Using Fuzzy Neural Network to Solve Short-term Load Forecasting Problems

John R. Birge
Dayong Li
Department of Industrial & Operations Engineering
University of Michigan
Ann Arbor, MI 48109

Technical Report 96-20

1996

Using Fuzzy Neural Network to Solve Short-term Load Forecasting Problems

John R. Birge Dayong Li
The University of Michigan
Department of Industrial and Operations Engineering
Ann Arbor, Michigan 48109-2117

Abstract: In this paper, a new fuzzy neural network based on new concepts about natural neural networks is presented. A simulation forecasting model is then established by using fuzzy neural network and discrete system concepts. Several key steps to enhance forecasting performance of the simulation forecasting model are discussed. They are how to choose and consider proper factors and concerned time-lag effects, how to smooth original data spanning several years and how to handle outputs from the fuzzy neural network. Numerical results show the present simulation model has high forecasting performance for daily forecasting and one-week lead forecasting.

Keywords: short-term load forecasting fuzzy neural network
simulation forecasting learning algorithm

I. INTRODUCTION

Short-term load forecasting is to predict load demand of a power system hour by hour for one day or one week. It plays a very important role in economical and reliable power system operation. Accurate load forecasting results in high power quality, large monetary savings and labor savings. Due to its great importance, many researchers have been attracted to develop high performance forecasting methods.

Various load forecasting methods with different advantages and defects have been published. Roughly classified, they can be cited as stochastic models [1,2,3], expert systems [4,5,6] and neural networks [7,8,9,10]. Stochastic models are good for normal days and have obvious defects in modeling special days such as holidays, weekends, and seasonal changes because of their theoretical limitations. Abnormal and real data may be taken as bad data and removed from the model.

To some extent, expert system methods have overcome some of these disadvantages with better forecasting performances than stochastic models. Extracting knowledge from experienced operation experts is, however, a lengthy process. In practice, the forecasting performance of the expert system mainly depends on the knowledge expressed from experts.

Recently, neural networks have been employed in load forecasting. Encouraging results have been achieved because of their rich nonlinearity and large capacity to simulate complex systems. However, most of this research is still theoretical. Few have practical applications[10]. The key reason is that most of these neural network based models use the back-propagation (BP) model as their core algorithm. Back-propagation is a good model for system simulation but has a time-consuming training process and local minima problem. These characteristics limit its capacity to learn from large-scale trained samples.

In short-term load forecasting, there is generally significant hour-by-hour historical recorded data spanning several years. Hence, training is quite difficult and sometimes unacceptable. It also makes model maintenance troublesome.

In this paper, a new neural network, which is called a fuzzy neural network, is applied to construct a simulation forecasting model for short-time load forecasting. The fuzzy neural network is based on some new concepts about natural neural networks [11]. The method employs fuzzy set theory and parallel neural network structure. In constructing a practically applicable simulation model, this paper introduces several methods for choosing input and output variables, data smoothing and decision rules for handling outputs of the fuzzy neural network.

One feature of this method is its flexibility and capacity in training large amounts of historical data. Hence, the construction of the simulation model becomes straightforward. There is no lengthy training process and no local minimum problem. The fuzzy neural network shows good performance in constructing the forecasting model according to numerical results for an existing power system.

II. FUZZY NEURAL NETWORK

The fuzzy neural network arises from the need to overcome the lengthy learning process and poor convergence of traditional neural networks (typically BP neural networks) and urgent needs to extract fine knowledge from a large amount of original data. The process is like sifting gold from sand. Its basic ideas come from a fuzzy membership function, a fuzzy decision [12,13,14] and the distributed and parallel structure of neural networks[15]. Association is realized by not only integrating the capacity of the network but also fuzzy generalization of the knowledge element. This is the result of combining the neural network and fuzzy logic.

There have been several efforts to realize this combination [16,17,18,20] with good performance. However most of these so-called neural networks still apply the BP neural network as a core program and use fuzzy set techniques to train input and output data. Thus the training speed and convergence problems have not yet been overcome. Recently, a few researchers have tried to make the connection weights fuzzy [19]. Though the concentration is still on adjusting connection weights, learning speed has been greatly enhanced and convergence guaranteed.

Considering new concepts for designing neural networks, we do not try to store knowledge in physical links (analogously biological links are supposed to transmit power and nutrients for control-end signals, while many signals are launched and transmitted wirelessly). We make neurons unto fuzzy input/output information processing units. Each unit possesses the ability to send and receive signals wirelessly. By connecting these fuzzy neurons properly in a distributed and parallel way, we can construct a high performance fuzzy neural network. These design thoughts have significant requirements for state-of-the-art hardware implementation. As motivation in the biology analogy, we believe that the human brain and neurons are complex and fine enough to realize this function.

A. Fuzzy Sets and Fuzzy Decision

For simplicity, the max-membership decision rule is applied for decision making. Other decision rules can also be used for certain problems.

Suppose the domain X, $X \in R$, $\underset{\sim}{A}$, $\underset{\sim}{B}$ are fuzzy subsets of X.

$\mu_A(x)$: membership function for fuzzy subset $\underset{\sim}{A}$.

$\mu_B(x)$: membership function for fuzzy subsets $\underset{\sim}{B}$.

We suppose the following max-membership fuzzy decision laws:

If $\mu_A(x) > \mu_B(x)$, then we judge x belongs to $\underset{\sim}{A}$;

If $\mu_A(x) < \mu_B(x)$, then we judge x belongs to $\underset{\sim}{B}$;

If $\mu_A(x) = \mu_B(x)$, then we judge x belongs to $\underset{\sim}{A}$ and $\underset{\sim}{B}$ at the same time.

B. Fuzzy Neural Network

1. Fuzzy Neuron

We suppose the following standard form for fuzzy neurons (illustrated in Fig. 1).

$$y = f(X, K, T) \tag{1}$$

X: input vector of a neuron
K: knowledge element (input part of a sample)
T: threshold for a neuron
y: output of a neuron

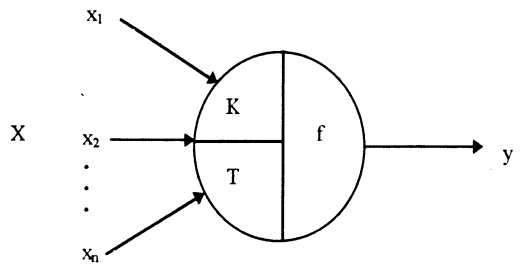

Fig. 1 An Illustration For A Fuzzy Neuron

In FNN, we have five types of neurons, input neurons (IN), knowledge neurons (KN), category neurons (CN), output neurons (ON) and a threshold neuron (TN). They are all derived from the standard form of a fuzzy neuron.

1) input neuron

An input neuron is placed in the input layer in FNN. It has only one input. No knowledge is stored in an input neuron. It has no threshold T. We suppose it as

$$y = x \tag{2}$$

So that output equals input for an input neuron. We use the notation x as an element of vector X while y is the output for an input neuron.

2) knowledge neuron

A knowledge neuron is placed in the front part of the knowledge layer. Its role is just to store knowledge. A knowledge element is generalized and extended around this point through a fuzzy membership function.

We suppose it has the form

$$y = e^{-\frac{(X-K)^2}{\sigma}} \tag{3}$$

We make element K fuzzy by adding a fuzzy membership function (3). This function can be taken as a membership function for knowledge element K. The parameter σ represents the degree of fuzziness. Generally, we take it as 1.0. We can adjust it according to the needs of real-world problems.

3) category neuron

A category neuron is placed in the knowledge layer. Its role is to produce the degree of membership for one knowledge category. This membership value for the knowledge category is equal to that of the point with the largest membership function value (possibility) in one knowledge category. So we suppose it has the form

$$y = \max_{i=1}^{n}(x_i) \tag{4}$$

Note: n is the number of inputs to a category neuron. It shows that there are n knowledge elements under this knowledge category.

4) output neuron

An output neuron is placed in the output layer. Its role is to produce a definite output 1 or 0 according to its own threshold t, where t is sent from a threshold neuron in the output layer.

In this case,

$$y = f(x,t) = \begin{cases} 1, & \text{if } x \geq t \\ 0, & \text{if } x < t \end{cases} \tag{5}$$

Each output neuron in the output layer corresponds to one output category. If the output representing a category is 1, then it says the current input vector belongs to this category, otherwise not.

5) threshold neuron

There is only one threshold neuron in the output layer. A threshold is to produce a dynamic and changeable threshold for each output neuron. Its function is the same as that of a category neuron in the knowledge layer. It has the form

$$y = \max_{i=1}^{n}(x_i), \tag{6}$$

where n is the input number of a threshold neuron.

2. Fuzzy Neural Network Structure

Consider the XOR problem in Table 1 as an example. Fig. 2 is the basic and original FNN network structure before learning the XOR problem. Fig. 3 is the final FNN network structure after 8 samples have been learned. From these two figures, it can be found that FNN has three layers. They are the input layer, the knowledge layer and the output layer.

Table 1 XOR problem

NO	Input			Output
1	0	0	0	0
2	0	0	1	1
3	0	1	0	1
4	0	1	1	0
5	1	0	0	1
6	1	0	1	0
7	1	1	0	0
8	1	1	1	1

The input layer receives the input vector and transmits it to the knowledge layer. The knowledge layer stores a knowledge and processes it .(Note: knowledge is fuzzified and extended here.) The output layer treats (defuzzifies) the output values from the knowledge layer.

In the input layer, there are three input neurons (IN) corresponding to three factors of the input vector.

The knowledge layer of FNN, which is a supposed place to store knowledge in a fuzzy way in the imaginary brain, is self-organized and self-improved. With the learning process going on, the knowledge layer has more and more knowledge elements until the training process stops. In the front part of the knowledge layer, there are eight knowledge neurons (KN) corresponding to 8 stored samples. The number in a neuron represents the number for a knowledge element. Here, all eight samples of the XOR problem are studied and stored.

The output layer is to treat (defuzzify) the output values from the knowledge layer by the max-membership decision rule. From Fig. 3, we can find two outputs (y1 and y2) and two output neurons (ON). This is because there are two output states (0 and 1) for the XOR problem. It can be also said that there are two categories from the eight samples. Thus there should be two category neurons (CN) in the knowledge layer. Note there is a threshold neuron (TN) in the output layer. It helps to produce final outputs from the output layer.

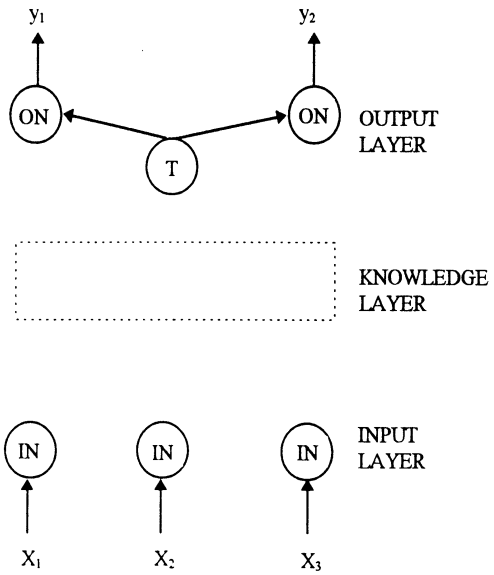

Fig. 2 Basic Network Structure for XOR

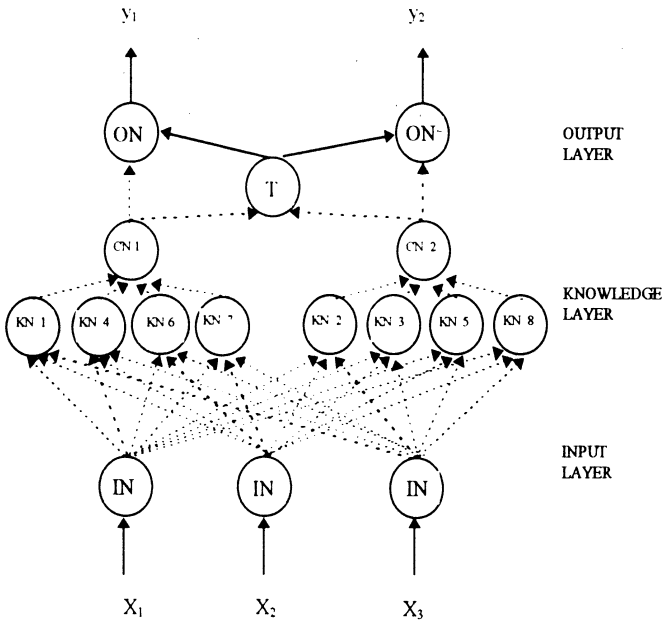

Fig. 3 Final Network Structure for XOR

III. LEARNING ALGORITHMS FOR FNN

A learning algorithm for a neural network is a process through which the neural network can realize all the input-output mappings of samples with acceptable learning error. Different neural networks have different learning algorithms. As most of the current neural networks assume that knowledge is stored in rich combinations of connection weights, the study process is just the process to adjust connection weights. Hence, training a neural network often becomes quite hard.

The FNN of this paper moves a different way on the basis of some new concepts of the human brain. The idea is to store knowledge in a knowledge layer that is supposed in a so-called imaginary brain. The study process is to select and store most typical knowledge elements from samples in a fuzzy way without work on connections.

The FNN can be trained from a zero-knowledge network or a network with learned knowledge. We suppose it develops from an original, basic and zero knowledge network. That is, we assume no knowledge elements (or knowledge neurons) in the knowledge layer. We take the XOR problem for example to express this explicitly. We have the original and basic network shown in Fig 2. Fig. 3 is a final structure of FNN when eight samples have been trained.

Two kinds of study algorithms exist in practical applications. They are called the standard study algorithm and the instant study algorithm, respectively. The standard study algorithm is for the normal learning process that has the capability to choose knowledge as typical as possible and hence to make memory size as small as possible. The instant study algorithm is a one-cycle algorithm. In this case, all the samples are turned into knowledge elements and then the formed knowledge elements placed into FNN directly. Instant study algorithm can be a basis for designing fuzzy neural network databases.

A. Standard Study Algorithm

For simplicity, the study algorithm has two stages, the learning stage and the test stage, respectively. In the learning stage, samples are chosen one by one. Typical samples are added to the knowledge layer under the relevant categories in the form of (3). In the test stage, the trained FNN is checked to confirm that it can cover all samples with acceptable learning error.

First, we input a sample vector and check if the network can produce the desired output for this sample. If yes, we do not adjust the network. Otherwise, we think the trained network cannot cover this sample . This sample is added as a new knowledge element under a related category in the knowledge layer. The new knowledge element is then fuzzified and extended around this point by adding a knowledge neuron with a chosen fuzzy membership function. In this way, the selected function for the knowledge neuron can guarantee that the new knowledge point has the greatest possibility (1.0) among the related categories and also the largest value among all the outputs of output neurons when the vector of this knowledge is input. This process guarantees the trained network can cover the sample. Then we can say this sample has been trained.

The complete study algorithm has the following steps and loops.

step 0 . Input the operation mode to decide to train or test.

step 1 Read in the input number and output number.
Read in the number of total samples.
Define the input vector and output vector.
Define the sample array.
Define a buffer to store all the samples.
Let the learning count = 0.

step 2 Let the learning count increase by 1.

step 3 Training Stage Circulation begins.
The circulation ends when all samples have been studied. Then, go to step 4.

step 3.1 Input one sample.

step 3.2 If the desired output for the sample is a new category, there are no knowledge elements under this category. Set a new category and put a new knowledge element under this category. Go to step 3.1.
If the desired output for the sample is not a new category, there is at least one knowledge element under this category. Go to step 3.3.

step 3.3 Compute the final output of FNN with equations

(2) (3) (4) (5) (6).

If the real output is the same as we expect, pass the sample to the stack for later test or training.

If not, add a new knowledge element in the form of (3) under the concerned category.

step 4 Test Stage Circulation begins.

The circulation ends when all the samples have been tested and the total learning error is computed.

Go to step 5.

step 5 If the total learning error is greater than 0.0, go to step 2. Otherwise, go to step 6.

step 6 Stop and output training or test results as the knowledge base.

B. Instant Study Algorithm

The instant study algorithm is simple. All samples are turned into standard knowledge elements and read as knowledge bases into FNN directly. This is the training process. As each sample has its own concerned knowledge element in FNN, it is definite that the trained FNN can cover all the samples with a learning error of 0.0!

IV. SIMULATION FORECASTING MODEL BASED ON FNN

Generally speaking, we take the forecasting problems as an open-ring system with its own state variables X(t+1), X(t), X(t-1), . . . X(t-p) and control variables U(t), U(t-1), . . . ,U(t-p), where p is the time-lag number. This is to say the discrete system model for STLF should have the following form(illustrated in Fig. 4).

$$X_{t+1} = F(X_t, X_{t-1}, \ldots, X_{t-p}, U_t, U_{t-1}, \ldots, U_{t-p}) \tag{7}$$

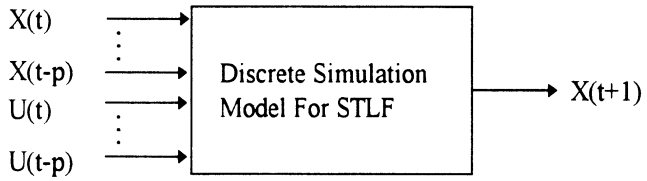

Fig. 4 Supposed Discrete Simulation Model for STLF

In essence , a discrete simulation model for STLF is a nonlinear, multi-period lag model. It is because of the complexity of STLF that we cannot construct a precise mathematical model, such as differential equation model, to simulate the practical problems. Hence, we try to use FNN (a comprehensive, parallel network rather than one group of differential equations) to realize the complicated nonlinear mapping of the discrete model.

The basic forecasting procedures to apply FNN for STLF are as follows:

(1) Define the input and output variables as state variables , control variables and output variables at time t+1, t, t-1, . . . , t-p. where p is the number of periods of time lag. In the proposed fuzzy neural network, there are 1000 outputs which represent 1000 possible output categories in the unit range [0,1] from 0.001, 0.02 , . . . , to 0.998, 0.999, 1.000.

(2) Set up a series of input/output samples from historical data.

(3) By training the model with formed samples, we can obtain an approximate discrete simulation model from limited (not unlimited) but rich historical data.

(4) Use the established simulation model to conduct load forecasting hour by hour.

What is mentioned above is just a principal procedure. It cannot produce satisfactory forecasting performance. Additionally the forecasting performance for a forecasting model depends on many other considerations such as

- how to select proper input variables,
- how to consider time-lag effects of input variables,
- how to smooth original data over different years,
- how to handle the outputs from FNN to produce the best forecasting value.

In Section V, we elucidate how to consider these factors.

V. KEY PROCEDURES TO ENHANCE FORECASTING PERFORMANCES

A. How To Select Proper Input Variables And How To Consider Time-Lag Effects Of Input Variables

Four types of factors affect load demand at time t+1. They are past load, past weather , day type, and time. The four kinds of factors can be further divided more exactly as follows.

load: load at time t, t-1, t-2, . . . ,t-p.
weather: weather type, temperature , humidity, wind speed, wind direction, sky cover (rain or not), snow(or not) at time t, t-1, t-2, . . . ,t-p.
day-type: weekdays-weekends, holidays at time t, t-1, t-2, . . . ,t-p
time: season, month, week, date, hour,

where p is the number for time-lag effect.

There are tradeoffs between computational efficiency and forecasting accuracy. Not all variables need be considered. To have better forecasting performance, we always hope to consider more variables and more time lag. The computation task, hence, becomes heavier. The problem is to find balance. Our suggested way is to apply a combination of the following three methods to find proper variables.

(1) correlation analysis,
(2) testing forecasting computation,
(3) checks with experienced operation experts.

In the forecasting computation for two large neighboring electric utilities, the inputs include

- month code,
- week code,
- hour code,
- Holiday code with 3 hour time lag,
- temperatures at five sites (three large cities and two metropolitan airports) with time lag of 5,
- relative humidity at the two airports,
- load with 27-hour time lag.

Actual computation tests show the above selected inputs variables are sufficient for good results.

B. How To Smooth Original Data From Different Years

We consider large amounts of historical original data spanning over three or four years. During this period, the number of consumers and the load of previous customers changes smoothly, sometimes rapidly. The load change depends on the prosperity of the economy, consumer structure, and macro weather changes. A problem is to smooth these data or convert the old data into a present value.

Two methods of smoothing the changes [10] are used. The first and simple one is to compute an average flat growth rate over the previous years. The historical load data is turned into the present values by multiplying by the concerned whole growth rate. A more reasonable method is to compute load

growth rates for different temperatures and hours. For the temperature of 55° to 85°, the occurrence of load for every two degree within this range is considered for each year. Then the average of each year is calculated. The ratios are calculated based on these averages using the equation:

$$ratio(T) = \frac{Load_{ave}^{year1}(T)}{Load_{ave}^{year2}(T)}, \tag{8}$$

where T denotes temperature. The procedure is applied to all 24 hours. This method takes into account the variation in load in different seasons. This method was implemented in [10] and was said to give consistent growth ratios for all the hours and to yield better results than other methods.

In this research, the above two methods have been tried. The results are not, however, very satisfactory. From careful investigation of the forecasting process, we found the above two methods to be based on the assumption that load changes by an average annual growth rate. Actually, this assumption is not exact enough. Suppose loads of different months and seasons have very different growth rates compared to the yearly load growth rates. Using yearly load growth rates to replace load growth rates of different months often results in forecasting accuracy fluctuations of about 2%.

In this paper, we smooth the data month by month (not year by year). We compute an average load for one month. Original loads of the month are then divided by double their concerned monthly average load. This process is repeated for different months of different years. Forecasting computations show this data smoothing method is better and more stable than the previous methods. The further advantage is that the latest samples can be turned into knowledge in a timely manner without the need to smooth the old knowledge again due to a new average year growth rate.

C. How To Handle The Outputs From FNN To Produce Better Forecasting Results

In this paper, FNN is employed as the core algorithm for the STLF model. To have a satisfactory forecasting performance, we establish 1000 categories in the output layer. The 1000 outputs represent possible forecasting values from 0.001, 0.002, . . . to 0.998, 0.999. Each output is accompanied by a membership value indicating the possibility of this output.

In view of the FNN decision rule, we should choose the output with the largest membership as the forecasted value. Basically, this decision rule can promise 80% correctness and average forecasting error of about 2.5%. However, there are also some forecasted values that are not very good and sometimes quite poor. With many computational tests and deep analysis of the forecasting process of FNN, we found there is always a satisfactory forecasted value among the best three forecasted scenarios with the largest three memberships. Hence the problem is how to select a final forecasted value from the best three outputs. In some cases, we may need a scenario forecast (all three scenario forecasts for stochastic unit commitment models [21]).

A so-called dual membership decision rule is suggested for this purpose. The rule uses input variables grouped into four parts. They correspond to time, holiday type, weather, and load, respectively. The membership for all the input variables is computed as a whole membership. The load membership is only for the load variables (see Fig. 5). The dual membership decision rule uses the two memberships to produce better forecasting performance.

The basic dual membership decision rule can be expressed as two steps:

step 1. select the best three outputs with the largest three whole memberships.

step 2. select a final output with the largest load membership among the best three outputs as the forecast value.

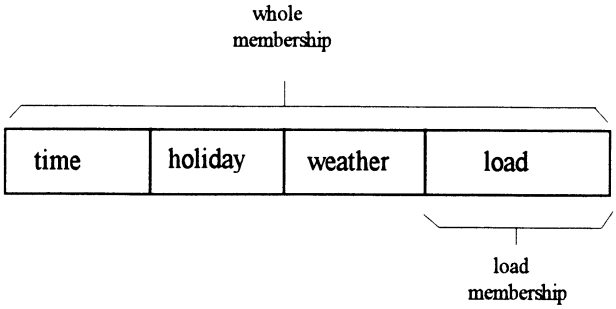

Fig. 5 Dual Membership

The dual membership decision rule finally makes the average daily forecasting error as low as below 2%.

VI. NUMERICAL RESULTS

Based on the forecasted method discussed above, a simulation forecasting model has been constructed in an object-oriented program design using C++. The computations are presently conducted on a Sun Workstation. The following numerical results are one part of the computational tests. From these results, it appears that the proposed forecasting method is correct and feasible.

A. One Day, One Week Forecasting, And 10 Day Forecasting

Unit commitment problems typically require daily forecasting. Purchases, sales and hydropower decisions often require one week forecasts and even 10 day forecasts (or longer). In this model, three types of forecasting can be output within tens of minutes.

Table 2 is a summary to show daily forecasting examples of April 11 through April 17. Table 3 is a summary with computational tests for April 1-20, 1994. From Table 3, we find

- average daily forecasting error is 1.87%,
- average one week forecasting error is 2.21%,
- average 10 day forecasting error is 2.34%,
- average peakload forecasting error is 1.62%

The forecasting errors are acceptable for most practical applications.

Fig. 6-1 through Fig. 6-7 are a group of charts showing forecasting results for one week of April 11 through 17 on the MEPCC. The first 24 hour result is for daily forecasting. The first 148 hour result is for seven day forecasting. The whole 240 hour result is for ten day forecasting.

Table 2 Daily Forecasting of April 11 through April 17 ,1994.

Hour	Mon	Tue	Wed	Thu	Fri	Sat	Sun
1	-0.01	2.73	-0.7	2.69	-0.13	-0	0.35
2	-0.22	3.85	3.87	2.89	-0.18	-2.5	2.72
3	-1.31	3.34	3.53	3.53	0.65	-2.5	3.12
4	0.09	2.67	1.88	3.07	-0.14	-1.1	2.51
5	2.36	3.57	0.51	2.85	-0.15	-1.5	3.25
6	-0.13	2.74	1.07	1.57	-1.9	-1.7	3.52
7	-0.82	3.85	2.04	2.97	-0.4	0.31	3.03
8	-2.88	1.86	-0.2	0.51	-2.49	-0.1	0.71
9	-2.36	0.66	-2.95	-1.65	-2.74	1.7	1.44
10	-2.7	1.29	-1.91	-2.67	-3.03	3.11	0.79
11	-3.76	2.18	-1.08	-2.35	-0.95	4.01	0.38
12	-5.08	0.87	-1.51	-2.67	-0.58	5.43	-0.27
13	-3.99	0.67	-1.79	-2.43	0.41	7.51	-0.14
14	-1.05	0.8	-0.8	-1.43	0.64	6.92	0.58
15	0.11	0.43	-2.11	-1.26	0.66	3.32	1.6
16	0.35	0.15	-0.66	-1.01	-0.46	1.38	1.51
17	1.32	0.3	1	-0.41	-0.26	4.43	1.63
18	2.98	0.64	0.58	-0.84	0.62	5.25	1.69
19	4.23	0.1	0.72	-0.11	3.85	2.69	2.23
20	4.74	-1.28	0.18	-1.21	4.66	-1.1	-1.86
21	5.56	1.04	1.99	0.49	5.56	0.5	-0.6
22	2.6	0.63	1.38	1.05	3.75	1.65	0.89
23	-0.01	0.17	1.01	0.35	0.83	1.08	0.03
24	0.93	1.74	2.41	0.85	2.55	-0.1	-0.13
Aver	2.07	1.56	1.49	1.7	1.57	2.5	1.46
week average(%)				1.76			

Table 3 Summary of Forecasting Results of April 1 through April 20, 1994

Fcst Type	Mon	Tue	Wed	Thu	Fri	Sat	Sun	Average
daily aver error (%)	2.15	1.76	1.53	1.95	1.53	2.73	1.46	1.87
peakload error (%)	1.68	1.91	1.28	1.67	1.36	2.63	0.83	1.62
7 day fcst error (%)	2.33	2.22	2.19	2.08	2.02	2.1	2.55	2.21
10 day fcst error (%)	2.16	2.07	2.1	2.59	2.6	2.59	2.33	2.34

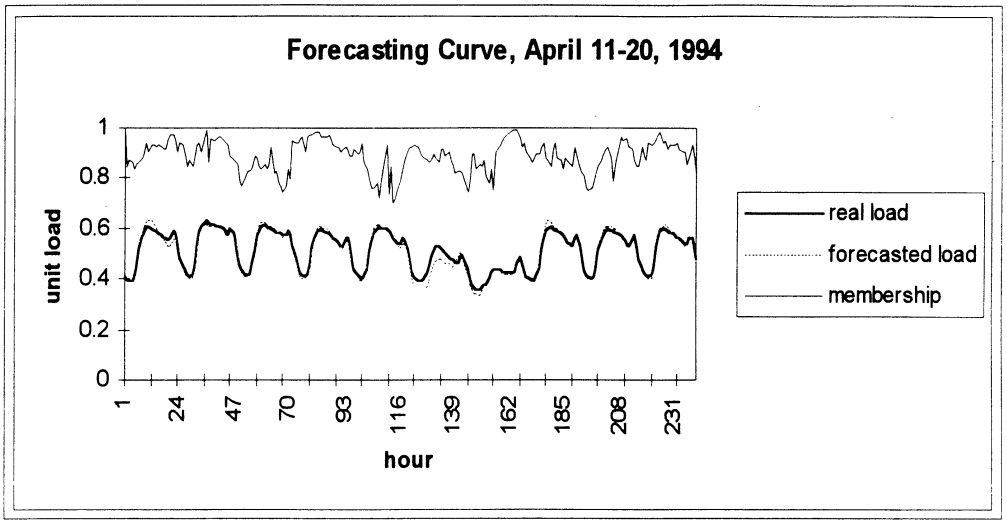

Fig. 6-1 Forecasting Curve for April 11, April 11-17, April 11-20

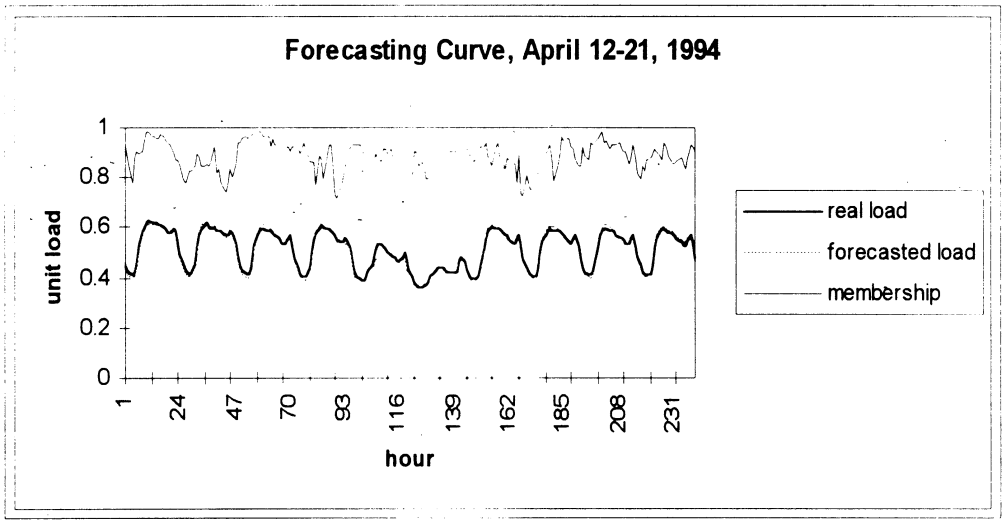

Fig. 6-2 Forecasting Curve for April 12, April 12-18, April 12-21

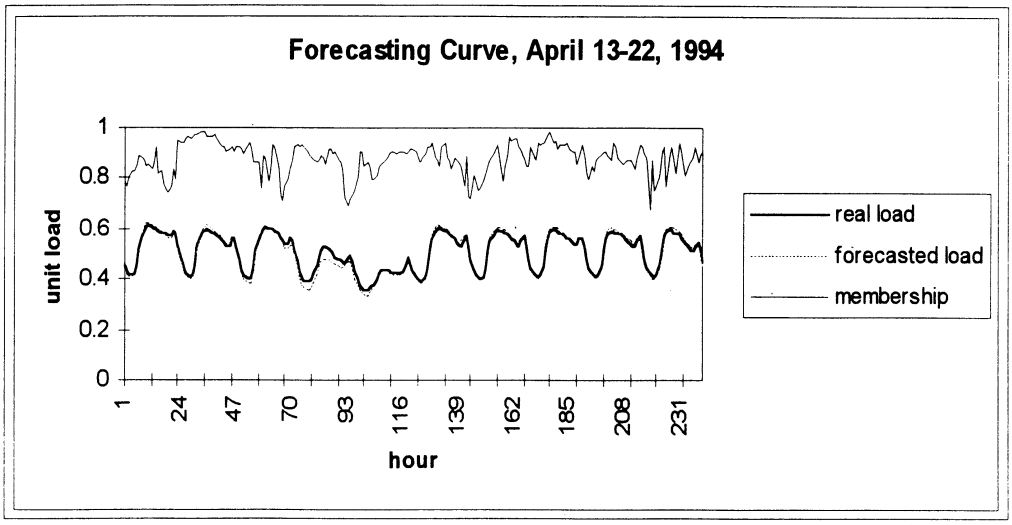

Fig. 6-3 Forecasting Curve for April 13, April 13-19, April 13-22

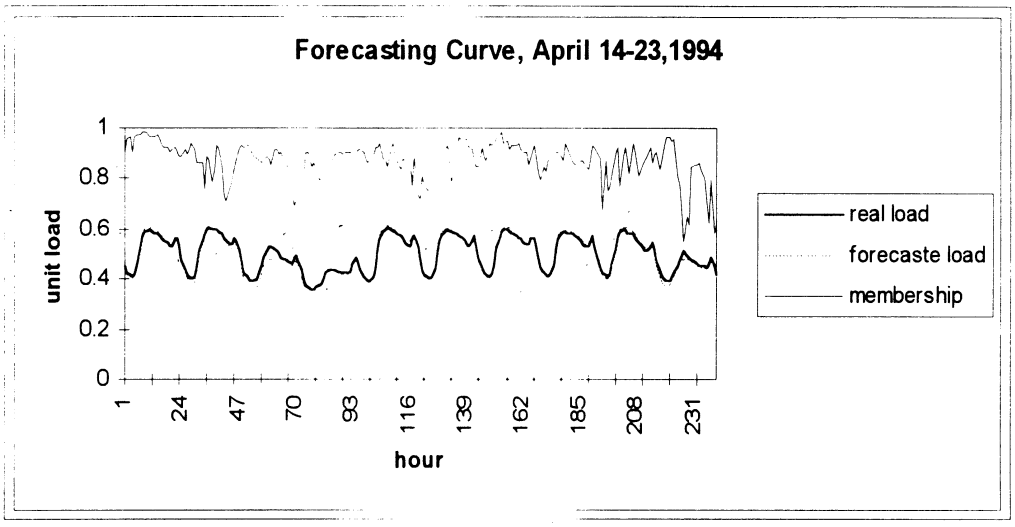

Fig. 6-4 Forecasting Curve for April 14, April 14-20, April 14-23

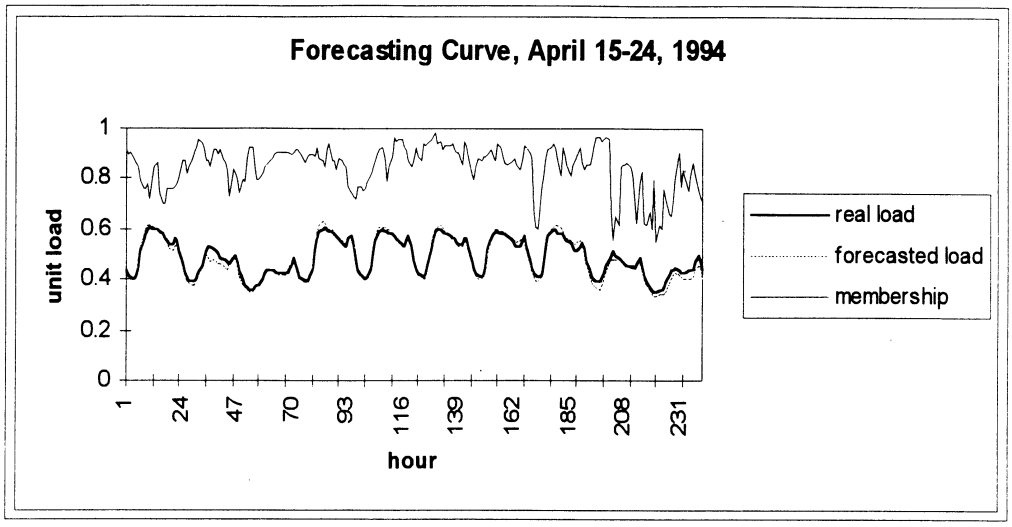

Fig. 6-5 Forecasting Curve for April 15, April 15-21, April 15-24

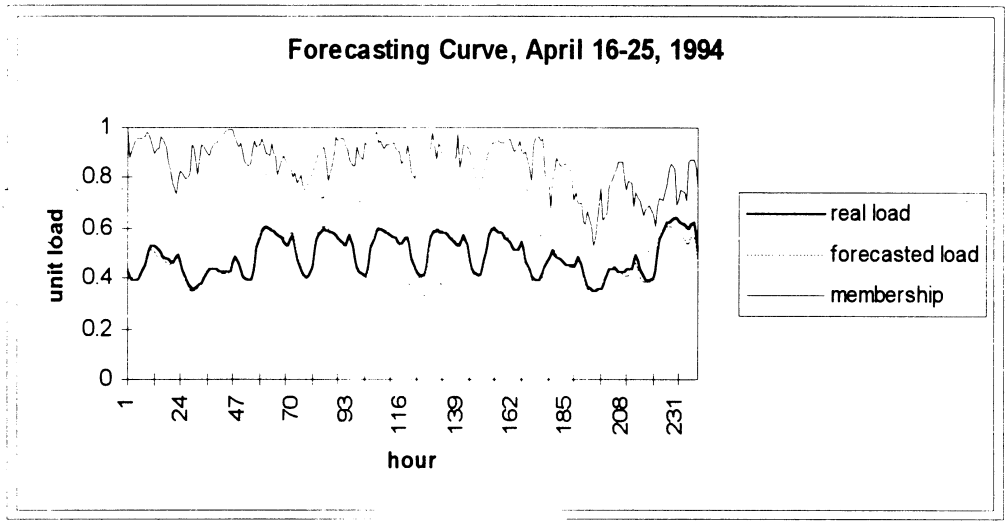

Fig. 6-6 Forecasting Curve for April 16, April 16-22, April 16-25

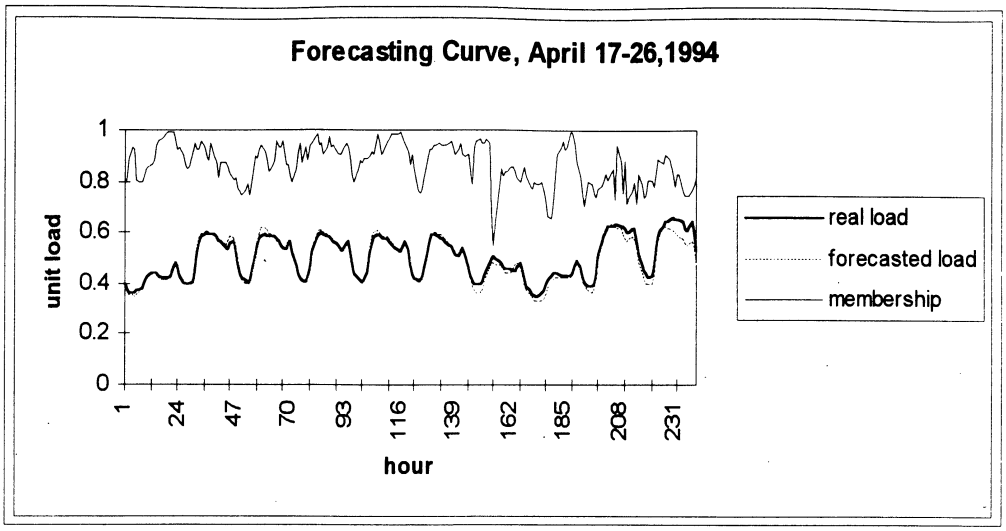

Fig. 6-5 Forecasting Curve for April 15, April 15-21, April 15-24

B. Holiday Forecasting

The simulation forecasting model can also predict loads in holidays with low forecasting error. For example, April 3 is an Easter holiday. The average forecasting error for this day is 1.67%. See Table 4 and Fig. 7.

Table 4 A Holiday Forecasting Example , April 3, 94

hour	real load	fcst load	error(%)
1	0.359	0.3692	-2.83
2	0.346	0.3523	-1.82
3	0.341	0.34	0.28
4	0.338	0.3335	1.34
5	0.336	0.3327	0.99
6	0.346	0.3388	2.08
7	0.351	0.3476	0.98
8	0.358	0.3635	-1.52
9	0.379	0.3826	-0.95
10	0.402	0.4002	0.44
11	0.414	0.4081	1.42
12	0.416	0.4123	0.9
13	0.413	0.4105	0.62
14	0.4	0.4011	-0.28
15	0.386	0.386	0.01
16	0.373	0.3723	0.19
17	0.363	0.361	0.56
18	0.36	0.3559	1.13
19	0.363	0.3542	2.43
20	0.375	0.3706	1.17
21	0.43	0.4031	6.25
22	0.444	0.4282	3.55
23	0.431	0.4175	3.12
24	0.408	0.387	5.15

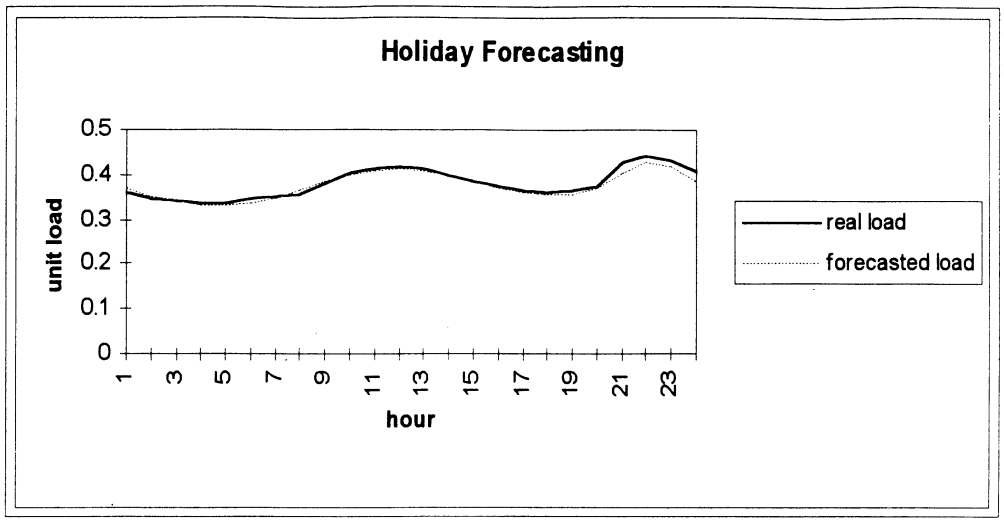

Fig. 7 A Holiday Forecasting Example (Easter Day)

C. Scenario Forecasting

Scenario forecasting allows multiple forecasts for the stochastic operation model [23]. Thanks to the fuzzy neural network, we can produce forecast scenarios easily. Taking the best three forecast scenarios with the largest memberships as the three possible forecast scenarios approximately, we can have a upper scenario, a middle scenario, and a lower scenario as shown in Fig. 8. The detailed results can be found in Table 2. As the memberships of three scenarios are very close, we can have each individual scenario possibility as approximately $\frac{1}{3}$.

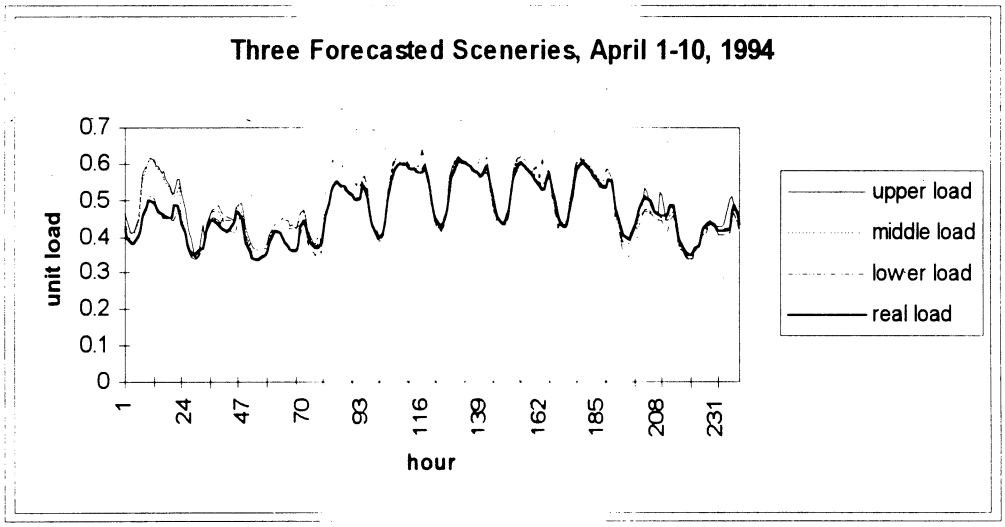

Fig. 8 Three Forecasting Scenarios for April 1-10

VII. CONCLUSIONS

Short-term load forecasting is one of the most difficult problems in power system planning. Though much research has been done in this field, only few find practical applications. Reasons include high forecasting error or difficulties in model construction and maintenance. In this paper, a fuzzy neural network has been applied to solve the problem. With FNN's capacity in simulating nonlinearity and its high flexibility in model maintenance, a new simulation forecasting model of short-term load forecasting has been created. After more investigating in data smoothing, variable selecting and output handling, the simulation forecasting model shows many advantages. They are acceptable forecasting accuracy (especially for peakload), fast training speed (compared to other ANN models), 100% training convergence, and flexibility to add and remove knowledge from the knowledge base that is vital for knowledge maintenance and renewal. An most important characteristic is FNN's capacity to handle large amounts of original data.

ACKNOWLEDGMENTS

The authors thank US National Science Foundation and Electric Power Research Institute for their sponsorship under Grant ECS-9216819. We are very grateful to Mr. Greg Ostrowski and the MEPCC for their kind assistance in providing the original data and valuable suggestions.

REFERENCES (mainly)

[1] M. T. Hagen. S. M. Behr, The Time Serials Approach To Short-Term Load Forecasting, IEEE Trans, PWRS, vol, 1987, pp 785-791

[2] A. D. Papalexopoulos, T. C. Hesterberg, A Regression-Based Approach To Short-Term Load Forecasting, IEEE Trans PWRS, vol 5, 1990, pp 1535-1544

[3] W. R. Christiaanse, Short-Term Load Forecasting Using General Exponential Smoothing, IEEE Trans PWRS, vol 90, 1971

[4] S. Rahman, R. Bhatnagar, An Expert System Algorithm For Short-Term Load Forecasting, IEEE Trans PWRS, vol 3, 1988.

[5] Kun-long Ho, etc, Short-Term Load Forecasting Of Taiwan Power System Using A Knowledge-Based Expert System, IEEE transactions on Power Systems, Vol. 5, No. 4., Nov. 1990.

[6] S. Rahman. O. Hazim, A Generalized Knowledge-Based Short-Term Load Forecasting Technique, IEEE Transactions on Power Systems, Vol. 8, No. 2, May 1993.

[7] C. N. Lu, H. T. Wu, S. Vemuri, Neural Network Short Term Load Forecasting, IEEE Transactions on Power System, Vol.8, No. 1, Feb. 1993

[8] Shin-Tzo Chen David C. Yu, A. R. Moghaddamjo, Weather Sensitive Short-Term Load Forecasting Using Nonfully Connected Artificial Neural Network, IEEE Transactions on Power Systems, Vol. 7, No.3, August 1992.

[9] Li Dayong , Research On Macro-Forecasting Of Electric Power Demand And Macro-Decision Of Power System Development, Ph. D. Dissertation of China's EPRI, July, 1993

[10] O. Mohammed, D. Park, R. Merchant, T. Dinh, C. Tong, A. Azeem and etc. Practical Experiences with An Adaptive Neural Network Short-term Load Forecasting System, IEEE Transactions on Power Systems, Vol. 10, No. 10, February 1995.

[11] Dayong Li, John R. Birge, New Concepts For Natural Human Neural Network And Their Embodiment: A New Fuzzy Neural Network And Study Algorithm, IOE Dept. of The University of Michigan (Ann Arbor) (to be appeared)

[12] Hans J. Zimmermann, Fuzzy Set Theory -- And Its Application, KLUWER-NIJHOFF PUBLISHING, 1985

[13] A. Kaufman, H.M. Gupta, Fuzzy Mathematical Models In Engineering And Management Science.

[14] Kurt J. Schmucker, Fuzzy Sets, Natural Language Computations And Risk Analysis, Computer Science Press. Inc. 1984.

[15] D. E. Rumehart and J. L. McClelland, Parallel Distributed Processing, vol. I. Cambridge, MA: MIT Press , 1988.
[16] Yan_hwang Kuo et al, A Fuzzy Neural Network Model And Its Hardware Implementation, IEEE Trans On Fuzzy Systems, Vol. 2. No. 3, August 1994.
[17] Sushmita Mitra and Sankar K. Pal, Fuzzy Multi-Layer Perception, Inference And Rule Generation, IEEE Trans On Neural Networks, Vol. 6, No. 1, August 1994
[18] S. HOrikawa et al, On Fuzzy Modeling Using Fuzzy Neural Networks With The Back-Propagation Algorithm, IEEE Trans on Neural Networks, vol. 3 ,Sept. 1992.
[19] Hon Keung Kwan, A Fuzzy Neural Network Model And Its Application To Pattern Recognition, IEEE Trans On Fuzzy Systems, Vol. 2, No. 3, August 1994
[20] Valluru B. Rao and Hayagriva V. Rao, C++ Neural Networks and Fuzzy Logic, MIS Press, 1993
[21] Samer Takriti, John R. Birge, Erik Long , Intelligent Unified Control Of Unit Commitment And Generation Allocation, Technical Report 94-26, with Department of Industrial and Operations Engineering, University of Michigan, Ann Arbor, September 1994 (to appear in IEEE Trans. Power Systems).