International Journal of Electrical, Electronic and Communication Sciences
ISSN: 2517-9438
Vol:2, No:9, 2008

Evolutionary Techniques Based Combined Artificial Neural Networks for Peak Load Forecasting

P. Subbaraj, V. Rajasekaran

Abstract**—**This paper presents a new approach using Combined Artificial Neural Network (CANN) module for daily peak load forecasting. Five different computational techniques –Constrained method, Unconstrained method, Evolutionary Programming (EP), Particle Swarm Optimization (PSO), and Genetic Algorithm (GA) – have been used to identify the CANN module for peak load forecasting. In this paper, a set of neural networks has been trained with different architecture and training parameters. The networks are trained and tested for the actual load data of Chennai city (India). A set of better trained conventional ANNs are selected to develop a CANN module using different algorithms instead of using one best conventional ANN. Obtained results using CANN module confirm its validity.

Abstract**—**Combined ANN, Evolutionary Programming, Particle Swarm Optimization, Genetic Algorithm and Peak load forecasting.

I. Introduction

LOAD forecasting becomes much more important in power system planning, operation and in a deregulated electricity market. Particularly, daily peak load forecasting plays a vital role in generation scheduling. The conventional load forecasting techniques use conventional smoothing techniques, regression models and statistical analysis. These techniques fail to produce an accurate load forecast because of their inherent limitations [19]. In the recent years, many studies have been reported and many models have been developed for load forecasting using the computational intelligence methods such as Fuzzy systems and Artificial Neural Networks [1-5]. Especially, several ANN approaches have been studied and successfully employed in many load forecasting applications [6-10]. The ANN possess the properties required for load forecasting applications, such as, non linear and smooth interpolation, ability to learn complex non linear complex mappings, adapting themselves to different statistical distributions and ability to learn from the past experiences to improve their performance.

P. Subbaraj is with the A.K. College of Engineering, Krishnankoil, Tamilnadu, India.(phone: +91-04563-289 042; fax: +91-04563-289 322; e-mail: subburaj_potti@ yahoo.com).

V. Rajasekaran is with the P.S.N.A. College of Engineering and Technology, Dindigul, Tamilnadu, India.(phone: +91-0451-255 4409; fax: +91-0451-255 4249; e-mail: rajasekaran_vm@ yahoo.co.in).

In this paper, a number of neural networks are trained with different architecture and different training parameters for the given input and output relationships. Of these networks, two cases, (i) five and (ii) ten neural networks with best performance are selected and combined together to develop a CANN module for load forecasting application rather than using only the single best trained ANN. The Optimal Linear Combination [11] of these trained networks is achieved by five different computational techniques, such as, Unconstrained and Constrained methods [11], and the proposed methods Evolutionary Programming, Particle Swarm Optimization and Genetic Algorithm. The developed CANN module, (i) with five trained neural networks (ii) with ten trained neural networks with the above five techniques is tested and the results of these combinations are compared themselves and with the conventional ANN with best performance.

The developed CANN module is proposed to achieve Medium Term Load Forecasting (MTLF) [3, 13], where the objective is to predict daily peak load for the months of May, June and July 2005 (summer) for the power system of Chennai city (Tamilnadu State - India).

II. System Data And Architecture of ANN

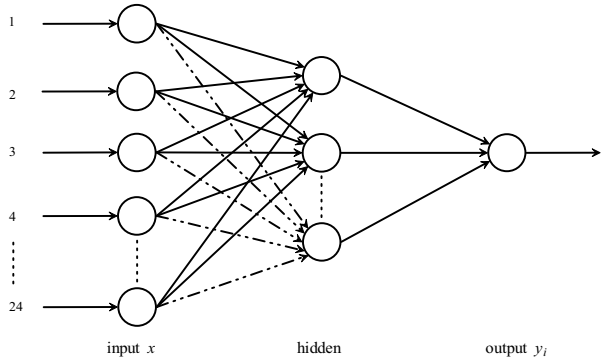

Fig. 1 General Architecture Representation of ANN

A three layer feed forward ANN with a sigmoid function is selected for ANN modeling [6-8]. The back propagation algorithm is adopted to train the ANN. Using past experience and heuristics the structure and the input variables [12] are

selected. Fig.1 shows the general architecture representation of ANN and Table 1 shows different input variables for ANN.

Table I shows the list of selected input variables. With those input variables selection, a number of ANNs are trained with different architectures and different training parameters. Of these trained networks, based on the error measures (performance) the best networks are selected and these are combined together to improve the accuracy of prediction. The daily load forecasting has been applied for Chennai city with the help of these selected network structures and by using the combined ANN modules. The Mean Absolute Percent Error (MAPE) and Root Mean Squared Error (RMSE) are the error measures used to analyze the results [3, 19]. They are defined by (1) and (2).

TABLE I
DIFFERENT INPUTS TO THE ANN

Input Variables	Index	
Peak Load of previous day	L(d–1)	(1)
Temperature of previous day(Mean,Max,Min)	T(d–1)	(2-4)
Relative Humidity of previous day(Mean,Max,Min)	RH(d–1)	(5-7)
Wind speed of previous day(Max)	WS(d–1)	(8)
Peak Load of previous week	L(d–7)	(9)
Temperature of previous week(Mean,Max,Min)	T(d–7)	(10-12)
Relative Humidity of previous week(Mean,Max,Min)	RH(d–7)	(13-15)
Wind speed of previous day(Max)	WS(d–7)	(16)
Temperature of forecast day(Mean,Max,Min)	T(d)	(17-19)
Relative Humidity of previous day(Mean,Max,Min)	RH(d)	(20-22)
Wind speed of previous day(Max)	WS(d)	(23)
Day index	d	(24)

$$MAPE = \left(\frac{1}{N} \sum_{i=1}^{N} \frac{|y_i - d_i|}{d_i} \right) \times 100 \quad (1)$$

$$RMSE = \sqrt{\frac{1}{N} \sum_{i=1}^{N} \left(y_i - d_i \right)^2} \quad (2)$$

where y_i is the predicted load and d_i is the actual (desired) load for a day i and N is the total number of test data.

III. COMBINATION OF ANNS

In this section, the Optimal Linear Combination (OLC) [11] problem is formulated for a set of *n*-trained neural networks. There are *n*-trained artificial neural networks for the given input-output relations.

x is the input to all the neural networks,

y_j is the predicted output for the input x, (j = 1, 2, … n),

d is the desired output for the given input x and

$e_j = d - y_j$ is the error of the jth neural network for the given input x,

$y_c = a_1 y_1 + a_2 y_2 + ... + a_n y_n$ is the linear combination of the outputs of n-trained neural networks for a given input x and the corresponding error for the input x is given by $e_c = d - y_c$.

a_j is the combination weight associated with ANNs outputs (j = 1, 2 … n).

Fig. 2 shows the general block diagram representation of combining the *n*-trained neural networks. The input x is applied to the all *n*-trained ANNs. The outputs $y_1, y_2, ...y_n$ are predicted and obtained from the *n*-trained ANNs and then given to the combining module. This combining module follows the five different computational algorithms given below and produces the combined output y_c.

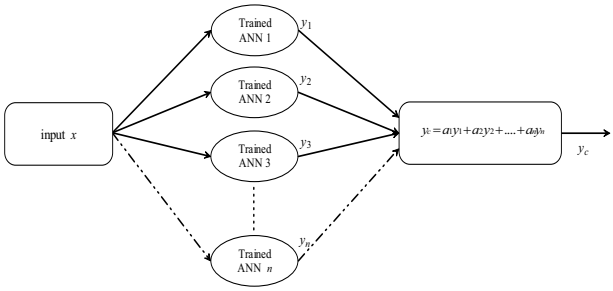

Fig. 2 General representation of CANN module for combining *n*-trained neural networks

The problem is to find best values for the combination weights a_j, (j= 1,2 … n) using the optimal linear combination of the outputs of *n*-trained ANNs. The OLC is defined by the optimal combination weights vector that minimizes the expected loss

$$\int_s l\left(d_c\left(X^c : a\right)\right) dF_X^c, \quad (3)$$

where s is the support of F_X^c and l is a loss function. The input x is as an observation of a random variable X^c from a multivariate distribution function F_X^c. Although various loss functions could be followed, here the loss function is restricted to squared-error loss, $l(e_c) = (e_c)^2$. The objective is then to minimize the mean squared error (MSE) of y_c given by,

$$\mathrm{MSE}\left(y_c\left(x : a\right)\right) = E\left(e_c\left(x : a\right)^2\right) \quad (4)$$

In this paper five different computational techniques are pursued to achieve the optimal linear combination of weights for *n*-trained neural networks by minimizing the MSE and so that to obtain the required performance measures (1, 2) for the selected load forecasting problem.

IV. IMPLEMENTATION OF COMPUTATION TECHNIQUES

A. Unconstrained Method

$$a = Z^{-1} \times b, \quad (5)$$

where Z is a $n \times n$ matrix and b is a $n \times 1$ vector and

$$Z_{ij} = \frac{1}{|N|} \sum_{k=1}^{|N|} y_i(x_k) \times y_j(x_k) \text{ for all } i,j \quad (6)$$

$$b_i = \frac{1}{|N|} \sum_{k=1}^{|N|} d \times y_i(x_k) \text{ for all } i \quad (7)$$

B. Constrained Method

$$\sum_{j=1}^{N} a_j = 1 \tag{8}$$

$$a = \frac{C^{-1} \times \vec{1}}{\vec{1} \times C^{-1} \times \vec{1}^t} \tag{9}$$

where C is a $n \times n$ matrix and $\vec{1}^t$ is a $n \times 1$ vector with all components equal to 1

$$C_{ij} = \frac{1}{|N|} \sum_{k=1}^{|N|} c_i(x_k) \times c_j(x_k) \text{ for all } i,j \tag{10}$$

$|N|$ is the cardinality of N and $y_i(x_k)$ is the output of the ith ANN for the kth input in the data set N.

C. Evolutionary Programming

Evolutionary Programming [14] searches for and finds the optimal solution by evolving a population of candidate solutions over a number of iterations. Evolutionary Programming needs an initial population to start with, like natural evolution. This is called parent population it is mutated using a combination rule to produce the off spring population. In this technique a strong behavioral link is sought between each parent and its offspring. It emphasizes probabilistic nature of solution by conducting a stochastic tournament for survival. The surviving candidates of this generation will form the parent population of next generation. The process continues until it reaches the global optimum solution.

The implementation of EP for the selected problem follows the sequence below.

1. An initial population of parent solutions p_k, $k = 1, 2, ...m$ (where m is the number of parents) is generated at random, from a feasible range of each variable, $a_j \left(j = 1, 2, ...n \right)$.

2. The objective function value (fitness) associated with each solution p_k is obtained by a fitness function according to $e_c = d - y_c$.

3. An off-spring $p_k^{'}$ is created from each parent by adding a Gaussian random variable $N\left(0, \sigma_k^2\right)$ to the elements a_j of parent p_k.

$$a_j^{'} = a_j + N\left(0, \sigma_k^2\right) \tag{11}$$

$$\sigma_k = \beta \times \frac{f\left(y_c\right)}{f_{\min}} \times \left(a_{j\max} - a_{j\min}\right) \tag{12}$$

where $f_{\min}$ is the fitness value associated with the best feasible solution in the population, β is the scaling factor, $a_{j\max}$ is the maximum limits of the jth element, $a_{j\min}$ is the minimum limits of jth element. The feasible solutions are evaluated and applied to the competition pool.

4. Each feasible off spring $p_k^{'}$ is evaluated according to $e_c = d - y_c$

5. For each feasible candidate, parent or off spring, a weight w_s is assigned to the competition.

$$w_s = \sum_{i=1}^{z} w_i$$

$$w_i = 1 \quad \text{if } f\left(p_k\right) \langle f\left(p_r\right)$$

$$w_i = 0 \quad \text{otherwise}$$

where z is the number of competitions w_i is a number of 0 or 1, which represents win 1, or loss 0 as p_k completes with a randomly selected individual p_r in the combined population and $f\left(p_r\right)$ is the fitness of a randomly selected individual p_r.

6. The feasible competitors are ranked in the descending order of w_s. The first m solutions survive and are transcribed along with their elements to form the basis of next generation.

7. The process is terminated when the given maximum iteration count is reached or when there is no appreciable change in the best solution for a certain number of iterations.

The selected control parameters for EP:

Population size	: 30
Scaling factor	: 0.2
Number of iterations	: 1000

D. Particle Swarm Optimization

Particle Swarm Optimization (PSO) is a stochastic global optimization technique developed by Dr. Eberhart and Dr. Kennedy in 1995 based on simulation of social behaviors of [15, 16] bird flocking and fish schooling. It uses a number of particles that constitute a swarm. During flight or swim each particle adjusts its position according to its own experience and the experiences of neighbors, which includes the current velocity and position and the best previous position encountered by it and its neighbors.

In a two dimensional search space, let x and v be the particle coordinates for position and velocity respectively. The best previous position of a particle is obtained and denoted as p_{best}. The index of the best particle among all the particles in the group is denoted as g_{best}. The modified velocity and the position can be calculated using the following formulas:

$$v_i^{k+1} = wv_i^k + c1rand(\,) \times \left(p_{best_i} - s_i^k\right) + c2rand(\,) \times \left(g_{best} - s_i^k\right) \tag{13}$$

where

v_i^k	: velocity of particle i at iteration k
w	: inertia weight factor
c_1, c_2	: acceleration constant
$rand(\,)$	: uniform random value between 0 and 1

s_i^k : current position of particle i at iteration k

p_{best_i} : p_{best} of particle i

g_{best} : g_{best} of the group

Proper selection of weight w provides a balance between global and local explorations, thus requiring less iteration on average to find a sufficiently optimal solution. The following is the equation to obtain weight w.

$$w = w_{\max} - \frac{w_{\max} - w_{\min}}{iter_{\max}} \times iter \quad (14)$$

where $w_{\max}$ is the initial weight
$w_{\min}$ is the final weight
$iter_{\max}$ is the maximum number of iterations
$iter$ is the current number of iteration,

using the above procedures, a certain velocity which gradually gets close to p_{best_i} and g_{best} can be calculated. The current position of ith particle can be modified by (15).

$$s_i^{k+1} = s_i^k + v_i^{k+1} \quad (15)$$

The convergence speed of each particle could be influenced by the acceleration constants (c_1, c_2). Low values of c_i minimize the speed of optimization process and require large number of iterations. Large values of c_i numerically unstablize the optimization process. Hence the acceleration constants c_1 and c_2 are often set at 10-20% according to the past experiences.

The sequences of PSO for the selected problem are:

1. Initialize the parameters such as the swarm size, inertia weight factor, acceleration constant etc., Generate the initial trial vectors for a_j, $(j = 1, 2, ...n)$ at random.
2. Calculate the fitness value for the each particle according to $e_c = d - y_c$ in the population pool.
3. Compare the fitness value of each particle with its p_{best}. The particle with the best fitness value among p_{best} is denoted as g_{best}.
4. Modify the velocity v of each particle according to (13).
5. Change the member position s of each particle according to (15).
6. If the fitness value of created off spring is better than the current p_{best} then the p_{best} value is replaced by the current value and if the p_{best} is better than the g_{best} then the best value is set to g_{best}.
7. The process of creating new trials and the modification of each particle will be repeated until the best objective value (minimum value of e_c) is not obviously improved or the given number of total iterations is reached. The PSO control parameters are chosen as:

Population size	: 30
Maximum no. of iteration	: 1000
$w_{\max}$ and $w_{\min}$	: 0.9 and 0.1
Acceleration constant (c_1, c_2)	: 2

E. Genetic Algorithm

Genetic Algorithm(GA) is a global search algorithm based on the principles inspired from the genetic and evolution mechanisms observed in natural systems and populations of living being [17, 18].

A typical GA is usually composed of three operators. The first operator Reproduction or selection is a process in which individual strings (chromosomes) are copied according to a high fitness value; otherwise the individual is eliminated from the solution pool. Recombination or Crossover is the second operator which selects a pair of parents from the population and divides two strings of bits into segments by setting a crossover point at random (single point crossover). The segments of bits from the parents behind the crossover point are exchanged with each other to generate their off spring. The idea of this operation is to create better performing off springs. Mutation is the third operator which plays a secondary role in Gas and prevents the premature stopping of the algorithm in a local solution point. Mutation is an occasional (with smaller probability) random alteration of the value of a string position. In the binary coding, this means changing a 1 to a 0 and vice versa.

Generally GA is designed to maximize the fitness function, which is a measure of quality of each candidate solution in the population. The fitness function designed depending upon system requirement. In this problem the fitness function is taken as,

$$fitval = \frac{1}{1 + objval}$$

The steps used to solve using GA are:

1. Randomly generate a number of chromosome strings for the initial population.
2. Evaluate the fitness of the each candidate solution and calculate the error of the given problem $e_c = d - y_c$
3. Apply the Genetic operations Selection, Recombination and Mutation to the coded binary string matrix to create a new population.
4. If the convergence conditions meet the specified standard, then export the best solution as the final optimization result. Otherwise go to the step 2 for the next iteration.

The following is the list of control parameters for GA:

Selection method	: Roulette Wheel Selection
Recombination method	: Single Point Crossover
Mutation method	: Constant mutation rate=0.001
Number of Individuals	: 50

V. Test Results

The entire work of this selected problem is carried out in AMD Sempron 1.4 GHz processor. The programs for all the algorithms are coded in MATLAB 6.5 software. Initially, the neural networks with the different architecture and different training parameters have been selected for this problem. For the case of architecture the number of hidden neurons has been varied from the range of 1 neuron to 80 neurons, so that 80 different neural networks in terms of architecture are modeled and created for training. These neural networks are

trained with different learning rates and it has been varied from 0.1 to 1.5 with step 0.1. Totally, 1200 networks are obtained and trained with different architecture and with different learning rates. The number of iterations has also been varied from 500 to 20,000 and finally it is set to 10,000 for all the networks.

All these neural networks are trained for the months from January to April 2005 (four months and 120 input data sets). The data sets of May, June and July 2005 (test data) are selected to test the trained networks. To improve the performance of training, the training data can be updated with subsequent months. It is understood that the better neural networks are obtained for the hidden neurons that are varying from 5 to 48, and for the learning rate 0.1 to 0.3. Based on the performance measures given in (1) and (2), for case (i) the outputs of first 5 ranked neural networks with best performance are selected for the combination module and for case (ii) first 10 ranked neural networks are selected to produce combined output and their performances are studied and it is tabulated in tables II and III.

It is found that when more number of networks included in the combination, the performance of the combination is also improved, hence the performance of case (ii) is analyzed hereafter. The results produced by the combined module of case (ii), using the above said five different algorithms are compared with each other and with the best conventional neural network with respect to the performance measures MAPE and RMSE.

TABLE II
COMPARISON OF PERFORMANCES OF DIFFERENT ALGORITHMS
CASE (I) WITH FIVE NE URAL NETWORKS

Method	May 2005		June 2005		July 2005	
	MAPE %	RMSE (MW)	MAPE %	RMSE (MW)	MAPE %	RMSE (MW)
Unconstrained	2.31	28.21	2.37	28.35	2.32	29.31
Constrained	2.30	27.95	2.33	27.96	2.27	28.57
EP based CANN	**2.14**	**23.17**	**2.17**	**22.55**	**2.18**	**23.68**
PSO based CANN	2.29	27.42	2.31	26.93	2.26	28.08
GA based CANN	2.44	32.01	2.42	30.69	2.39	32.38

TABLE III
COMPARISON OF PERFORMANCES OF DIFFERENT ALGORITHMS
CASE (II) WITH TEN NE URAL NETWORKS

Method	May 2005		June 2005		July 2005	
	MAPE %	RMSE (MW)	MAPE %	RMSE (MW)	MAPE %	RMSE (MW)
Unconstrained	2.30	28.17	2.33	28.12	2.29	29.09
Constrained	2.28	27.91	2.31	27.85	2.26	28.51
EP based CANN	**2.12**	**23.12**	**2.14**	**22.46**	**2.15**	**23.57**
PSO based CANN	2.26	27.35	2.27	26.82	2.24	27.98
GA based CANN	2.42	31.94	2.39	30.55	2.36	32.10

Table IV shows the details of selected neural networks with the tested results. From Table IV it is understood that the conventional ANN with the structure of 24-19-1 and with the learning rate of 0.1 produced the best performance values of MAPE and RMSE as 2.92 and 48.86 MW respectively. The ANN with the structure 24-05-1 and the learning rate of 0.3 produced the least performance values among the selected 10 networks as 3.56 and 58.25 MW for the given set of test data of the month MAY 2005.

TABLE IV
BEST SELECTED CONVENTIONAL ANNS

Networks	Topology i/p-hid-o/p	Learning rate	May 2005		June 2005		July 2005	
			MAPE %	RMSE (MW)	MAPE %	RMSE (MW)	MAPE %	RMSE (MW)
ANN1	24-48-1	0.2	3.25	53.17	3.37	55.35	3.52	59.22
ANN2	24-32-1	0.1	3.49	59.97	3.42	57.67	3.35	54.15
ANN3	24-30-1	0.1	3.18	51.75	3.24	53.46	3.21	51.96
ANN4	24-24-1	0.1	3.16	50.26	**2.99**	**49.13**	3.19	50.76
ANN5	24-19-1	0.1	**2.92**	**48.86**	3.10	49.92	3.14	49.55
ANN6	24-18-1	0.2	3.29	56.82	3.16	51.78	**2.96**	**49.02**
ANN7	24-17-1	0.1	3.20	51.58	3.25	53.73	3.22	52.01
ANN8	24-14-1	0.3	3.21	51.60	3.19	52.11	3.11	49.43
ANN9	24-12-1	0.1	3.19	52.67	3.46	58.20	3.41	55.63
ANN10	24-05-1	0.3	3.56	58.25	3.39	56.92	3.29	53.17

Iterations =10,000

Table V gives the details of results produced by the CANN module using the referenced techniques and the best conventional ANN. In comparison with the results in terms of performance measures, all the CANN modules produce satisfactory performance than the best conventional ANN. And among the different algorithms, the EP based CANN module displays a better performance (i.e., the error measures MAPE and RMSE are considerably reduced from the values of 2.92% and 48.86 MW to 2.12% and 23.12 MW respectively) than the other four algorithms and then PSO based CANN module produces a good performance than the other three CANN modules for the selected problem.

TABLE V
COMPARISON OF PERFORMANCES OF DIFFERENT ALGORITHMS

Method	May 2005		June 2005		July 2005	
	MAPE %	RMSE (MW)	MAPE %	RMSE (MW)	MAPE %	RMSE (MW)
Best Conventional ANN	2.92	48.86	2.99	49.13	2.96	49.02
Unconstrained	2.31	28.21	2.37	28.35	2.32	29.31
Constrained	2.30	27.95	2.33	27.96	2.27	28.57
EP based CANN	**2.14**	**23.17**	**2.17**	**22.55**	**2.18**	**23.68**
PSO based CANN	2.29	27.42	2.31	26.93	2.26	28.08
GA based CANN	2.44	32.01	2.42	30.69	2.39	32.38

Figures 3 to 7 show the variations between the actual and forecasted loads using the CANN modules and Fig. 8 shows the variation of actual and forecasted loads using best conventional ANN. Figure 9 shows the comparison between the forecasted loads using EP based CANN module and the best conventional ANN with the structure 24-19-1.

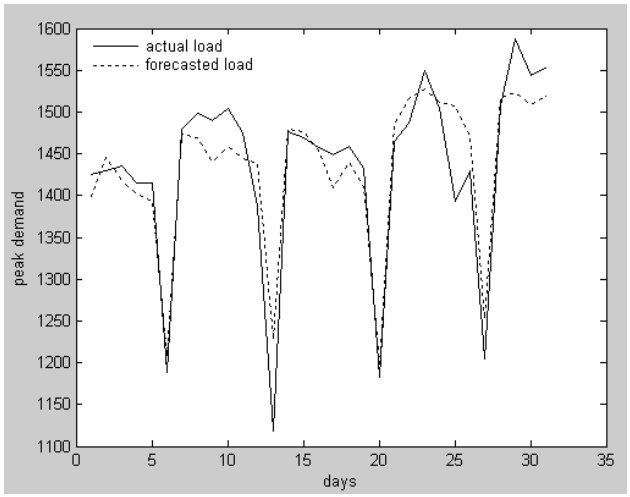

Fig. 3 Variation between actual and forecasted loads for May 2005 using constrained method based CANN module

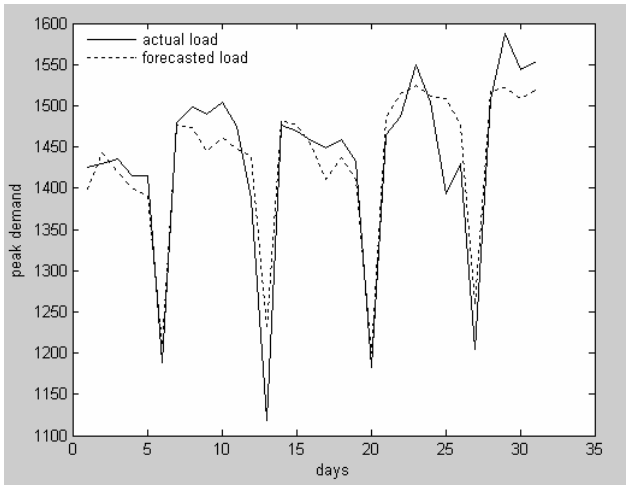

Fig. 4 Variation between actual and forecasted loads for May 2005 using unconstrained method based CANN module

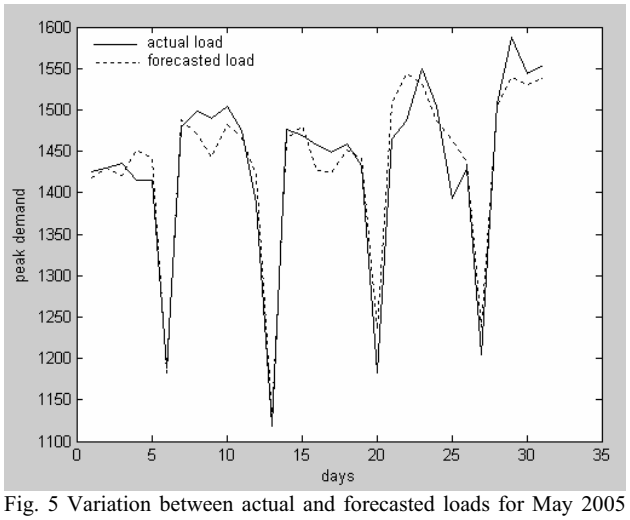

Fig. 5 Variation between actual and forecasted loads for May 2005 using EP based CANN module

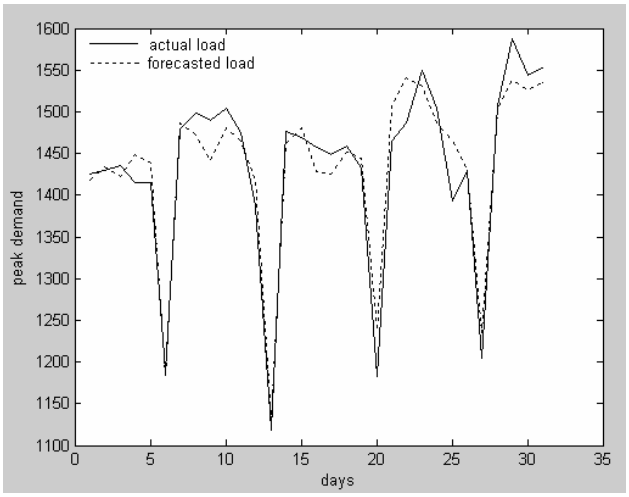

Fig. 6 Variation between actual and forecasted loads for May 2005 using PSO based CANN module

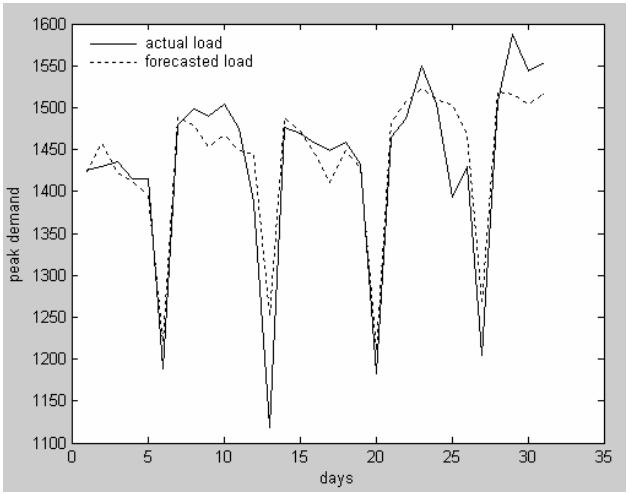

Fig. 7 Variation between actual and forecasted loads for May 2005 using GA based CANN module

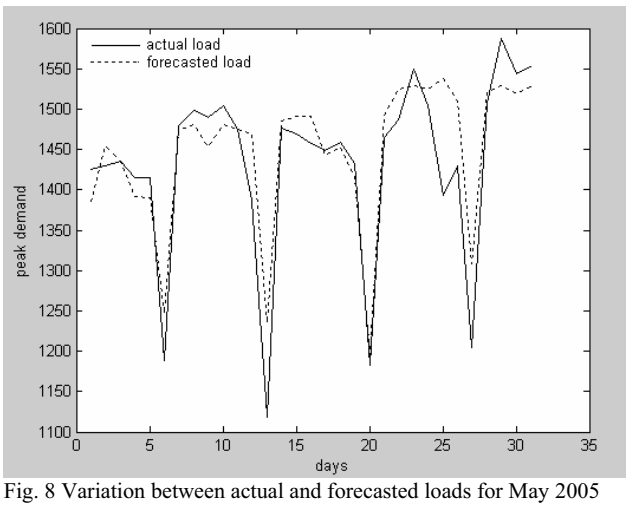

Fig. 8 Variation between actual and forecasted loads for May 2005 using the best conventional ANN with the structure 24-19-1

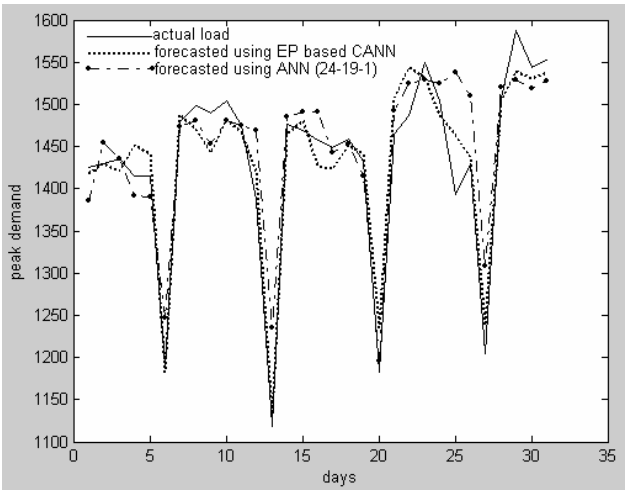

Fig. 9 Comparison between EP based CANN module and best conventional ANN with the structure 24-19-1 for the forecasted loads of May 2005.

VI. Conclusion

This paper proposes a daily peak load forecasting module using five different algorithms. A set of neural networks are trained and some of the networks with best performance are selected for combination. Five different approaches have been discussed and applied to develop an optimal linear combination module that combines the outputs of the selected networks. The obtained results imply that all the proposed methods can provide power system engineer with the reason of forecasting results. Among the five approaches EP based algorithm has shown a bit of higher accuracy with reduced error values of MAPE and RMSE. The proposed combined ANN module using the referenced algorithms can forecast the daily peak load demand more accurately than the single best trained ANN and other conventional methods.

References

[1] S.E. Papadakis,J.B. Theocharis, S.J. Kiartzis and A.G. Bakirtzis, "A novel approach to short-term load forecasting using fuzzy neural networks", *IEEE Transactions on Power Systems*, Vol. 13, No. 2, pp. 480-492, 1998.

[2] Tomonobu Senjyu, Hitoshi Takara, Katsumi Uezato and Toshihisa Funabashi, "One-Hour-Ahead load forecasting using fuzzy neural network", *IEEE Transactions on Power Systems*, Vol. 17, No. 1, pp. 113-118, 2002.

[3] A.A.EI Desouky and M.M. EIKateb, "Hybrid adaptive techniques for electric-load forecast using ANN and ARIMA", *IEE Proceedings*, *Gener. Transm. Distrib*. Vol. 147, No. 4, pp. 213-217, 2000.

[4] A.G. Bakirtzis, J.B. Theocharis, S.J. Kiartzis and K.J. Satsios, "Short term load forecasting using fuzzy neural networks", *IEEE Transactions on Power Systems*, Vol. 10, No. 3, pp.1518-1524, 1995.

[5] S. Rahman and O. Hazim, "A generalized knowledge-based Short-Term load forecasting technique", *IEEE Transactions on Power Systems*, Vol. 8, No. 2, pp.508-514, 1993.

[6] C.N. Lu, H.T. Wu and S. Vemuri, "Neural network based short term load forecasting", *IEEE Transactions on Power Systems*, Vol. 8, No.1, pp. 336-342, 1993.

[7] Alex D. Papalexopoulos, Shangyou Hao and Tie-Mao Peng, "An Implementation of a neural network based load forecasting model for the EMS", *IEEE Transactions on Power Systems*, Vol. 9, No. 4, pp. 1956-1962, 1994.

[8] K.Y. Lee and J.H. Park, "Short-term load forecasting using an artificial neural network", *IEEE Transactions on Power Systems*, Vol. 7, No. 1, pp. 124-132, 1992.

[9] A.S. AlFuhaid, M.A. El-Sayed and M.S. Mahmoud, "Cascaded Artificial Neural Networks for Short-term load forecasting", *IEEE Transactions on Power Systems*, Vol.12, No. 4, pp. 1524-1529, 1997.

[10] I. Drezga and S. Rahman, "Short-Term load forecasting with local ANN predictors", *IEEE Transactions on Power Systems*, Vol. 14, No. 3, pp. 844-858, 1998.

[11] Sherif Hashem and Bruce Schmeiser, "Improving model accuracy using optimal linear combinations of trained neural networks", *IEEE Transactions on Neural Networks* Vol.6,no.3, pp. 792-794,1995.

[12] I. Drezga and S. Rahman, "Input variable selection for ANN based short-term load forecasting", *IEEE Transactions on Power Systems*, Vol. 13, No. 4, pp. 1238-1244,1998.

[13] Tetsuro Matsui, Tatsuya Iizaka and Yoshikazu Fukuyama, "Peak load forecasting using analyzable structured neural network", *Proceedings of IEEE PES 2001*, Winter Meeting.

[14] L.L.Lai, "*Intelligent System Applications in Power Engineering – Evolutionary Programming and Neural Networks*", John Wiley and Sons, Inc., New York, 1998.

[15] J. Kennedy, R. Eberhart, "Particle swarm optimization", *Proceedings of IEEE International Conference on Neural Networks* (ICNNN'95), Perth, Australia, Vol. IV, pp. 1942-1948, 1995.

[16] Konstantinos E. Parsopoulos and Michale N. Vrahatis, "Particle Swarm Optimization method for constrained optimization problems", Department of Mathematics, University of Patras Artificial Intelligence Research Center (UPAIRC), Greece.

[17] Lawrence Davis, "*Hand book of Genetic Algorithms*", International Thomson Computer Press, London.

[18] David E. Goldberg, "*Genetic Algorithms in search, Optimization and Machine learning*", Addison-wesley, 2000.

[19] P.K. Dash &S. Dash and S. Rahman, "A Fuzzy Adaptive Correction Scheme for Short term load forecasting using fuzzy layered neural network", ANNPS'93,*IEEE Proceedings of the second international forum on application of neural networks to power systems*, pp 432-437, 1993.

Biographies

Dr. P. Subbaraj was born in Kalingapatti, Tamilnadu, India in 1957. He received his BE (Electronics and Communication Engineering) in 1979 and ME (Control Systems) in 1981 from PSG College of Technology, Coimbatore, India. He received his PhD (Optimization) in 1991 from IIT-Madras, Chennai, India. He has been with Department of Electrical and Electronics Engineering, Thiagarajar College of Engineering, Maduai, India, since 1981. He is also a member of professional bodies. He is one of the recognized research supervisors in Madurai Kamaraj University, Madurai and Anna University Chennai, India. He is at present with Kalasalingam University as Director Academics. His fields of interest are Computer Control, Signal Processing, Optimization and Evolutionary Computing.

V. Rajasekaran was born in Madurai, Tamilnadu, India in 1971. He received his BE (Electrical and Electronics Engg) in 1994 and ME (Power Systems) in 1997 from Thiagarajar College of Engineering, Madurai, India. He has been with Dept. of Electrical and Electronics Engineering, PSNA College of Engineering, Dindigul, Tamilnadu, India since 1998. His fields of interest are Power System Analysis, Distribution, Artificial Neural Networks and Fuzzy Logic.