International Journal of Computer and Communication Technology

Volume 4 | Issue 4 Article 7

October 2013

Optimization of ANN Structure Using Adaptive PSO & GA and Performance Analysis Based on Boolean Identities

Amaresh Sahu
Computer Science Department, Siksha O Anusandhan University, Bhubaneswar, India,
amaresh_sahu@yahoo.com

Sushanta Panigrahi
ctcsushanta@gmail.com

Sabyasachi Pattnaik
I & CT, Fakir Mohan University Vyasa Vihar, Balasore Odisha, India, Spattnaik40@yahoo.co.in

Optimization of ANN Structure Using Adaptive PSO & GA and Performance Analysis Based on Boolean Identities

Amaresh Sahu [1],Sushanta Kumar Panigrahi [2] , Sabyasachi Pattnaik[3]
Computer Science Department, Siksha O Anusandhan University, Bhubaneswar, India [1]
ICT Department ,Fakir Mohan University , Balasore, India [2,3]
E-mail : amaresh_sahu@yahoo.com[1] , spattnaik40@yahoo.co.in[2], ctcsushanta@gmail.com [3]

Abstract—**In this paper, a novel heuristic structure optimization technique is proposed for Neural Network using Adaptive PSO & GA on Boolean identities to improve the performance of Artificial Neural Network (ANN). The selection of the optimal number of hidden layers and nodes has a significant impact on the performance of a neural network, is decided in an adhoc manner. The optimization of architecture and weights of neural network is a complex task. In this regard the use of evolutionary techniques based on Adaptive Particle Swarm Optimization (APSO) & Adaptive Genetic Algorithm (AGA) is used for selecting an optimal number of hidden layers and nodes of the neural controller, for better performance and low training errors through Boolean identities. The hidden nodes are adapted through the generation until they reach the optimal number. The Boolean operators such as AND, OR, XOR have been used for performance analysis of this technique.**

***Keywords—ANN*, adaptive PSO, adaptive GA, structure optimization, boolean identities**

I. INTRODUCTION

Increasing productivity, decreasing costs, and maintaining high product quality at the same time are the main challenges of industries today. The proper selection of machining parameters is an important step towards meeting these goals and thus gaining a competitive advantage. Several researchers have attempted to design and meet these goals. Artificial Neural Network (ANN) is the best option.

ANN is an information processing paradigm that is inspired by the way biological neural network or nervous systems process information, such as brain. The processing power of ANN allows the network to learn and adapt. Time to time, the application of Artificial Neural Network (ANN) in industries has been accepted very rapidly. In the growing interest of using ANN is to assist building model structure with particular characteristics such as the ability to learn or adapt, to organize or to generalize data. Up-to-date designing optimal network architecture is made by a human expert and requires a tedious trial and error process. Especially automatic determination of the optimal number of hidden layers and nodes in each hidden layer is the most critical task [1, 2, 3].

Computational models using ANN are dependent on the network structure (topology, connections, neurons number) and their operational parameters (learning rate, momentum, etc). In other words, the form in which the network architecture is defined affects significantly its performance that can be classified in learning speed, generalization capacity, fault tolerance and accuracy in the learning. It is hard to project an efficient Neural Network (NN), even there are some techniques those come empirical knowledge, which is not always get right. This happens due to inherent particularities to the physical processes in which the networks are applied [1, 2, 3, 4, 5, 6, 7].

In this paper evolutionary system, such as Adaptive Particle Swarm Optimization (APSO) & Adaptive Genetic Algorithm (AGA) is used for the training and the network for better generalization performances and low training errors through Boolean identities.

II. EVOLUTIONARY SYSTEM

Evolutionary System is a research area of Computer Science, which draws inspiration from the process of natural evolution. Evolutionary computation, offers practical advantages to the researcher facing difficult optimization problems. These advantages are multifold, including the simplicity of the approach, its robust response in changing circumstance, its flexibility and many other facets. The evolutionary approach can be applied to problems where heuristic solutions are not available or generally lead to unsatisfactory results. Thus evolutionary computing is needed for Developing automated problem solvers, where the most powerful natural problem solvers are human Brain and evolutionary process. In this work evolutionary system such as adaptive PSO & GA is used to design architecture of ANN and is used for low training errors of ANN's as well as to find the appropriate network architecture [5, 6, 7].

A. Adaptive PSO

Dr. Russell C. Eberhart and Dr.James Kennedy first introduced particle Swarm Optimization in 1995. As described by Eberhart and Kennedy, the PSO algorithm is an adaptive algorithm based on a social-psychological metaphor; a population of individuals (referred to as particles) adapts by returning stochastically toward previously successful regions. In PSO algorithm, supposed in D dimensions search space, a particle swarm have m particles. The position of swarm i is x_i = (x_{i1}, x_{i2}, x_{i3}, …, x_{iD} ,) , i =1,2, ….,m . The best position of i is individual historical best position Pi, and the best fitness is individual historical best fitness of i. V_i is flight velocity of the swarm i. The best position of the particle swarm is the position of global best particle and the best fitness of the

swarm is the fitness of global best particle. In general, the update formulas of particle i are as follows:

$$v^{n+1}_{id} = \chi(\omega v^{n}_{id} + c_1 r_1(p^{n}_{id} - x^{n}_{id}) + c_2 r_2(p^{n}_{gd} - x^{n}_{id})) \quad (1)$$

$$x^{n+1}_{id} = x^{n}_{id} + v^{n+1}_{id} \quad (2)$$

In the formulas: χ is contracting factor, ω is inertia weight, c_1 and c_2 are acceleration factors; r_1 and r_2 are both random numbers in [0, 1] [21, 22, 23, 24, 25] .

As evolution goes on, the swarm might undergo an undesired process of diversity loss. Some particles become *inactively* while lost both of the global and local search capability in the next generations. For a particle, the lost of global search capability means that it will be only flying within a quite small space, which will be occurs when its location and *pbest* is close to *gbest* (if the *gbest* has not significant change) and its velocity is close to zero (for all dimensions) according to the equation (1); the lost of local search capability means that the possible flying cannot lead perceptible effect on its fitness. From the theory of self-organization, if the system is going to be in equilibrium, the evolution process will be stagnated. If *gbest* is located in a local optimum, then the premature convergence occurring as all the particles become inactively. To stimulate the swarm with sustainable development, the inactive particle should be replaced by a fresh one adaptively so as to keep the non-linear relations of feedback in equation (1) efficiently by maintaining the social diversity of swarm. This is known as adaptive particle swarm optimization (APSO) [10, 11, 13, 19, 32, 40, 41, 45]. However, it is hard to identify the inactive particles, since the local search capability of a particle is highly dependent on the specific location in the complex fitness landscape for different problems. Based on these facts, we adopt the adaptive particle swarm optimization (APSO) to obtain the optimal solution as follows:

APSO Algorithm:

```
Initialize Error_Goal, Max_Iteration, No_of_ hidden_ layers
and No_Nodes_ in_each_layer
Initialize Local_Error = 1
While (No_ of_ hidden _layers > = 0)
     While (No_of_ nodes_ in (Output_ Layers-1)>1)
//Initialization Iteration=0;
Calculate the Dimension = No_of_ Weights + No_ of_ Biases;
//initialize velocity, Accelerating parameters, lower and upper
bound of Population_ values
Calculate the best_Error; //Population having the Minimum
Error
//Initialization Iteration=0;
       While (best_Error >ErrGoal & Iteration<Max_Iteration)
 Train neural network through PSO;
        Iteration = Iteration + 1;
        Calculate the best_Error;
      End While;
 If (best_ Error < Local _Error)
    Then
    Save population, No_of_ hidden layers & No_of_Nodes_
in_each_layer
     Local _Error= best_ Error;
       End if;
 If (best_Error < Error_Goal or Iteration = = Max_Iteration)
    Then
 No_of_ nodes_ in (Output_ Layers -1) = No_of_ nodes_ in
(Output_ Layers -1)-1;
      Break;
      End if;
 End While;
 If (No_of_ nodes_ in (Output_ Layers -1) = = 1)
       Then
        No_of_ hidden_layers = No_of_ hidden_layers – 1;
 No_of_ nodes_ in (Output_ Layers -1) = No_of_ nodes_ in
(Output_ Layers);
      Discard Output_layer
     End if;
 End While;
```

B. Adaptive GA

Optimization is a general tool used in numerous problems of management, engineering and sciences. The life has been shaped by evolution during billions of years. Hence there seems to be things shared or sharable between different sciences that can be beneficial in many ways in the context of optimization and search. In order to implement, analyze, and utilize methods similar to evolution to solve optimization and search problems that are or seem to be computationally hard and/or complex i.e. have many parameters and/or are non-continuous and/or discrete or combinatorial. This has led to the study of genetic algorithms (GA). Genetic algorithms (GA) first described by John Holland in 1960s and further developed by Holland and his students and colleagues at the University of Michigan in the 1960s and 1970s. GA used Darwinian Evolution to extract nature optimization strategies that uses them successfully and transforms them for application in mathematical optimization theory to find the global optimum in defined phase space. GA is used to find approximate solutions to difficult problems through a set of methods or techniques inheritance or crossover, mutation, natural selection, and fitness function. Such methods are principles of evolutionary biology applied to computer science. GA is useful for, solving difficult problems and modeling the natural system that inspired design [38, 61, 62, 63, 64, 66, 71, 74].

If the multimode functions wants to keep the global search ability it must have balanced search ability. Crossover probability p_c and mutation probability p_m are the main factors in affecting balanced search ability (global search ability and local search ability). While we strengthen one ability by increasing or decreasing p_c, p_m, we may weaken other abilities.

Both p_c and p_m in the simple genetic algorithm (SGA) are invariant, so for the complex optimal problem the GA's efficiency is not high. In addition, immature convergence maybe caused. Therefore, the goals with adaptive probabilities of crossover and mutation are to maintain the genetic diversity in the population and prevent the genetic algorithms to converge prematurely to local minima. The basic idea behind adaptive genetic algorithm (AGA) is to adjust p_c and p_m according to the individual fitness. This algorithm can better solve the problem of adjusting p_c and p_m dynamically and also fits to all kinds of optimal problem [11, 32, 40, 41, 47, 67, 68, 76, 77, 78]. Based on these facts, we adopt the adaptive genetic algorithm to obtain the optimal solution as follows:

AGA Algorithm:

```
Initialize Error_Goal, Max_Iteration, No_of_ hidden_ layers
and No_Nodes_ in_each_layer
Initialize Local_Error = 1
While (No_ of_ hidden _layers > = 0)
      While (No_of_ nodes_ in (Output_ Layers-1)>1)
//Initialization Iteration=0;
Calculate the Dimension = No_of_ Weights + No_ of_ Biases;
//initialize mutation _rate, crossover _rate, lower and upper
bound of Population_ values
Calculate the best_Error; //Population having the Minimum
Error
//Initialization Iteration = 0;
While (best_Error >ErrGoal && Iteration < Max_Iteration)
Train neural network through GA using Crossover and
Mutation operation;
Iteration = Iteration +1;
Calculate the best_Error;
       End While;
   If (best_ Error < Local _Error)
        Then
Save population, No_of_ hidden layers and No_Nodes_
in_each_layer
        Local _Error = best_ Error;
        End if;
   If (best_Error < Error_Goal or Iteration = = Max_Iteration)
        Then
No_of_ nodes_ in (Output_ Layers -1) = No_of_ nodes_ in
(Output_ Layers -1) -1;
        Break;
        End if;
 End While;
 If (No_of_ nodes_ in (Output_ Layers -1) = = 1)
        Then
 No_of_ hidden_layers = No_of_ hidden_layers – 1;
 No_of_ nodes_ in (Output_ Layers -1) = No_of_ nodes_ in
(Output_ Layers);
 Discard Output_layer
 End if;
End While;
```

III. STRUCTURE OPTIMIZATION USING APSO & AGA

The performance of neural networks does not depend only on the choice of the weights, but also strongly on the structure, i.e., the number of neurons and the way the neurons are connected. In particular, the task of fast learning or learning with a small amount of data demands a suitable architecture. Evolutionary structure optimization of the neural networks has proven to be a very efficient approach in choosing the structure as well as the weights. In principle, it is possible to embed the structure optimization of the approximate model into the design optimization algorithm. In this work, we perform structure optimization of the neural network in offline, i.e., before the neural networks are employed as meta-models in the design optimization algorithm. The data for this offline optimization stems from previous design optimization runs. Only the weights will be adapted in every single control cycle during the design optimization. The standard optimization task is to structure a NN such that it represents the input-output mapping induced by a given set of data with a minimum error, including the ability to generalize towards other data stemming from the same process. As such kind of optimization tries to find structure and weight configuration suitable for all available data, the optimization would aim at an approximate model for the whole fitness landscape of the design evolution. Typical structure optimization techniques i.e. APSO & AGA with using of Boolean operators like AND, OR and XOR is employed in our investigations, for optimizing the architecture and the weights of NN by means of minimum training error and selection of the optimal number of hidden layers and nodes [10, 11, 13, 14, 17, 19, 20, 26, 36, 37, 44, 51, 53, 58, 59, 60, 63, 64, 66, 68, 69, 70, 72, 73, 75, 79, 80].

A. Experimental Setup for Adaptive PSO

To develop an accurate process model using ANN, the training, and validation processes are among the important steps. In the training process, a set of input-output patterns is repeated to the ANN. From that, weights of all the interconnections between neurons are adjusted until the specified input yields the desired output. Through these activities, the ANN learns the correct input-output response behavior. The model training stage includes choosing a criterion of fit (MSE) and an iterative search algorithm to find the network parameters that minimizes the criterion. APSO is used in an effort to formalize a systematic approach to training ANN, and to insure creation of a valid model. They are used to perform global search algorithms to update the weights and biases of neural network. The control parameters used for running APSO is shown in Table I given below:

TABLE I. THE CONTROL PARAMETERS USED FOR RUNNING APSO

Parameters	Value
Number of Population	30

Parameters	Value
Number of Generation	1000
Weight in Network	[50,50]
Error Goal	0.001
Acceleration Constant[C_1, C_2]	[1.55: 2.55]
Random Numbers in Range	[0, 1]
Fitness	MSE

B. Experimental Setup for Adaptive GA

To develop an accurate process model using ANN, the training, and validation processes are among the important steps. In the training process, a set of input-output patterns is repeated to the ANN. From that, weights of all the interconnections between neurons are adjusted until the specified input yields the desired output. Through these activities, the ANN learns the correct input-output response behavior. The model training stage includes choosing a criterion of fit (MSE) and an iterative search algorithm to find the network parameters that minimizes the criterion. AGA is used in an effort to formalize a systematic approach to training ANN, and to insure creation of a valid model. They are used to perform global search algorithms to update the weights and biases of neural network. The control parameters used for running AGA is shown in Table II given below:

TABLE II. THE CONTROL PARAMETERS USED FOR RUNNING AGA

Parameters	Value
Number of Population	30
Number of Generation	1000
Mutation Rate	0.09
Crossover Rate	0.08
Max Error	0.001
Random Numbers in Range	[0, 1]
Fitness	MSE

IV. EXPERIMENTAL RESULTS & ANALYSIS USING APSO & AGA

A. Structure Optimization Results of Adaptive PSO

The performance of the proposed models in tracking the actual process data during the training and validation testing stages of ANN by taking [2, 3, 1] for input layer, hidden layer and output layer respectively for APSO-NN using AND operator, [2, 4, 1] for input layer, hidden layer and output layer respectively for APSO-NN using OR operator, [2, 4, 1] for input layer, hidden layer and output layer respectively for APSO-NN using XOR operator, which can be seen from Figure 1, Figure 2, Figure 3 respectively. The final convergence value of the proposed models by all operators reached at 0.001, which can be seen from Figure 4.

TABLE III. RESULTANT VALUE OF APSO-NN USING ALL OPERATOR

Parameter	APSO-NN(AND)	APSO-NN(OR)	APSO-NN(XOR
Time(s) in Sec.	0.898121	0.2103675	0.681487
MSE	0.0015	0.0090	0.0055

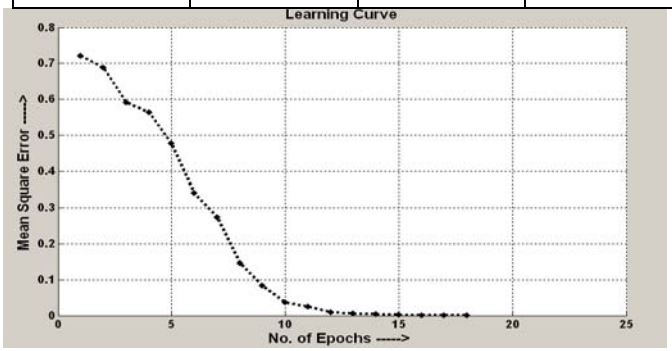

Figure 1. Resultant Figure of APSO-NN *(Using AND Operator)*

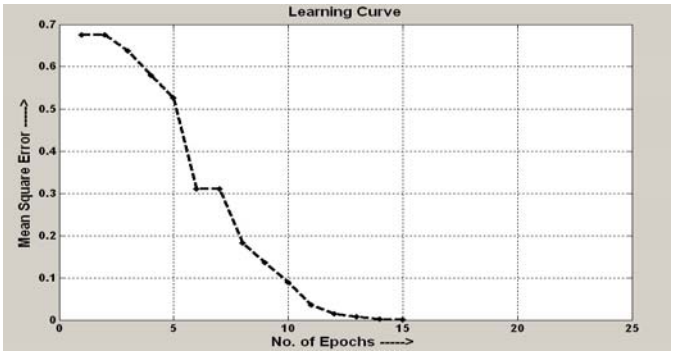

Figure 2. Resultant Figure of APSO-NN *(Using OR Operator)*

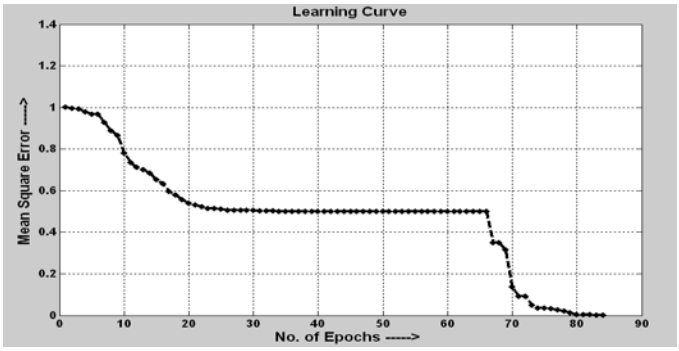

Figure 3. Resultant Figure of APSO-NN *(Using XOR Operator)*

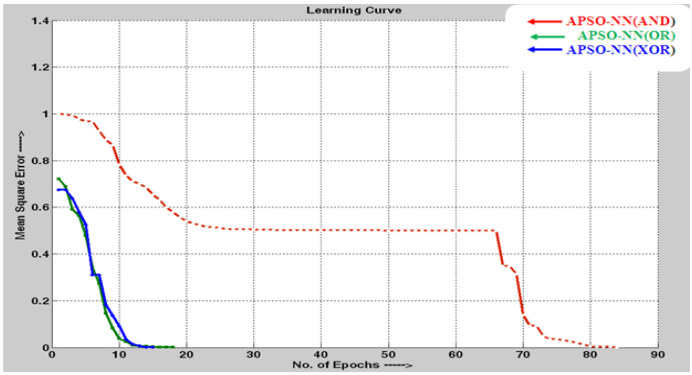

Figure 4. Resultant Figure of APSO-NN *(Using All Operator)*

B. *Structure Optimization Results of Adaptive GA*

The performance of the proposed models in tracking the actual process data during the training and validation testing stages of ANN by taking [2, 3, 1] for input layer, hidden layer and output layer respectively for AGA-NN using AND operator, [2, 4, 1] for input layer, hidden layer and output layer respectively for AGA-NN using OR operator, [2, 4, 1] for input layer, hidden layer and output layer respectively for AGA-NN using XOR operator, which can be seen from Figure 5, Figure 6, Figure 7 respectively. The final convergence value of the proposed models by all operators reached at 0.001, which can be seen from Figure 8.

TABLE IV. RESULTANT VALUE OF AGA-NN USING ALL OPERATOR

Parameter	AGA-NN(AND)	AGA-NN(OR)	AGA-NN(XOR)
Time(s) in Sec.	18.2103675	17.898121	20.1861487
MSE	$9.9740e^{-004}$	$9.7322e^{-004}$	$9.8418e^{-004}$
No of Epochs	423	424	405

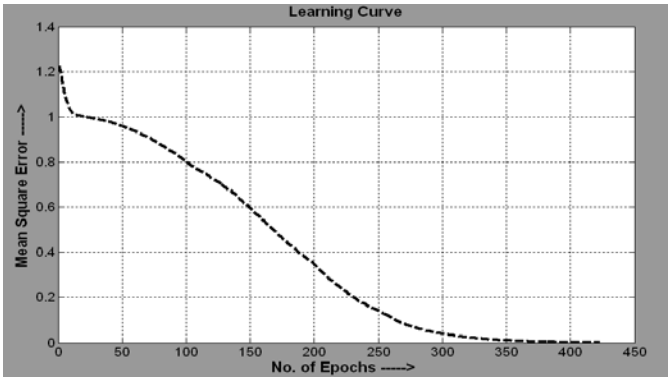

Figure 5. Resultant Figure of AGA-NN *(Using AND Operator)*

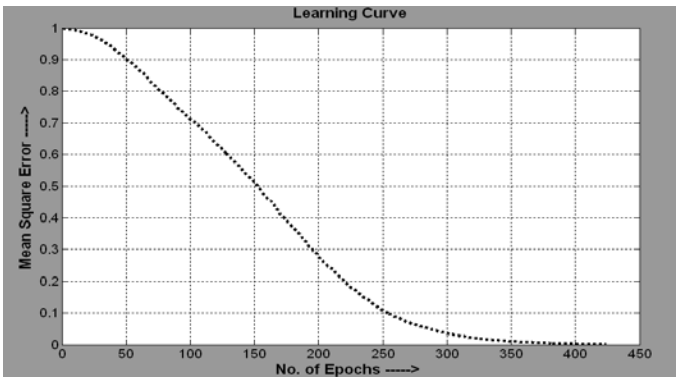

Figure 6. Resultant Figure of AGA-NN *(Using OR Operator)*

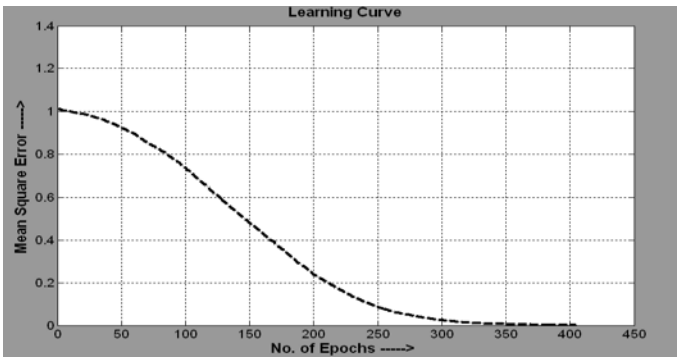

Figure 7. Resultant Figure of AGA-NN *(Using XOR Operator)*

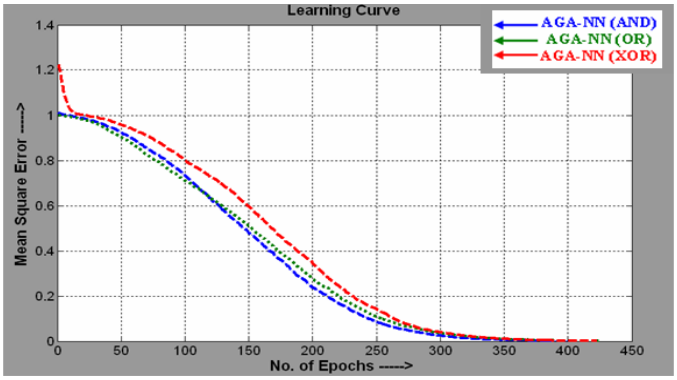

Figure 8. Resultant Figure of AGA-NN *(Using All Operator)*

V. CONCLUSION & FUTURE WORK

This work has proposed an enhancement to neural network model. Aiming to improve the model robustness, evolution in network connection weights using EAs was explored. Based on the results obtained from the 'APSO' & 'AGA' structure optimization approaches suggest that the approach for adaptive optimization of ANNs is more successful in obtaining higher accuracy levels. In this study,

the main conclusions of this paper are: adaptive PSO-NN & adaptive GA-NN is an efficient and effective empirical modeling tool for estimating the any process variable by using other easily available process measurements and the use of multilayer feed forward network with delay values in model input variables are sufficient to give estimation to any arbitrary accuracy. Also, this paper has taken advantage of flexibility EAs techniques by applying it to the problem not only to training neural networks but also optimize neural network structure. In particular the optimization methods like APSO & AGA is employed to provide a sense of the directivities of optimization the weights and biases of neural networks, and seek a good starting weight vector for subsequent neural networks learning algorithm. The preliminary results give a positive indication of the potential offered by EAs; this ensures the ability to effectively train the neural networks using the optimization techniques. In addition, PSO & GA offered an increased level of adaptability of neural networks and is more preferable as the optimal solution searching to any model. Despite the encouraging finding was obtained, there are still several further works to be considered. These include: The inclusion of adaptive feature using EAs to improve model robustness can be extended to evolution of a network architecture, which is typically number of hidden nodes as well as number of hidden layers; also the evolution of network architecture requires new set of connection weights.

In this research, evolution of only the number of hidden layers and hidden nodes has been considered with regard to the adaptive optimization of an ANN. But it is well known that these are not the only parameters that can be optimized in a given ANN. Therefore in the future, this research work can include the adaptive optimization of other ANN parameters like the learning rate, learning momentum and activation functions, in order to realize the goal of achieving a completely optimized ANN.

REFERENCES

[1] C. Zhang, H. Shao. "An ANN's Evolved by a New Evolutionary System and Its Application" in In Proceedings of the 39th IEEE Conference on Decision and Control, vol. 4, pp. 3562-3563, 2000.

[2] E. Eiben and J. E. Smith, Introduction to Evolutionary Computing. Natural Computing Series. MIT Press. Springer. Berlin. (2003).

[3] L. Prechelt "Proben1 - A set of neural network benchmark problems and benchmark rules", Technical Report 21/94, Fakultät für Informatik, Universität Karlsruhe, Germany, September, 1994.

[4] X. Yao, "Evolving artificil neural networks", Proceedings of the IEEE, vol. 87, pp. 1423-1447, 1999.

[5] X. Yao and Y. Liu, "A new evolutionary system for evolving artificial neural networks", IEEE Transactions on Neural Networks, vol. 8, pp. 694-713, 1997.

[6] X. Yao, A review of evolutionary artificial neural networks, Int. J. Intell. Syst., 8(4) (1993) 539–567.

[7] P.J. Angeline, G.M. Sauders, J.B. Pollack, An evolutionary algorithm that constructs recurrent neural networks, IEEE Trans. Neural Networks 5 (1) (1994) 54–65.

[8] K. Homik, Multilayer feedforward networks are universal approximators, Neural Networks 2 (1989) 359–366.

[9] Hassoun, M. H. (1995). Fundamentals of artificial neural networks. Cambridge, MA: MIT Press.

[10] Jinn-Tsong Tsai, Jyh-Horng Chou, and Tung-Kuan Liu, "Tuning the Structure and Parameters of a Neural Network by Using Hybrid Taguchi-Genetic Algorithm" IEEE TRANSACTIONS ON NEURAL NETWORKS, VOL. 17, NO. 1, JANUARY 2006

[11] N. P. Suraweera, D. N. Ranasinghe, "Adaptive Structural Optimisation of Neural Networks" *The International Journal on Advances in ICT for Emerging Regions 2008 01 (01*): *33 – 41*

[12] Jing-Ru Zhang , Jun Zhang , Tat-Ming Lok , Michael R. Lyu, "A hybrid particle swarm optimization–back-propagation algorithm for feedforward neural network training", Applied Mathematics and Computation 185 (2007) 1026–1037

[13] Niranjan Subrahmanya,YungC.Shinn, "Constructive training of recurrent neura l networks using hybrid optimization ",Neurocomputing 73 (2010) 2624–2631

[14] Ji-Xiang Du, De-Shuang Huang, Guo-Jun Zhang, Zeng-Fu Wang, "A novel full structure optimization algorithm for radial basis probabilistic neural networks", Neurocomputing 70 (2006) 592–596

[15] Pavel Kordík, Jan Koutník , Jan Drchal, Oleg Kováík, Miroslav .epek, Miroslav .norek, "Meta-learning approach to neural network optimization", 2010 Special Issue

[16] Serkan Kiranyaz, Turker Ince, Alper Yildirim, Moncef Gabbouja, "Evolutionary artificial neural networks by multi-dimensional particle swarm optimization", Neural Networks(Elsevier).

[17] Marcio Carvalho, Teresa B. Ludermir, "Particle Swarm Optimization of Neural Network Architectures and Weights", Seventh International Conference on Hybrid Intelligent Systems

[18] Pingzhou Tang, Zhaocai Xi, "The Research on BP Neural Network Model Based on Guaranteed Convergence Particle Swarm Optimization", Second International Symposium on Intelligent Information Technology Application

[19] De-Shuang Huang, Senior Member, and Ji-Xiang Du, "A Constructive Hybrid Structure Optimization Methodology for Radial Basis Probabilistic Neural Networks", IEEE TRANSACTIONS ON NEURAL NETWORKS, VOL. 19, NO. 12(2008- 2009)

[20] Guangbin Yu, Guixian Li , Xiangyang Jin, Yanwei Bai, "Tuning of the Structure and Parameters of Dynamic Process Neural Network Using Improved Chaotic PSO", Third International Conference on Natural Computation (2007) IEEE

[21] J. Kennedy and R. C. Eberhart, "A discrete binary version of the particle swarm algorithm," in Proc. Conf. Syst., Man Cybern., Piscataway, NJ, 1997, pp. 4104–4108.

[22] Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. In Proc. of IEEE int. conf. on neural networks. vol. 4 (pp. 1942_1948).

[23] Y. Shi, R.C. Eberhart, Empirical study of Particle Swarm Optimization, in: Proc. of IEEE World Conference on Evolutionary Computation (1999) 6–9.

[24] R. C. Eberhart and Y. Shi, "Comparing inertia weights and constriction factors in particle swarm optimization," in Proc. Congr. Evolut.Comput., 2000, vol. 1, pp. 84–88.

[25] M. Clerc and J. Kennedy, "The particle swarm-explosion, stability, and convergence in a multidimensional complex space," IEEE Trans.Evolut. Comput., vol. 6, no. 1, pp. 58–73, Feb. 2002.

[26] A. S. Weigend, D. E. Rumelhart, and B. A. Huberman, "Generalization by weight-elimination with application to forecasting", in Advances in Neural Information Processing (3), R. Lippmann, J. Moody, and D. Touretzky, Eds., 1991, pp. 875-882.

[27] D. Rumelhart, G.E. Hilton and R.J.Williams, "Learning representations of backpropagation errors", Nature (London), vol.323, pp. 523-536, 1986.

[28] E. Cantu-Paz and C. Kamath, "An empirical comparison of combinations of evolutionary algorithms and neural networks for

classification problems", IEEE Transactions on Systems, Man, and Cybernetics, Part B, vol. 35, pp. 915-927, 2005.

[29] M. Carvalho, T.B. Ludermir, "Particle Swarm Optimization of Feed-Forward Neural Networks with Weight Decay," his, p. 5, in Sixth International Conference on Hybrid Intelligent Systems (HIS'06), 2006.

[30] Marco Gori, Alberto Tesi, On the problem of local minima in back-propagation, IEEE Trans. Pattern Anal. Mach. Intell. 14 (1) (1992) 76–86.

[31] A. van Ooyen, B. Nienhuis, Improving the convergence of the back-propagation algorithm, Neural Network 5 (4) (1992) 465–471.

[32] M.K. Weirs, A method for self-determination of adaptive learning rates in back propagation, Neural Networks 4 (1991) 371–379.

[33] B. Irie, S. Miyake, Capability of three-layered perceptrons, in: Proc. of IEEE Int. Conf. On Neural Networks, San Diego, USA (1998) 641–648.

[34] S. Shaw, W. Kinsner, Chaotic simulated annealing in multilayer feedforward networks, in: Proc. of Canadian Conf. on Electrical and Computer Engineering, vol. 1 (1996) 265–269.

[35] Chunkai Zhang, Huihe Shao, Yu Li, Particle swarm optimization for evolving artificial neural network, in: Proc. of IEEE Int. Conf. on System, Man, and Cybernetics, vol. 4 (2000) 2487–2490.

[36] J. Salerno, Using the particle swarm optimization technique to train a recurrent neural model, in: Proc. of Ninth IEEE Int. Conf. on Tools with Artificial Intelligence (1997) 45–49.

[37] R.C. Eberhart, Y. Shi, Comparing Inertia Weights and Constriction Factors in Particle swarm Optimization, in: Proc. of 2000 congress on Evolutionary Computing, vol. 1 (2000) 84–88.

[38] D.W. Boeringer, D.H. Werner, Particle swarm optimization versus genetic algorithms for phased array synthesis, IEEE Trans. Antennas Propagation 52 (3) (2004) 771–779.

[39] D.S. Chen, R.C. Jain, A robust Back-propagation Algorithm for Function Approximation, IEEE Trans. Neural Network 5 (1994) 467–479.

[40] Cheng-Jian Lin, Cheng-Hung, Chi-Yung Lee, A self-adaptive quantum radial basis function network for classification applications, in: Proc. of 2004 IEEE International Joint Conference on Neural Networks, vol. 4, 25–29 July (2004) 3263–3268.

[41] J.J.F. Cerqueira, A.G.B. Palhares, M.K. Madrid, A simple adaptive back- propagation algorithm for multilayered feedforward perceptrons, in: Proc. of IEEE International Conference on Systems, Man and Cybernetics, volume 3, 6–9 October (2002) 6.

[42] C.M. Kuan, K. Hornik, Convergence of learning algorithms with constant learning rates, IEEE Trans. Neural Networks 2 (5) (1991) 484–489.

[43] Angeline, P. J., Sauders, G. M., & Pollack, J. B. (1994). An evolutionary algorithm that constructs recurrent neural networks. IEEE Transactions on Neural Networks, 5, 54_65.

[44] Carvalho, M., & Ludermir, T. B. (2007). Particle swarm optimization of neural network architectures and weights. In Proc. of the 7th int. conf. on hybrid intelligent systems (pp. 336_339).

[45] Clerc, M. (1999). The swarm and the queen: Towards a deterministic and adaptive particle swarm optimization. In Proc. of the IEEE congress on evolutionary computation. vol. 3 (pp. 1951_1957).

[46] Fahlman, S. E. (1988). An empirical study of learning speed in backpropagation. Technical report, CMU-CS-88-162. Carnegie-Mellon University.

[47] Frean, M. (1990). The upstart algorithm: A method for constructing and training feedforward neural networks. Neural Computation, 2(2), 198_209.

[48] Gudise, V. G., & Venayagamoorthy, G. K. (2003). Comparison of particle swarm optimization and backpropagation as training algorithms for neural networks. Swarm intelligence symposium, 2003. Proc. of the 2003 IEEE (pp. 110_117).

[49] Lovberg, M., & Krink, T. (2002). Extending particle swarm optimisers with selforganized

[50] Meissner, M., Schmuker, M., & Schneider, G. (2006). Optimized particle swarm optimization (OPSO) and its application to artificial neural network training. BMC Bioinformatics, 7, 125.

[51] Prados, D. L. (1992). Training multilayered neural networks by replacing the least fit hidden neurons. In Proc. IEEE SOUTHEASTCON'92. vol. 2 (pp. 634_637).

[52] N.P. Suraweera, D.N. Ranasinghe 40 Yao X. (1999, September). Evolving Artificial Neural Networks, Proceedings of the IEEE, Vol. 87, No.9.

[53] Balakrishnan K. and Honavar V. (1995). Evolutionary Design of Neural Architectures – A Preliminary Taxonomy and Guide to Literature, Tech Report no CS TR 95-01, Artificial Intelligence Research group, Iowa State University,

[54] Zhang C., Shao H., Li Y. (2000). Particle Swarm Optimisation for Evolving Artificial Neural Network. *IEEE International Conference on Systems, Man and Cybernetics,* Vol 4

[55] Van den Bergh F., (1999, September) Particle Swarm Weight Initialization in Multi-layer Perceptron Artificial Neural Networks, *Development and Practice of Artificial Intelligence Techiniques,* pp. 41-45, Durban, South Africa.

[56] Eberhart R.C., Hu X.. (1999). Human tremor analysis using particle swarm optimization, *Evolutionary Computation*.

[57] Van den Bergh F., Engelbrecht A.P., Cooperative Learning in Neural Networks using Particle Swarm Optimizers, *South African Computer Journal,*

[58] Liu B., Wang L., Jin Y., Huang D. (2005). Designing neural networks using hybrid particle swarm optimization, *LNCS 3496*, pp 391-397.

[59] Xian-Lun T., Yon-Guo L., Ling Z., (2007). A hybrid particle swarm algorithm for the structure and parameter optimization of feedforward neural networks, *LNCS 4493*, pp 213-218.

[60] Niu B., Li L., (2008). A hybrid particle swarm optimization for feed forward neural network training, *LNAI 5227*, pp. 494-501.

[61] Kathryn, A.D., "Genetic Algorithms a Tool for OR?", Journal of the operational Research Society 47, pp. 550-561, (1996).

[62] "Genetic algorithms and financial applications", Davis L., Deoeck, 1994

[63] Liyi Zhang Ting Liu Yunshan Sun Lei Chen, "Research on Neural Network Blind Equalization Algorithm with Structure Optimized by Genetic Algorithm", Sixth International Conference on Natural Computation (ICNC- 2010).

[64] Marwan. A. Ali, Mat Sakim. H. A, Rosmiwati Mohd-Mokhtar, "Structure Optimization of Neural Controller Using Genetic Algorithm Technique", European Journal of Scientific Research, Vol.38 No.2 (2009), pp.248-271

[65] F. H. F. Leung, H. K. Lam, S. H. Ling, and P. K. S. Tam, "Tuning of the structure and parameters of a neural network using an improved genetic algorithm," IEEE Trans. Neural Netw., vol. 14, no. 1, pp. 79–88, Jan. 2003.

[66] N. Garcia-Pedrajas, D. Ortiz-Boyer and C. Hervas-Martinez, "An alternative approach for neural network evolution with a genetic algorithm: crossover by combinatorial optimization", Neural Networks, vol. 19, pp. 514-528, 2006.

[67] R.A. Jacobs, Increased rates of convergence through learning rate adaptation, Neural Networks 1 (1988) 295–307.

[68] Goldberg, D. (1989). Genetic algorithms in search, optimization and machine learning. Reading, MA: Addison-Wesley (pp. 1_25).

[69] Koza, J. (1992). Genetic programming: On the programming of computers by means of natural selection. Cambridge, MA: MIT Press.

[70] Miller, G. F., Todd, P. M., & Hegde, S. U. (1989). Designing neural networks using genetic algorithms. In Proc. 3rd int. conf. genetic algorithms their applications (pp. 379_384).

[71] D.E. Goldberg, Genetic Algorithm in Search Optimization and Machine Learning (Addison-Wesley, Reading, MA, 1989) 1-23.

[72] J. Sun, W.I. Grosky and M.H. Hassoun, A fast algorithm for finding global minima of error functions in layered neural networks, *Proc. IEEE IJCNN,* Vol. 1 (1990) 715-720.

[73] T. Ueyama, T. Fukuda and F. Arai, Structure configuration using genetic algorithm for cellular robotic system, *Proc. IEEE IROS,* Vol. 3 (1992) 1542-1549.

[74] D. Whitley, T. Starkweather and C. Bogart, Genetic algorithms and neural networks: optimizing connection and connectivity, *Parallel Comput.* 14 (1990) 347-361.

[75] J. T. Tsai, T. K. Liu, and J. H. Chou, "Hybrid Taguchi-genetic algorithm for global numerical optimization," *IEEE Trans. Evol. Comput.*, vol. 8, no. 4, pp. 365–377, Aug. 2004.

[76] H. F. Leung, H. K. Lam, S. H. Ling, and K. S. Tam, "Tuning of the structure and parameters of neural network using an improved genetic algorithm," *Proc. 27th Annu. Conf. IEEE Industrial Electronics Society*, pp. 25–30, 2001.

[77] Y.W. Leung and Y.Wang, "An orthogonal genetic algorithm with quantization for global numerical optimization," *IEEE Trans. Evol. Comput.*, vol. 5, no. 1, pp. 41–53, Feb. 2001.

[78] J. Yen and B. Lee, "A simplex genetic algorithm hybrid," in *Proc.IEEE Int. Conf. Evolutionary Computation* , Indianapolis, IN, 1997, pp. 175–180.

[79] S. A. Harp and T. Samad, "Optimizing Neural Networks with Genetic Algorithms," Proc. *American Power Conference,* vol. 2.1992 pp. 1138-1 143.

[80] N. Dodd, "Optimisation of Network Structure using Genetic Techniques," Proc. *Internat'l Joir~CO @. Neural Networks,* vol. 3,1990, pp. 965-970.