Short-Term Electric Load Forecasting based on a Neural Fuzzy Network

S.H. Ling[1], F.H.F. Leung *Senior Member, IEEE*, H.K. Lam *Member, IEEE* and P.K.S. Tam *Member, IEEE*

Centre for Multimedia Signal Processing
Department of Electronic and Information Engineering
The Hong Kong Polytechnic University
Hong Kong

Abstract – **Electric load forecasting is essential to improve the reliability of the AC power line data network and provide optimal load scheduling in an intelligent home system. In this paper, a short-term load forecasting realized by a neural fuzzy network (NFN) and a modified genetic algorithm (GA) is proposed. It can forecast the hourly load accurately with respect to different day types and weather information. By introducing new genetic operators, the modified GA performs better than the traditional GA under some benchmark test functions. The optimal network structure can be found by the modified GA when switches in the links of the network are introduced. The membership functions and the number of rules of the NFN can be obtained automatically. Results for a short-term load forecasting will be given.**

Index Terms- **Home networking, load forecasting, genetic algorithm, neural fuzzy network.**

I. INTRODUCTION

Nowadays, homes should have smart features to ensure a high degree of security, entertainment and comfort. To realize these features, reliable channels for the communication among electrical appliances and users should be present. Moreover, with a home network, electrical appliances can be used in an efficient way and the wastage of energy can be reduced. This paper is based on an intelligent home system [15]. In this system, the AC power line network is used not only for supplying electrical power, but also serving as the data communication channel for electrical appliances. Once an electrical

---
[1] *Corresponding author: S.H. Ling, address: CD514, The Hong Kong Polytechnic University, Hong Kong.*
fax number: (852)-23628439, email: ensteve@eie.polyu.edu.hk

appliance is plugged into a power socket, digital data can be transferred through the socket. With this AC power line data network, a short-term load forecasting can be realized. An accurate load forecasting can bring the following benefits to the intelligent home.

1) Increasing the reliability of the AC power line data network - On using the AC power line as the networking medium, we may suffer from the possible low impedance of the power line in the operating bandwidth [16-17] for data transmission. When this occurs, the maximum transmission rate, the reliability and the throughput of the AC power line data network will decrease. The attenuation of the data signal in an AC power line is proportional to the load connected to it. The reliability of the power line data network can be enhanced if the load is kept at an optimal level through forecasting and power backup. We can also adaptively set a suitable data transmission rate based on the forecasted load condition in order to reduce the overhead of data retransmission.

2) Optimal load scheduling - At present, the peak demand of electricity is met by operating costly auxiliary generators, or by purchasing power from other utility companies. The cost for supplying peak power is therefore much higher than that for supplying the average power. A reduction in the peak value of electricity demand can be achieved if we can realize load forecasting, and schedule the demands on the utility company accordingly. This has to be supported by batteries installed in the intelligent home to share the load demand.

Computational intelligence techniques have been applied in daily load forecasting. Neural networks have been considered as a very promising tool to short-term load forecasting [18-25], but its slow convergence time and poor ability of processing linguistic information may cause some problems. In recent year, fuzzy logic has been used to deal with variable linguistic information in load forecasting [26-27]. By processing fuzzy information, reasoning with respect to a linguistic knowledge base can be done. In [18-25],

gradient-descent (GD) algorithm was used to train the neural network parameters. However, the common problems of convergence to local minima and sensitivity to initial values persist. Global search technique such as Genetic Algorithm (GA) may solve these problems. GA is a powerful searching algorithm to handle optimization problems [1-2, 5]. It is particularly useful for complex optimization problems with a large number of tuned parameters. GA has been widely applied in different areas such as fuzzy control [7-9, 13], path planning [10], greenhouse climate control [11], modeling and classification [12] etc.

In this paper, we develop a neural fuzzy system with a modified GA for short-term load forecasting in an intelligent home. New genetic operators are introduced in the modified GA. It will be shown that the modified GA performs better than the traditional GA [1-2, 5] based on some benchmark test functions [3-4, 6, 14]. The modified GA needs only one user-input parameter (population size), instead of three, for its implementation. This makes the modified GA simple and easy to use, especially for those users who do not have too much knowledge on tuning. A neural fuzzy network (NFN) with rule switches is proposed. For a common NFN, the number of possible rules may be too high. This makes the network complex while some rules may be unnecessary. Thus, the rule switches are proposed to facilitate the tuning for the optimal number of rules using the modified GA. This implies that the cost of implementing the proposed NFN can be reduced.

This paper is organized as follows. The modified GA will be introduced in Section II. The performance of the modified GA with respect to some test functions will be discussed in Section III. The proposed NFN is presented in Section IV. A short-term load forecasting realized by the proposed NFN tuned by the modified GA will be presented in Section V. Simulation results will be given. A conclusion will be drawn in Section VI.

II. MODIFIED GENETIC ALGORITHM

The traditional GA process [1-2, 5] is shown in Fig. 1. First, a population of chromosomes is created. Second, the chromosomes are evaluated by a defined fitness function. Third, some of the chromosomes are selected for performing genetic operations. Forth, genetic operations of crossover and mutation are performed. The produced offspring replace their parents in the initial population. This GA process repeats until a user-defined criterion is reached. In this paper, the traditional GA is modified and new genetic operations are introduced to improve the performance. Such a modified GA process is shown in Fig. 2. Its details are given as follows.

A. Initial Population

The initial population is a potential solution set P. The first set of population is usually generated randomly.

$$P = \{\mathbf{p}_1, \mathbf{p}_2, \cdots, \mathbf{p}_{pop\_size}\} \tag{1}$$

$$\mathbf{p}_i = \begin{bmatrix} p_{i_1} & p_{i_2} & \cdots & p_{i_j} & \cdots & p_{i_{no\_vars}} \end{bmatrix}, i = 1, 2, \ldots, pop\_size; j = 1, 2, \ldots, no\_vars \tag{2}$$

$$para_{min}^j \le p_{i_j} \le para_{max}^j \quad i = 1, 2, \ldots, pop\_size; j = 1, 2, \ldots, no\_vars \tag{3}$$

where pop_size denotes the population size; no_vars denotes the number of variables to be tuned; p_{i_j}, $i = 1, 2, \ldots, pop\_size$; $j = 1, 2, \ldots, no\_vars$, are the parameters to be tuned; $para_{min}^j$ and $para_{max}^j$ are the minimum and maximum values of the parameter p_{i_j}. It can be seen from (1) to (3) that the potential solution set P contains some candidate solutions $\mathbf{p}_i$ (chromosomes). The chromosome $\mathbf{p}_i$ contains some variables p_{i_j} (genes).

B. Evaluation

Each chromosome in the population will be evaluated by a defined fitness function. The better chromosomes will return higher values in this process. The fitness function to evaluate a chromosome in the population can be written as,

4

$$fitness = f(\mathbf{p}_i) \tag{4}$$

The form of the fitness function depends on the application.

C. Selection

Two chromosomes in the population will be selected to undergo genetic operations for reproduction. It is believed that the high potential parents will produce better offspring (survival of the best ones). The chromosome having a higher fitness value should therefore have a higher chance to be selected. The selection is done by first assigning a probability q_i to the chromosome $\mathbf{p}_i$:

$$q_i = \frac{f(\mathbf{p}_i)}{\sum\limits_{j=1}^{pop\_size} f(\mathbf{p}_j)} , \, i = 1, 2, \ldots, pop\_size \tag{5}$$

The cumulative probability $\hat{q}_i$ for the chromosome $\mathbf{p}_i$ is defined as,

$$\hat{q}_i = \sum\limits_{j=1}^{i} q_j , \, i = 1, 2, \ldots, pop\_size \tag{6}$$

Based on a randomly generated nonzero floating-point number, $d \in \begin{bmatrix} 0 & 1 \end{bmatrix}$, for each chromosome, the chromosome $\mathbf{p}_i$ is selected if $\hat{q}_{i-1} < d \leq \hat{q}_i$, $i = 1, 2, \ldots, pop\_size$, and $\hat{q}_0 = 0$. Thus, a chromosome having a larger $f(\mathbf{p}_i)$ will have a higher chance to be selected. Consequently, the best chromosomes will get more copies, the average will stay and the worst will die off. In the selection process, two chromosomes will be selected to undergo the genetic operations.

D. Genetic Operations

The genetic operations are to generate some new chromosomes (offspring) from their parents after the selection process. They include the averaging and the mutation operations. The averaging operation is mainly for exchanging information from the two parents obtained in the selection process. The operation is realized by taking the average of the parents. For

5

instance, if the two selected chromosomes are $\mathbf{p}_1$ and $\mathbf{p}_2$, the offspring generated by the averaging process is given by,

$$\mathbf{os} = \begin{bmatrix} os_1 & os_2 & \cdots & os_{no\_vars} \end{bmatrix} = \frac{\mathbf{p}_1 + \mathbf{p}_2}{2} \tag{7}$$

This offspring (7) will then undergo the mutation operation that changes the genes of the chromosomes. Consequently, the features of the chromosomes inherited from their parents can be changed. Three new offspring will be generated by the mutation operation as defined by,

$$\mathbf{nos}_j = \begin{bmatrix} os_1^j & os_2^j & \cdots & os_{no\_vars}^j \end{bmatrix} + \begin{bmatrix} b_1 \Delta nos_1 & b_2 \Delta nos_2 & \cdots & b_{no\_vars} \Delta nos_{no\_vars} \end{bmatrix}, j = 1, 2, 3 \tag{8}$$

where b_i, $i = 1, 2, \ldots, no\_vars$, can only take the value of 0 or 1, Δnos_i, $i = 1, 2, \ldots, no\_vars$, are randomly generated floating numbers such that $para_{min}^i \leq os_i^j + \Delta nos_i \leq para_{max}^i$. The first new offspring ($j = 1$) is obtained according to (8) with that only one b_i (i being randomly generated within the range) is allowed to be 1 and all the others are 0. The second new offspring is obtained according to (8) with that some b_i chosen randomly are set to be 1 and others are zeros. The third new offspring is obtained according to (8) with all $b_i = 1$. These three new offspring will then be evaluated using the fitness function of (4). The one with the largest fitness value f_l will replace the chromosome with the smallest fitness value f_s in the population if $f_l > f_s$.

After the operation of selection, averaging, and mutation, a new population is generated. This new population will repeat the same process. Such an iterative process can be terminated when the result reaches a defined condition, e.g. the change of the fitness values between the current and the previous iteration is less than 0.001, or a defined number of iteration has been reached. For the traditional GA process depicted in Fig. 2, the offspring generated may not be better than their parents. This implies that the searched target is not

necessarily approached monotonically after each iteration. Under the proposed modified GA process, however, if $f_l < f_s$, the previous population is used again in the next genetic cycle. A more efficient search may then be obtained.

III.   BENCHMARK TEST FUNCTIONS

De Jong's Test Functions [3-4, 6, 17] are used as the test functions to examine the applicability and efficiency of the modified GA   A brief description of each function and the problem it represents are given as follows. f_1 is a sphere function, which is probably the most widely used test function. It is smooth, unimodal and symmetric. The performance on this function is a measure of the general efficiency of an algorithm. f_2 is a Rosenbrock function of which the optimum is located in a very narrow ridge. The tip of the ridge is very sharp, and it runs around a parabola. Algorithms that cannot discover good directions will perform poorly in this problem. f_3 is a step function, which is a representative of flat surfaces. Flat surfaces are obstacles for optimisation algorithms because they do not give any information about the search direction. Unless the algorithm has a variable step size, it can be stuck in one of the flat surfaces. f_4 is a quartic function, which is a simple unimodal function padded with noise. The Gaussian noise causes the algorithm never getting the same value at the same point. Algorithms that do not do well in this function will perform poorly on noisy data. f_5 is a foxholes function that has many local minima (25 in this case). Many standard algorithms can be stuck in the first maximum they find.

The test functions are denoted by $f_i(\mathbf{x})$, i = 1, 2, 3, 4, 5, where $\mathbf{x} = \begin{bmatrix} x_1 & x_1 & \cdots & x_{no\_x} \end{bmatrix}$. no_x is an integer denoting the dimension of the vector $\mathbf{x}$.

$$f_1(\mathbf{x}) = \sum_{i=1}^{n} x_i^2 , \ -5.12 \le x_i \le 5.12 \tag{9}$$

where n = 3 and the minimum point is at $f_1(0, 0, 0) = 0$

$$f_2(\mathbf{x}) = \sum_{i=1}^{n-1} \left(100 \times \left(x_{i+1} - x_i^2 \right)^2 + \left(x_i - 1 \right)^2 \right), \quad -2.048 \le x_i \le 2.048 \tag{10}$$

where $n = 2$ and the minimum point is at $f_2(1, 1) = 0$.

$$f_3(\mathbf{x}) = \sum_{i=1}^{n} floor\left((x_i + 0.5)^2 \right), \quad -5.12 \le x_i \le 5.12 \tag{11}$$

where $n = 5$ and the minimum point is at $f_3(0, \dots, 0) = 0$. The value of the floor function, $floor(\cdot)$, is obtained by rounding down the argument to the nearest smaller integer.

$$f_4(\mathbf{x}) = \sum_{i=1}^{n} i \times x_i^4 + Gauss(0,1), \quad -1.28 \le x_i \le 1.28 \tag{12}$$

where $n = 30$ and the minimum point is at $f_4(0, \dots, 0) = 0$. $Gauss(0, 1)$ is obtained by randomly generating a floating-point number between 0 and 1.

$$f_5(\mathbf{x}) = \left[\frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2} (x_i - a_{ij})^6} \right]^{-1}, \quad -65.356 \le x_i \le 65.356 \tag{13}$$

where
$$\mathbf{a} = \{a_{ij}\} = \begin{bmatrix} -32 & -16 & 0 & 16 & 32 & -32 & -16 & 0 & 16 & 32 \\ 32 & 32 & 32 & 32 & 32 & -16 & -16 & -16 & -16 & -16 \\ -32 & -16 & 0 & 16 & 32 & -32 & -16 & 0 & 16 & 32 & -32 & -16 & 0 & 16 & 32 \\ 0 & 0 & 0 & 0 & 0 & 16 & 16 & 16 & 16 & 16 & 32 & 32 & 32 & 32 & 32 \end{bmatrix},$$

$k = 500$ and the minimum point is at $f_5(-32, -32) \approx 1$.

It should be noted that the minimum values of all functions in the defined domain are zero except for $f_5(\mathbf{x})$. The fitness function for f_1 to f_4 is defined as,

$$fitness = \frac{1}{1 + f_i(\mathbf{x})}, \quad i = 1, 2, 3, 4. \tag{14}$$

and the fitness function for f_5 is defined as,

$$fitness = \frac{1}{f_5(\mathbf{x})} \tag{15}$$

The modified GA goes through these 5 test functions. The results are compared with those obtained by the traditional GA [5]. For each test function, the population size is 20.

Each parameter of the traditional GA is encoded into a 40-bit number in the chromosome, and the probabilities of crossover and mutation are 0.25 and 0.03 respectively. The initial values of **x** in the population for a test function are set to be the same. For tests 1 to 5, the initial value are $\begin{bmatrix} 1 & 1 & 1 \end{bmatrix}$, $\begin{bmatrix} 0.5 & 0.5 \end{bmatrix}$, $\begin{bmatrix} 1 & \cdots & 1 \end{bmatrix}$, $\begin{bmatrix} 0.5 & \cdots & 0.5 \end{bmatrix}$ and $\begin{bmatrix} 10 & 10 \end{bmatrix}$ respectively. The results of the average fitness values over 30 times of simulations of the modified and traditional GAs are shown in Fig. 3 and tabulated in Table I. It can be seen from Fig. 3 that the performance of the modified GA is better than that of the traditional GA.

IV. TUNING OF NFN USING THE MODIFIED GA

In this section, tuning of the membership functions and the number of rules of a neural fuzzy network (NFN) using the modified GA will be presented. The optimal number of rules can be found by introducing switches in some links of the NFN.

A. Neural Fuzzy Network with Rule Switches

We use a fuzzy associative memory (FAM) [28] type of rule base for the NFN. An FAM is formed by partitioning the universe of discourse of each fuzzy variable according to the level of fuzzy resolution chosen for the antecedents, thereby generating a grid of FAM elements. The entry at each grid element in the FAM corresponds to a fuzzy premise. An FAM may, then be interpreted as a geometric or tabular representation of a fuzzy logic rule base. For an NFN, the number of possible rules may be too large. This makes the network complex while some rules may be not necessary. The implementation cost is also unnecessarily high. Thus, a multiple-input-single-output NFN (Fig. 4) is proposed which can have an optimal number of rules and membership functions. The main difference between the proposed network and the traditional network is that a unit step function is introduced to some links of the NFN. The unit step functions is defined as,

$$\delta(\varsigma) = \begin{cases} 0 \text{ if } \varsigma \leq 0 \\ 1 \text{ if } \varsigma > 0 \end{cases}, \varsigma \in \Re \tag{16}$$

This is equivalent to adding a switch to each rule in the NFN. Referring to Fig. 4, we define the input and output variables as x_i and y respectively; where $i = 1, 2, \ldots, n$ and n is the number of input variables. The behaviour of the NFN is governed by p fuzzy rules in the following format;

R_g: IF $x_1(t)$ is $A_{1g_1}(x_1(t))$ AND $x_2(t)$ is $A_{2g_2}(x_2(t))$ AND ... AND $x_n(t)$ is $A_{ng_n}(x_n(t))$

   THEN $y(t)$ is w_g, $t = 1, 2, \ldots, u$ \hfill (17)

where u denotes the number of input-output data pairs; $g = 1, 2, \ldots, p$, is the rule number; w_g is the output singleton of rule g. From Fig. 4, it can be seen that

$$p = \prod_{i=1}^{n} m_i \tag{18}$$

where m_i is the number of membership functions of input variable x_i and $g_i \in [1, \ldots, m_i]$, $i = 1, \ldots, n$.

In this network, the membership function is a bell-shaped function as given by,

$$A_{ig_i}(x_i(t)) = e^{\frac{-(x_i(t) - \bar{x}_{ig_i})^2}{2\sigma_{ig_i}^2}} \tag{19}$$

where parameter $\bar{x}_{ig_i}$ and σ_{ig_i} are the mean value and the standard deviation of the membership function respectively. The grade of the membership of each rule is defined as,

$$\mu_g(t) = A_{1g_1}(x_1(t)) \cdot A_{2g_2}(x_2(t)) \cdot \ldots \cdot A_{ng_n}(x_n(t)) \tag{20}$$

The output of the neural fuzzy network $y(t)$ is defined as,

$$y(t) = \frac{\sum_{g=1}^{p} \mu_g(t) w_g \delta(\varsigma_g)}{\sum_{g=1}^{p} \mu_g(t)} \tag{21}$$

where ς_g denotes the rule switch parameter of the g-th rule.

B. Tuning

The proposed NFN can be employed to learn a input-output relationship of an application using the modified GA. The desired input-output relationship is described by,

$$y^d(t) = \mathbf{q}(\mathbf{x}^d(t)), \ t = 1, 2, \ldots, u \tag{22}$$

where $y^d(t)$ is the desired output, $\mathbf{x}^d(t) = \begin{bmatrix} x_1^d(t) & x_2^d(t) & \cdots & x_n^d(t) \end{bmatrix}$ is the desired input vector and $\mathbf{q}(\cdot)$ is an unknown non-linear function. The fitness function is defined as,

$$fitness = \frac{1}{1 + err} \tag{23}$$

where $err = \dfrac{1}{u} \displaystyle\sum_{t=1}^{u} \dfrac{\left| y^d(t) - y(t) \right|}{y^d(t)}$ \hfill (24)

The objective is to minimize the mean absolute percentage error (MAPE) of (24) using the modified GA by setting the chromosome to be [$\bar{x}_{ig_i}$ σ_{ig_i} ς_g] for all *i*, g_i, *g*. The range of *fitness* in (23) is [0, 1]. A larger value of *fitness* indicates a smaller *err*. By using the proposed neural fuzzy network and the modified GA, an optimal neural fuzzy network in terms of the number of rules and the membership functions can be obtained.

V. SHORT-TERM LOAD FORECASTING SYSTEM

It is desired to forecast the load demand in a home with respect to the week's day number and the hour number. The load forecasting system involves 168 multi-input-single-output NFNs, one for a given week's day number and an hour number ($7 \times 24 = 168$). The most important task in the short-term load-forecasting problem is to select the input variables. The forecasting result is affected by two main kinds of information. One is the historical load data and the other is the uncertain information such as the average temperature and rainfall index (weather condition) [21, 29-30].

Historical load data − the hourly load values for yesterday were used as historical load

inputs. These historical hourly loads provides the shape and magnitude reference for the forecasted load. They reflect the habit of the family on power consumption.

Temperature – the average temperature at the previous day and the present day are used as inputs in this forecasting system. The value of the average temperature of the present day is got from the temperature forecast of the weather observatory.

Rainfall index – the average rainfall indexes of the previous day and the present day are used as two inputs in this forecasting system. The range of the rainfall index is from 0 and 1. 0 represents no rain and 1 represents heavy rain.

One of the 168 proposed NFNs for daily load forecasting is shown in Fig. 5. It is a 7-input-1-output network with rule switches. The inputs, z_i, of the proposed NFN are:

$z_1 = L^d(d-1, h-1)$ which represents the load value at the previous hour of the previous day, $z_2 = L^d(d-1, h)$ which represents the load value at the forecasting hour of the previous day, $z_3 = L^d(d-1, h+1)$ which represents the load value at the next hour of the previous day, z_4 = average temperature at the previous day, z_5 = average temperature at the present day, z_6 = average rainfall index at the previous day, z_7 = average rainfall index at the present day. The output $y(t) = L(d, h)$, where $d = 1, 2, \ldots, 7$ is the week's day number (e.g. $d = 1$ for Monday, $d = 7$ for Sunday), $h = 1, 2, \ldots, 24$ is the hour number. One should note the special case that if $d = 1$, $(d-1)$ should be 7. $L(d, h)$ is the forecasted load for day-d, hour-h.

Data of 12 weeks (week 1 to week 12) for learning and data of 2 weeks (week 13 to week 14) for testing are prepared. The number of membership function for each input variables is 2 (i.e. $m_i = 2$, $i = 1, 2, \ldots, 7$) such that the number of rules is $p = 2^7 = 128$. Referring to (21), the proposed NFN used for the load forecasting of a particular hour is governed by,

$$y(t) = \frac{\sum\limits_{g=1}^{128} \mu_g(t) w_g \delta(\varsigma_g)}{\sum\limits_{g=1}^{128} \mu_g(t)} \qquad (25)$$

The fitness function for training is defined as follows,

$$fitness = \frac{1}{1 + err} \qquad (26)$$

$$err = \frac{1}{12} \sum_{t=1}^{12} \frac{\left| y^d(t) - y(t) \right|}{y^d(t)} \qquad (27)$$

(25) is one of the 168 NFNs in the proposed load forecaster.

The modified GA is employed to tune the parameters and structure of the NFNs. The population size is 10. The bounds of parameters are set at $0 \le \bar{x}_{ig_i} \le 1$, $0 \le \sigma_{ig_i} \le 0.4$ and $-1 \le \varsigma_g \le 1$. The chromosomes used for the modified GA are [$\bar{x}_{ig_i}$ σ_{ig_i} ς_g], $i = 1, \ldots, 7$; $g_i = 1, 2$; $g = 1, \ldots, 128$. Initial values of $\bar{x}_{ig_i}, \sigma_{ig_i}, \varsigma_g$ of 0.5, 0.2 and 1 respectively are used. The number of the iterations to train the NFN is 2000. For comparison, another proposed NFN trained by the traditional GA, and a 7-inputs-1-output NFN without rule switches trained by the modified GA and traditional GA, are also applied for the load forecasting. The common network parameters are kept unchanged. In addition, a bit length of 9 is used for each parameter coding. The probabilities of crossover and mutation for the traditional GA are 0.65 and 0.05 respectively.

The load forecasting results are tabulated in Table II to Table V. Table II shows the load forecasting results for Wednesday using the proposed NFNs trained by the modified GA and traditional GA respectively. Table III shows the load forecasting results for Wednesday using traditional NFNs without rule switches trained by the modified GA and traditional GA respectively. Table IV shows the load forecasting results for Sunday using the proposed NFNs trained by the modified GA and traditional GA respectively. Table V shows the load forecasting results for Sunday using traditional NFNs without rule switches

trained by the modified GA and traditional GA respectively. From these four tables, we observe that the proposed NFN provides better results than the traditional NFN in term of the fitness value and number of rules. Besides, the proposed GA also produces better results than the traditional GA. The average numbers of rules of the proposed NFNs trained by the modified GA for load forecasting on Wednesday and Sunday are 67.3 and 69.6 respectively. These imply a 47.4% and 45.64% reduction of the number of rules after learning.

Table VI to Table X show the average training error (MAPE) based on data of week 1 to week 12 and the average forecasting error (MAPE) based on data of week 13 to week 14 for Wednesday and Sunday respectively. From these tables, we can see that the proposed NFN trained by the modified GA gives the best results. Fig. 6 and Fig. 7 show the forecasted daily load curve on Wednesday and Sunday of Week 13 respectively. We can conclude that the proposed NFN offers a satisfactory performance in load forecasting.

VI. CONCLUSION

In this paper, a modified GA with new genetic operations has been proposed. Based on the benchmark De Jong's test functions, it has been shown that the modified GA performs better than the traditional GA. A neural fuzzy network has been proposed in which a switch is introduced in each fuzzy rule. Thus, the number of rules can be optimized by applying the modified GA. The cost of implementing the NFN can be reduced. A short-term load forecasting in an intelligent home has been realized using the proposed network. The optimal number of rules and the network parameters are tuned by the modified genetic algorithm. The performance of the proposed network is satisfactory as the average errors are lower than 2%.

REFERENCES

[1] J.H. Holland, *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI: University of Michigan Press, 1975.

[2] D.T. Pham and D. Karaboga, *Intelligent Optimization Techniques, Genetic Algorithms, Tabu Search, Simulated Annealing and Neural Networks*. Springer, 2000.

[3] Y. Hanaki, T. Hashiyama, and S. Okuma, "Accelerated evolutionary computation using fitness estimation," in *Proc. IEEE Int. Conf. Systems, Man, and Cybernetics*, vol. 1, 1999, pp. 643-648.

[4] K.A. De Jong, *Ph.D. Thesis*: *An Analysis of the Behavior of a Class of Genetic Adaptive Systems*. Ann Arbor, MI: University of Michigan, 1975.

[5] Z. Michalewicz, *Genetic Algorithm + Data Structures = Evolution Programs*, 2nd ed. Springer-Verlag, 1994.

[6] G.X. Yao and Y. Liu "Evolutionary programming made faster," *IEEE Trans. Evolutionary Computation*, vol. 3, no. 2, pp.82-102, July 1999.

[7] B.D. Liu, C.Y. Chen, and J.Y. Tsao, "Design of adaptive fuzzy logic controller based on linguistic-hedge concepts and genetic algorithms," *IEEE Trans. Systems, Man and Cybernetics, Part B,* vol. 31 no. 1, pp. 32-53, Feb. 2001.

[8] Y.S Zhou and L.Y, Lai "Optimal design for fuzzy controllers by genetic algorithms," *IEEE Trans. Industry Applications*, vol. 36, no. 1, pp. 93-97, Jan.-Feb. 2000.

[9] C.F. Juang, J.Y. Lin, and C.T. Lin, "Genetic reinforcement learning through symbiotic evolution for fuzzy controller design," *IEEE Trans. Systems, Man and Cybernetics, Part B*, vol. 30, no. 2, pp. 290–302, April 2000.

[10] H. Juidette and H. Youlal, "Fuzzy dynamic path planning using genetic algorithms," *Electronics Letters*, vol. 36, no. 4, pp. 374-376, Feb. 2000.

[11] R. Caponetto, L. Fortuna, G. Nunnari, L. Occhipinti, and M. G. Xibilia, "Soft computing for greenhouse climate control," *IEEE Trans. Fuzzy Systems*, vol. 8, no. 6, pp. 753-760, Dec. 2000.

[12] M. Setnes and H. Roubos, "GA-fuzzy modeling and classification: complexity and performance," *IEEE. Trans. Fuzzy Systems*, vol. 8, no. 5, pp. 509–522, Oct. 2000.

[13] K. Belarbi and F. Titel, "Genetic algorithm for the design of a class of fuzzy controllers: an alternative approach," *IEEE Trans. Fuzzy Systems*, vol. 8, no. 4, pp. 398-405, Aug. 2000.

[14] S. Amin and J.L. Fernandez-Villacanas, "Dynamic local search," in *Proc. 2nd Int. Conf. Genetic Algorithms in Engineering Systems: Innovations and Applications*, 1997, pp. 129-132.

[15] L.K. Wong, S.H. Ling, F.H.F. Leung, Y.S. Lee, S.H. Wong, T.H. Lee, H.K. Lam, K.H Ho, D.P.K. Lun, and T.C. Hsung, "An intelligent home," in *Proc. Workshop on Service Automation and Robotics, Hong Kong*, June 2000, pp. 111-119.

[16] G. Schickhuber and O. McCarthy, "Control using power lines: a European view," *Computing and Control Engineering Journal*, pp.180-184, Aug. 1997.

[17] D. Liu, E. Flint, B. Gaucher, and Y. Kwark, "Wide band AC power line characterization," *IEEE Trans. Consumer Electronics,* vol. 45, no. 4, pp. 1087-1097, Nov. 1999.

[18] K.Y. Lee, Y.T. Cha, and J.H. Park, "Short-term load forecasting using an artificial neural network," *IEEE Trans. Power Systems*, vol. 7, no. 1, pp.124-132, 1992.

[19] Y.Y. Hsu and C.C. Yang, "Design of artificial neural networks for short-term load forecasting. Part 1: Self-organizing feature maps for day type identification," *IEE Proceedings-C*, vol. 138, no. 5, pp. 407-418, 1991.

[20] I. Drezga and D.S. Rahman, "Short-term load forecasting with local ANN predictors,"

IEEE Trans. Power System, vol. 14, no. 3, pp. 844-850, Aug. 1999.

[21] J.A. Momoh, Y. Wang, and M. Elfayoumy, "Artifical neutal network based load forecasting," in *Proc. IEEE Int. Conf. Systems, Man, and Cybernetics 1997,* vol. 4, 1997, pp. 3443-3451.

[22] D. Part, M. El-Sharkawi, R. Marks, L. Atlas, and M. Damborg, "Electric load forecasting using an artificial neural network," *IEEE Trans. Power System*, vol. 6, no. 2, pp.442-449, 1991.

[23] C.N. Lu, H.T. Wu, and S. Vemuri, "Neural network based short-term load forecasting," *IEEE Trans. Power Systems*, vol. 8, no. 1, pp.336-342, Feb.1993.

[24] A.P. Rewagad and V.L. Soanawane, "Artificial neural network based short term load forecasting," in *Proc. 1998 IEEE Region 10 Int. Conf. Global Connectivity in Energy, Computer, Communication and Control,* vol. 2, 1998, pp.588-595.

[25] A.G. Bakirtzls, V. Petridls, S.J. Klartzis, M.C. Alexladls, and A.H. Malssls, "A neural network short term load forecasting model for Greed power system," *IEEE Trans. Power Systems*, vol.11, no.2, pp.858-863, May1996.

[26] Y.Y. Hse and K.L. Ho, "Fuzzy expect systems: an application to short-term load forecasting," *IEE Proceedings-C*, vol. 139, no.6, pp. 471-477, Nov 1992.

[27] H.C. Wu and C.N. Lu, "Automatic fuzzy model identification for short-term load forecast," *IEE Proceedings-Gener. Transm. Distrib.*, vol. 146, no.5, pp.477-482, Sept. 1999.

[28] B. Kosko, *Neural Networks and Fuzzy System: A Dynamical Systems Approach to Machine Intelligence.* Prentice Hall, 1991.

[29] K.H. Kim, J.L. Park, K.J. Hwang, and S.H. Kim," Implementation of hybrid short-term load forecasting system using artificial neural networks and fuzzy expert systems," *IEEE Trans. Power Systems*, vol. 10, no. 3, pp.1534-1539, Aug. 1995.

[30] D. Srinivasan, C.S. Chang, and S.S. Tan," One-day ahead electric load forecasting with hybrid fuzzy-neural networks," in *Proc. Biennial Conf. North American Fuzzy Information Processing Society (NAFIPS),* Berkeley, CA, USA, 19-22 Jun 1996, pp.160 −163.
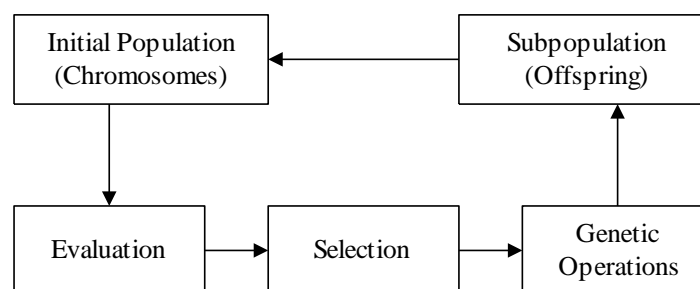
Fig. 1. Traditional GA.
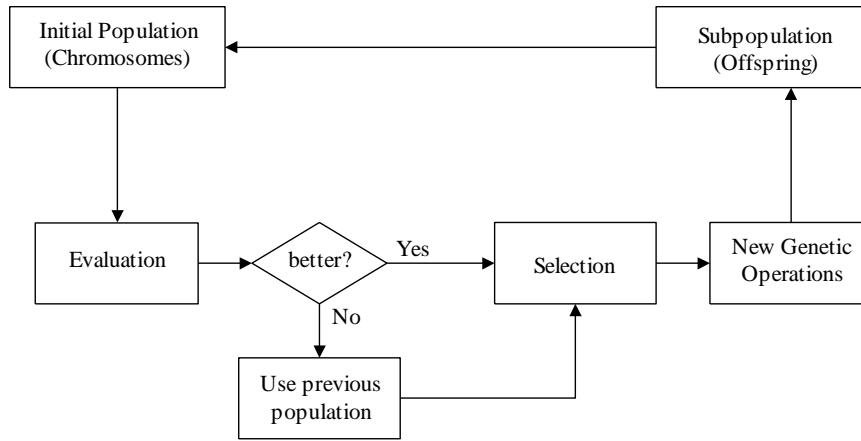
Fig. 2. Modified GA.
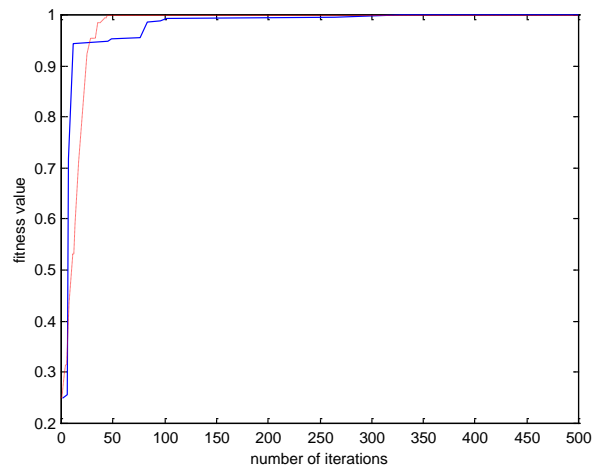
(a). The average fitness value of the test function $f_1(\mathbf{x})$ obtained by the modified (solid line) and traditional (dotted line) GAs.
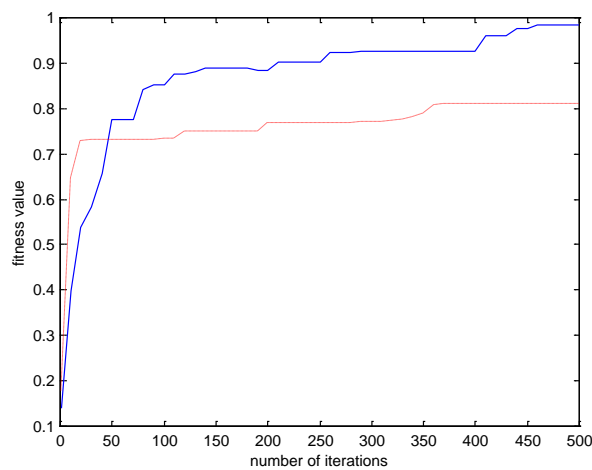
(b). The average fitness value of the test function $f_2(\mathbf{x})$ obtained by the modified (solid line) and traditional (dotted line) GAs.

17
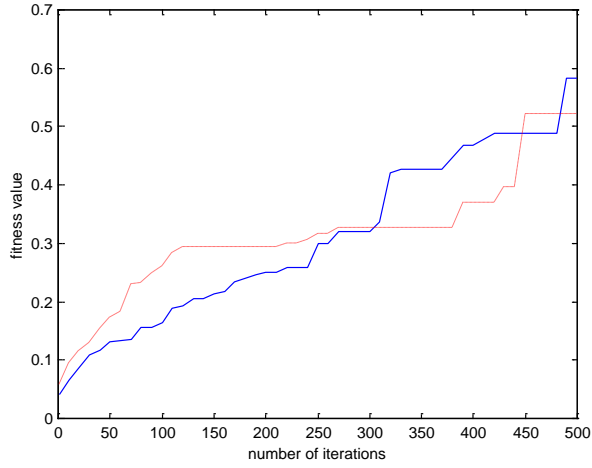
(c). The average fitness value of the test function $f_3(\mathbf{x})$ obtained by the modified (solid line) and traditional (dotted line) GAs.
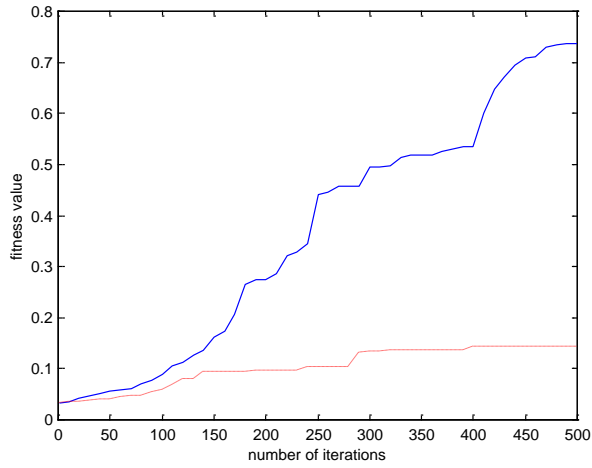
(d). The average fitness value of the test function $f_4(\mathbf{x})$ obtained by the modified (solid line) and traditional (dotted line) GAs.
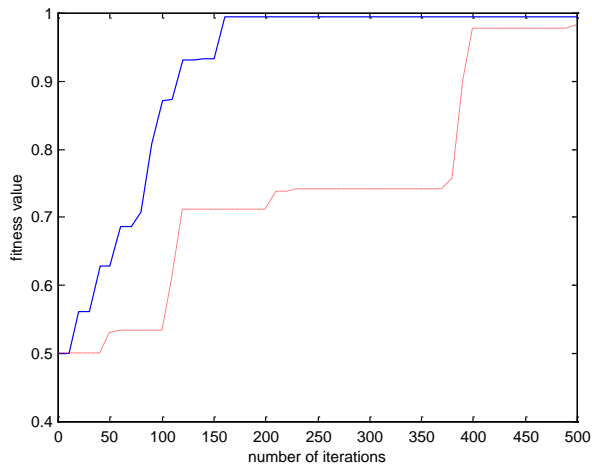
(e). The average fitness value of the test function $f_5(\mathbf{x})$ obtained by the modified (solid line) and traditional (dotted line) GAs.

Fig. 3. Simulation results of the modified and traditional GAs.
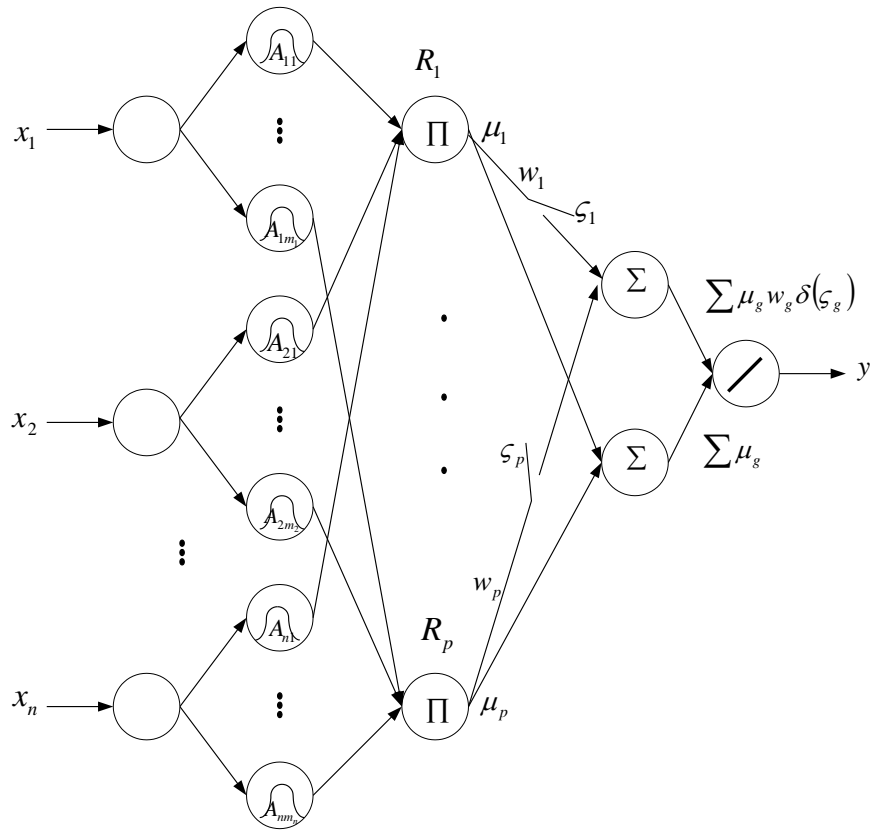
Fig. 4. Proposed neural fuzzy network.
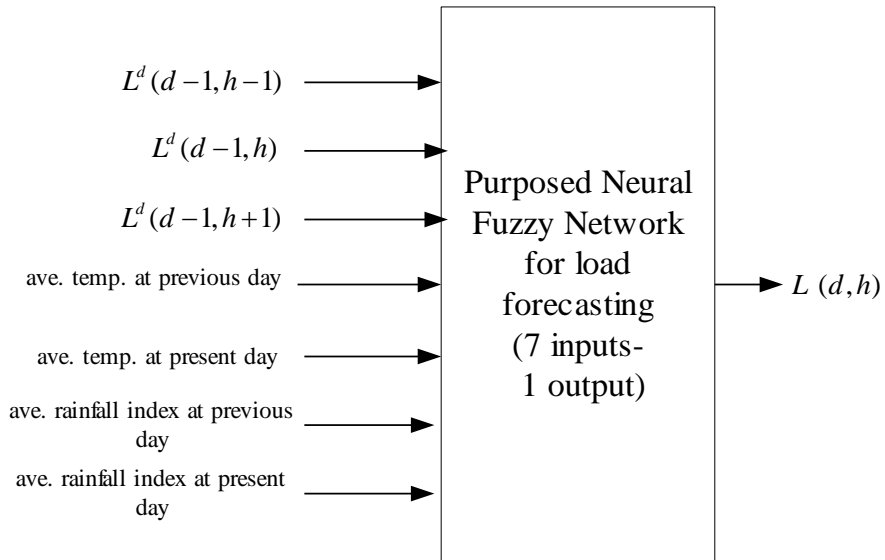
Fig. 5. Proposed neural fuzzy network for load forecasting
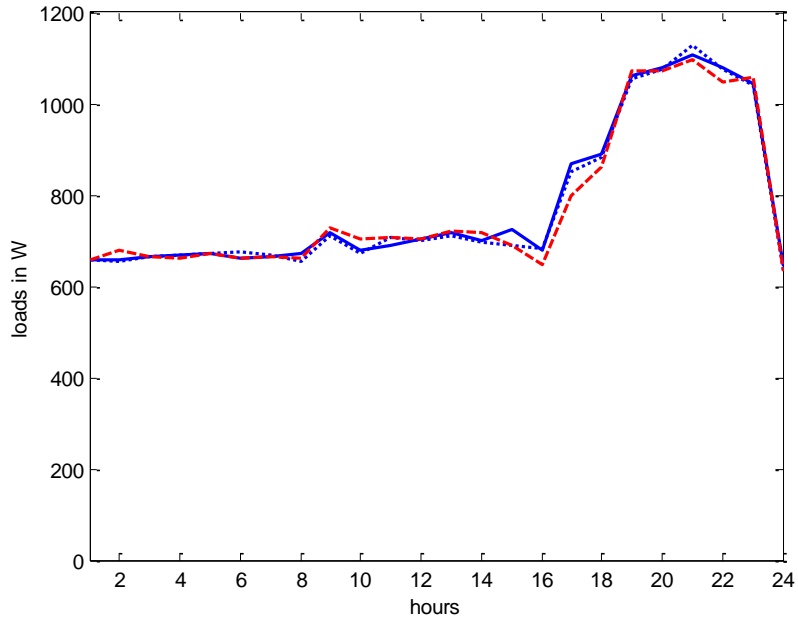
Fig.6. Actual load (solid line) and forecast results for Wednesday (Week13) from the proposed forecasting system (dashed line) and the traditional forecasting system.
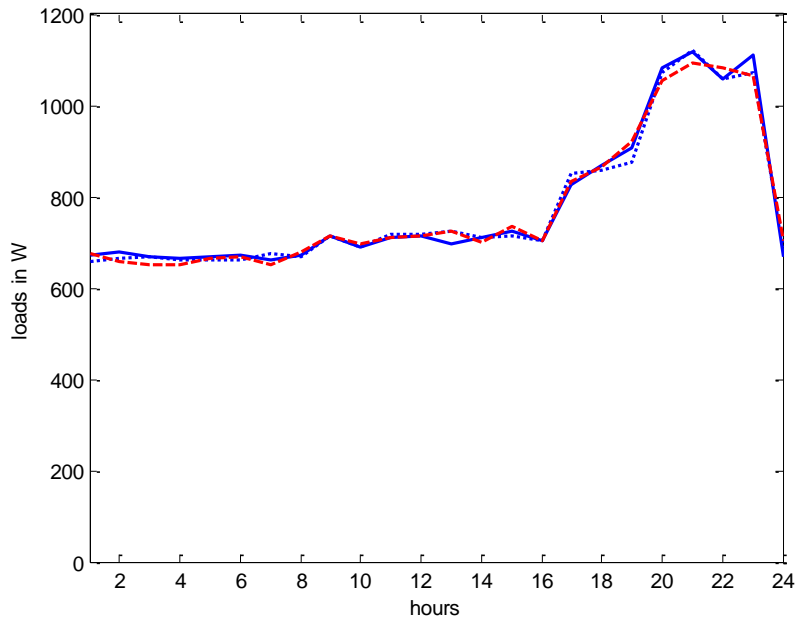
Fig.7. Actual load (solid line) and forecast results for Sunday (Week13) from the proposed forecasting system (dashed line) and the traditional forecasting system.

	Modified GA	Traditional GA
Test Functions	Fitness Value	Fitness Value
$f_1(\mathbf{x})$	0.999955	0.999382
$f_2(\mathbf{x})$	0.984039	0.810813
$f_3(\mathbf{x})$	0.583333	0.520833
$f_4(\mathbf{x})$	0.737526	0.14211
$f_5(\mathbf{x})$	0.995509	0.982912

Table I. Simulation results of the modified and the traditional GAs based on the De Jong's test functions.

	Trained with modified GA		Trained with traditional GA	
Hour	Fitness value	No. of rules	Fitness value	No. of rules
1	0.988313	66	0.987183	61
2	0.991696	68	0.990321	67
3	0.994460	63	0.992122	58
4	0.988092	67	0.987987	69
5	0.991691	74	0.990212	71
6	0.991463	67	0.989102	63
7	0.992417	75	0.990124	70
8	0.981700	71	0.977821	68
9	0.986421	57	0.982099	62
10	0.983249	74	0.978106	73
11	0.988425	66	0.984239	65
12	0.987979	63	0.982205	63
13	0.982555	69	0.978265	70
14	0.984378	70	0.980639	68
15	0.984158	74	0.980243	77
16	0.981027	63	0.975093	69
17	0.983318	67	0.979036	65
18	0.987679	70	0.985643	66
19	0.979401	57	0.977232	62
20	0.982569	66	0.979023	65
21	0.984503	68	0.982637	63
22	0.988063	70	0.985302	64
23	0.978526	67	0.974009	70
24	0.980531	64	0.977875	68
Average:	**0.985942**	**67.3**	**0.982772**	**66.5**

Table II. Load forecasting results for Wednesday using the proposed NFN with modified and traditional GAs after learning.

	Trained with modified GA		Trained with traditional GA	
Hour	Fitness value	No. of rules	Fitness value	No. of rules
1	0.985742	128	0.979566	128
2	0.989217	128	0.987982	128
3	0.982123	128	0.981278	128
4	0.977821	128	0.973886	128
5	0.984974	128	0.982847	128
6	0.984820	128	0.982845	128
7	0.980012	128	0.978329	128
8	0.984933	128	0.989166	128
9	0.977932	128	0.977387	128
10	0.980231	128	0.979038	128
11	0.988219	128	0.986572	128
12	0.979236	128	0.977390	128
13	0.975237	128	0.972502	128
14	0.982367	128	0.984302	128
15	0.974743	128	0.976310	128
16	0.973432	128	0.970608	128
17	0.980023	128	0.976288	128
18	0.981623	128	0.980906	128
19	0.977834	128	0.975917	128
20	0.980323	128	0.977832	128
21	0.983234	128	0.986197	128
22	0.988346	128	0.988051	128
23	0.984437	128	0.986947	128
24	0.977732	128	0.977180	128
Average:	**0.981441**	**128**	**0.980389**	**128**

Table III. Load forecasting results for Wednesday using the traditional NFN with modified and traditional GAs after learning.

	Trained with modified GA		Trained with traditional GA	
Hour	Fitness value	No. of rules	Fitness value	No. of rules
1	0.993858	64	0.992123	66
2	0.989306	64	0.988364	63
3	0.987975	80	0.983323	77
4	0.994966	80	0.992310	79
5	0.991833	74	0.987832	79
6	0.989173	67	0.985623	65
7	0.989657	66	0.987345	69
8	0.981039	76	0.977438	70
9	0.987334	75	0.985434	76
10	0.980280	70	0.977435	74
11	0.982896	66	0.983484	68
12	0.987385	60	0.985435	65
13	0.978656	80	0.976546	75
14	0.976452	68	0.974504	74
15	0.983945	75	0.985645	79
16	0.978974	69	0.979450	64
17	0.975966	74	0.974771	69
18	0.982075	56	0.983054	60
19	0.977009	68	0.976859	64
20	0.983535	62	0.984095	67
21	0.989151	70	0.986780	65
22	0.986421	64	0.982067	68
23	0.978060	68	0.978103	63
24	0.985058	74	0.983088	75
Average:	**0.984625**	**69.6**	**0.982963**	**69.7**

Table IV. Load forecasting results for Sunday using the proposed NFN with modified and traditional GAs after learning.

	Trained with modified GA		Trained with traditional GA	
Hour	Fitness value	No. of rules	Fitness value	No. of rules
1	0.991876	128	0.991465	128
2	0.989032	128	0.990133	128
3	0.982231	128	0.979135	128
4	0.985364	128	0.981797	128
5	0.989093	128	0.988750	128
6	0.985749	128	0.983563	128
7	0.988763	128	0.988964	128
8	0.979126	128	0.978053	128
9	0.987328	128	0.987016	128
10	0.976432	128	0.975352	128
11	0.982983	128	0.982571	128
12	0.980234	128	0.975890	128
13	0.972349	128	0.971394	128
14	0.975095	128	0.974831	128
15	0.981370	128	0.978184	128
16	0.976061	128	0.972019	128
17	0.973525	128	0.971706	128
18	0.970192	128	0.961152	128
19	0.975578	128	0.974105	128
20	0.977232	128	0.976875	128
21	0.981096	128	0.978810	128
22	0.984092	128	0.981486	128
23	0.984536	128	0.986915	128
24	0.982311	128	0.981015	128
Average:	**0.981319**	**128**	**0.979633**	**128**

Table V. Load forecasting results for Sunday using the traditional NFN with modified and traditional GAs after learning.

	Trained with modified GA		Trained with traditional GA	
Hour	Ave. training error (Week 1-12)	Ave. forecasting error (Week13-14)	Ave. training error (Week 1-12)	Ave. forecasting error (Week13-14)
1	1.1826	0.7216	1.2983	0.8802
2	0.8374	0.3664	0.9774	0.4636
3	0.5564	0.0665	0.7941	0.0996
4	1.2072	0.6502	1.2159	0.8007
5	0.8379	1.5986	0.9885	1.9593
6	0.8611	1.2652	1.1018	1.5427
7	0.7641	0.5917	0.9975	0.7219
8	1.8641	1.7350	2.2682	2.1156
9	1.3765	1.6626	1.8227	2.0226
10	1.7037	1.7857	2.2384	2.1871
11	1.1710	2.8853	1.6013	3.5075
12	1.2167	0.3422	1.8117	0.4246
13	1.7755	1.6810	2.2218	2.0492
14	1.5870	0.5044	1.9743	0.6243
15	1.6097	1.8949	2.0155	2.3165
16	1.9340	0.7740	2.5543	0.9512
17	1.6965	2.5373	2.1413	3.0840
18	1.2475	2.0180	1.4566	2.4696
19	2.1033	1.8034	2.3298	2.2057
20	1.7740	0.8074	2.1426	0.9876
21	1.5741	1.6664	1.7670	2.0261
22	1.2082	1.2375	1.4917	1.5115
23	2.1945	1.1769	2.6685	1.4408
24	1.9856	2.7392	2.2626	3.3325
Average:	**1.4279**	**1.3546**	**1.7559**	**1.6552**

Table VI. Training error and forecasting error (in MAPE) for Wednesday under the proposed NFN trained by the modified and traditional GAs.

	Trained with modified GA		Trained with traditional GA	
Hour	Ave. training error (Week 1-12)	Ave. forecasting error (Week13-14)	Ave. training error (Week 1-12)	Ave. forecasting error (Week13-14)
1	1.4464	1.3342	2.0860	1.4611
2	1.0901	2.6273	1.2164	2.9000
3	1.8202	1.2012	1.9079	1.3257
4	2.2682	0.7619	2.6814	0.8206
5	1.5255	1.6326	1.7452	1.8025
6	1.5414	1.0802	1.7454	1.1866
7	2.0396	0.7136	2.2151	0.7691
8	1.5297	0.9747	1.0953	1.0584
9	2.2566	1.5496	2.3136	1.6954
10	2.0168	1.8352	2.1411	2.0148
11	1.1921	4.2288	1.3611	4.6699
12	2.1204	0.8581	2.3132	0.9426
13	2.5392	0.5808	2.8275	0.6353
14	1.7950	2.7824	1.5948	3.0533
15	2.5911	3.2574	2.4265	3.5760
16	2.7293	4.6023	3.0282	5.0777
17	2.0384	2.4358	2.4288	2.6843
18	1.8721	4.6344	1.9466	5.1140
19	2.2668	2.2953	2.4677	2.5192
20	2.0072	1.0399	2.2671	1.1266
21	1.7052	0.9028	1.3996	0.9833
22	1.1791	3.5312	1.2094	3.8948
23	1.5809	0.8890	1.3226	0.9808
24	2.2775	2.6093	2.2828	2.8691
Average:	**1.8910**	**2.0149**	**2.0010**	**2.2150**

Table VII. Training error and forecasting error (in MAPE) for Wednesday under the traditional NFN trained by the modified and traditional GAs.

	Trained with modified GA		Trained with traditional GA	
Hour	Ave. training error (Week 1-12)	Ave. forecasting error (Week13-14)	Ave. training error (Week 1-12)	Ave. forecasting error (Week13-14)
1	0.6180	1.5162	0.7940	1.7460
2	1.1081	1.7387	1.1773	1.9986
3	1.2171	2.3829	1.6960	2.7326
4	0.5060	0.5548	0.7750	0.6384
5	0.8235	0.7548	1.2318	0.8785
6	1.0945	2.0746	1.4587	2.3817
7	1.0451	2.1525	1.2817	2.4751
8	1.9327	1.4970	2.3083	1.7193
9	1.2828	0.2303	1.4781	0.2696
10	2.0117	0.0847	2.3086	0.1037
11	1.7402	1.1305	1.6793	1.3086
12	1.2776	0.4350	1.4780	0.5023
13	2.1809	1.9984	2.4017	2.2984
14	2.4116	1.3677	2.6163	1.5667
15	1.6317	2.3624	1.4564	2.7121
16	2.1478	0.5613	2.0981	0.6569
17	2.4626	2.6652	2.5882	3.0579
18	1.8252	1.1180	1.7238	1.2810
19	2.3532	2.1587	2.3689	2.4838
20	1.6741	0.6842	1.6162	0.7937
21	1.0969	1.2435	1.3397	1.4267
22	1.3766	2.1614	1.8260	2.4830
23	2.0194	1.7320	2.2387	1.9944
24	1.5169	3.4776	1.7203	3.9980
Average:	**1.5564**	**1.5034**	**1.7332**	**1.7294**

Table VIII. Training error and forecasting error (in MAPE) for Sunday under the proposed NFN trained by the modified and traditional GAs.

	Trained with modified GA		Trained with traditional GA	
Hour	Ave. training error (Week 1-12)	Ave. forecasting error (Week13-14)	Ave. training error (Week 1-12)	Ave. forecasting error (Week13-14)
1	0.8191	0.2025	0.8606	0.2011
2	1.1090	2.1124	0.9965	2.2285
3	1.8090	5.6686	2.1310	5.9949
4	1.4853	1.3020	1.8541	1.3666
5	1.1027	0.7114	1.1378	0.7444
6	1.4457	4.9620	1.6711	5.2338
7	1.1365	2.2162	1.1159	2.3351
8	2.1319	0.3739	2.2439	0.3854
9	1.2835	0.7060	1.3155	0.7347
10	2.4137	1.8500	2.5271	1.9451
11	1.7312	0.6123	1.7738	0.6324
12	2.0165	0.1286	2.4705	0.1246
13	2.8437	2.9658	2.9448	3.1241
14	2.5541	1.3580	2.5819	1.4254
15	1.8984	3.0919	2.2302	3.2611
16	2.4526	0.4710	2.8787	0.4975
17	2.7195	3.2347	2.9118	3.4216
18	3.0724	1.4232	4.0418	1.4938
19	2.5033	2.5159	2.6583	2.6583
20	2.3298	2.6656	2.3672	2.8132
21	1.9268	1.6891	2.1649	1.7761
22	1.6165	2.1979	1.8863	2.3166
23	1.5707	2.1157	1.3258	2.2288
24	1.8008	3.3468	1.9352	3.5402
Average:	**1.9037**	**1.9967**	**2.0844**	**2.1035**

Table X. Training error and forecasting error (in MAPE) for Sunday under the traditional NFN trained by the modified and traditional GAs.