SHORT TERM LOAD FORECASTING USING NEURO GENETIC HYBRID APPROACH: RESULTS ANALYSIS WITH DIFFERENT NETWORK ARCHITECTURES

[1] PRADEEPTA KUMAR SARANGI, [2] NANHAY SINGH, [3] DEEPAK SWAIN, [4] DR. R. K. CHAUHAN, [5] DR. RAGHURAJ SINGH

[1] Lecturer, School of Computer Science, Apeejay Institute of Technology, Greater Noida(India)
[2] Lecturer, Department of Computer, Science & Engineering, HBTI ,Kanpur (India)
[3] Lecturer, School of Computer Science, Apeejay Institute of Technology, Greater Noida(India)
[4] Professor & Chairman, Department of Computer Science & Applications, Kurukshetra University (India)
[5] Professor & Head, Department of Computer Science & Engineering, HBTI, Kanpur (India)

ABSTRACT

Load forecasting is an important component for energy management system. Precise load forecasting helps the electric utility to make unit commitment decisions including decisions on purchasing and generating electric power, load switching, and infrastructure development. Besides playing a key role in reducing the generation cost, it is also essential to the reliability of power systems. Short-term load forecasting (STLF) can help to estimate load flows and to make decisions that can prevent overloading. Timely implementations of such decisions lead to the improvement of network reliability and to the reduced occurrences of equipment failures and blackouts. Load forecasting is also important for contract evaluations and evaluations of various sophisticated financial products on energy pricing offered by the market. In the deregulated economy, decisions on capital expenditures based on long-term forecasting are also more important than in a non-deregulated economy where a rate increase could be justified by capital expenditure projects. Data mining plays the key role to infer the information which is important to make the right decision. In this article we examine and analyze the use of genetic algorithm (GA) techniques for the determination of weights in a back propagation network (BPN) for short-term load forecasting.

Keywords: *Genetic algorithm based backpropagation network (GA-BPN), Short term load forecasting (STLF).*

1. INTRODUCTION & BACKGROUND

Forecasting is a phenomenon of knowing what may happen to a system when certain trends or conditions continue or continue to change. In electrical power systems, there is a great need for accurately forecasting the load and energy requirements because electricity generations as well as distribution are a great financial liability to the state exchequer. Accurate load forecast provides system dispatchers with timely information to operate the system economically and reliably. It is also necessary because availability of electricity is one of the most important factors for industrial development, especially for a developing country like India. Over the years, the application of Artificial Neural Network (ANN) in power industries has been growing in acceptance. Given sufficient input-output data, ANN is able to approximate any continuous function to arbitrary accuracy. Also, while the network must be sufficiently trained to extract a sufficient set of general features applicable to both seen and unseen instances, overtraining the network may lead to undesired effects. Several attempts have been proposed by various researchers to alleviate this training problem.

These include imposing constraints on the search space, restarting training at many random points, adjusting training parameter and restructuring the ANN architecture [1]. However, some approaches are problem-specific and not well accepted and different researchers tend to prefer different methodologies. Among these, one of the more promising techniques is by introducing adaptation of network training using genetic algorithm (GA).

Unlike backpropagation, genetic algorithm is a global search algorithm based on the principle of "survival of fittest". It simultaneously searches for solutions in several regions, thus increasing the

probability of global convergence. Furthermore, since it is impossible to formulate a priori exact model of the system, a more practical approach is off-line set up a rough model, followed by on-line update of the model using GA. In this way, the merging of GA and ANN will gain adaptability to dynamic environment and lead to significantly better intelligent systems than relying on ANN or GA alone.

2. RESEARCH OBJECTIVE & METHODOLOGY

In this research work, an attempt has been made to develop different GA based backpropagation network (GA-BPN) models and to analyze the results.

The methodologies adopted in this research are explained below:

- Construction of different ANN architectures having (i) three neurons in input layer, two neurons in hidden layer and one neuron in output layer (3-2-1and 3-2-2-1) and (ii) three neurons in input layer, three neurons in hidden layer and one neuron in output layer (3-3-1 and 3-3-3-1) .
- Hybridization of backpropagation network (BPN) with GA. i.e extraction of weights for BPN by implementing genetic algorithm.
- Training of the GA-BPN network with different values of population size.
- Finding the best value of population size for GA-BPN forecasting.
- Forecasting using the best value of population.

3. THEORITICAL FRAMEWORK

Genetic algorithm (GA) is a computerized search and optimization algorithm based on the mechanics of natural genetics and natural selection. Genetic algorithm is very different from most of the traditional optimization methods. Genetic algorithm needs design space to be converted into genetic space. So, genetic algorithm works with a coding of variables. Similar to other evolutionary algorithms, GA is based on the principle of "survival of fittest", as in the natural phenomena of genetic inheritance and Darwinian strife for survival. In other words, GA operates on a population of individuals which represent potential solutions to a given problem. Mimicking the biological principles in nature, a single individual of a population usually is affected by other individuals as well as the environment. Normally, the better an individual performs under these competitive conditions the greater is the change for the individual to survive and reproduce. This in turn inherits the good parental genetic information. Hence, after several generations, the bad individual will be eliminated and better individuals are produced.

3.1 Genetic Terms and Operators

- **Genetic Terms:**

Chromosomes: Symbols from some finite alphabet in the form of strings. In case of binary alphabet (0, 1) the chromosomes are binary strings and in the case of real alphabet (0-9) the chromosomes are decimal strings.

Genes: The symbols that form the chromosomes are known as genes.

Population: A set of solutions represented by chromosomes.

Fitness Function: The criteria of goodness expressed in terms of an objective function to find the best alternative solution is called fitness function.

Fitness Value: It is the figure of merit, which is to be either maximized or minimized.

- **Genetic Operators:**

Reproduction: Reproduction also known as selection operator is used to select the best chromosomes for parents from the population into the mating pool to cross over and produce offspring.

Cross over: Cross over is a recombination operator applied to the mating pool, proceeds in three steps. First, the reproduction operator selects at random a pair of two individual strings for mating. Then a cross-site is selected at random along the string length and the position values are swapped between two strings following the cross site.

Mutation: Mutation operator involves flipping a bit in the string from 0 to 1 and vice versa.

3.2 Genetic Algorithm Based Back Propagation Networks (GA-BPN)

GA Based BPN (GA-BPN) is a Neuro-Genetic hybrid approach which makes use of GA to determine the weights of a multilayer feed-forward network with back propagation learning.

The learning algorithm behind BPN is a kind of gradient descent technique with backward error (gradient) propagation called back propagation. Back propagation searches on the error surface by means of the gradient descent technique in order to minimize the error criterion. It is therefore likely to get stuck in a local minimum.

$$E=1/2\sum(T_j-O_j)^2 \qquad (1)$$

Where E is the error, T_j is the target output and O_j is the output calculated by the network.

Conventionally, a BPN determines its weights based on a gradient search technique and there fore runs the risk of encountering the local minimum problem. GA on the other hand , though not guaranteed to find global optimum solution to problems, have been found to be good at finding "acceptably good" solutions to problems "acceptably quickly".

3.3 GA Based Weight Determination

Genetic algorithm which uses a direct analogy of natural behavior, work with a population of individual strings, each representing a possible solution to the problem considered. Each individual string is assigned a fitness value which is an assessment of how good a solution is, to a problem. The high-fit individuals participate in "reproduction" by cross-breeding with other individuals in the population. This yields new individuals strings as offspring which share some features with each parent. The least-fit individuals are kept out from reproduction and so "die out". A whole new population of possible solutions to the problem is generated by selecting the best (high-fit) individuals from the current generation. This new generation contains characteristics which are better than their ancestors. Processing in this way, after many generations, owing to mixing and exchange of good characteristics, the entire population inherits the best characteristics and therefore turns out to be fit solution to the problem.

The success of any neural network (NN) architecture depends on the search for the optimized weights for given training data set. GA has been shown in practice that it is very effective at function optimization and can perform efficient searching for the approximate global minima. Thus, GA can be effectively utilized for NN weight selection. The mathematical expression for extraction of weights using GA for BPN is as below:

$$w_k = + \frac{x_{kd+2}10^{d-2} + x_{kd+3}10^{d-3} + \ldots. x_{(k+1)d}}{10^{d-2}} \text{ if } 5 \le x_{kd+1} \le 9 \qquad (2)$$

and

$$w_k = - \frac{x_{kd+2}10^{d-2} + x_{kd+2}10^{d-2} + \ldots. x_{(k+1)d}}{10^{d-2}} \text{ if } 0 \le x_{kd+1} \le 5 \qquad (3)$$

where w_k is the actual weight extracted from the chromosome, x_1, x_2…represents chromosomes and x_{kd+1}, $x_{kd+2\ldots}$ $x_{(k+1)d}$ represents the k^{th} gene($k \ge 0$) in the chromosome [2].

3.4 Pseudo Code for implementation of GA-BPN

Algorithm FITGEN ( )

{

Let (I_i,T_i), i = 1,2,. . . ,N where $I_i = (I_{1i}, I_{2i}, \ldots, I_{li})$ and $T_i = (T_{1I}, T_{2I}, \ldots, T_{ni})$ represent the input–output pairs of the problem to be solved by BPN with a configuration l - m - n. (the architecture and l>0, m≥0, n>0)

For each chromosome C_i, i = 1,2,. . . ,p belonging to the current population P_i whose size is 'p'

{

Extract weights W_i from C_i using equation 2 & 3.

Keeping W_i as a fixed weight setting, train the BPN for the N input instances;

Calculate error E_i for each of the input instances using the formula,

$$E_i = \sum_J (T_{ji} - O_{ji})^2 \qquad (4)$$

Where O_{ji} is the output vector calculated by BPN;

Find the mean square E of the errors E_i, i = 1,2,. . . ,N;

$$E = \sqrt{\frac{\sum_I E_i}{N}} \qquad (5)$$

Calculate the fitness value F_i for each of the individual string of the population as

$$F_i = \frac{1}{E} \qquad (6)$$

}
Output F_i for each C_i, i = 1, 2. . . p }
END FITGEN

- **GA based weight determination**

Algorithm GA-BPN-WEIGHT ()

{
i ← 0;
Generate the initial population P_i of real-coded chromosomes C_{Ji} each representing a weight set for the BPN;
While the current population P_i has not converged

{
Generate fitness values F_I^J for for each $C_I^J \in P^i$ using the Algorithm FITGEN ();
Get the mating pool ready by terminating worst fit individuals and duplicating high fit individuals;
Using the cross over mechanism, reproduce offspring from the parent chromosomes;
i← i + 1;
Call the current population P_I;
Calculate fitness values F_I^J for each $C_I^J \in P^i$

}
Extract weights from P_i to be used by the BPN

}
END GA-BPN-WEIGHT

The methodology for computation of fitness value for the (initial) population is given in figure-1 below.

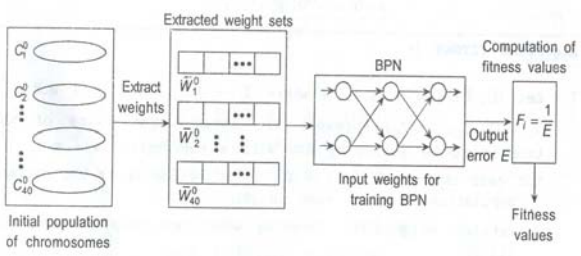


Figure-1: Methodology for computation of fitness value

4. SIMULATION DESIGN AND RESULTS

Here we design the models for electrical load forecasting. This section describes research data, Construction of Network Architecture, Requirement of minimum number of patterns and Selection of Input Variables etc.

4.1 Research Data

The data used in this research is the daily load (un restricted demand) data for the state of Delhi (INDIA). The data (un-normalized) have been collected from State Load Despatch Centre (SLDC), Delhi (India) from their website "http://www.delhisldc.org/report/monthly-power-data/July-2008 (block year 2008-09) in the month of February, 2009. The first fifteen days data have been used in this research. For training, the first nine days data have been used and next six days data have been used for testing purposes.

4.2 Construction of Network Architecture

Structure of the network affects the accuracy of the forecast. Network configuration mainly depends on the number of hidden layers, number of neurons in each hidden layer and the selection of activation function. No clear cut guide lines exist up to date for deciding the architecture of ANN. Mostly it is problem dependent. However, Gowri T. M et. al [3] suggest that for a three layer ANN, the number of hidden neurons can be selected by one of the following thumb rules:

1. (i-1) hidden neurons, where 'i' is the number of input neurons
2. (i+1) hidden neurons, where 'i' is the number of input neurons.
3. For every 5 input neurons, 8 hidden neurons can be taken. This is developed seeing the performance of network within 5 inputs, 8 hidden neurons and 1 output.
4. Number of input neurons / number of output neurons
5. Half the sum of input and output neurons.
6. P/i neurons, where 'i' is the input neurons and 'P' represents number of training samples.

In this research, we have used four different architectures. First one is the 3-2-1 architecture where there is one input layer having 3 input neurons, one hidden layer having 2 hidden neurons and the output layer having 1 neuron. The second one is the 3-3-1 architecture where there is one input layer having 3 input neurons, one

hidden layer having 3 hidden neurons and the output layer having 1 neuron. Others are 3-2-2-1 and 3-3-31 architectures.

4.3 Requirement of minimum number of patterns

Gowri T. M. and Reddy V.V.C. also suggest that the minimum numbers of patterns required are half the number of input neurons and the maximum is equal to the product of number of input neurons and number of hidden neurons or five times the number of input neurons, which ever is less. However, no justifications can be given to such thumb rules. In this work, performance of configurations is studied with six numbers of training patterns and three numbers of test patterns.

In this research work we have used a total number of 15 input data divided into two sets, training set consisting of first 9 days data and the second one is the test set consisting of last 6 days data.

4.4 Selection of Input Variables

Papalexopoulos A.D. et. al [4] state that, there is no general rule for selecting number of input variables. It largely depends on engineering judgment, experience and is carried out almost entirely on trial and error basis.

The importance of the factors (in the input variables) may vary for different customers. According to Wang X. et al [5], for most customers historical data (such as weather data, weekend load, and previous load data) is most important for predicting demand in STLF. In practice, it is neither necessary nor useful to include all the historical data as input. Autocorrelation helps to identify the most important historical data. Wang describes the importance of recent daily loads in daily load prediction based on correlation analysis is described as below:

Factors	X_1	X_2	X_3	X_4	X_5	X_6	X_7
Importanc	1	2	3	5	7	6	4

Table-1 Selection of input variables

The first line shows the recent daily load history. X_i represents the load consumed and the subscripts show the time index, i.e., 1 means yesterday (one day ago), 2 means the day before yesterday (two days ago), and 7 means the same day of last week (7 days ago). In the second line the importance is described. 1 means the most important and 7 means the least important. These information and results help us to decide what factors should be included in input. According to table-1, when we perform daily load prediction, we should include X_1, X_2, X_3, X_7 as our inputs.

4.5 GA-Based Back Propagation Algorithm

Our genetic algorithm here works with a fixed value of populations selected from a set of population size. Since sometimes it could happen for a single population scheme that though the neural network could theoretically solve a certain problem, the system may not return a correct solution. This is because of the random nature of the algorithm and its reliance on natural selection, mutation and cross-over. First, at the initialization stage, the neural network structure, including number of input nodes, hidden nodes, and output nodes are specified. Connection weights corresponding to this structure are encoded in GA's chromosomes; each chromosome represents one structure with given connection weights contained in its genes. Second, at the GA-based weight connection training stage, these initialized chromosomes which may belong to different populations are evolved generation to generation by using GA according to the fitness and RMSE performance.

4.6 GA-BPN Training Methodology

The training of present network has been accomplished using the back propagation algorithm (BP). Conventionally, A BPN determines its weights based on a gradient search technique and hence runs the risk of encountering the local minimum problem. GA on the other hand is found to be good at finding ''acceptably good'' solutions. The idea to hybridize the two networks has been successful to enhance the accuracy of forecasting. In the present work, the initial weights for the BPN have been obtained by using a GA. Genetic algorithm (GA) which uses a direct analogy of natural behavior, work with a population of individual strings, each representing a possible solution to the problem considered. Each individual string is assigned a fitness value, which is an assessment of how good a solution is to a problem. The high-fit individuals participate in ''reproduction'' by crossbreeding with other individuals in the population. This yields new individual strings as off-spring, which share some

features with each parent. The least-fit individuals are kept out from reproduction and so they ''die out''. A whole new population of possible solutions to the problem is generated by selecting the high-fit individuals from the current generation. This new generation contains characteristics, which are better than their ancestors. The parameters, which represent a potential solution to the problem, genes, are joined together to form a string of values referred as a chromosome. In this research, a real coding system has been adopted for coding the chromosomes. The network configuration of the BPN for the present work is 3-2-1. Therefore, the numbers of weights (genes) that are to be determined are 3 X 2 + 2 X 1 = 8. With each gene being a real number, and taking the gene length as 5, the string representing the chromosomes of weights will have a length of 8 X 5 = 40. This string represents the weight matrices of the input-hidden-output layers. An initial population of chromosomes is randomly generated. Weights from each chromosome have been extracted. For cross over, we have used a two-point cross over selected at random and the selection is made on the basis of fitness value (1/MSE) . Similar training process has also been used for other architectures. The strategy for implementation of GA-BPN algorithm is explained in figure-2 below.

Figure-2 implementation strategy for GA-BPN

4.7 Results from GA-BPN Method

The GA-BPN has been implemented by taking different architectures with different population size. For each value of population, the program has been executed 5 times and the root mean square error (RMSE) value has been calculated. The best result (the case having lowest RMSE value) among all these cases (indicated as bold italic letters) has been considered for forecasting.

Various parameters used in this research are:

Architecture: 3-2-1 and 3-3-1
Number of hidden layers: 01
Number of input neurons: 03
Number output neurons: 01
Selection: Based on fitness value
Fitness function: 1/ (Mean Square Error)
Crossover: Two point Crossover
Stopping criteria: when fitness converged

The training results are shown in table-2 and table-3.

Table-2 contains the RMSE and average RMSE values of GA-BPN 3-2-1 architecture with various population sizes. The GA-BPN program has been executed five times and the RMSE values are given in Run-1 to Run-5 column. Finally the average RMSE has been calculated. This table is used to find a suitable value of population size that produces the lowest error for experiment of GA-BPN 3-2-1 architecture. It can be observed from the table that, our architecture GA-BPN-3-2-1

P	Run-1 RMSE	Run-2 RMSE	Run-3 RMSE	Run-4 RMSE	Run-5 RMSE	Average RMSE
30	0.0669	0.0669	0.0669	0.0669	0.0669	0.0669
40	0.0783	0.0783	0.0783	0.0783	0.0783	0.0783
50	0.0794	0.0794	0.0794	0.0794	0.0794	0.0794
60	0.0703	0.0703	0.0703	0.0703	0.0703	0.0703
70	0.0604	0.0604	0.0604	0.0604	0.0604	0.0604
80	0.0713	0.0713	0.0713	0.0713	0.0713	0.0713
90	***0.0205***	***0.0205***	***0.0205***	***0.0205***	***0.0205***	***0.0205***
100	0.0454	0.0454	0.0454	0.0454	0.0454	0.0454

Table-2: Training Results of GA-BPN 3-2-1 architecture with different population size(**P**)

performs better with population size of 90 by producing lowest value of RMSE.

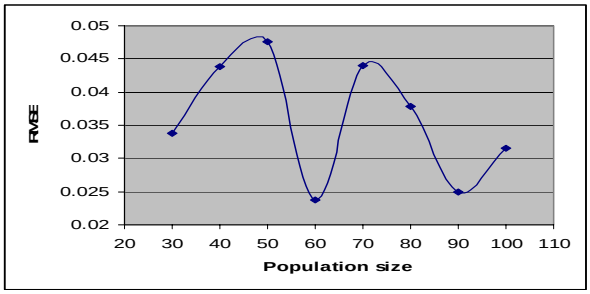
Figure-3 population size versus error rate (GA-BPN 3-2-1)

Similarly the training results have been calculated for the architecture 3-3-1. Results are shown below.

P	Run-1 RMSE	Run-2 RMSE	Run-3 RMSE	Run-4 RMSE	Run-5 RMSE	Average RMSE
30	0.0338	0.0338	0.0338	0.0338	0.0338	0.0338
40	0.0439	0.0439	0.0439	0.0439	0.0439	0.0439
50	0.0476	0.0476	0.0476	0.0476	0.0476	0.0476
60	**0.0237**	**0.0237**	**0.0237**	**0.0237**	**0.0237**	***0.0237***
70	0.0440	0.0440	0.0440	0.0440	0.0440	0.0440
80	0.0377	0.0377	0.0377	0.0377	0.0377	0.0377
90	0.0249	0.0249	0.0249	0.0249	0.0249	0.0249
100	0.0315	0.0315	0.0315	0.0315	0.0315	0.0315

Table-3: Training Results of GA-BPN 3-3-1 architecture with different population size (**P**)

It can be observed from the table that, our architecture GA-BPN-3-3-1 performs better with population size of 60.



Forecasted Values (GA-BPN)

Once the training is over, the forecasted values for next days have been calculated. The values are given in table-4 below.

Techniques	Actual /Target Value	Forecasted Value	Squared Error	Error (RMSE)
GA-BPN 3-2-1, Population size=90	0.5876	0.585405	4.82E-06	0.020585
	0.6672	0.687654	0.000418	
	0.6892	0.718321	0.000848	
GA-BPN 3-3-1, Population size=60	0.5876	0.574182	0.00018	0.023733
	0.6672	0.668198	9.96E-07	
	0.6892	0.728043	0.001509	

Table-4 Forecasted values, GA-BPN 3-2-1 and 3-3-1 architecture

Similarly the results have been calculated for the architecture 3-2-2-1 and 3-3-3-1. The forecasted values are given in table-5 below.

Techniques	Actual /Target Value	Forecasted Value	Squared Error	Error (RMSE)
GA-BPN 3-2-2-1, Population size=60	0.5876	0.605177	0.000309	0.010400
	0.6672	0.666389	6.58E-07	
	0.6892	0.693057	1.49E-05	
GA-BPN 3-3-3-1, Population size=70	0.5876	0.601256	0.000186	0.011996
	0.6672	0.653031	0.000201	
	0.6892	0.682530	4.45E-05	

Table-5 Forecasted values, GA-BPN 3-2-2-1 and 3-3-3-1 architectures

The Root Mean Square Error (RMSE) has been calculated as:
RMSE=Square root of [(∑ (Target Value – Obtained Value)^2)/number of patterns]

5. RESULTS ANALYSIS AND CONCLUSIONS

From table-4 and table-5, it can be analyzed that, the RMSE obtained is approximately in between 0.01 to 0.02, which is quite appreciable. In our case the architecture 3-2-1 produces a better result than the architecture 3-3-1 and the architecture 3-2-2-1 produces a better result than the architecture 3-3-3-1. Also from table-4 and table-5, it can be observed that the structure of the network, selection of other parameters affect the result. In

our case the architecture performs better with two hidden layers having two neurons in each hidden layer than that of other architectures used in this research.

This research also has some limitations. They are:

- Training and Testing of all models are conducted offline.
- All the results given in this research are experimental basis. These have been obtained only as the out put of the program developed for this purpose. No mathematical proof could be given for the results obtained.
- We have considered only one input factor for forecasting i.e the previous three days load data. However other factors such as weather data and temperature could also be used as input parameters.
- All these results given are only problem specific i.e they are applicable only to the specific data used in this research. It can not be generalized. Achievements of this research need to be verified in other fields of applications to make these models more generalized.

REFERENCES:

[1] Sexton R.S., Dorsey R.E. and Sikander N.A. (2002), 'Simultaneous Optimization of Neural Network Function and Architecture Algorithm', Decision Support Systems. 1034: 1-13.

[2] Rajasekaran S. & Vijayalakshmi P. G.A. (2007), "Neural Networks, Fuzzy Logic and Genetic Algorithms – Synthesis and Applications" , Prentice-Hall of India Private Limited.

[3] Gowri T. M. and Reddy V.V.C.(2008), 'Load Forecasting by a Novel Technique Using ANN', ARPN Journal of Engineering and Applied Sciences, Vol.3, No. 2.

[4] Papalexopoulos A.D., Hao S., and Peng T.M.(1994), 'An Implementation of a Neural Network Based Load Forecasting Model for the EMS', IEEE Transactions on Power Systems, 9:1956–1962.

[5] Wang X. and Tsoukalas L. H (2004),'Load Analysis and Prediction for Unanticipated Situations', Bulk Power System Dynamics and Control - VI, 2004, Cortina d'Ampezzo, Italy.

[6] Khan A.U, Bandopadhyaya T.K and Sharma S. (2008), ' Genetic Algorithm Based Back Propagation Neural Network Performs better than Back propagation Neural Network in Stock Rates Prediction', International Journal of Computer Science and Network Security, Vol. 8, No.7.

[7] Mishra S. (2008), 'Short Term Load Forecasting using Neural Network Trained with Genetic Algorithm & Particle Swarm Optimization' ,First International Conference on Emerging Trends in Engineering and Technology, IEEE Computer Society,978-0-7695-3267-7/08.

[8] Montana D. and Davis L. (1989), 'Training Feedforward Neural Networks Using Genetic Algorithms', Proceedings of Eleventh International Joint Conference on Artificial Intelligence. 762-767.

[9] Peng M., Hubele N.F., and Karady G.G.(1992), 'Advancement in the Application of Neural Networks for Short-Term Load Forecasting', IEEE Transactions on Power Systems, 7:250–257.

[10] Asar A. and Syed E., Hassnain R. (2000), 'Short term load forecasting from an artificial neural network perspective', INMIC Fourth IEEE national Multi topic conference, Islamabad, Pakistan.

[11] Jain A.K., Mao J. and Mohiuddin K.M. (1996), 'Artificial Neural Networks: A Tutorial', IEEE Computer Special Issue on Neural Computing. 31-44.