ORIENTAL JOURNAL OF
COMPUTER SCIENCE & TECHNOLOGY
An International Open Free Access, Peer Reviewed Research Journal
Published By: **Oriental Scientific Publishing Co., India.**
www.computerscijournal.org

ISSN: 0974-6471
December 2012,
Vol. 5, No. (2):
Pgs. 231-239

Using Artificial Bee Colony Algorithm for MLP Training on Software Defect Prediction

SOLMAZ FARSHIDPOUR[1] and FARSHID KEYNIA[2]

[1]Department of Science and Research branch , Islamic Azad University, Kerman, Iran.
[2]Graduate University of Advanced Research, Kerman, Iran.

(Received: August 04, 2012; Accepted: October 15, 2012)

ABSTRACT

Defects in software systems continue to be a major problem. Defect prediction is an important topic in software quality research and could help on planning, controlling and executing software development activities. Nowadays, computer scientists have shown the interest in the study of social insect's behaviour in neural networks area for solving different prediction problems.Chief among these is the Artificial Bee Colony (ABC) algorithm. This paper investigates the use of ABC algorithm that simulates the intelligent foraging behaviour of a honey bee swarm. Multilayer Perceptron (MLP) trained with the standard back propagation algorithm normally utilises computationally intensive training algorithms. One of the crucial problems with the backpropagation (BP) algorithm is that it can sometimes yield the networks with suboptimal weights because of the presence of many local optima in the solution space. To overcome ABC algorithm used in this work to train MLP learning the complex behaviour of software defect prediction data trained by BP, the performance of MLP-ABC is benchmarked against MLP training with the standard BP. The experimental result shows that MLP-ABC performance is better than MLP-BP.

Key words: Artificial Bee Colony algorithm, Backpropagation, Multilayer Perceptron

INTRODUCTION

Quality of software is often measured by the number of defects in the final product. Minimizing the number of defects—maximizing software quality—requires a thorough testing of the software in question. On the other hand, testing phase requires approximately 50% of the whole project schedule[1]. This means testing is the most expensive, time and resource consuming phase of the software development lifecycle. An effective test strategy should therefore consider minimizing the number of defects while using resources efficiently. Defect prediction models are helpful tools for software testing. Accurate estimates of defective modules may yield decreases in testing times and project managers may benefit from defect predictors in terms of allocating the limited resources[2].Many modeling techniques have been developed and applied for software quality prediction.These include, logistic regression[3], discriminant analysis[4,5], the discriminative power techniques[6], Optimized Set Reduction[7], artiûcial neural network[8], fuzzy classiûcation[9], Bayesian

Belief Networks[10] , genetic algorithms[11], classiûcation trees[12-15], and recently Dempster-Shafer Belief Networks[16]. For all these software quality models, there is a tradeoff between the defect detection rate and the overall prediction accuracy. Thus, a performance comparison of various models, if based on only one criterion (either the defect detection rate or the overall accuracy), may render the comparison only partially relevant. A model can be considered superior over its counterparts if it has both a higher defect detection rate, and a higher overall accuracy.

Artificial Neural Networks (ANNs) are the most novel and powerful artificial tool suitable for solving combinatorial problems such as prediction and classification[17-20]. ANNs are being used extensively for solving universal problems intelligently like continuous,discrete, and clustering [21,22]. ANNs are being applied for different optimisation and mathematical problems such as classification, object and image recognition, signal processing and prediction. Different techniques are used for optimal network performance for training ANNs such as evolutionary algorithms (EA), genetic algorithms (GA), partial swarm optimisation (PSO), differential evolution (DE), ant colony, and backpropagation algorithm[23-29].These techniques are used for initialisation of optimal weights, parameters, activation function, and selection of proper network. Backpropagation (BP) algorithm is accepted algorithm used for MLP training .The main task of BP algorithm is to update the network weights for minimising output error using backpropagation processing because the accuracy of any approximation depends on the selection of proper weights for the neural networks (NNs). It has high success rate in solving many complex problems but it still has some drawbacks especially when setting parameter values like initial values of connection weights, value for learning rate, and momentum. If the network topology is not carefully selected, the NNs algorithm can get trapped in local minima or it might lead to slow convergence or even network failure. In order to overcome the disadvantages of standard BP, many global optimisation population- based techniques have been proposed for MLP training such as EA, GA, improved GA, DE, BP-ant colony, and PSO[11-13,16,17].Artificial Bee Colony (ABC) algorithm is a population- based algorithm that can provide the best possible solutions for different mathematical problems by using inspiration techniques from nature[30]. A common feature of population-based algorithms is that the population consisting of feasible solutions to the difficulty is customised by applying some agents on the solutions depending on the information of their robustness. Therefore, the population is encouraged towards improved solution areas of the solution space. Population-based optimisation algorithms are categorised into two sections namely evolutionary algorithm (EA) and SI-based algorithm[31,32].In EA,the major plan underlying this combination is to take the weight matrices of the ANNs as individuals, to change the weights by means of some operations such as crossover and mutation, and to use the error produced by the ANNs as the fitness measure that guides selection. In SI-based algorithm, ABC has the advantage of global optimisation and easy recognition. It has been successfully used in solving combinatorial optimisation problems such as clustering and MLP training for XOR problem[33,34]. This is one of the self-organising and highly approachable solutions for different computational and mathematical problems. ABC algorithm is an easily understandable technique for training MLP on classification problems[35]. This algorithm uses randomly selected natural techniques with colony to train NNs by optimal weights . In this study, ABC algorithm is used successfully to train MLP on software defect data for prediction task. The performance of the algorithm is compared with standard BP algorithm , KNN and Naïve Bayes algorithm. This paper is organised as follows: A brief review on ANN and ABC and BP algorithms is given in Section 2 and Section 3, respectively. The proposed ABC algorithm and the training and testing of the network using ABC algorithm are detailed in Section 4. Section 5 contains the prediction of software defect. Results and discussion are discussed in Section 6. Finally, the paper is concluded in Section 7.

Artificial Neural Networks

Training of Mlp Neural Networks

MLP was introduced in 1957 to solve different combinatorial problems[36]. MLP, which is also known as feed forward neural networks was first introduced for the non-linear XOR, and was then successfully applied to different combinatorial problems. MLP is mostly used for information processing and pattern recognition in prediction of seismic activities. In this section, MLP's characteristics and interaction with the seismic signals are explained. MLP works as a universal approximation in which inputs signal propagates in forward direction. It is highly used and tested with different problems such as in prediction and function approximation[1,3,4,37]. Figure 1 shows the architecture of MLP with two hidden layers, one output layer, and one input layer.

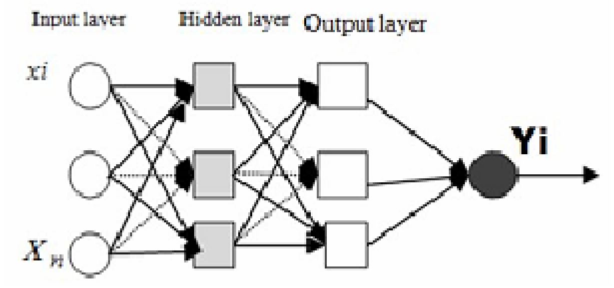


Fig. 1: Multi Layer Prediction Neural Network

$$Y_i = f_i\left(\sum_{j=1}^{n} w_{ij} x_j + b_i\right) \quad ...(1)$$

where Yi is the output of the node, Xi is the jth input to the node, w_{ij} is the connection weight between the input node and output node, èi is the threshold (or bias) of the node, and fi is the node transfer function. Usually, the node transfer function is a non-linear function such as a sigmoid function, a Gaussian function, and etc. The network error function E will be minimised as

$$E(w(t)) = \frac{1}{n}\sum_{j=1}^{n}\sum_{k=1}^{k}(d_k - O_t) \quad ... (2)$$

where E(w(t)) is the error at the *t*th iteration; w(t) is the weights in the connections at the t th iteration;d_k is the desired output node; ok is the actual value of the kth output node; K is the number of output nodes; and n is the number of patterns. T is the optimisation target to minimise the objective function by optimising the network weights w(t).

Artificial Bee Colony

Swarm Intelligence

Since the last two decades, swarm intelligence (SI) has been the focus of many researches because of its unique behaviour inherent from the social insects[13,14,22,25]. Bonabeau has defined the SI as "any attempt to design algorithm or distributed problem-solving devices inspired by the collective behaviour of social insect colonies and other animal societies"[38]. He mainly focused on the behaviour of social insects alone such as termites, bees, wasps, and different ant species. However, swarm can be considered as any collection of interacting agents or individuals. Ants are individual agents of ACO . An immune system can be considered as a group of cells and molecules as well as a crowd is a swarm of people . PSO and ABC are popular population-based stochastic optimisation algorithms adapted for the optimisation of non-linear functions in multidimensional space[39].

Bees in nature

A colony of honey bees can extend itself over long distances (more than 10 km) and in multiple directions simultaneously to exploit a large number of food sources[40,41]. A colony prospers by deploying its foragers to good fields. In principle, flower patches with plentiful amounts of nectar or pollen that can be collected with less effort should be visited by more bees, whereas patches with less nectar or pollen should receive fewer bees[42,43]. The foraging process begins in a colony by scout bees being sent to search for promising flower patches. Scout bees move randomly from one patch to another. During the harvesting season, a colony continues its exploration, keeping a percentage of the population as scout bees[41]. When they return to the hive, those scout bees that found a patch which is rated above a certain quality threshold (measured as a combination of some constituents, such as sugar content) deposit their nectar or pollen and go to the "dance floor" to perform a dance known as the "waggle dance"[40]. This mysterious dance is essential for colony communication, and contains three pieces of information regarding a flower patch: the direction in which it will be found, its distance from the hive and its quality rating (or fitness)[40,43]. This information helps the colony to send its bees to flower patches precisely,without using guides or maps.Each individual's knowledge of the outside environment is gleaned solely from the waggle dance. This dance enables the colony to evaluate the relative merit of different patches according to both the quality of the food they provide and the amount of energy needed to harvest it[43]. After waggle dancing on the dance floor, the dancer (i.e. the scout bee) goes back to the flower patch with follower bees that were waiting inside the hive. More follower bees are sent to more promising patches. This allows the colony to gather food quickly and efficiently.While harvesting from a patch, the bees monitor its food level. This is necessary to decide upon the next waggle dance when they return to the hive[43]. If the patch is still good enough as a food source,then it will be advertised in the waggle dance and more bees will be recruited to that source.

Artificial Bee Colony Algorithm

Artificial Bee Colony algorithm (ABC) was proposed for optimisation, classification, and NNs problem solution based on the intelligent foraging behaviour of honey bee swarm[20-22]Therefore, ABC is more successful and most robust on multimodal functions included in the set with respect to DE, PSO, and GA[16,21,23]. ABC algorithm provides solution in organised form by dividing the bee objects into different tasks such as employed bees, onlooker bees, and scout bees.

Employed bees

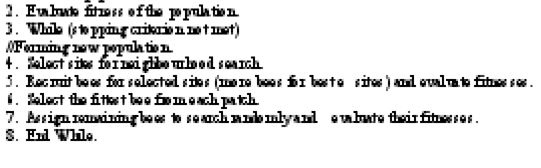
Employed bees use multidirectional search space for food source with initialisation of thearea. They get information and all possibilities to find food source and solution space. Sharing of information with onlooker bees is performed by employee bees. An employed bee produces a modification on the source position in her memory and discovers a new food source position. Provided that the nectar amount of the new source is higher than that of the previous source, the employed bee memorizes the new source position and forgets the old one.

Onlooker bees

Onlooker bees evaluate the nectar amount obtained by employed bees and choose a food source depending on the probability values calculated using the fitness values. For this purpose, a fitness-based selection technique can be used. Onlooker bees watch the dance of hive bees and select the best food source according to the probability proportional to the quality of that food source.

Scout bees

Scout bees select the food source randomly without experience. If the nectar amount of a food source is higher than that of the previous source in their memory, they memorise the new position and forget the previous position. Whenever employed bees get a food source and use the food source very well again, they become scout bees to find new food source by memorising the best path. Figure 2 shows the pseudo code for the algorithm in its simplest form. The algorithm requires a number of parameters to be set, namely: number of employed bees (n), number of sites selected out of n visited sites(scout bees) (m), number of bees recruited for best site (nep), number of bees recruited for the other selected sites (nsp1, nsp2 ,..), initial size of patches (ngh) which includes site and its neighbourhood and stopping criterion.



```
1. Initialise population with random solutions.
2. Evaluate fitness of the population.
3. While (stopping criterion not met)
//Forming new population.
4. Select sites for neighbourhood search.
5. Recruit bees for selected sites (more bees for best e  sites) and evaluate fitnesses.
6. Select the fittest bee from each patch.
7. Assign remaining bees to search randomly and   evaluate their fitnesses.
8. End While.
```

Fig. 2: Pseudo code of the Artificial Bee Colony algorithm

The algorithm starts with the n employed bees being placed randomly in the search space.The fitnesses of the sites visited by the bees are evaluated in step 2. The fitnesses can be found by the following formula:

$$\mathrm{fit}_i = \begin{cases} \dfrac{1}{1+f_i} & f_i >= 0 \\ 1+\mathrm{abs}(f_i) & f_i < 0 \end{cases} \quad ...(3)$$

In step 4, bees that have the highest probability fitnesses are chosen as "selected bees" and sites visited by them are chosen for neighbourhood search. the probability values pi for the solutions by means of their fitness values by using formula:

$$p_i = \frac{\mathrm{fit}_i}{\sum_{k=1}^{N} \mathrm{fit}_n} \quad ...(4)$$

Then, in steps 5 and 6, the algorithm conducts searches in the neighbourhood of the selected sites, assigning more bees to search near to the best sites. The bees can be chosen directly according to the fitnesses associated with the sites they are visiting.Alternatively, the fitness values are used to determine the probability of the bees being selected.Searches in the neighbourhood of the best sites which represent more promising solutions are made more detailed by recruiting more bees to follow them than the other selected bees. Together with scouting,this differential recruitment is a key operation of the Bees Algorithm.However, in step 6, for each patch only the bee with the highest fitness will be selected to form the next bee population. In nature, there is no such a restriction.This restriction is introduced here to reduce the number of points to be explored. In step 7, the remaining bees in the population are assigned randomly around the search space scouting for new potential solutions. These steps are repeated until a stopping criterion is met. At the end of each iteration, the colony will have two parts to its new population representatives from each selected patch and other scout bees assigned to conduct random searches.

The Proposed Framework For MLP-ABC

The proposed flowchart of the ABC algorithm for software defect prediction is given in Figure 3.

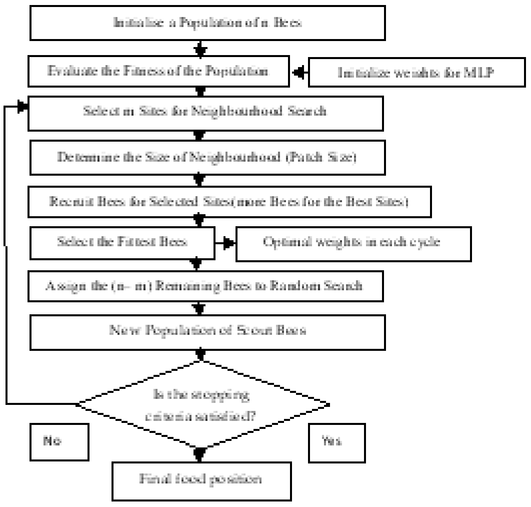

Initialise a Population of n Bees
Evaluate the Fitness of the Population
Initialize weights for MLP
Select m Sites for Neighbourhood Search
Determine the Size of Neighbourhood (Patch Size)
Recruit Bees for Selected Sites(more Bees for the Best Sites)
Select the Fittest Bees
Optimal weights in each cycle
Assign the (n– m) Remaining Bees to Random Search
New Population of Scout Bees
Is the stopping criteria satisfied?
No
Yes
Final food position

Fig. 3: Initialise a Population of n Bees

The basic idea of ABC scheme is to use agents of bees to search the best combination of weights for network. All steps in finding optimal weights for network are shown in proposed ABC algorithm framework in Figure 3. The figure shows how to find and select the best weights .The initialisation of weights was compared with output and the best weight cycle was selected by scout bees' phase. The bees (employed bees, onlooker bees) would continue searching until the last cycle to find the best weights for networks. The food source of which the nectar was neglected by the bees was replaced with a new food source by the scout bees. Every bee (employed bees, onlooker bees) would produce new solution area for the network and the Greedy Selection would decide the best food source position.The proposed frameworks can easily train software defect data for prediction task by finding optimal network weights for MLP.

Prediction of Software Defect

In this research,software defect data of NASA promise was selected for training and testing. We use three real-world data sets for our experiments.our data sets are publicly available from the Promise Software Engineering Repository (PSER) at http://promise.site.uottawa.ca/ SERepository .Our data sets come from a C programming application called CM1,KC1 and KC2 .The Data for data sets, comes from McCabe and Halstead features extractors of source code. The McCabe and Halstead measures are "module"-based where a "module" is the smallest unit of functionality. In C or Smalltalk, "modules" would be called "function" or "method" respectively. There are 22 attributes in our data sets .The 22 attributes consist of 5 different lines of code measure, 3 McCabe metrics, 4 base Halstead measures, 8 derived Halstead measures, one branch-count variable, and one binary decision variable denoting whether defects were found in a module or not.Table 1 details the different attributes available in the CM1 ,KC1 and KC2 product datasets. The networks were tested for the prediction of software defects by MLP-ABC and MLP-BP.

Table 1: Data description of attributes for the CM1, KC1 and KC2 products

Attribute	Description
1.loc	McCabe's line count of code
2.v(g)	McCabe "cyclomatic omplexity"
3.ev(g)	McCabe "essential complexity"
4.iv(g)	McCabe "design complexity"
5.n	Halstead total operators +operands
6.v	Halstead "volume"
7.l	Halstead "program length"
8.d	Halstead "difficulty"
9.i	Halstead "intelligence"
10.e	Halstead "effort"
11.b	Halstead
12.t	Halstead's time estimator
13. lOCode	Halstead's line count
14.lOComment	Halstead's count of lines of comments
15.lOBlank	Halstead's count of blank lines
16.lO	CodeAndComment
17.uniq_Op	unique operators
18.uniq_Opnd	unique operands
19.total_Op	total operators
20.total_Opnd	total operands
21.branchCount	of the flow graph
22.defects	module has/has not one or more

Simulation Results

In order to evaluate the performance of the proposed ABC to train MLP for benchmark software defect data scheme for prediction techniques, simulation experiments were performed on a 1.66 GHz Core 2 Duo Intel Workstation with 1GB RAM using Matlab software. The comparison of standard BP training and ABC algorithms is discussed based on the simulation results implemented in Matlab. The data were divided into two datasets: 70% for training and 30% for testing. The weight values of MLP-ABC were initialised, evaluated, and fitted using ABC algorithm, while the weight values of MLP-BP were adjusted from the range [1,-1] randomly. The MLP was trained with one input layer(21 nodes),two hidden layers(10 nodes),and one output layer(1 node). MLP-BPwas stopped on maximum cycle number(MCN). All the simulation parameters were taken as given in Table 2.Besides that, the Accuracy and Precision were selected for testing.The simulation results showed the effectiveness and efficiency of ABC algorithm.

Table 2: Artificial Bee Colony algorithm parameters

parameters	
n	50
m	3
nep	25
nsp1	15
nsp2	10
ngh	5
stopping criteria	50(maximum cycle number)

Finally,Accuracy and Precision were calculated for MLP-BP and MLP-ABC algorithms.These measures are based on the classification results in the confusion matrix(Figure 4).

	True class	
Classified as	Defective	Not Defective
Defective	TP	FP
Not Defective	FN	TN

Fig. 4: Confusion matrix

Each cell in the Figure 4 compares the overlap (or lack of) between the predicted and actual set. A defect can be classiûed as defective when it truly is defective (true positive, TP); it can be classiûed as defective when actually it is not defective (false positive, FP); it can be classiûed as not defective when it is actually defective (false negative, FN); orit can be classiûed as not defective and it truly is not defective (true negative, TN). Accuracy and Precision can be found by the following formulae:

$$\textbf{Precision} = \frac{TP}{TP+FP} \quad ...(5)$$

$$\textbf{Accuracy} = \frac{TP+TN}{TP+FP+TN+FN} \quad ...(6)$$

The simulation results of training and testing MLP-BP and MLP-ABC are given in Table 3 and Table4.

Table 3: Results of Accuracy

Dataset	MLP-BP (%)	MLP-ABC (%)
CM1	77.87	79.51
KC1	73.47	74.07
KC2	80.86	82.78

Table 4: Results of Precision

Dataset	MLP-BP (%)	MLP-ABC (%)
CM1	78.47	80.00
KC1	74.91	77.13
KC2	85.25	87.15

The simulation results showed the effectiveness and efficiency of ABC algorithm.

CONCLUSION

ABC algorithm has the powerful ability of searching global optimal solution. So, the proper weights of the algorithms may speed up the initialisation and improve the prediction accuracy of the trained NNs. The simulation results show that the proposed ABC algorithm can successfully train software defect data for prediction purpose, which further extends the quality of the given approach. The performance of ABC is compared with the traditional BP algorithm. ABC shows significantly higher results than backpropagation during experiment. ABC also shows higher accuracy and Precision in prediction.The proposed frameworks have successfully predicted the software defects.

REFERENCES

1. Halpern, S.D., Ubel, P.A., Caplan, A.L. Solid-organ transplantation in HIV-infected patients. *N. Engl. J. Med.,* 347(4): 284-7 (2002).
2. Tahat,B.V., Korel,B.,Bader,A.Requirement-Based Automated Black-Box Test Generation. In Proceedings of 25th Annual International Computer Software and Applications Conference, Chicago, Illinois, pp489-495 (2001).
3. Song,O,Shepperd,M.,Cartwright,M. ,Mair,C.Software Defect Association Mining and Defect Correction Effort Prediction, *IEEE Transactions on Software Engineering.* 32: 69-82 (2006).
4. Basili,V. R ., Briand, L. C., Melo, W. A Validation of Object-oriented Design (1996).
5. Khoshgoftaar, T.M, Allen,E.B,Kalaichelvan, K.S., Goel, N.Early Quality Prediction: A Case Study in Telecommunications, IEEE Software, **13**: 65-71 (1996).
6. Munson, J. C.,Khoshgoftaar,T. M. The Detection of Fault-prone Programs,IEEE Trans. *Software Eng.,* **18**(5): 423-433 (1992).
7. Schneidewind, N. F.Methodology For Validating Software Metrics, *IEEE Trans.Software Eng.* **18**(5): 410-422 (1992).
8. Briand, L. C.,Basili, V. R., Hetmanski, C. J.Developing Interpretable Models, with Optimaized Set Reduction for Identifying High-Risk Software Components, *IEEE Trans. Software Eng.* **19**(11): 1028-1044 (1993).
9. Khoshgoftaar, T.M.,Lanning, D. L..A Neural Network Approach for Early Detection of Program Modules Having High Risk in the Maintenance Phase, *Journal of Systems and Software,* **29**(1): 85-91 (1995).
10. Ebert,.C, Classiûcation Techniques for Metric-based Software Development, *Software Quality Journal,* 5(4): 255-272 (1996).
11. Fenton, N. , NeilSoftware, M. , Metrics and Risk, Proc. 2nd European Software Measurement Conference, TI-KVIV, Amsterdam (1999).
12. Azar, D., Bouktif, S., Sahraoui, H., Precup, D. Combining AndAdapting Software Quality Predictive Models By Genetic Algorithms, Proc. 17th IEEEInternational Conference on Automated Software Engineering (ASE2002), pp285 (2002).
13. Gokhale, S., Lyu, M. R.Regression tree modeling for the prediction of software quality, Proc. the Third ISSAT International Conference on Reliability and Quality in Design, Anaheim, CA, pp31–36 (1997).
14. Khoshgoftaar, T. M., Seliya, N. Tree-Based Software Quality Estimation Models For Fault Prediction, Proc. the Eighth IIIE Symposium on Software Metrics (METRICS'02), pp203 (2002).
15. Selby, R. W., Porter,A.,Learning from Examples: Generation and Evaluation of Decision Trees for Software Resource Analysis, *IEEE Trans. Software Eng.,* **4**(12): 1743-1756 (1988).
16. Troster, J., Tian, J.Measurement and Defect Modeling for a Legacy Software System, Annals of Software Eng., **1**: 95- 118 (1995).
17. Guo, L., Cukic, B., Singh, H., Predicting Fault Prone Modules by the DempsterShafer Belief Networks, Proc. 18th IEEE International Conference on Automated Software Engineering (ASE 2003) (2003).
18. Panakkat, A., Adeli, H.Neural network models for earthquake magnitude prediction using multiple seismicity indicators*, International Journal of Neural Systems,* **17**(1): 13-33 (2007).
19. Liao, S.-H., Wen, C. H. Artificial neural networks classification and clustering of methodologies and applications - literature analysis from 1995 to 2005. *Expert Systems with Applications.* **32**(1): 1-11 (2007).
20. Ghazali, R., Jaafar Hussain, A. Non-stationary and stationary prediction of financial time series using dynamic ridge polynomial neural network. *Neurocomputing,* **72**(10-12): 2359-2367 (2009).
21. Connor, J.,Atlas, L., Recurrent Neural Networks andTime Series Prediction, IEEE International Joint conference on Neural

networks, New York, USA, I 301- I 306 (2005).

22. Du, K. L., Clustering: A neural network approach.Neural Networks. **23**(1): 89-107 (2010).
23. Carrasco, M. P., Pato, M. V. A comparison of discrete and continuous neural network approaches to solve the class/teacher timetabling problem. *European Journal of Operational Research.* **153**(1): 65-79 (2004).
24. Leung, C., Chow, W.S.A Hybrid Global Learning Algorithm Based on Global Search and Least Squares Techniques for Backpropagation Networks, Neural Networks,1997. In: International Conference on, **3**: pp 1890-1895 (1997).
25. Yao, X . Evolutionary artificial neural networks. *International Journal of Neural Systems,* **4**(3): 203-222 (1993).
26. Mendes, R. ,Cortez, P., Rocha, M., Neves, J. Particle swarm for feedforward neural network training. In: Proceedings of the *International Joint Conference on Neural Networks,* **2**: 1895–1899 (200) .
27. VanderBerghand, F.,Engelbrecht, A.Cooperative learning in neural networks using particle swarm optimizers. *South African Computer Journal,*; **26**: 84-90 (2000).
28. Ilonen, J .,Kamarainen, J.I., Lampinen, J. Differential Evolution Training Algorithm for Feed-Forward Neural Networks (2007).
29. B,Y., He, X. Training Radial Basis Function Networks with Differential Evolution, Granular Computing, 2006. In: IEEE International Conference on, 369–372 (2006).
30. Liu, Y., Wu, G.Evolving Neural Networks Using the Hybrid of Ant Colony Optimization and BP Algorithms. Advances in Neural Networks - ISNN 2006 (2006).
31. Karaboga, D., Akay, B., A comparative study of Artificial Bee Colony algorithm *Applied Mathematics and Computation.***214**: 108-132 (2009).
32. Eiben, A.E. , Smith, J.E. Introduction to Evolutionary Computing, Springer (2003).
33. Eberhart, R.C., Shi, Y. , Kennedy, J. Swarm Intelligence, Morgan Kaufmann (2001).
34. Karaboga, D. , Aka, B. “Artificial Bee Colony (ABC) Algorithm on Training Artificial Neural Networks” signals Processing and Communications Applications, SIU 2007. IEEE 15th (2007).
35. Zhang, C., Ouyang, D. An artificial bee colony approach for clustering.” Expert Systems with Applications , 2010;37(7): 4761-4767.
36. Karaboga, D.An idea based on honey Bee Swarm for Numerical Optimization Technique. Report-06, Erciyes University Engineering Faculty,Computer Engineering Department (2005).
37. Rosenblatt, F.A Probabilistic Model for Information Storage and Organization in the Brain, *Cornell Aeronautical Laboratory,* 65: pp.386-108 (1958).
38. Michele, R.Artificial neural network for tsunami forecasting*, Journal of Asian Earth Sciences* **36**: 29-37 (2009).
39. Bonabeau, E. ,Dorigo, M. , Theraulaz, G. Swarm Intelligence:From Natural to Artificial Systems, Oxford University Press, NY (1999).
40. Eberhart, R.C., Shi, Y. , Kennedy, J. Swarm Intelligence, Morgan Kaufmann (2001).
41. Von Frisch, K. Bees: Their Vision, Chemical Senses and Language. (Revised edn) Cornell University Press, N.Y., Ithaca (1976).
42. Seeley, TD. The Wisdom of the Hive: The Social Physiology of Honey Bee Colonies. Massachusetts: Harvard University Press, Cambridge, (1996).
43. Bonabeau, E .,Dorigo, M. , Theraulaz, G. Swarm Intelligence: from Natural to Artificial Systems. Oxford University Press, New York (1999).
44. Camazine, S.,Deneubourg, J., Sneyd , J., Theraula, G ., Bonabeau , E. Self-Organization in Biological Systems. Princeton: Princeton University Press (2003).