Gain Estimation for an AC Power Line Data Network Transmitter Using a Neural-Fuzzy Network and an Improved Genetic Algorithm[1]

H.K. Lam[2], S.H. Ling, F.H.F. Leung, P.K.S. Tam, Y.S. Lee
Centre for Multimedia Signal Processing, Department of Electronic and Information Engineering, The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong

Abstract: This paper presents the estimation of the transmission gain for an AC power line data network in an intelligent home. The estimated gain ensures the transmission reliability and efficiency. A neural-fuzzy network with rule switches is proposed to perform the estimation. An improved genetic algorithm is proposed to tune the parameters and the rules of the proposed neural-fuzzy network. By turning on or off the rule switches, an optimal rule base can be obtained. An application example will be given.

I. Introduction

Nowadays, homes [1] should offer smart features to ensure higher security, more entertainment and comfort to residents. A reliable communication channel among electrical appliances and users has to be present to support these features. The energy can also be used in a more efficient manner. At present, many researchers and companies are developing intelligent home systems. A phone-based remote controller facilitates home users to issue control commands to their home appliances through telephones [2]. A small two-arm mobile robot in a home can be controlled via an ISDN link [3]. In the U.S., X-10 systems are commonly used to support low-cost and slow-rate AC power line data networks.

Without relying on the manufacturers of electrical appliances and installing a LAN, one simple way to realize the communication channel [4] for home appliances and users is to make use of the AC power line. We have successfully realized a power line data network [6] based on spread-spectrum technology [5], which supports communications at 10 Kbps in the noisy and signal-distorting environment of AC power lines. This network serves as a backbone for an intelligent home system through which electrical appliances can be controlled via line/mobile phones, PDAs, keypads or personal computers anytime and anywhere, inside or outside the home. One of the major issues of the power line data network is the reliability, which ensures the sent information to be correctly received. However, the electric power line at home has many appliances connected to it, and each appliance has its own characteristics that affect the power line conditions. When using the AC power line as a networking medium [6], one has to deal with problems such as electromagnetic interference, varying impedance, narrow frequency impairments (due to noise), and signal attenuation. To increase the network reliability, a higher gain for transmitters should be used; however, the transmission rate has to be reduced and the power for data transfer will increase. Thus, the gain of the transmitters in a power line data network is an important factor to ensure the transmission reliability and efficiency. In this paper, a neural-fuzzy network (NFN) with rule switches [7-8] is proposed to estimate the transmitter gain. Thanks to the rule switches, an optimal rule base can be obtained after training. An improved genetic algorithm (GA) is employed to tune the parameters and the rule base of the proposed NFN.

II. Neural-Fuzzy Network with Rule Switches

We use a fuzzy associative memory (FAM) [11] typed rule base for the NFN. The main difference between the proposed NFN and the traditional NFN is that a unit step function is introduced to each rule. The unit step functions is defined as,

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

This is equivalent to adding a switch to each rule in the NFN. The rule is used if the corresponding rule switch is closed; otherwise, the rule is not needed. An NFN with rule switches is shown in Fig. 1. Referring to this figure, we define the input and output variables as x_i and y_j respectively; where $i = 1, 2, \ldots, n_{in}$; n_{in} is the number of input variables; $j = 1, 2, \ldots, n_{out}$; n_{out} is the number of output variables. The behavior of y_j of the NFN is governed by m_f fuzzy rules of the following format:

R_g: IF $x_1(t)$ is $A_{1_g}(x_1(t))$ AND $x_2(t)$ is $A_{2_g}(x_2(t))$ AND … AND $x_{n_{in}}(t)$ is $A_{n_{in}g}(x_{n_{in}}(t))$

THEN $y_j(t)$ is w_{j_g}, $g = 1, 2, \ldots, m_f$; $t = 1, 2, \ldots, n_d$ (2)

where n_d is the number of input-output data pairs; w_{j_g}, $j = 1, 2, \ldots, n_{out}$, is the output singleton of the rule g; $g = 1, 2, \ldots, m_f$. The NFN membership functions are bell-shaped ones given by,

$$A_{i_g}(x_i(t)) = e^{\frac{-(x_i(t)-\bar{x}_{i_g})^2}{2\sigma_{i_g}^2}} \tag{3}$$

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

$$\mu_g(t)\in[0\quad 1] = A_{1_g}(x_1(t))\times A_{2_g}(x_2(t))\times\ldots\times A_{n_{in_g}}(x_{n_{in}}(t)),\ g = 1, 2, \ldots, m_f \tag{4}$$

The j-th output of the NFN, $y_j(t)$, is defined as,

[1] The work described in this paper was fully supported by a grant from the Research Grants Council of the Hong Kong Special Administration Region, China (Project No. PolyU 5098/01E).

[2] email: hkl@eie.polyu.edu.hk

$$y_j(t)=\frac{\sum_{g=1}^{m_f}\mu_g(t)w_{j_g}\delta\left(\varsigma_{j_g}\right)}{\sum_{g=1}^{m_f}\mu_g(t)} \quad (5)$$

where ς_{j_g} denotes the rule switch parameter of the g-th rule.

It should be noted that for this partially connected NFN, the number of rules is equal to the number of membership functions of each input variables, m_f. Comparing with that of a fully connected NFN, the number of rules required is smaller.

III. Improved Genetic Algorithm

Genetic algorithm (GA) is a directed random search technique [9] that is widely applied in optimization problems. Much research effort has been put to improve the performance of GA. Different selection schemes and genetic operators have been proposed [9]. Selection schemes such as rank-based selection, elitist strategies, steady-state election and tournament selection were reported [9]. In this paper, the standard GA is modified and new genetic operations are introduced to improve its performance. The improved GA process is shown in Fig. 2. Its details will be given as follows.

A. Initial Population

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

$$P=\{\mathbf{p}_1,\mathbf{p}_2,\cdots,\mathbf{p}_{pop\_size}\} \quad (6)$$

$\mathbf{p}_i=\begin{bmatrix}p_{i_1} & p_{i_2} & \cdots & p_{i_j} & \cdots & p_{i_{no\_vars}}\end{bmatrix}$, $i=1,2,\ldots,pop\_size; j = 1, 2, \ldots, no\_vars$ (7)

$$para^j_{\min}\le p_{i_j}\le para^j_{\max} \quad (8)$$

where pop_size is the population size; no_vars is 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; $para^j_{\min}$ and $para^j_{\max}$ are the minimum and maximum values of p_{i_j} respectively for all i. From (6) to (8), we see 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,

$$fitness=f(\mathbf{p}_i) \quad (9)$$

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 by the method of spinning the roulette wheel [9]. It is believed that high potential parents produce better offspring (survival of the best ones). The chromosome having a higher fitness value should therefore have a higher chance to be selected. First, we assign probability q_i to the chromosome $\mathbf{p}_i$:

$$q_i=\frac{f(\mathbf{p}_i)}{\sum_{k=1}^{pop\_size}f(\mathbf{p}_k)},\ i=1,2,\ldots,pop\_size \quad (10)$$

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

$$\hat{q}_i=\sum_{k=1}^{i}q_k,\ i=1,2,\ldots,pop\_size \quad (11)$$

Then, we randomly generate a nonzero floating-point number, $d\in\begin{bmatrix}0 & 1\end{bmatrix}$. The chromosome $\mathbf{p}_i$ is selected if $\hat{q}_{i-1}<d\le\hat{q}_i$ ($\hat{q}_0=0$). Hence, a chromosome having a larger $f(\mathbf{p}_i)$ will have a higher chance to be selected. The best chromosomes will get more offspring, 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 crossover and mutation operations.

1. Crossover

The crossover operation exchanges information from the two parents, chromosomes $\mathbf{p}_1$ and $\mathbf{p}_2$, obtained in the selection process. The two parents will produce one offspring. First, four chromosomes will be generated:

$$\mathbf{os}^1_c=\begin{bmatrix}os^1_1 & os^1_2 & \cdots & os^1_{no\_vars}\end{bmatrix}=\frac{\mathbf{p}_1+\mathbf{p}_2}{2} \quad (12)$$

$$\mathbf{os}^2_c=\begin{bmatrix}os^2_1 & os^2_2 & \cdots & os^2_{no\_vars}\end{bmatrix}=\mathbf{p}_{\max}(1-w)+\max(\mathbf{p}_1,\mathbf{p}_2)w \quad (13)$$

$$\mathbf{os}^3_c=\begin{bmatrix}os^3_1 & os^3_2 & \cdots & os^3_{no\_vars}\end{bmatrix}=\mathbf{p}_{\min}(1-w)+\min(\mathbf{p}_1,\mathbf{p}_2)w \quad (14)$$

$$\mathbf{os}^4_c=\begin{bmatrix}os^4_1 & os^4_2 & \cdots & os^4_{no\_vars}\end{bmatrix}=\frac{(\mathbf{p}_{\max}+\mathbf{p}_{\min})(1-w)+(\mathbf{p}_1+\mathbf{p}_2)w}{2} \quad (15)$$

$$\mathbf{p}_{\max}=\begin{bmatrix}para^1_{\max} & para^2_{\max} & \cdots & para^{no\_vars}_{\max}\end{bmatrix} \quad (16)$$

$$\mathbf{p}_{\min}=\begin{bmatrix}para^1_{\min} & para^2_{\min} & \cdots & para^{no\_vars}_{\min}\end{bmatrix} \quad (17)$$

where $w\in\begin{bmatrix}0 & 1\end{bmatrix}$ is a weight to be determined by users, $\max(\mathbf{p}_1,\mathbf{p}_2)$ denotes the vector with each element obtained by taking the maximum among the corresponding element of $\mathbf{p}_1$ and $\mathbf{p}_2$, e.g. $\max(\begin{bmatrix}1 & -2 & 3\end{bmatrix},\begin{bmatrix}2 & 3 & 1\end{bmatrix})=\begin{bmatrix}2 & 3 & 3\end{bmatrix}$. Similarly, $\min(\mathbf{p}_1,\mathbf{p}_2)$ gives a vector by taking the minimum value, e.g. $\min(\begin{bmatrix}1 & -2 & 3\end{bmatrix},\begin{bmatrix}2 & 3 & 1\end{bmatrix})=\begin{bmatrix}1 & -2 & 1\end{bmatrix}$. Among $\mathbf{os}^1_c$ to $\mathbf{os}^4_c$, the one with the largest fitness value is used as the offspring $\mathbf{os}$ of the crossover operation:

$$\mathbf{os}\equiv\begin{bmatrix}os_1 & os_2 & \cdots & os_{no\_vars}\end{bmatrix}=\mathbf{os}^{i_{os}}_c \quad (18)$$

where i_{os} is the index i that gives a maximum value of $f\left(\mathbf{os}^i_c\right)$, $i = 1, 2, ,3 ,4$. If the crossover operation can provide a good offspring, a higher fitness value can be reached in less iteration. As seen from (12) to (15), the potential offspring after the crossover operation spreads over the domain. While (12) and (15) result in searching around the centre region of the domain (a value of w near to 1 in (15) can move $\mathbf{os}_c^4$ to be near $\frac{\mathbf{p}_1+\mathbf{p}_2}{2}$), (13) and (14) move the potential offspring to be near the domain boundary (a small value of w in (13) and (14) can move $\mathbf{os}_c^2$ and $\mathbf{os}_c^3$ to be near $\mathbf{p}_{\max}$ and $\mathbf{p}_{\min}$ respectively).

2. Mutation

The offspring (18) will then undergo the mutation operation, which changes the genes of the chromosomes. In this paper, a different process of mutation is proposed. The details are as follows. Every gene of the offspring $\mathbf{os}$ of (18) will have a chance to mutate governed by a probability of mutation, $p_m \in [0 \quad 1]$, which is defined by users. This probability gives an expected number ($p_m \times no\_vars$) of genes that undergo the mutation. For each gene, a random number between 0 and 1 will be generated such that if it is less than or equal to p_m, the operation of mutation will take place on that gene and updated instantly. The gene of the offspring of (18) is then mutated by:

$$\hat{o}_{s_k} = \begin{cases} o_{s_k} + \Delta o_{s_k}^U \text{ if } f(\mathbf{o}_s + \Delta\mathbf{o}_{s_k}^U) \geq f(\mathbf{o}_s - \Delta\mathbf{o}_{s_k}^L) \\ o_{s_k} - \Delta o_{s_k}^L \text{ if } f(\mathbf{o}_s + \Delta\mathbf{o}_{s_k}^U) < f(\mathbf{o}_s - \Delta\mathbf{o}_{s_k}^L) \end{cases}, k = 1, 2, \ldots, no\_vars \quad (19)$$

where

$$\Delta\mathbf{o}_{s_k}^U = [0 \quad 0 \quad \cdots \quad \Delta o_{s_k}^U \quad \cdots \quad 0] \quad (20)$$

$$\Delta\mathbf{o}_{s_k}^L = [0 \quad 0 \quad \cdots \quad \Delta o_{s_k}^L \quad \cdots \quad 0] \quad (21)$$

$$\Delta o_{s_k}^U = w_{m_k} r\left(para_{\max}^k - o_{s_k}\right) \quad (22)$$

$$\Delta o_{s_k}^L = w_{m_k} r\left(o_{s_k} - para_{\min}^k\right) \quad (23)$$

$r \in [0 \quad 1]$ is a randomly generated number; $w_{m_k} \in (0 \quad 1]$ is a weight governing the magnitudes of $\Delta o_{s_k}^U$ and $\Delta o_{s_k}^L$. The value of w_{m_k} is varied by the value of $\frac{\tau}{T}$, where τ is the iteration number and T is the total number of iteration. In order to search more locally at the later stage, the value of w_{m_k} should be small as $\frac{\tau}{T}$ increases so as to reduce the significance of the mutation. Based on this idea, a monotonic decreasing function governing w_{m_k} is proposed as follows,

$$w_{m_k} = w_f\left(1-\frac{\tau}{T}\right)^{\frac{1}{w_\tau}} \geq 0 \quad (24)$$

where $w_f \in [0 \quad 1]$ and $w_\tau > 0$ determine the initial value and the decay rate respectively. Their values are chosen by users. For a large w_f, from (22) and (23), $\Delta o_{s_k}^U \approx r\left(para_{\max}^k - o_{s_k}\right)$ and $\Delta o_{s_k}^L \approx r\left(o_{s_k} - para_{\min}^k\right)$ initially as $\left(1-\frac{\tau}{T}\right)^{\frac{1}{w_\tau}} \approx 1$, which ensure a large search space. When $\left(1-\frac{\tau}{T}\right)^{\frac{1}{w_\tau}} \approx 0$, $\Delta o_{s_k}^U$ and $\Delta o_{s_k}^L$ are small to ensure a small search space for fine-tuning.

E. Reproduction

The new offspring will be evaluated using the fitness function of (9). This new offspring will replace the chromosome with the smallest fitness value among the population if a randomly generated number within 0 to 1 is smaller than $p_a \in [0 \quad 1]$, which is the probability of acceptance defined by users. Otherwise, the new offspring will replace the chromosome with the smallest fitness value only if the fitness value of the offspring is greater than the fitness value of that chromosome in the population. p_a is effectively the probability of accepting a bad offspring in order to reduce the chance of converging to a local optimum. Hence, the possibility of reaching the global optimum is kept.

After the operation of selection, crossover, mutation and reproduction, 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. a defined number of iteration has been reached.

F. Benchmark Test Functions

Some benchmark test functions [10] are used to examine the applicability and efficiency of the improved GA. Six test functions, $f_i(\mathbf{x})$, i = 1, 2, 3, 4, 5, 6 will be used, where $\mathbf{x} = [x_1 \quad x_2 \quad \cdots \quad x_n]^T$, n is an integer denoting the vector's dimension. The six test functions are defined as follows.

$$f_1(\mathbf{x}) = \sum_{i=1}^{n} x_i^2, \ -5.12 \leq x_i \leq 5.12 \quad (25)$$

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

$$f_2(\mathbf{x}) = \sum_{i=1}^{n-1}\left(100\left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2\right), \ -2.048 \leq x_i \leq 2.048 \quad (26)$$

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

$$f_3(\mathbf{x}) = \sum_{i=1}^{n} floor(x_i + 0.5)^2, \ -5.12 \leq x_i \leq 5.12 \quad (27)$$

where n = 5 and the minimum point is at $f_3([5.12, 5], \ldots, [5.12, 5]) = 0$. The value of the floor function, $floor(\cdot)$, is obtained by rounding down its argument to the nearest smaller integer.

$$f_4(\mathbf{x}) = \sum_{i=1}^{n} i x_i^4 + Gauss(0,1), \ -1.28 \leq x_i \leq 1.28 \quad (28)$$

where n = 3 and the minimum point is at $f_4(0, 0, 0) = 0$. *Gauss*(0, 1) is a function to randomly generate a floating-point number between 0 and 1.

$$f_5(\mathbf{x}) = \frac{1}{k} + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2}\left(x_i - a_{ij}\right)^6}, \quad -65.356 \leq x_i \leq 65.356 \quad (29)$$

where

$$\mathbf{a} = \{a_{ij}\} = \left[\begin{matrix} -32 & -16 & 0 & 16 & 32 & -32 & -16 & 0 & 16 & 32 \\ 32 & 32 & 32 & 32 & 32 & -16 & -16 & -16 & -16 & -16 \end{matrix}\right. \\ \left.\begin{matrix} -32 & -16 & 0 & 16 & 32 & -32 & -16 & 0 & 16 & 32 & -32 & -16 & 0 & 16 & 32 \\ 0 & 0 & 0 & 0 & 0 & 16 & 16 & 16 & 16 & 16 & 32 & 32 & 32 & 32 & 32 \end{matrix}\right],$$

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

$$f_6(\mathbf{x}) = \sum_{i=1}^{n}\left[x_i^2 - 10\cos(2\pi x_i) + 10\right],\ -5.12 \le x_i \le 5.12 \qquad (30)$$

where $n = 3$ and the minimum point is at $f_6(0, 0, 0) = 0$. 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 functions for $f_1(\mathbf{x})$ to $f_4(\mathbf{x})$ and $f_6(\mathbf{x})$ are defined as,

$$fitness = \frac{1}{1 + f_i(\mathbf{x})},\ i = 1, 2, 3, 4, 6. \qquad (31)$$

and the fitness function for $f_5(\mathbf{x})$ is defined as,

$$fitness = f_5(\mathbf{x}) \qquad (32)$$

The proposed GA is used to find the minimum points of these 6 test functions. The results are compared with those obtained by the standard GA with arithmetic crossover and non-uniform mutation [9]. The control parameters of the proposed GA and the standard GA with arithmetic crossover and non-uniform mutation are tabulated in Table I. These parameters are selected by trial and error through experiments for good performance. The initial values of **x** in the population for a test function are set to be the same for both GAs. The number of iteration for each test function is listed in Table II. For test functions 1 to 6, the initial values are $[1\ \ 1\ \ 1]$, $[0.5\ \ 0.5]$, $[1\ \cdots\ 1]$, $[0.5\ \cdots\ 0.5]$, $[10\ \cdots\ 10]$ and $[1\ \ 1\ \ 1]$ respectively. The results of the average fitness values over 50 times of simulations based on the proposed and standard GAs are shown in Fig. 3 and tabulated in Table II. It can be seen that the performance of the proposed GA is better.

IV. Tuning of Membership Functions and Rule Base

By introducing the rule switches, the parameters and the rule base of the NFN proposed in section II can be tuned using the improved GA. To perform the tuning process, some control parameters of the improved GA, namely $w, p_m, p_a, pop\_size, w_f$ and w_τ have to be chosen first. The parameters of the NFN to be tuned are $[\ \bar{x}_{ig_i}\ \ \sigma_{ig_i}\ \ \varsigma_g\ ]$ for all i, g_i, g. This is the chromosome for the GA process. $\bar{x}_{ig_i}$ and σ_{ig_i} are the parameters of the membership functions and ς_g is the parameter of the rule switch. The proposed NFN is used to learn the input-output relationship of an application. The desired input-output relationship can be written as,

$$\mathbf{y}^d(t) = \mathbf{q}(\mathbf{z}^d(t)),\ t = 1, 2, \ldots, u \qquad (33)$$

where $\mathbf{y}^d(t) = [y_1^d(t)\ \ y_2^d(t)\ \ \cdots\ \ y_{n_{out}}^d(t)]$ is the desired output vector corresponding to the input vector, $\mathbf{z}^d(t) = [z_1^d(t)\ \ z_2^d(t)\ \ \cdots\ \ z_n^d(t)]$ and $\mathbf{q}(\cdot)$ is an unknown non-linear function. The fitness function is defined as,

$$fitness = \frac{1}{1 + err} \qquad (34)$$

$$err = \frac{1}{u}\sum_{t=1}^{u}\left|\mathbf{y}^d(t) - \mathbf{y}(t)\right| \qquad (35)$$

The objective is to minimize the mean absolute error (MAE) of (35) using the improved GA The range of *fitness* in (34) is [0, 1]. A larger *fitness* indicates a smaller *err*. By using the NFN and the improved GA, an optimal NFN in terms of the number of rules and the membership functions can be obtained.

V. Application Example

The proposed NFN will be employed to estimate the gain of the transmitter in a power line data network. A network with 48 inputs and 48 outputs is employed to perform the estimation. The inputs of the proposed NFN are the gains in every half-hour of the previous day, while the outputs are the estimated gains in every half-hour of the present day. Seven NFNs will be used for estimating the transmission gain. These NFNs are named Sunday-Monday, Monday-Tuesday, Tuesday-Wednesday, Wednesday-Thursday, Thursday-Friday, Friday-Saturday and Saturday-Sunday NFNs. For instance, the Sunday-Monday NFN makes use of the transmission gains of Sunday to estimate the transmission gains of Monday. To perform the training, we have to collect some testing patterns. 48 transmission gains for every half-hour at each day (24 hours) will be measured. To measure the optimal transmission gain, a data packet is continuously sent from the transmitter to the receiver while the transmission gain value is increased gradually. The increment in the gain value will stop at the point where the data packet can be correctly received. At that point, the value of the transmission gain is the smallest possible one that is not susceptible to interference, and the optimal transmission gain would be set at a value a bit higher than this smallest one.

In this application example, transmission gains for 7 weeks are collected as the testing patterns to train the proposed NFN using the proposed GA. Take the Sunday-Monday NFN for instance, the measured transmission gains for Sundays of the previous 7 weeks will serve as the inputs and the desired outputs are the measured transmission gains of Mondays in the previous 7 weeks respectively. The rationale for this arrangement of training is based on the assumption that the relation between the gain pattern of Sunday and that of Monday is approximately constant in the most recent seven weeks. Seven NFNs will be used to derive 7 different relations for the days in a week. The number of membership functions for each input variable is chosen to be 5. For the improved GA process, $w = 0.5$, $p_m = 0.1$, $p_a = 0.1$, $w_f = 1$ and $w_\tau = 1$. The training will last for 2000 iterations. The upper and lower bounds of each parameter are 1 and –1 respectively. The initial values of all the parameters are generated randomly. All the 7 NFNs are trained in the same way. The input and output patterns are normalized such that the elements of the input vector and the desired output vector are between 0 and 1. The fitness function for training is defined in (34) and (35). By minimizing the errors between the desired outputs and the proposed NFN's outputs, the characteristics of the transmission gain pattern are learnt. After training, the proposed NFN will be employed to estimate the transmission gain. The 48 transmission gains of the previous day will be fed to the trained NFN. The 48 NFN outputs will be the estimated transmission gains for every half-hour of the present day. The transmission gain employed by the transmitter in every minute is obtained by a linear interpolation equation:

$$\tilde{g}(\xi+30\tau)=\frac{\hat{g}(\tau+1)-g(\tau)}{30}\xi+g(\tau)\text{, }\xi=1,2,\ldots,30,\ \tau=0,1,\ldots,47 \qquad (36)$$

where $\tilde{g}(\xi+30\tau)$ denotes the transmission gain employed by the transmitter at $\xi + 30\tau$ minutes counted from 00:00 am; $\hat{g}(\tau+1)$ denotes the τ+1-th output of the corresponding NFN (i.e. the estimated transmission gain at $30(\tau + 1)$ minutes counted from 00:00 am); $g(\tau)$ denotes the measured transmission gain at 30τ minutes counted from 00:00 am.

For comparison purpose, a traditional 3-layer NFN [11] is also trained by the improved GA. The proposed NFN and the traditional 3-layer NFN will also be trained by standard GA with arithmetic crossover and non-uniform mutation [9] under the same condition. For the standard GA approach, the probabilities of crossover and mutation are selected to be 0.6 and 0.01 respectively, and the shaping parameter of the GA for non-uniform mutation [9] is selected to be 1. Table III shows the results of the proposed and traditional approaches for the Sunday-Monday NFN. To obtain the fitness value for the testing pattern, the measured transmission gains for Sunday of week 8 are fed to the trained NFN to obtain the estimated transmission gains for Monday of week 8. The fitness value is then obtained by (33) and (34) based on the estimated and measured transmission gains.

Fig. 4 shows the actual (solid line) and predicted (dotted line) normalized gains for Monday of week 8 using the proposed and traditional NFNs respectively with the improved GA. Fig. 5 shows the actual (solid line) and predicted (dotted line) normalized gains for Monday of week 8 using the proposed and the traditional NFNs respectively with standard GA. It can be seen that the performance of the proposed approach is better than that of other approaches in terms of the fitness values and size of the rule base.

VI. Conclusion

A neural-fuzzy network with rule switches has been proposed to estimate the transmission gain for the AC power line data network in an intelligent home. The proposed NFN facilitates the learning of the network parameters and the rule base. An improved GA has been proposed for the training process. An application example has been given to illustrate the design process and the merits of the proposed approach.

References

[1] C. Douligenis, "Intelligent home systems," *IEEE Communications Magazine*, vol. 31, pp. 52-61, Oct. 1993.

[2] E.M.C. Wong, "A phone-based remote controller for home and office automation," *IEEE Transactions on Consumer Electronics*, vol. 40, no. 1, pp. 148-152, Feb. 1994.

[3] J. Gray, D.G. Caldwell, and N. Linge, "A new perspective on the intelligent home," *Impact of Multimedia Services on the Home Environment, IEE Colloquium*, 1996, pp. 7/1-7/4.

[4] D.R. Amitava, "Networks for homes," *IEEE Spectrum*, vol. 36, pp.26-33, Dec. 1999.

[5] D. Radford, "Spread-spectrum data leap through ac power line wiring," *IEEE Spectrum*, pp. 48-53, Nov. 1996.

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

[7] H.K. Lam, K.F. Leung, S.H. Ling, F.H.F. Leung and P.K.S. Tam, "On interpretation of graffiti digits and commands for eBooks: neural-fuzzy network and genetic algorithm approach," in *Proc. IEEEFUZZ 2002, WCCI*, Honolulu, Hawaii, 12-17, May, 2002, pp.443-449.

[8] S.H. Ling, H.K. Lam, F.H.F. Leung and P.K.S. Tam, "A neural fuzzy network with optimal number of rules for short-term load forecasting in an intelligent home," in *Proc. 10th Int. Conf. Fuzzy Systems, IEEEFUZZ'2001*, Melbourne, Australia, 2-5 Dec, 2001, pp. 1456-1459.

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

[10] 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.

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

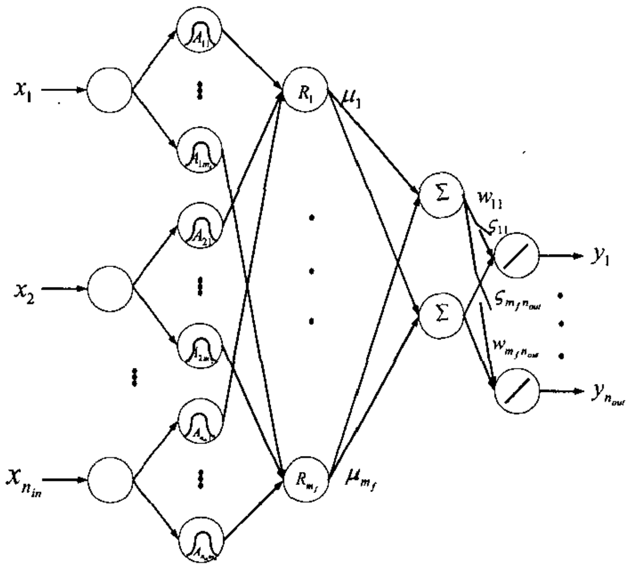
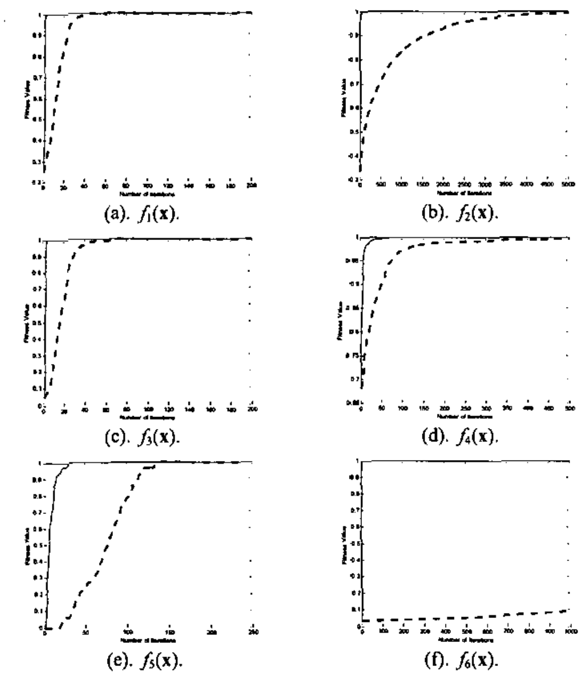


Fig. 1. Three-layer neural-fuzzy network with rule switches.

```
Procedure of the improved GA
begin
        $\tau \to 0$ // $\tau$: number of iteration
        initialize P($\tau$)      //P($\tau$): population for iteration $\tau$
        evaluate $f$(P($\tau$))    // $f$(P($\tau$)):fitness function
while (not termination condition) do
    begin
    $\tau \to \tau+1$
    select 2 parents $p_1$ and $p_2$ from P($\tau-1$)
    perform crossover operation according to equations (12) - (18)
    perform mutation operation according to equations (19) - (24) to
    generate the offspring os
    // reproduce a new P($\tau$)
    if random number < $p_a$  // $p_a$: probability of acceptance
                    os replaces the chromosome with the smallest
                    fitness value in the population
    else if $f$(os) > smallest fitness value in the P($\tau-1$)
          os replaces the chromosome with the smallest fitness value
    end
    evaluate $f$(P($\tau$))
    end
end
```

Fig. 2. Procedure of the improved GA.

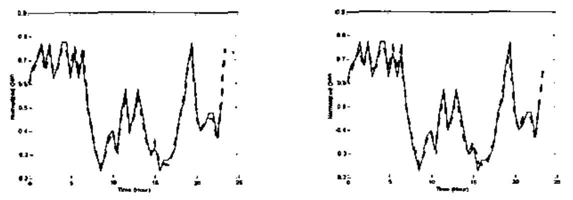


(a). $f_1(x)$. (b). $f_2(x)$.

(c). $f_3(x)$. (d). $f_4(x)$.

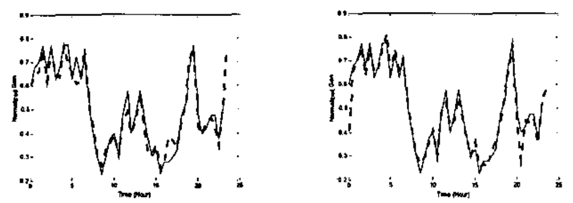
(e). $f_5(x)$. (f). $f_6(x)$.

Fig. 3. Results of the improved (solid line) and standard (dotted line) GAs.



(a) (b)

Fig. 4. The actual (solid line) and predicted (dotted line) normalized gains for week 8 using the NFNs (a) with, and (b) without rule switches (both trained by the improved GA).



(a). (b)

Fig. 5. The actual (solid line) and predicted (dotted line) normalized gains for week 8 using the NFNs (a) with, and (b) without rule switches (both trained by the GA with arithmetic crossover and non-uniform mutation).

Test function	w	p_m	w_f	w_τ	p_a
$f_1(x)$	0.1	0.3	0.001	0.001	0.1
$f_2(x)$	0.5	0.5	0.01	10	0.1
$f_3(x)$	0.1	0.8	1	1000	0.1
$f_4(x)$	0.5	0.35	0.001	10	0.1
$f_5(x)$	0.5	0.8	0.1	0.1	0.1
$f_6(x)$	0.01	0.1	0.01	0.01	0.1

(a)

Test function	b, (Shape parameter)	p_c (Probability of crossover)	p_m (Probability of mutation)
$f_1(x)$	5	0.7	0.9
$f_2(x)$	1	0.8	0.8
$f_3(x)$	0.1	0.7	0.6
$f_4(x)$	1	0.8	0.35
$f_5(x)$	0.1	0.8	0.5
$f_6(x)$	0.9	0.7	0.4

(b)

Table I. Control parameters of GAs for the benchmark test functions: (a) improved GA, (b) standard GA with arithmetic crossover and non-uniform mutation.

Test function	Average fitness value from proposed GA	Average fitness value from standard GA	Number of iterations
$f_1(x)$	1.0000	1.0000	100
$f_2(x)$	0.9997	0.6393	5000
$f_3(x)$	1.0000	1.0000	200
$f_4(x)$	0.9997	0.8037	500
$f_5(x)$	1.0000	1.0000	250
$f_6(x)$	1.0000	0.7297	200

Table II. Average fitness values obtained from the proposed GA and the traditional GA for the benchmark test functions.

	NFN with rule switches trained by the improved GA	NFN with rule switches trained by the standard GA	NFN without rule switches trained by the improved GA	NFN without rule switches trained by the standard GA
Fitness Value in MAE (Training)	0.9829	0.9813	0.9724	0.9703
Fitness Value in MAE (Testing)	0.9815	0.9793	0.9715	0.9697
Number of Rules	85	91	96	96

Table III. Results based on the proposed and traditional approaches.