
mse(i)= F(Swarm) // F is the MSE predefined function
End For
Sort the grass population (Swarm) ascending according to mse.
For iter=1 : maxit // Iteration and global search starting
Evaluate $C_2$ according to (8)
Generate the new population (NewSw) according to (4).
Bound NewSw : $\mathbf{lb} \leq \mathbf{NewSw} \leq \mathbf{ub}$ where:$\mathbf{NewSw}, \mathbf{ub}, \mathbf{lb} \in R^{ndim}$
For i=1: pop // check the NewPar for the best fitness particle loop
Mse(i)=F(NewSw)
End For
 Minmse=min(Mse) // Save minimum mean square error
 Index the grass with the Minmse // Index the position of the best particle in the population.
Evaluate GN as in (1). // GN is the number of stolons branches.
Evaluate $\mathbf{Grass_{Branches}}$ as in (2).
Evaluate the $\mathbf{Grass_{Survived}}$ from the ascending sorted Swarm.
Evaluate $\mathbf{Survived_{deviated}}$ as in (3).
If Minmse < $best_{minmse}$ then
$best_{minmse}$=Minmse
Gbest=best indexed grass
$stack_g$=0
Else If (7) is true then
increase $stack_g$ by 1
If $stack_g$ is at its maximum then
$stack_g$= 0 // Begin the local search
```

```
For q=1:random integer less than D // secondary root loop
Stack_l=0
For j=1:ndim // hair root loop
mgbest=Gbest //initial mgbest
evaluate mgbest as in (8)
localmin = F(mgbest)
If localmin<best_minmse then
best_minmse=localmin
Gbest=mgbest
stack_l=0
Else If (7) (with localmin instead of  minmse)  is true then
Increase stack_l by one
Else
stack_l=0
End If
If stack_l is at its maximum then
Break For
End If
End For (j loop)
End For (q loop)
End If
End If
best_minmse=F(Gbest)
If best_minmse ≤ GlobalError then
break For (iter loop)
End If
End For (iter loop)
```

suppose ***weight***(*1,ndim*) is the NN weight vector, then the input to hidden weight vector could be represented by (13) and the hidden to output weight is represented by (14), while (15) represents the ***BIAS*** vector.

$$\boldsymbol{W} = [weight(1), weight(2), \ldots\ldots, weight(n*l+l)] \quad (13)$$

$$\boldsymbol{V} = [weight(n*l+l+1), weight(n*l+l+2),\ldots,weight(ndim)] \quad (14)$$

$$\boldsymbol{BIAS} = [bias*ones(N,1)] \quad (15)$$

Reshape ***W***, and ***V*** vectors into matrix form, then they will be ***Wx***(*n+1, l*), and ***Vy***(*l+1,m*). The input and vector matrix could be represented by:

$$\boldsymbol{X} = [\,\boldsymbol{BIAS}, \boldsymbol{Input}] \qquad (16)$$

Applying log sigmoid to the output of the hidden layer and add ***BIAS*** vector to the hidden output, and evaluating the hyperbolic tangent of the network output as in (17), (18), and (19) respectively.

$$\boldsymbol{Z} = \frac{1}{1+\, e^{-(\boldsymbol{X} * \boldsymbol{W}\boldsymbol{x})}} \qquad (17)$$

$$\boldsymbol{H} = [\,\boldsymbol{BIAS}, \boldsymbol{Z}\,] \qquad (18)$$

$$\boldsymbol{Y} = \frac{2}{1+e^{-2(\boldsymbol{H} * \boldsymbol{V}\boldsymbol{y})}} - 1 \qquad (19)$$

The **MSE** between the NN output and the desired target is shown in (20), while the Average MSE (*AMSE*) of all network neurons output is shown in (21).

$$\boldsymbol{MSE} = \frac{1}{pop} \sum_{i=1}^{i=pop} (target(i) - Y(i))^2 \qquad (20)$$

$$AMSE = \frac{1}{m} \sum_{i=1}^{i=m} MSE^T(i) \qquad (21)$$

3. Experiments and Results

We have carried out two experiments to test GRA, and the proposed training method also compares it with other familiar meta-heuristic population-based algorithms. The first experiment was to classify four classes XOR, with 100 instances for each class, and 3 attributes. The second experiment has applied to classify real iris data, which consists of 3 classes each with 50 instances and four attributes. For both experiments we have divided the input pattern data into two sets; 80% of input data for training, and 20% for testing, so that we can discover if an overfitting occurs during training process. Recording the Average Classification Rate (ACR), Average Training Mean Square Error (ATRMSE), Average Testing Mean Square Error (ATEMSE), Average Error between MSE of Training and Testing data sets (AETT), Average Processing Time (APT) in seconds, and the Average number of required Iterations (AITER) to reach 100% classification rate for both of the experiments. For the XOR experiment we have used a NN with 3 input, 6 hidden, and 2 output neurons, while for the 2nd experiment we have used a network with 4 input, 5 hidden and 2 output neurons. If any algorithm reached 100% classification rate then the training will be stopped immediately. We have carried out 10 epochs for each algorithm each epoch has 100 maximum training iterations. The algorithms used to train the NN are; PSO, GA, BCA, DEA, Wind Driven Optimization (WDO) [15], Cuckoo Search Algorithm (CSA) [16], and the proposed GRA. Table (1) stands for algorithms performance for the 1st experiment it shows that GRA has gotten the highest ACR with the minimum required; iterations number and average processing time. Table (2) stands for the 2nd experiment algorithms performance it shows that GRA has gotten the highest ACR with the minimum required iterations, but with the highest required average processing time. Figs. 1, and 2 show the convergence curves for single epoch (100 iterations) of the tested algorithms and for both experiments, they show that GRA algorithm has much faster convergence than other tested algorithms, also it reaches an acceptable solution within fewer iterations number.

Table 1: 1st experiment algorithms performance

Alg.	ATRMSE	ATEMSE	AETT	ACR %	APT (sec)	AITER
BCA	0.1414	0.1470	7.64E-03	75.63	3.03	100
CSA	0.1593	0.1601	9.92E-03	67.08	2.64	100
DEA	0.1981	0.2110	1.37E-02	55.97	4.75	100
GA	0.1333	0.1404	7.11E-03	83.44	1.51	100
GRA	**0.1018**	**0.1048**	**4.39E-03**	**100.0**	**1.39**	**16.67**
PSO	0.1775	0.1864	9.06E-03	57.26	1.51	100
WDO	0.1528	0.1599	7.16E-03	69.31	1.46	100

Table 2: 2nd experiment algorithms performance

ALG.	ATRMSE	ATEMSE	AETT	ACR%	APT (sec)	AITER
BCA	0.09094	0.09513	0.00612	86.02	2.43	100
CSA	0.08357	0.0884	0.00747	79.17	2.09	100
DEA	0.12359	0.12365	**0.00417**	57.22	3.78	100
GA	0.07634	0.08022	0.00717	84.44	**1.28**	100
GRA	**0.05567**	**0.0569**	0.00477	**95.56**	3.29	**96.3**
PSO	0.09746	0.09924	0.0051	70.46	1.165	100
WDO	0.08833	0.09293	0.00693	80.28	1.158	100

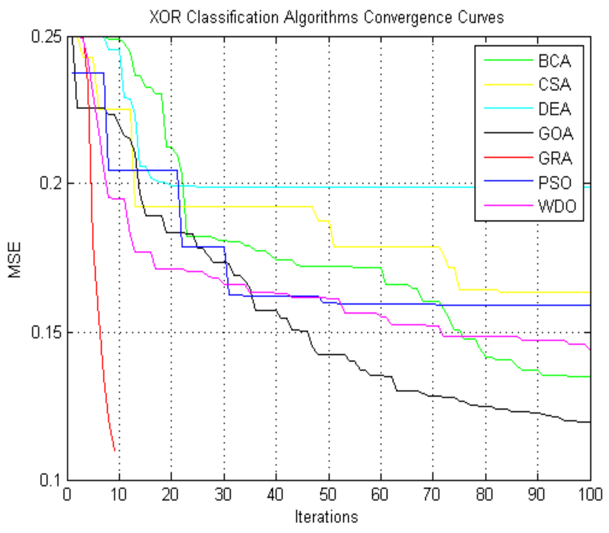

Figure 1: 1st experiment algorithms convergence curves .

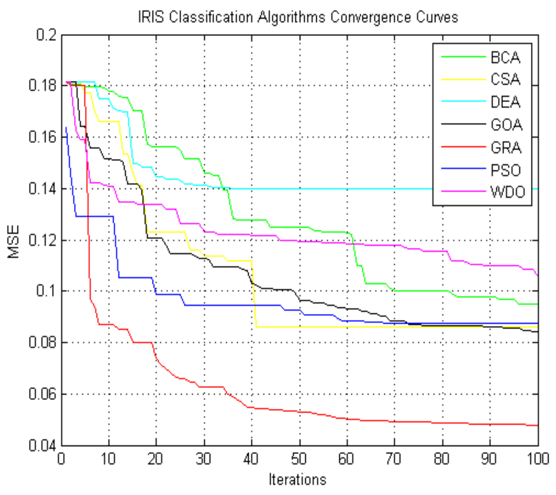

Figure 2: 2nd experiment algorithms convergence curves.

4. Conclusions

This paper has proposed a new meta-heuristic population-based algorithm for training ANN to classify 4 classes XOR and Iris data set comparing it with another familiar algorithm in that field. The proposed algorithm has been inspired from the reproduction and root system of general grass root. In the 1st experiment, we have found that GRA has gotten the highest 100% ACR, with the minimum required iteration number of 16.67 hence it has recorded the minimum required processing time with the minimum average training MSE and the average testing MSE. For the second experiment, GRA has gotten the highest 95.56% ACR, the minimum iteration number of 96.3, and the minimum average training and testing MSE. On the other hand, it has recorded a high average processing time 3.29 sec as compared to other algorithms (1.28 sec and 2.43 sec) that have gotten an acceptable ACR (84.44 % and 86.02%).

References

[1] E. Tileylioglu and A. Yilmaz, "Application of neural based estimation algorithm for gait phases of above knee prosthesis," Engineering in Medicine and Biology Society (EMBC), 2015 37th Annual International Conference of the IEEE, pp. 4820-4823, Milan, 2015.

[2] W. Shuhui, "A Novel Company Financial Risk Warning Method Based on BP Neural Network," Intelligent Computation Technology and Automation (ICICTA), 2014 7th International Conference on, pp. 32-35, Changsha, 2014.

[3] J. Wang, F. Zhang, F. Liu and J. Ma, "Hybrid forecasting model-based data mining and genetic algorithm-adaptive particle swarm optimisation: a case study of wind speed time series," in IET Renewable Power Generation, vol. 10, no. 3, pp. 287-298, 2016.

[4] A. Rubaai and P. Young, "Hardware/Software Implementation of Fuzzy-Neural-Network Self-Learning Control Methods for Brushless DC Motor Drives," in IEEE Transactions on Industry Applications, vol.

52, no. 1, pp. 414-424, Jan.-Feb. 2016.

[5] M. Gori and A. Tesi, "On the problem of local minima in backpropagation," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 14, no. 1, pp. 76-86, Jan 1992.

[6] 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," in IEEE Sensors Journal, vol. 16, no. 5, pp. 1158-1167, March1, 2016.

[7] Slowik A.; Bialko M., "Training of artificial neural networks using differential evolution algorithm," Human System Interactions, 2008 Conference on , vol.60, no.65, pp. 25-27, May 2008.

[8] K. Bai and J. Xiong, "A Method of Improved BP Neural Algorithm Based on Simulated Annealing Algorithm," Genetic and Evolutionary Computing, 2009. WGEC '09. 3rd International Conference on, Guilin, 2009, pp. 765-768.

[9] C. Worasucheep, "Forecasting currency exchange rates with an Artificial Bee Colony-optimized neural network," Evolutionary Computation (CEC), 2015 IEEE Congress on, Sendai, 2015, pp. 3319-3326.

[10] L. Zhang, J. Ma, Y. Wang and S. Pan, "PSO-BP Neural Network in Reservoir Parameter Dynamic Prediction," Computational Intelligence and Security (CIS), 2011 Seventh International Conference on, Hainan, 2011, pp. 123-126.

[11] Gaurang P.; Amit G.; Y P K.; and Devyani P., " Behaviour Analysis of Multilayer Perceptron's with Multiple Hidden Neurons and Hidden Layers," International Journal of Computer Theory and Engineering, vol. 3, no. 2, April 2011.

[12] E. Talbi, " Common Concepts for Metaheuristics", in Metaheuristics : from design to implementation, John Wiley & Sons, Inc., Hoboken, New Jersey, 2009, ch.1, pp.23-25.

[13] E. Talbi, " Population-Based Metaheuristics, in Metaheuristics : from design to implementation, John Wiley & Sons, Inc., Hoboken, New Jersey, 2009, ch.3, pp.190-200.

[14] C. Stichler, " Grass Growth and Development," Texas Cooperative Extension, Texas A&M University, SCS-2002-22.

[15] Z. Bayraktar, M. Komurcu and D. H. Werner, "Wind Driven Optimization (WDO): A novel nature-inspired optimization algorithm and its application to electromagnetics," Antennas and Propagation Society International Symposium (APSURSI), 2010 IEEE, Toronto, ON, 2010, pp. 1-4.

[16] R. A. Vazquez, "Training spiking neural models using cuckoo search algorithm," Evolutionary Computation (CEC), 2011 IEEE Congress on, New Orleans, LA, 2011, pp. 679-686.