2014 IEEE Congress on Evolutionary Computation (CEC)
July 6-11, 2014, Beijing, China

Electromagnetic Algorithm for tuning the structure and parameters of Neural Networks

Ayad Mashaan Turky, Salwani Abdullah and Nasser R. Sabar

***Abstract*—Electromagnetic algorithm is a population based meta-heuristic which imitates the attraction and repulsion of sample points. In this paper, we propose an electromagnetic algorithm to simultaneously tune the structure and parameter of the feed forward neural network. Each solution in the electromagnetic algorithm contains both the design structure and the parameters values of the neural network. This solution later will be used by the neural network to represents its configuration. The classification accuracy returned by the neural network represents the quality of the solution. The performance of the proposed method is verified by using the well-known classification benchmarks and compared against the latest methodologies in the literature. Empirical results demonstrate that the proposed algorithm is able to obtain competitive results, when compared to the best-known results in the literature.**

I. Introduction

Neural networks (NNs) are computing techniques inspired by nature, which has been successfully used to solve a wide variety of problems such as pattern recognition [1], signal processing [2] and optimisation [2]. The use of neural network to solve real world problems requires some critical decisions, which may has negative effect on solving certain problem. For example, the optimum network structure and parameters are the most important attributes of the network networks, which have direct effect on the solution quality, to be determined [3]. In multidimensional space, the tuning of the neural network structure and parameters can be considered as a complex optimisation problem, since each point represents a potential neural network with different network structures and link weights [3]. Hence, the use of fixed parameters and structure at the overall connectivity between the neurons may not produce good results. A network that has small neurons may not achieve good performance due to its limited information processing power. On the other hand, a network that consists large number of neurons may contain many redundant connections and also computationally expensive [4, 5].

During the last decade, several population-based methods have been developed to generate the appropriate structure and parameters of a neural network. Ilonen et al. [6] employed a differential evolution for training the feed forward neural network. The proposed algorithm produced promising results when tested on pattern classification and function approximation. Leung et al. [4] proposed a genetic algorithm to tune the structure and parameter values of a neural network. In this approach, a fully connected three-layer feed forward neural network with switches are used and the number of hidden nodes is manually determined, starting with a small number and iteratively increased until the learning performance is achieved. The algorithm is tested on sunspots and associative memory. Tsai et al. [7] employed a hybrid Taguchi-genetic algorithm to tune the structure and the parameter values of a neural network. In this approach, the same model which is proposed by Leung et al. [4] is used. The proposed algorithm has shown excellent results when used to estimate the number of sunspots and to realize the associative memory. Dai et al. [8] introduced a new population-based heuristic search algorithm called seeker optimisation algorithm, which is used to tune the structure and the parameter values of a neural network. The proposed algorithm has shown good results when tested on pattern classification and function approximation. Zhao et al. [9] applied a cooperative binary-real particle swarm optimisation to tune the structure and parameter values of a neural network. In this method, binary particle swarm optimisation (PSO) is used to tackle the switches set, where each switch has either 0 or 1 value, whilst the basic particle swarm optimisation is used to optimise the weight values. The proposed binary-real PSO algorithm has been able to achieve the state-of-the-art results when tested to estimate the number of sunspots.

The successes of the mentioned population-based methods are the main motivating factors for proposing a new population-based method based on Electromagnetic algorithm (EM) for tuning both structure and parameter values of a feed forward neural network. EM is a population based meta-heuristic method that imitates the attraction and repulsion of the sample points and moves them towards a high quality solution while avoiding the local optima [10]. It has been successfully employed to solve several optimisation problems such as examination timetabling problems [11], vehicle routing problems [12] and job shop scheduling [13], which made it a worthy candidate to consider for solving real world problems. In addition to our knowledge, it has not been comprehensively studied in the context of tuning both the structure and parameter values of a feed forward neural network [14-16].

This paper is organised as follows. Section 2 presents the proposed algorithm. Experimental results are discussed

Ayad Mashaan Turky and Salwani Abdullah are with Universiti Kebangsaan Malaysia, 43600 UKM Bangi, Selangor, Malaysia (e-mail: ayadalrashid@gmail.com, salwani@ftsm.ukm.my).

Ayad is also affiliated with Swinburne University of Technology, Victoria, Australia.

Nasser R. Sabar is with The University of Nottingham Malaysia Campus, Jalan Broga, 43500 Semenyih, Selangor, Malaysia (e-mail: Nasser.Sabar@nottingham.edu.my).

978-1-4799-1488-3/14/$31.00 ©2014 IEEE

in Section 3. Finally, some brief concluding comments are provided in Section 4.

II. THE PROPOSED ALGORITHM

In this section, the NN with link switches, followed by the description of the EM algorithm and its applications to tune the structure and parameters of the neural network are presented.

A. The NN with link switches

In general, the structure of a neural network starts with a fixed number of inputs, hidden and outputs nodes. These include the set of the parameters and network structure. Hence, the use of fixed parameters and structure may not yield good results within a given training period. Small networks get into the local minima too easily, and may not achieve good results due to its limited information processing power. On the other hand, large networks may take a long time to learn the characteristics of the data and computationally too expensive [4] [7] [8].

In this study, we used a fully connected three layer neural network with link switches that was proposed by Leung et al. [4]. In order to choose an optimal number of hidden nodes, their numbers are first fixed between three and seven, in order to test the learning performance. The input-output relationship can be defined as follows:

$$y_k(t)=\sum_{j=1}^{n_h}\left(w_{jk}t_{jk}\text{logsig}\left(\sum_{i=1}^{n_i}v_{ij}s_{ij}z_i(t)-b_{j1}^1\delta_{j1}^1\right)-\delta_{i1}^2\text{logsig}\left(b_{i1}^2\right)\right),\ \text{k=1,2,…, no.} \quad (1)$$

where $z_1,z_2,\ldots,z_{n_i}$ and $y_1,\ldots y_{n_o}$ are the inputs and outputs of neural network, respectively; n_i represents the number of inputs; n_o represents the number of outputs; n_h represents the number of hidden nodes; v_{ij} denotes the weight of the link between the jth hidden node and the ith input node; w_{jk} denotes the weight of the link between the kth output node and the jth hidden node; b_{j1}^1 and b_{i1}^2 denote the biases for the hidden and output nodes, respectively; s_{ij} represents the switch link between the ith input node and the jth hidden node; t_{jk} represents the switch link between the jth hidden node and the kth output node; δ_{i1} and δ_{j1} denote the bias switches of hidden and output nodes, respectively. In case the switch value is equal to 1, there is a link between two nodes from different layers, otherwise there is no link between these two nodes. Logsig (·) is a sigmoid function that is defined in Eq. (2):

$$\text{logsig}(\alpha)=\frac{1}{1+e^{-\alpha}} \qquad \alpha\in\Re. \quad (2)$$

For each dimension, the connection weight values are tuned to be within [-2, 2] and the link switch bit is either 0 or 1 as in [4]. Along with [4], a unit step function is introduced to each link that is defined as in Eq. (3):

$$\delta(\alpha)=\begin{cases}0 & \text{if} \quad \alpha<0.5\\ 1 & \text{if} \quad \alpha\geq 0.5\end{cases} \quad (3)$$

B. Electromagnetic Algorithm (EM) for tuning the neural networks

Nature inspired algorithms have proven to be an effective solution method for various optimization problems [17, 18]. Electromagnetic algorithm (EM) is a recent nature inspired population based metaheuristic algorithm introduced by Birbil and Fang [10]. EM simulates an attraction–repulsion mechanism of electromagnetic theory in exploring the multi-dimensional solution space of a given problem. Each point represents a solution and each solution is associated with a charge, which represents the quality of the solution. The solutions will exert some force (attraction or repulsion) on other solutions. The attraction-repulsion force is used to explore a solution search space. The main idea behind the attraction-repulsion is that, a bad quality solution repels other solutions from moving towards its direction. On the other hand, a good quality solution will attract other solutions to move towards its direction. EM has four steps as follows (see Fig. 1):

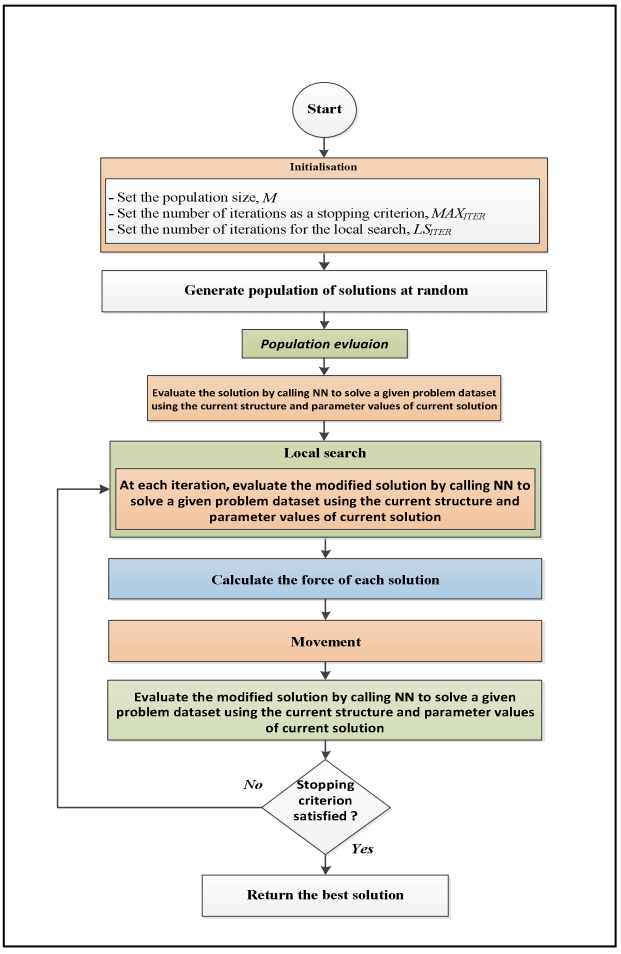

Fig 1. Flowchart of the EM algorithm for tuning neural networks (EM-NN).

1) ***Initialisation*****:** In this step, EM parameters are initialised i.e., the population size (M), number of iterations (MAX_{ITER}) and number of iterations for local search (LS_{ITER}). A population of solution is randomly created. In this study, a one-dimensional vector represents the solution. The size of the vector is equal to the number of decision variables in the given problem. Each cell in the vector represents one decision variable. In this study, the solution is divided into two parts: network structure, and connected weights. Two types of representations (binary and real) are used. Binary representation represents the network structure, whilst the real representation represents the regularization parameter and connected weights. For binary representation, the solution is generated by randomly assigning either zero or one for each variable. For real representation, the solution is randomly generated by assigning a random value to each decision variable within its upper and lower range as calculated in Eq. (4).

$$x^j_i = {}_Lx_i + rand\,[0,1] \bullet ({}_Ux_i - {}_Lx_i) \tag{4}$$

where *rand* return a random number between [0,1], ${}_Ux_i$ and ${}_Lx_i$ are the upper and lower bounds of the decision parameter, respectively. The proposed method is employed to learn the neural network model for approximating the given input-output relationships:

$$y^d(t) = g(z^d(t)),\ t=1,2,\ldots,n_d \tag{5}$$

where $z^d(t)= [z^d_1(t)\ \ z^d_2(t) \ldots z^d_{n_i}(t)]$ and $y^d(t)=[y^d_1(t)\ \ y^d_2(t) \ldots y^d_{n_o}(t)]$ represent the inputs and the desired outputs of an unknown nonlinear function g(.), respectively. n_d is the number of the input-output pairs. The quality of each solution is calculated by creating the neural network using current structure and the weight values encoded in the current solution using Eq. (6). The objective is to maximise the fitness value as defined in Eq. (6) and to minimise the error rate as in Eq. (7).

$$f = \frac{1}{1+\mathrm{err}} \tag{6}$$

with

$$\mathbf{err} = \sum_{k=1}^{n_o} \frac{\sum_{t=1}^{n_d} |\boldsymbol{y}_k^d(t) - \boldsymbol{y}_k(t)|}{\boldsymbol{n_d n_o}} \tag{7}$$

where the *err* represents mean absolute error (MAE). The sample of the solution representation is given in Fig.2.

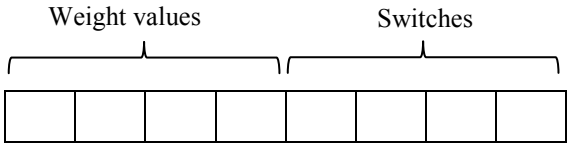

Fig 2.The representation of the solution showing weight and switches values.

2) ***Local search:*** In this step, a local search procedure is conducted to improve the quality of the solutions in the population that are generated in the initialisation step. The local search procedure has two parameters i.e., the number of iteration (LS_{ITER}) and the multiplier for the neighborhood search (δ), which is referred to as a changing amount when the solution is updated by adding or reducing by δ amount. At each iteration, the local search procedure generates a neighbourhood solution by adding or subtracting δ from the current solution for the real code representation. Both adding and subtracting have the same probability rate which is fixed to 0.5, i.e., if the generated random number is less than 0.5, then δ is subtracted from the current solution, otherwise, δ is added to the current solution (Birbil and Fang [10]). The neighbourhood solution is generated by flipping-flopping the decision variable value from zero to one or one to zero in the binary representation. If the neighbourhood solution is better than the current one, it will be accepted and becomes the current solution for the next iteration. Otherwise, it will be rejected. This process is repeated for a pre-defined number of iterations (LS_{ITER}).

3) ***Total force calculation:*** In this step, the charge of each solution based on the objective function is calculated. Based on the charge value, the solution will either follow attraction or repulsion. The charge of each solution is calculated as stated in Eq. (8):

$$q^i = \exp\left(-n\frac{f(x^i)-f(x^{best})}{\sum_{k=1}^{m}(f(x^k)-f(x^{best}))}\right), \forall \mathrm{i}. \tag{8}$$

This implies that the good quality solutions have a higher charge and consequently will have a strong attraction. Thus, the solutions will be attracted to the good quality solutions and will be repel from the bad quality solutions. Based on the calculated charge, the total force, *F*, exerted on each solution is calculated as shown in Eq. (9):

$$F^i = \sum_{j\neq i}^{m} \begin{cases} (x^J - x^i)\dfrac{q^i q^j}{\left\|x^j - x^i\right\|^2} & if\ f(x^j) < f(x^i) \\ (x^i - x^j)\dfrac{q^i q^j}{\left\|x^j - x^i\right\|^2} & f\ f(x^j) \geq f(x^i) \end{cases} \tag{9}$$

From Eq. (9), the calculated total force between the solutions is proportional to the product of the charges and is inversely proportional to the distance between the solutions.

*4) **Movement:*** Based on the calculated total force in Eq. (9), the solutions moved in the direction of the total force by a random step length using Eq. (10). The random step length takes a random value between zero and one. A positive force will move the solution towards the upper bound, whilst a negative force will move the solution towards a lower bound. Note that, the best quality solution will not move and will only attract other solutions. For the real code representation, the movement is calculated based on Eq. (10).

$$x^i = x^i + \lambda \frac{F^i}{\left\| F^i \right\|} (RNG)\ i=1,2,...,m \quad (10)$$

For the binary representation, the solution is moved based on the given probability as follows:

- For the attraction case: a solution is moved according to the attraction probability (*PA*), which is fixed to a large number between zero and one.
- For the repulsion case: a solution is repelled based on the repulsion probability (*PR*), which is fixed to small number between zero and one.

For example, assume that *PA*=0.8 and *PR* =0.01. If the state is attracting, the solution will be moved as follows: generate a random number *PR* between zero and one. If *PR* is less than *PA*, flip-flop the current decision variable. Otherwise, keep it unchanged.

III. EXPERIMENTAL RESULTS

In this section, the proposed algorithm EM-NN performance is analysed using five datasets obtained from University of California at Irvine (UCI) Machine Learning Repository (http://mlearn.ics.edu//MLRepository.html). These datasets are chosen because they represent a variety of important real world problems as well as many researchers have used them to evaluate the performance of their algorithms. The characteristics of these datasets are presented in Table 1.

TABLE 1 CHARACTERISTIC OF THE DATASETS USED

Dataset	*Number of attributes*	*Number of instances*	*Number of classes*
Australian	14	690	2
Breast cancer	9	699	2
German	30	1000	2
Iris	4	150	3
Pima	8	768	2 class

In order to determine the appropriate values for the EM parameters (i.e., population size, stopping criterion, and number of iterations for the local search), some preliminary experiments were conducted. The obtained preliminary results in terms of the classification accuracy are presented in Table 2, where the algorithm performs the best with the population size = 100 (presented in bold) on two datasets.

TABLE 2 RESULTS OF USING DIFFERENT POPULATION SIZE

Dataset	*Population size, M*			
	30	50	100	150
Pima	55.3	73.8	**96.5**	89.1
Breast cancer	68.2	87.7	**98.8**	90.6

We then tuned the number of iterations for the stopping criterion (MAX_{ITER}) and the local search (LS_{ITER}) by fixing the value of the population size as 100. We examined MAX_{ITER} with three different values i.e., 50, 100 and 150, and LS_{ITER} with 5, 10 and 15 as shown in Table 3.

TABLE 3 CLASSIFICATION ACCURACY WITH DIFFERENT VALUES OF MAX_{ITER} AND LS_{ITER}

MAX_{ITER}	LS_{ITER}	*Datasets*	
		Pima	Breast cancer
50	5	52.4	75.1
	10	73.8	87.7
	15	75.3	87.8
100	5	75.2	88.6
	10	**96.5**	**98.8**
	15	84.1	91.8
150	5	76.8	89.1
	10	89.1	90.6
	15	87.8	92.3

Table 3 presents the results of the classification accuracy. The best results are presented in bold. From Table 3, it is clearly shown that the best classification accuracy are obtained when MAX_{ITER} = 100 and LS_{ITER} = 10. The final parameter settings for the EM algorithm are presented in Table 4.

TABLE 4 PARAMETER SETTING

Parameter	*Value*
Population size, *M*	100
No. of iterations for the local search (LS_{ITER})	10
EM stopping condition (MAX_{ITER})	100

In our experiments, each dataset has been divided into training data (75%) and test set (25%) as in [8]. The experiments were executed for 50 times with different seed numbers on a PC with 2.4 GHz speed and 4 GB RAM under Windows 7 Operating System.

In this study, the performance of the proposed algorithm is compared with the state-of-the-art approaches, which are summarised in Table 5. Note that, these approaches are chosen based on their ability to produce the best-known results in the literature.

TABLE 5 ACRONYMS OF STATE-OF-THE-ART APPROACHES IN COMPARISONS

Symbol	***References***	***Description***
PSO+SVM	[19]	Particle swarm optimization for parameter determination and feature selection of support vector machines.
GA+SVM	[20]	A GA-based feature selection and parameters optimization for support vector machines.

SS-based ensemble	[21]	Enhancing the classification accuracy by scatter-search-based ensemble approach.
SA-SVM	[22]	Parameter determination of support vector machine and feature selection using simulated annealing approach.
Ensembles	[23]	Creating diversity in ensembles using artificial data.

The results of the comparison are presented in Table 6, where the best results are presented in bold. The comparison shows that our proposed approach (EM-NN) is able to obtain three new best results out of five tested datasets. The idea of attraction-repulsion mechanisms within the EM algorithm that aims to move the solutions toward the high quality solutions and avoid the solution from being trapped in a local optimum helps in achieving higher classification accuracy on the tested datasets. This shows that EM is one of the appropriate methods to simultaneously tune the structure and parameters of the NN for the classification problems.

TABLE 6 CLASSIFICATION ACCURACY FOR EM-NN AND OTHER METHODS

Datasets	*Algorithms*					
	EM-NN	PSO+ SVM	GA+ SVM	SS-based ensemble	SA-SVM	Ensembles
Australian	**93.56**	88.09	88.09	91.74	88.34	85.93
Breast cancer	98.80	97.95	94.23	**99.46**	97.95	96.31
German	**86.91**	79.00	84.24	85.49	-	-
Iris	98.23	98.00	97.56	**99.23**	-	94.67
Pima	**96.50**	80.19	82.98	83.92	80.19	-

'-' indicates that datasets have not been attempted.

The results obtained are further analysed by conducting a Friedman's multi comparison statistical tests with a significant interval of 95% ($\alpha = 0.05$) to see if there is any significant difference between EM-NN and the compared methods (PSO+SVM, GA+SVM and SS-based ensemble) [24]. Note that, only those methods that have been tested on all datasets are considered in this statistical test. If significant differences are detected (based on Friedman's test), post hoc methods (Holm's and Hochberg's tests) are conducted to obtain the adjusted *p*-values for each comparison between the control algorithm (the best-performing one in the comparison is the one that has 1st rank according to Friedman's test) and the rest of algorithms.

The *p*-value computed by the Friedman's test is 0.000, which is below the significant interval of 95% ($\alpha = 0.05$). This indicates that there is a significant difference among the observed results. Table 7 summarises the ranking obtained by the Friedman's test where EM-NN is ranked first.

TABLE 7 AVERAGE RANKING OF FRIEDMAN'S TEST.

#	*Algorithm*	*Ranking*
1	EM-NN	1.4
2	PSO+SVM	3.5
3	GA+SVM	3.5
4	SS-based ensemble	1.6

A post hoc method is conducted to obtain the adjusted *p*-values for each comparison between EM-NN (as the controlling method) and PSO+SVM, GA+SVM and SS-based ensemble algorithms. Table 8 shows the adjusted *p*-values which reveals that EM-NN is better than (PSO+SVM and GA+SVM) with $\alpha = 0.05$. However, there is no significant difference between EM-NN and SS-based ensemble (adjusted *p*-value is higher than 0.05). Nevertheless, the results reported in Table 6 clearly show that EM-NN is able to obtain three new best results out of five tested datasets as compared to SS-based ensemble that is only marginally better on two datasets.

TABLE 8 ADJUSTED *P*-VALUE OF THE COMPARED METHODS

Algorithm	*Unadjusted P*	*P Holm*	*P Hochberg*
PSO + SVM	0.010112	0.030337	0.020225
GA + SVM	0.010112	0.030337	0.020225
SS-based ensemble	0.806496	0.806496	0.806496

IV. CONCLUSION

In this paper, we have presented a methodology that handles the problem of tuning the structure and parameters of the three layers fully connected feed forward neural network with link switches based on the principle of the electromagnetic-like mechanism. The performance of the approach is tested on classification of benchmark datasets. We have made comparison with a set of state-of-the-art approaches from the literature. This approach produces three best results and is consistently good across the all the benchmark problems in comparison with other approaches studied in the literature. With the help of the electromagnet that moves sample points (solutions) towards a high quality solution while avoiding the local optima by utilising a calculated force value, our approach is capable of finding better solutions for the classification problems. We are confident that a significant contribution in producing high quality solutions to the classification problems has been made.

REFERENCES

[1] M. M. Gupta, *et al.*, *Static and Dynamic neural networks*: Wiley Online Library, 2003.

[2] A. Cochocki and R. Unbehauen, *Neural networks for optimization and signal processing*: John Wiley & Sons, Inc., 1993.

[3] T. Kavzoglu, "Determining optimum structure for artificial neural networks," in *Proceedings of the 24 th Annual Technical Conference and Exhibition of the Remote Sensing Society*, 1999, pp. 675-682.

[4] F. H. F. Leung, *et al.*, "Tuning of the structure and parameters of a neural network using an improved genetic algorithm," *IEEE Transactions on Neural Networks, ,* vol. 14, pp. 79-88, 2003.

[5] I. Tsoulos, *et al.*, "Neural network construction and training using grammatical evolution," *Neurocomputing,* vol. 72, pp. 269-277, 2008.

[6] J. Ilonen, *et al.*, "Differential evolution training algorithm for feed-forward neural networks," *Neural Processing Letters,* vol. 17, pp. 93-105, 2003.

[7] J. T. Tsai, *et al.*, "Tuning the structure and parameters of a neural network by using hybrid Taguchi-genetic algorithm," *IEEE Transactions on Neural Networks,,* vol. 17, pp. 69-80, 2006.

[8] C. Dai, *et al.*, "Seeker optimization algorithm for tuning the structure and parameters of neural networks," *Neurocomputing,* vol. 74, pp. 876-883, 2011.

[9] L. Zhao and F. Qian, "Tuning the structure and parameters of a neural network using cooperative binary-real particle swarm optimization," *Expert Systems with Applications,* vol. 38, pp. 4972-4977, 2011.

[10] Ş. İ. Birbil and S. C. Fang, "An electromagnetism-like mechanism for global optimization," *Journal of global optimization,* vol. 25, pp. 263-282, 2003.

[11] S. Abdullah, *et al.*, "A hybridization of electromagnetic-like mechanism and great deluge for examination timetabling problems," *Hybrid Metaheuristics,* pp. 60-72, 2009.

[12] A. Yurtkuran and E. Emel, "A new hybrid electromagnetism-like algorithm for capacitated vehicle routing problems," *Expert Systems with Applications,* vol. 37, pp. 3427-3433, 2010.

[13] R. Tavakkoli-Moghaddam, *et al.*, "A hybridization of simulated annealing and electromagnetic-like mechanism for job shop problems with machine availability and sequence-dependent setup times to minimize total weighted tardiness," *Soft Computing-A Fusion of Foundations, Methodologies and Applications,* vol. 13, pp. 995-1006, 2009.

[14] P. Wu, *et al.*, "An electromagnetism algorithm of neural network analysis—an application to textile retail operation," *Journal of the Chinese Institute of Industrial Engineers,* vol. 21, pp. 59-67, 2004.

[15] X. J. Wang, *et al.*, "Electromagnetism-like mechanism based algorithm for neural network training," *Advanced Intelligent Computing Theories and Applications. With Aspects of Artificial Intelligence,* pp. 40-45, 2008.

[16] Q. Wu, *et al.*, "Training neural networks by electromagnetism-like mechanism algorithm for tourism arrivals forecasting," in *Bio-Inspired Computing: Theories and Applications (BIC-TA), 2010 IEEE Fifth International Conference on*, 2010, pp. 679-688.

[17] M. Hadwan, *et al.*, "A harmony search algorithm for nurse rostering problems," *Information Sciences,* vol. 233, pp. 126-140, 2013.

[18] A. M. Turky and S. Abdullah, "A multi-population harmony search algorithm with external archive for dynamic optimization problems," *Information Sciences,* 2014.

[19] S. W. Lin, *et al.*, "Particle swarm optimization for parameter determination and feature selection of support vector machines," *Expert Systems with Applications,* vol. 35, pp. 1817-1824, 2008.

[20] C. L. Huang and C. J. Wang, "A GA-based feature selection and parameters optimizationfor support vector machines," *Expert Systems with Applications,* vol. 31, pp. 231-240, 2006.

[21] S. C. Chen, *et al.*, "Enhancing the classification accuracy by scatter-search-based ensemble approach," *Applied Soft Computing,* vol. 11, pp. 1021-1028, 2011.

[22] S. W. Lin, *et al.*, "Parameter determination of support vector machine and feature selection using simulated annealing approach," *Applied Soft Computing,* vol. 8, pp. 1505-1512, 2008.

[23] P. Melville and R. J. Mooney, "Creating diversity in ensembles using artificial data," *Information Fusion,* vol. 6, pp. 99-111, 2005.

[24] S. García, *et al.*, "Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power," *Information Sciences,* vol. 180, pp. 2044-2064, 2010.