An Improved Genetic Algorithm Based Fuzzy-Tuned Neural Network

S.H. Ling[1], F.H.F. Leung, and H.K. Lam.
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 a fuzzy-tuned neural network, which is trained by an improved genetic algorithm (GA). The fuzzy-tuned neural network consists of a neural-fuzzy network and a modified neural network. In the modified neural network, a neuron model with two activation functions is used so that the degree of freedom of the network function can be increased. The neural-fuzzy network governs some of the parameters of the neuron model. It will be shown that the performance of the proposed fuzzy-tuned neural network is better than that of the traditional neural network with a similar number of parameters. An improved GA is proposed to train the parameters of the proposed network. Sets of improved genetic operations are presented. The performance of the improved GA will be shown to be better than that of the traditional GA. Some application examples are given to illustrate the merits of the proposed neural network and the improved GA.**

***Index Terms* — Fuzzy logic, genetic algorithm, neural-fuzzy network, neural network.**

I. INTRODUCTION

Neural networks (NNs) are widely applied in areas such as prediction [3, 8], system modeling and control [1]. Owing to its particular structure, a 3-layer feed-forward neural network can approximate any nonlinear continuous function to an arbitrary accuracy [1]. It can be trained [9] using some algorithms such as Genetic Algorithm (GA) [5, 10] and back propagation [9]. Traditionally, the three layers (input, hidden and output layers) of nodes are connected in a layer-to-layer manner [2].

Neural-fuzzy network [1] has been used to deal with variable linguistic information, and it is much closer in spirit to human thinking and natural language. By processing fuzzy information, reasoning with respect to a linguistic knowledge base can be realized.

GA is a directed random search technique [10] that is widely used to find the global solution of optimization problems over a domain [9-11]. It has been applied in different areas such as fuzzy control [7, 12-13], forecasting [3-4], path planning [14], greenhouse climate control [15], modeling and classification [6] etc. Efforts have been spent to improve the performance of GA. Different

[1] corresponding e-mail: ensteve@eie.polyu.edu.hk

selection schemes and genetic operations have been proposed. Selection schemes such as rank-based selection, elitist strategies, steady-state election and tournament selection have been reported [16]. There are two kinds of genetic operations, namely crossover and mutation. Apart from random mutation and crossover, other crossover and mutation mechanisms have been proposed [10, 16-18].

In this paper, a fuzzy-tuned neural network, which consists of a traditional neural-fuzzy network (NFN) [1] and a modified neural network [4], is proposed. Inside the modified neural network, each neuron of the hidden layer has two different activation functions. In this way, the degree of freedom of the network function can be increased. Some parameters of these activation functions will be tuned by the NFN. The proposed fuzzy-tuned neural network can give a better performance than traditional feed-forward neural networks [2] with a similar number of parameters. We also propose in this paper an improved GA to train the neural network parameters. Modified genetic operations of crossover and mutation will be introduced. On realizing the crossover operation, the offspring spreads over the domain so that a higher chance of reaching the global optimum can be obtained. A fuzzy mutation operation, which is modified from the non-uniform mutation [11, 17], incorporates human knowledge on mutation into fuzzy rules. By employing these genetic operations, the improved GA performs more efficiently and provides a faster convergence than the traditional GA in eight benchmark test functions [19-20]. Three application examples are used to test the proposed network and the improved GA, and good results are obtained.

This paper is organized as follows. In section II, the proposed fuzzy-tuned neural network is presented. In section III, the improved GA will be presented. In section IV, it will be shown that the improved GA performs more efficiently than the traditional GA based on some benchmark test functions. In section V, the training of the parameters of the proposed fuzzy-tuned neural network using the improved GA will be presented. In section VI, some application examples will be given. A conclusion will be drawn in session VII.

II. Fuzzy-Tuned Neural Network

The block diagram of the proposed fuzzy-tuned neural network is shown in Fig. 1. It consists of a modified neural network [4] and a neural-fuzzy network. Inside the modified neural network, each neuron in the hidden layer has two activation functions: a static activation function (SAF) and a dynamic activation function (DAF). The parameters of the SAF are fixed. For the DAF, the parameters are obtained from the neural-fuzzy network.

A. Modified Neural Network

Fig. 2 shows the proposed neuron model. With fixed parameters, the SAF has its output depending on the inputs of the neuron. The output of the SAF is processed by the DAF, of which the parameters depend on the outputs of the neural-fuzzy network. With this proposed neuron in the

hidden layer, the connection of the modified neural network is shown in Fig. 3. $\mathbf{p^U} = \left[p_1^U, \quad p_2^U, \quad \ldots \quad p_{n_h}^U\right]$ and $\mathbf{p^L} = \left[p_1^L, \quad p_2^L, \quad \ldots \quad p_{n_h}^L\right]$ are the parameters of the DAFs obtained from the neural-fuzzy network, where n_h denotes the number of hidden nodes.

Proposed neuron model

Let v_{ik} be the synaptic connection weight from the *i*-th input node z_i to the *k*-th neuron, the output κ_k of the *k*-th neuron's SAF is defined as,

$$\kappa_k = net_s^k\left(\sum_{i=1}^{n_{in}} z_i v_{ik}\right),\ i = 1, 2, \ldots, n_{in};\ k = 1, 2, \ldots, n_h \tag{1}$$

where n_{in} denotes the number of inputs, n_h denotes the number of hidden nodes, and $net_s^k(\cdot)$ is the static activation function defined as:

$$net_s^k\left(\sum_{i=1}^{n_{in}} z_i v_{ik}\right) = \begin{cases} e^{\frac{-\left(\sum_{i=1}^{n_{in}} z_i v_{ik} - m_s^k\right)^2}{2{\sigma_s^k}^2}} - 1 & \text{if } \sum_{i=1}^{n_{in}} z_i v_{ik} \le m_s^k \\ 1 - e^{\frac{-\left(\sum_{i=1}^{n_{in}} z_i v_{ik} - m_s^k\right)^2}{2{\sigma_s^k}^2}} & \text{otherwise} \end{cases} \tag{2}$$

where m_s^k and σ_s^k are the mean and standard deviation for the *k*-th SAF respectively. The parameters m_s^k and σ_s^k are fixed after the training process. From (2), the output value is ranged from –1 to 1. The shapes of the SAFs are shown in Fig. 4. The static mean is used to control the bias as shown in Fig. 4a, and the static standard deviation influences the sharpness as shown in Fig. 4b.

The output ζ_k of the *k*-th neuron is the DAF output defined as,

$$\zeta_k = net_d^k(\kappa_k, p_k^U, p_k^L),\ k = 1, 2, \ldots, n_h \tag{3}$$

and,

$$net_d^k(\kappa_k, p_k^U, p_k^L) = \begin{cases} e^{\frac{-\left(\kappa_k - p_k^U\right)^2}{2{p_k^L}^2}} - 1 & \text{if } \kappa_k \le p_k^U \\ 1 - e^{\frac{-\left(\kappa_k - p_k^U\right)^2}{2{p_k^L}^2}} & \text{otherwise} \end{cases} \tag{4}$$

p_k^U and p_k^L are the parameters of the DAF, which are effectively the dynamic mean and the dynamic standard deviation (that depend on z_i) respectively for the *k*-th DAF. From (1) to (4), the input-output relationship of the *k*-th neuron is given by,

$$\zeta_k = net_d^k\left(net_s^k\left(\sum_{i=1}^{n_{in}} z_i v_{ik}\right)\right) \tag{5}$$

Connection of the modified neural network

The proposed neural network (Fig. 3) has n_{in} nodes in the input layer, n_h nodes in the hidden layer, and n_{out} nodes in the output layer. In the hidden layer, the proposed neuron model is employed. In the output layer (Fig. 5), a static activation function is used. Considering an input-output pair $(\mathbf{z}, \mathbf{y})$ for the neural network, from (5), the *l*-th output of the modified neural network is given by,

$$y_l = net_o^l\left(\sum_{k=1}^{n_h} \zeta_k w_{kl}\right) \tag{6}$$

$$= net_o^l\left(\sum_{k=1}^{n_h} net_d^k\left(net_s^k\left(\sum_{i=1}^{n_{in}} z_i v_{ik}\right)\right) w_{kl}\right) \tag{7}$$

where w_{kl}, $k = 1, 2, \ldots, n_h$; $l = 1, 2, \ldots n_{out}$ denotes the weight of the link between the *k*-th hidden and the *l*-th output nodes; $net_o^l(\cdot)$ denotes the activation function of the output neuron:

$$net_o^l\left(\sum_{k=1}^{n_h} \zeta_k w_{kl}\right) = \begin{cases} e^{\frac{-\left(\sum_{k=1}^{n_h} \zeta_k w_{kl} - m_o^l\right)^2}{2\sigma_o^{l\,2}}} - 1 & \text{if } \sum_{k=1}^{n_h} \zeta_k w_{kl} \le m_o^l \\ 1 - e^{\frac{-\left(\sum_{k=1}^{n_h} \zeta_k w_{kl} - m_o^l\right)^2}{2\sigma_o^{l\,2}}} & \text{otherwise} \end{cases} \tag{8}$$

where m_o^l and σ_o^l are the mean and the standard deviation of the output node activation function respectively.

Of this network structure, the first layer simply distributes the input variables. The SAFs in the hidden layer effectively determine hyper-planes as switching surfaces. Owing to the DAF, the input $\mathbf{p}^{\mathbf{U}}$ concerns the bias term while the input $\mathbf{p}^{\mathbf{L}}$ influences the sharpness of the edges of these hyper-planes. They eventually combine into convex regions by the output layer. Some of the modified neural network parameters are trained by GA, and some others are obtained from the NFN.

B. Neural-Fuzzy Network

By using the NFN, the parameters of $\mathbf{p}^{\mathbf{U}}$ and $\mathbf{p}^{\mathbf{L}}$ of the DAFs are governed by some fuzzy rules. The NFN is shown in Fig. 6. The input and output variables of the NFN are z_i and p_j respectively; where $i = 1, 2, \ldots, n_{in}$; $j = 1, 2, \ldots, q$; and $q = 2n_h$ is the number of output variables. The behavior of the NFN is governed by *m* fuzzy rules of the following format:

R_h: IF $z_1(t)$ is A_1^h AND $z_2(t)$ is A_2^h AND … AND $z_{n_{in}}(t)$ is $A_{n_{in}}^h$

THEN $p_j(t)$ is g_{hj}, $t = 1, 2, \ldots, u$ (9)

where u denotes the number of input-output data pairs; h = 1, 2, …, m, is the rule number; g_{hj} is the j-th output singleton of the rule h. The membership functions are bell-shaped ones given by,

$$\mu_{A_i^h}(z_i(t)) = e^{\frac{-(z_i(t)-\bar{z}_i^h)^2}{2\sigma_i^{h^2}}} \tag{10}$$

where the parameter $\bar{z}_i^h$ and σ_i^h are the mean value and the standard deviation of the membership function $\mu_{A_i^h}$ respectively. The grade of membership of each rule is defined as,

$$\mu_h(t) = \prod_{i=1}^{n} \mu_{A_i^h}(z_i(t)) \tag{11}$$

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

$$p_j(t) = \frac{\sum_{h=1}^{m} \mu_h(t) g_{hj}}{\sum_{h=1}^{m} \mu_h(t)} \tag{12}$$

which is a parameter of the DAF. The number of outputs of the NFN doubles the number of hidden node of the modified neural network. Referring to Fig. 3 and Fig. 6, $p_1 = p_1^U$, $p_2 = p_1^L$, …, $p_{q-1} = p_{n_h}^U$, $p_q = p_{n_h}^L$. The weights of the NFN are tuned by the improved GA.

III. Improved Genetic Algorithm

The traditional GA is modified and new genetic operations are introduced to improve its performance. The improved GA process is shown in Fig. 7. Its details are discussed as follows.

A. Initial Population

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

$$P = \{\mathbf{p}_1, \mathbf{p}_2, \cdots, \mathbf{p}_{pop\_size}\} \tag{13}$$

$$\mathbf{p}_i = \begin{bmatrix} p_{i_1} & p_{i_2} & \cdots & p_{i_j} & \cdots & p_{i_{no\_vars}} \end{bmatrix}, \quad i = 1, 2, \ldots, pop\_size; \; j = 1, 2, \ldots, no\_vars \tag{14}$$

$$para_{\min}^j \le p_{i_j} \le para_{\max}^j \tag{15}$$

where *pop_size* denotes the population size; *no_vars* denotes the number of variables in each chromosome to be tuned; p_{i_j}, i = 1, 2, …, *pop_size*; j = 1, 2, …, *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}

respectively for all *i*. It can be seen from (13) to (15) 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 can be written as,

$$fitness = f(\mathbf{p}_i) \tag{16}$$

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 [11]. It is believed that 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. A probability q_i is first assigned to the chromosome $\mathbf{p}_i$:

$$q_i = \frac{f(\mathbf{p}_i)}{\sum_{k=1}^{pop\_size} f(\mathbf{p}_k)},\ i = 1, 2, \ldots, pop\_size \tag{17}$$

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

$$\hat{q}_i = \sum_{k=1}^{i} q_k,\ i = 1, 2, \ldots, pop\_size \tag{18}$$

The selection process starts by randomly generating a nonzero floating-point number $d \in [0 \quad 1]$. Then, the chromosome $\mathbf{p}_i$ is chosen if $\hat{q}_{i-1} < d \le \hat{q}_i$, $i = 1, 2, \ldots, pop\_size$, and $\hat{q}_0 = 0$. It can be seen that a chromosome having a larger $f(\mathbf{p}_i)$ will have a higher chance to be selected. Consequently, the best chromosomes will get more offspring, the average will stay and the worst will die off. The process is repeated so that two chromosomes are selected to undergo the genetic operations.

D. Genetic Operations

The genetic operations generate some new chromosomes (offspring) from their parents after the selection process. They include the crossover and the mutation operations.

Crossover

The crossover operation is mainly for exchanging 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. To realize the crossover operation, four chromosomes will first be generated as follows,

$$\mathbf{o}_{\mathbf{s}_c^1} = \begin{bmatrix} o_{s_1^1} & o_{s_2^1} & \cdots & o_{s_{no\_vars}^1} \end{bmatrix} = \frac{\mathbf{p}_1 + \mathbf{p}_2}{2} \tag{19}$$

$$\mathbf{o}_{\mathbf{s}_c^2} = \begin{bmatrix} o_{s_1^2} & o_{s_2^2} & \cdots & o_{s_{no\_vars}^2} \end{bmatrix} = \mathbf{p}_{\max}(1-w) + \max(\mathbf{p}_1, \mathbf{p}_2)w \tag{20}$$

$$\mathbf{o}_{\mathbf{s}_c^3} = \begin{bmatrix} o_{s_1^3} & o_{s_2^3} & \cdots & o_{s_{no\_vars}^3} \end{bmatrix} = \mathbf{p}_{\min}(1-w) + \min(\mathbf{p}_1, \mathbf{p}_2)w \tag{21}$$

$$\mathbf{o}_{\mathbf{s}_c^4} = \begin{bmatrix} o_{s_1^4} & o_{s_2^4} & \cdots & o_{s_{no\_vars}^4} \end{bmatrix} = \frac{(\mathbf{p}_{\max} + \mathbf{p}_{\min})(1-w) + (\mathbf{p}_1 + \mathbf{p}_2)w}{2} \tag{22}$$

$$\mathbf{p}_{\max} = \begin{bmatrix} para_{\max}^1 & para_{\max}^2 & \cdots & para_{\max}^{no\_vars} \end{bmatrix} \tag{23}$$

$$\mathbf{p}_{\min} = \begin{bmatrix} para_{\min}^1 & para_{\min}^2 & \cdots & para_{\min}^{no\_vars} \end{bmatrix} \tag{24}$$

where $w \in [0 \quad 1]$ denotes the weight to be determined by users, $\max(\mathbf{p}_1, \mathbf{p}_2)$ denotes the vector with each element obtained by taking the maximum between the corresponding element of $\mathbf{p}_1$ and $\mathbf{p}_2$. For instance, $\max([1 \quad -2 \quad 3], [2 \quad 3 \quad 1]) = [2 \quad 3 \quad 3]$. Similarly, $\min(\mathbf{p}_1, \mathbf{p}_2)$ gives a vector by taking the minimum value. For instance, $\min([1 \quad -2 \quad 3], [2 \quad 3 \quad 1]) = [1 \quad -2 \quad 1]$. Among $\mathbf{o}_{\mathbf{s}_c^1}$ to $\mathbf{o}_{\mathbf{s}_c^4}$, the one with the largest fitness value is used as the offspring. The offspring $\mathbf{o}_\mathbf{s}$ is defined as,

$$\mathbf{o}_\mathbf{s} \equiv \begin{bmatrix} o_{s_1} & o_{s_2} & \cdots & o_{s_{no\_vars}} \end{bmatrix} = \mathbf{o}_{\mathbf{s}_c^{i_{os}}} \tag{25}$$

where i_{os} denotes the index i that gives the maximum value of $f(\mathbf{o}_{\mathbf{s}_c^i})$, i = 1, 2, 3 ,4.

If the crossover operation can provide a good offspring, a higher fitness value can be reached in less iteration. In general, two-point crossover, multipoint crossover, arithmetic crossover or heuristic crossover can be used to realize the crossover operation [9-11]. However, the offspring generated by these methods may not be better than that of our approach. As seen from (19) to (22), the offspring spreads over the domain: (19) and (22) will move the offspring near the centre region of the concerned domain (as w in (22) approaches 1, $\mathbf{o}_{\mathbf{s}_c^4}$ approaches $\frac{\mathbf{p}_1 + \mathbf{p}_2}{2}$), and (20) and (21) will move the offspring near the domain boundary (as w in (20) and (21) approaches 0, $\mathbf{o}_{\mathbf{s}_c^2}$ and $\mathbf{o}_{\mathbf{s}_c^3}$ approaches $\mathbf{p}_{\max}$ and $\mathbf{p}_{\min}$ respectively).

Mutation

The offspring (25) may then undergo a fuzzy mutation operation, which changes the genes of the offspring chromosomes. Every gene of $\mathbf{o}_\mathbf{s}$ of (25) will have a chance to mutate governed by a probability of mutation, $p_m \in [0 \quad 1]$, which is defined by the user. 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. The gene of the offspring of (25) is then mutated by:

$$\hat{o}_{s_k} = \begin{cases} o_{s_k} + \Delta o_{s_k}^U \text{ if } f(\mathbf{o_s} + \Delta \mathbf{o}_{\mathbf{s}_k}^U) \geq f(\mathbf{o_s} - \Delta \mathbf{o}_{\mathbf{s}_k}^L) \\ o_{s_k} - \Delta o_{s_k}^L \text{ if } f(\mathbf{o_s} + \Delta \mathbf{o}_{\mathbf{s}_k}^U) < f(\mathbf{o_s} - \Delta \mathbf{o}_{\mathbf{s}_k}^L) \end{cases}, k = 1, 2, \ldots, no\_vars \quad (26)$$

where

$$\Delta o_{s_k}^U = r^{\frac{1}{w_{m_k}}} \left(para_{\max}^k - o_{s_k} \right) \quad (27)$$

$$\Delta o_{s_k}^L = r^{\frac{1}{w_{m_k}}} \left(o_{s_k} - para_{\min}^k \right) \quad (28)$$

$$\Delta \mathbf{o}_{\mathbf{s}_k}^U = \begin{bmatrix} 0 & 0 & \cdots & \Delta o_{s_k}^U & \cdots & 0 \end{bmatrix} \quad (29)$$

$$\Delta \mathbf{o}_{\mathbf{s}_k}^L = \begin{bmatrix} 0 & 0 & \cdots & \Delta o_{s_k}^L & \cdots & 0 \end{bmatrix} \quad (30)$$

$r \in [0 \quad 1]$ 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 weight w_{m_k} is determined by two factors: the rate of change of the fitness with respect to o_{s_k}, i.e. $\left| \frac{\partial f(\mathbf{o_s})}{\partial o_{s_k}} \right|$, and the ratio of the current iteration number to the total number of iterations $\frac{\tau}{T}$. A large value of $\left| \frac{\partial f(\mathbf{o_s})}{\partial o_{s_k}} \right|$ implies the gene o_{s_k} has a large search space. A large weight w_{m_k} is thus necessary when $\left| \frac{\partial f(\mathbf{o_s})}{\partial o_{s_k}} \right|$ is large in order to obtain a significant mutation (large $\Delta o_{s_k}^U$ or $\Delta o_{s_k}^L$). On the other hand, the value of $\frac{\tau}{T}$ affects the fine-tuning of the optimum. The value of weight w_{m_k} should approach 0 as $\frac{\tau}{T}$ increases in order to reduce the significance of the mutation. Based on these two factors, the weight w_{m_k} is governed by the following fuzzy rules:

Rule j: IF $\left| \frac{\partial f(\mathbf{o_s})}{\partial o_{s_k}} \right|$ is N_1^j AND $\frac{\tau}{T}$ is N_2^j THEN $w_{m_k} = w_{s_j}$, j = 1, 2, …, r_m; k = 1, 2, …, *no_vars* (31)

where N_1^j and N_2^j are fuzzy terms of rule j, r_m denotes the number of rules, $w_{s_j} \in [0 \quad 1]$ is a singleton to be determined. The final value of w_{m_k} is given by

$$w_{m_k} = r_s \sum_{j=1}^{r_m} m_j w_{s_j},\ k = 1, 2, \ldots, no\_vars \tag{32}$$

where

$$m_j = \frac{\mu_{\mathrm{N}_1^j}\left(\left|\frac{\partial f(\mathbf{o_s})}{\partial o_{s_k}}\right|\right) \times \mu_{\mathrm{N}_2^j}\left(\frac{\tau}{T}\right)}{\sum_{k=1}^{r_m}\left(\mu_{\mathrm{N}_1^k}\left(\left|\frac{\partial f(\mathbf{o_s})}{\partial o_{s_k}}\right|\right) \times \mu_{\mathrm{N}_2^k}\left(\frac{\tau}{T}\right)\right)} \tag{33}$$

$$\sum_{j=1}^{r_m} m_j = 1,\ m_j \in [0,\ \ 1] \text{ for all } j \tag{34}$$

$\mu_{\mathrm{N}_1^j}\left(\left|\frac{\partial f(\mathbf{o_s})}{\partial o_{s_k}}\right|\right)$ and $\mu_{\mathrm{N}_2^j}\left(\frac{\tau}{T}\right)$ are the membership functions corresponding to N_1^j and N_2^j respectively, $r_s \in [0\ \ 1]$ is a gain affecting the searching area. A maximum searching area is provided when r_s=1.

E. Reproduction

After going through the fuzzy mutation process, the new offspring will be evaluated using the fitness function of (16). 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 $p_a \in [0\ \ 1]$, which is the probability of acceptance defined by the user. 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.

After the operations of selection, crossover and fuzzy mutation, a new population is generated. This new population will repeat the same process to produce another offspring. Such an iterative process can be terminated when a defined condition is met, e.g. a sufficiently large number of iteration has been reached.

F. Choosing the parameters

The GA process is effectively seeking a balance between the exploration of new regions and the exploitation of already sampled regions in the search space. This balance, which critically affects the performance of the GA, is governed by the right choices of control parameters: the probability of fuzzy mutation (p_m), the probability of acceptance (p_a), the population size (*pop_size*), the weight in the crossover operation (w) and the gain affecting the searching area (r_s). Some views about these parameters are included as follows:

- Increasing p_m tends to transform the genetic search into a random search. This probability gives an expected number ($p_m \times no\_vars$) of genes that undergo the mutation. When $p_m = 1$, all genes will mutate. The value of p_m therefore depends on the desirable number of genes that undergo the mutation operation.
- Increasing p_a will increase the chance that a poor offspring joins the population. This reduces the probability that the GA prematurely converges to a local optimum. From experience, a p_a of 0.1 is a good enough choice for many optimization problems.
- Increasing *pop_size* will increase the diversity of the search space, and reduce the probability that the GA prematurely converges to a local optimum. However, it also increases the time required for the population to converge to the optimal region in the search space. From experience, a population size of 10 is an acceptable choice.
- Changing the value of the weight *w* will change the characteristics of the crossover operations. It is chosen by trial and error, which varies in different optimization problems.
- Increasing the gain r_s will increase the search space (i.e. increase $\Delta o_{s_k}^U$ or $\Delta o_{s_k}^L$). It is chosen by trial and error.
- In fuzzy mutation, the value of w_{s_j} are determined by $\mu_{\mathrm{N}_1^j}\left(\left|\frac{\partial f(\mathbf{o}_s)}{\partial o_{s_k}}\right|\right)$ and $\mu_{\mathrm{N}_2^j}\left(\frac{\tau}{T}\right)$. When the value of the $\mu_{\mathrm{N}_2^j}\left(\frac{\tau}{T}\right)$ is large, which implies the current iteration number is approaching the total number of iteration, a fine-tuning processing is necessary. Then, the value of w_{s_j} should be set small. On the other hand, when the value of the $\mu_{\mathrm{N}_1^j}\left(\left|\frac{\partial f(\mathbf{o}_s)}{\partial o_{s_k}}\right|\right)$ is large, which implies the rate of change of the fitness with respect to the gene is large, a large w_{s_j} should be used. An example of the relationship between them is shown in Fig. 9, where L represents "Low", M represents "Medium", and H represents "High".

IV. Benchmark Test Functions

A suite of eight benchmark test functions [19-20] is used to test the performance of the improved GA. Many kinds of optimization problems are covered by these benchmark test functions. They are divided into three main categories: unimodal functions, multimodal functions with only a few local minima, and multimodal functions with many local minima. Functions f_1 to f_4 are unimodal functions. Function f_1 is a sphere model, which is probably the most widely used test

function. It is smooth and symmetric. The performance on this function is a measure of the general efficiency of an algorithm. Function f_2 is a step function, which is a representative of flat surfaces. Flat surfaces are obstacles for optimization algorithms because they do not give any information about the search direction. Unless the algorithm has a variable step size, it can get stuck in one of the flat surfaces. Function f_3 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. Many algorithms that do not do well in this function are due to the noisy data. Function f_4 corresponds to the Schwefel's problem. Each parameter affects the overall performance deeply. Functions f_5 to f_6 are multimodal functions with only a few local minima, and the dimension of each function is small. Function f_5 is a Shekel's foxholes function. Function f_6 is a Goldstein-Price's function. Functions f_7 to f_8 are multimodal functions with many local minima, and the dimension of each function is relatively large. Function f_7 is a generalized Rastrigin's function. Function f_8 is a generalized Griewank function. The eight test functions are defined as follows,

$$f_1(\mathbf{x}) = \sum_{i=1}^{n} x_i^2 \text{ , } -5.12 \le x_i \le 5.12 \tag{35}$$

where $n = 30$ and min(f_1)= $f_1(\mathbf{0}) = 0$, The fitness function for f_1 is defined as,

$$fitness = \frac{1}{1 + f_1(\mathbf{x})}.$$

$$f_2(\mathbf{x}) = \sum_{i=1}^{n} floor((x_i + 0.5)^2), \ -5.12 \le x_i \le 5.12 \tag{36}$$

where $floor(\cdot)$ is obtained by rounding down the argument to the nearest smaller integer, $n = 30$ and min(f_2)= $f_2(\mathbf{0}) = 0$. The fitness function for f_2 is defined as,

$$fitness = \frac{1}{1 + f_2(\mathbf{x})}.$$

$$f_3(\mathbf{x}) = \sum_{i=1}^{n} i x_i^4 + random[0, \ 1), \ -1.28 \le x_i \le 1.28 \tag{37}$$

where $random[0, \ 1)$ will give a randomly generated floating-point number between 0 and 1, $n = 30$ and min(f_3)= $f_3(\mathbf{0}) = 0$. The fitness function for f_3 is defined as,

$$fitness = \frac{1}{1 + f_3(\mathbf{x})}.$$

$$f_4(\mathbf{x}) = \max_i \{|x_i|, 1 \le i \le n\}, -100 \le x_i \le 100 \tag{38}$$

where $n = 30$ and min(f_4)= $f_4(\mathbf{0}) = 0$. The fitness function for f_4 is defined as,

$$fitness = \frac{1}{1 + f_4(\mathbf{x})}.$$

$$f_5(\mathbf{x}) = \left[\frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2} (x_i - a_{ij})^6} \right]^{-1}, \quad -65.536 \le x_i \le 65.536 \tag{39}$$

where $a_{ij} = \begin{pmatrix} -32 & -16 & 0 & 16 & 32 & -32 & \cdots & 0 & 16 & 32 \\ -32 & -32 & -32 & -32 & -32 & -16 & \cdots & 32 & 32 & 32 \end{pmatrix}$, $\min(f_5) = f_5([-32, \quad 32]) \approx 1$.

The fitness function for f_5 is defined as,

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

$$f_6(\mathbf{x}) = \left[1 + (x_1 + x_2 + 1)^2 \cdot (19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)\right] \cdot \left[30 + (2x_1 - 3x_2)^2 \cdot (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)\right],$$

$$-2 \le x_1, x_2 \le 2 \tag{40}$$

where $\min(f_6) = f_6([0, \quad -1]) = 3$. The fitness function for f_6 is defined as,

$$fitness = \frac{1}{-2 + f_6(\mathbf{x})}.$$

$$f_7(\mathbf{x}) = \sum_{i=1}^{n} \left[x_i^2 - 10\cos(2\pi x_i) + 10\right], \quad -5.12 \le x_i \le 5.12 \tag{41}$$

where $n = 30$ and $\min(f_7) = f_7(\mathbf{0}) = 0$. The fitness function for f_7 is defined as,

$$fitness = \frac{1}{1 + f_7(\mathbf{x})}.$$

$$f_8(\mathbf{x}) = \frac{1}{4000} \sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1, \quad -600 \le x_i \le 600 \tag{42}$$

where $n = 30$ and $\min(f_8) = f_8(\mathbf{0}) = 0$. The fitness function for f_8 is defined as,

$$fitness = \frac{1}{1 + f_8(\mathbf{x})}.$$

The improved GA goes through the eight test functions. The results are compared with those obtained by the traditional GA with arithmetic, heuristic or one-point crossover and non-uniform mutation, depending on which one gives the best result in each iteration. For each test function, the population size is 10. Fig. 8 shows the membership functions for the fuzzy mutation. Considering the discussion on the mutation operation in Section III, the fuzzy rule table is designed and shown in Fig. 9. From this table, the output singleton values for the fuzzy mutation are set as 0.2 for L, 0.7 for M, and 1.0 for H. All the simulation results are averaged ones out of 50 runs. The best fitness value is equal to one (*fitness* = 1). The probability of acceptance (p_a) is set at 0.1 for all test functions. The simulation results (averaged, maximum and minimum fitness values and the standard deviation) for f_1

to f_8 obtained by the improved GA and the traditional GA, and the number of iteration are tabulated in Table I. The parameter settings for them are tabulated in Table II. All settings are chosen by trial and error. The convergence of f_1 to f_8 obtained by the improved GA and the traditional GA are shown in Fig. 10. It can be seen that the improved GA performs more efficiently, gives smaller standard deviations, and provides a faster convergence than the traditional GA.

V. Training Fuzzy-Tuned Neural Networks Using Improved GA

The proposed fuzzy-tuned neural network can learn the input-output relationship of an application using the improved GA. Let the input-output relationship be described by,

$$\mathbf{y}^d(t) = \mathbf{g}\left(\mathbf{z}^d(t)\right),\ t = 1, 2, \ldots, n_d \tag{43}$$

where $\mathbf{z}^d(t) = \begin{bmatrix} z_1^d(t) & z_2^d(t) & \cdots & z_{n_{in}}^d(t) \end{bmatrix}$ and $\mathbf{y}^d(t) = \begin{bmatrix} y_1^d(t) & y_2^d(t) & \cdots & y_{n_{out}}^d(t) \end{bmatrix}$ are the given inputs and the desired outputs of an unknown nonlinear function $\mathbf{g}(\cdot)$ respectively, n_d denotes the number of input-output data pairs, the fitness function is defined as,

$$fitness = \frac{1}{1+err} \tag{44}$$

$$err = \frac{\sum_{t=1}^{n_d}\sum_{k=1}^{n_{out}}\left|y_k^d(t) - y_k(t)\right|}{n_d n_{out}} \tag{45}$$

The objective is to maximize the fitness value of (44) (minimize *err* of (45)) using the improved GA by setting the chromosome to be $\begin{bmatrix} v_{ik} & m_s^k & \sigma_s^k & w_{kl} & m_o^l & \sigma_o^l & \bar{z}_i^h & \sigma_i^h & g_{hj} \end{bmatrix}$ for all *h*, *i*, *j*, *k*, *l*. In this paper, $v_{ik}, w_{kl}, g_{hj} \in \begin{bmatrix} -1 & 1 \end{bmatrix}$, $m_s^k, m_o^l, \bar{z}_i \in \begin{bmatrix} -0.5 & 0.5 \end{bmatrix}$ and $\sigma_s^k, \sigma_o^l, \sigma_i \in \begin{bmatrix} 0.01 & 0.5 \end{bmatrix}$. The range of the fitness value of (44) is [0, 1].

VI. Application Examples

Three application examples will be given in this section to illustrate the merits of the proposed neural networks tuned by the improved GA. They are the XOR problem, forecasting the sunspot number, and pattern recognition.

A. XOR Problem

A 3-input XOR function, which is not linearly separable, has the following input-output relationship:

$$\begin{aligned} &(-1,-1,-1), (-1,+1,+1), (+1,-1,+1), (+1,+1,-1) \rightarrow -1 \\ &(-1,-1,+1), (-1,+1,-1), (+1,-1,-1), (+1,+1,+1) \rightarrow +1 \end{aligned} \tag{46}$$

The three inputs of the proposed neural network are defined as $x_i(t)$, i = 1, 2, 3, and $y(t)$ is the network output. The number of hidden nodes (n_h) in the modified neural network is set at 2 and the number of membership functions for each input (m) in the NFN is set at 3. The total number of parameter is 44. Referring to (7), the input-output relationship of the proposed neural network to realize the 3-input XOR function is given by,

$$y(t) = net_o^1(\sum_{k=1}^{2} net_d^k(net_s^k(\sum_{i=1}^{3} x_i v_{ik}))w_k) \tag{47}$$

The fitness function is defined as follows,

$$fitness = \frac{1}{1+err} \tag{48}$$

$$err = \frac{\sum_{t=1}^{8} \left| y^d(t) - y\ (t) \right|}{8} \tag{49}$$

The improved GA is employed to tune the parameters of the proposed fuzzy-tuned neural network. The objective is to maximize the fitness function of (47). The best fitness value is 1 and the worst one is 0. The population size used for the GA is 10. The chromosomes used for the improved GA are $\left[v_{ik} \quad m_s^k \quad \sigma_s^k \quad w_{kl} \quad m_o^l \quad \sigma_o^l \quad \bar{z}_i^h \quad \sigma_i^h \quad g_{hj} \right]$ for all h, i, j, k, l. The initial values of the parameters are randomly generated. For comparison purpose, the proposed network trained by the traditional GA, traditional feed-forward neural networks trained by the proposed and the traditional GAs, are also used to realize the XOR function. The number of hidden nodes of the traditional neural network is 8 so that the total number of trained parameters is 41. For all approaches, the number of iteration to train the neural networks is 1000 and the results are averaged ones out of 30 runs. For the improved GA, the probability of acceptance is set at 0.1 for both networks. The parameter settings for all approaches are tabulated in Table III. Results of the proposed and traditional neural networks trained by the proposed and traditional GAs are tabulated in Table IV. The training results for the proposed network trained by the improved GA and the traditional GA are shown in Fig. 11. The training results for the traditional neural network trained by the improved GA and the traditional GA are shown in Fig. 12. After training, the network outputs based on the proposed approach (the proposed network trained with the improved GA) and the traditional approach (the traditional network trained with traditional GA) are compared with the desired output, and are tabulated in Table V. It can be seen from Tables IV and V, Figs. 11 and 12 that the performance of the proposed approach is better in terms of the fitness value and the convergence rate.

B. Forecasting of the Sunspot Number

The sunspot numbers [3, 8] from 1700 to 1980 are shown in Fig. 13. The cycles generated are non-linear, non-stationary, and non-Gaussian which are difficult to model and predict. We use the proposed fuzzy-tuned neural network for the sunspot number forecasting. The inputs, x_i, of the proposed network are defined as $x_1(t) = y^d(t-1)$, $x_2(t) = y^d(t-2)$ and $x_3(t) = y^d(t-3)$ where t denotes the year and $y^d(t)$ is the sunspot number at the year t. The sunspot numbers of the first 180 years (i.e. $1705 \le t \le 1884$) are used to train the proposed neural network. In this network, the number of hidden nodes (n_h) in the modified neural network is set at 3 and the number of membership functions for each input (m) in the NFN is set at 2. Then, the total number of parameters is 44. Referring to (7), the input-output relationship of the proposed network is governed by,

$$y(t) = net_o^1\left(\sum_{k=1}^{3} net_d^k\left(net_s^k\left(\sum_{i=1}^{3} x_i v_{ik}\right)\right) w_k\right) \tag{50}$$

The fitness function is defined as follows,

$$fitness = \frac{1}{1+err} \tag{51}$$

$$err = \sum_{t=1705}^{1884} \frac{\left|y^d(t) - y(t)\right|}{180} \tag{52}$$

The improved GA is employed to tune the parameters of the proposed fuzzy-tuned neural network of (50). The objective is to maximize the fitness function of (51). The population size used for the GA is 10. The chromosomes used for the improved GA are $\left[v_{ik} \quad m_s^k \quad \sigma_s^k \quad w_{kl} \quad m_o^l \quad \sigma_o^l \quad \bar{z}_i^h \quad \sigma_i^h \quad g_{hj}\right]$ for all h, i, j, k, l. The initial values of the parameters of the neural network are randomly generated. For comparison purpose, the proposed network trained by the traditional GA, and traditional feed-forward neural networks trained by the proposed and the traditional GAs, are also applied in forecasting of the sunspot number. The number of hidden node of the traditional neural network is 9 so that the total number of parameters is 46. For all approaches, the number of iteration to train the neural network is 1000 and the results are averaged ones out of 25 runs. For the improved GA, the probability of acceptance is set at 0.1 for both networks. The parameter settings for all approaches are tabulated in Table VI.

The trained network is used to forecast the sunspot number during the years 1885-1979. Fig. 14 shows the results of the forecasting using the proposed network trained with the improved GA (dashed lines) and the traditional network trained with the traditional GA (dotted lines), as compared with the actual sunspot numbers (solid lines). The average fitness value, the average training error (governed by (52)) and the average forecasting error (governed by $\sum_{t=1885}^{1980} \frac{\left|y^d(t) - y\ (t)\right|}{96}$) are tabulated in

Table VII. It can be observed from Table VII that our approach performs better than the traditional approaches. The training error and the forecasting error of the proposed approach in terms of mean absolute error (MAE) are 9.40 and 12.34 respectively.

C. Pattern Recognition

An application on hand-written graffiti pattern recognition will be presented. Numbers are assigned to pixels on a two-dimensional plane, and 10 numbers are used to characterize the 10 uniformly sampled pixels of a given graffiti. A 10-input-3-output network is used. The ten inputs nodes, x_i, i = 1, 2, …, 10, are the numbers representing the graffiti pattern. Three standard patterns are to be recognized: rectangle, triangle and straight line (Fig. 15). We use 300 sets of 10 normalized sample points for each pattern to train the neural network. Hence, we have 900 sets of data for doing the training. The three outputs, $y_l(t)$, l = 1, 2, 3, indicates the similarity between the input pattern and the three standard patterns respectively. The desired outputs of the pattern recognition system are $\mathbf{y}^d(t) = [1 \quad 0 \quad 0]$, $\mathbf{y}^d(t) = [0 \quad 1 \quad 0]$ and $\mathbf{y}^d(t) = [0 \quad 0 \quad 1]$ for rectangles, triangles and straight lines respectively. After training, a larger value of $y_l(t)$ implies that the input pattern matches more closely to the corresponding graffiti pattern. For instance, a large value of $y_1(t)$ implies that the input pattern is near to a rectangle. Referring to (7), the input-output relationship of the proposed network used for the pattern recognition is governed by,

$$y_l(t) = net_o^l\left(\sum_{k=1}^{n_h} net_d^k\left(net_s^k\left(\sum_{i=1}^{10} x_i v_{ik}\right)\right) w_{kl}\right), \; l = 1, 2, 3. \tag{53}$$

The improved GA is employed to tune the parameters of the proposed network of (53). The fitness function is defined as follows,

$$fitness = \frac{1}{1 + err} \tag{54}$$

$$err = \frac{\sum_{t=1}^{300}\sum_{k=1}^{3}\left(\left(\frac{y_k(t)}{\|\mathbf{y}(t)\|}\right)^2 - \left(\frac{y_k^d(t)}{\|\mathbf{y}^d(t)\|}\right)^2\right)}{300 \times 3} \tag{55}$$

The value of *err* indicates the mean square error (MSE) of the recognition system. In this network, the number of hidden nodes (n_h) in the modified neural network is set at 6 and the number of membership functions for each input (m) in the NFN is set at 4, which are chosen by trial and error through experiments for good performance. Then, the total number of parameter is 224. The improved GA is employed to tune the parameters of the proposed fuzzy-tuned neural network of (53). The population size is 10. The chromosomes are $\left[v_{ik} \quad m_s^k \quad \sigma_s^k \quad w_{kl} \quad m_o^l \quad \sigma_o^l \quad \bar{z}_i^h \quad \sigma_i^h \quad g_{hj}\right]$ for all h, i, j, k, l. The initial values of the parameters of the neural network are randomly generated. For comparison,

the proposed network trained by the traditional GA, and a traditional network trained by the proposed and the traditional GAs, are also used to recognize the patterns. The number of hidden node of the traditional neural network is 16 so that the total number of parameter is 227. For all approaches, the number of iteration to train the neural network is 2000, and the results are averaged ones out of 25 runs. For the improved GA, the probability of acceptance is set at 0.1 for both networks. The parameter settings for all approaches are tabulated in Table VIII.

After training, we use 600 (200×3) sets of data for testing. The results are tabulated in Table IX. From this Table, it can be seen that the average training error (governed by (55)) and average forecasting error (governed by $err = \dfrac{\sum_{t=1}^{200}\sum_{k=1}^{3}\left(\left(\dfrac{y_k(t)}{\|\mathbf{y}(t)\|}\right)^2 - \left(\dfrac{y_k^d(t)}{\|\mathbf{y}^d(t)\|}\right)^2\right)}{200\times 3}$) of the proposed network are smaller. The recognition accuracy is governed by

$$\text{Accuracy} = \left(\frac{n_{recognized}}{n_{testing}}\right)\times 100\%. \tag{56}$$

where $n_{testing}$ and $n_{recognized}$ are the total number of testing patterns and the number of successfully recognized patterns respectively. The accuracy of the proposed network is good.

VII. Conclusion

A fuzzy-tuned neural network has been proposed. It consists of a modified neural network and a neural-fuzzy network. A neuron model with two activation functions has been introduced in the hidden layer of the modified neural network. Some parameters of the neuron are tuned by the neural-fuzzy network. By employing this neuron model and the network structure, the performance of the proposed network is found to be better than that of the traditional neural network. The parameters of the proposed network can be tuned by an improved GA, in which modified genetic operations have been introduced. By using the proposed crossover operation, the offspring spreads over the domain so that the probability of reproducing good offspring is increased. The proposed fuzzy mutation incorporates human knowledge on mutation into fuzzy rules. Based on some benchmark test functions, it has been shown that the improved GA performs more efficiently and provides a faster convergence than the traditional GA. Examples on implementing XOR function, sunspot forecasting and pattern recognition have been given. The performance of the proposed network in these examples is good.

Acknowledgement

The work described in this paper was substantially supported by a grant from the Hong Kong Polytechnic University (PhD Student Account Code RG9T).

References

[1] M. Brown and C. Harris, *Neural Fuzzy Adaptive Modeling and Control.* Prentice Hall, 1994.

[2] A.E. Bryon and Y.C. Ho, *Applied Optimal Control*. Blaisdell, 1969.

[3] F.H.F. Leung, H.K. Lam, S.H. Ling, and P.K.S. Tam, "Tuning of the structure and parameters of neural network using an improved genetic algorithm," *IEEE Trans. Neural Networks*, vol.14, no. 1, pp.79-88, Jan. 2003.

[4] S.H. Ling, F.H.F. Leung, H.K. Lam, Y.S. Lee, and P.K.S. Tam, "A novel GA-based neural network for short-term load forecasting," *IEEE Trans. Industrial Electronics*, vol. 50, no. 4, pp. 793-799, Aug. 2003.

[5] S.H. Ling, H.K. Lam, F.H.F. Leung, and P.K.S Tam, "Learning of neural network parameters using a fuzzy genetic algorithm," in *Proc.2002 Congress on Evolutionary Computation (CEC 2002), World Congress on Computational Intelligence (WCCI 2002)*, Honolulu, Hawaii, May 12-17, 2002, pp. 1928 - 1933.

[6] H.K. Lam, F.H.F. Leung and P.K.S. Tam, "Design and stability analysis of fuzzy model based nonlinear controller for nonlinear systems using genetic algorithm," *IEEE Trans. Syst., Man and Cybern, Part B: Cybernetics*, vol. 33, no. 2, pp. 250-257, Feb. 2003

[7] H.K. Lam, S.H. Ling, F.H.F. Leung, and P.K.S. Tam, "Optimal and stable fuzzy controllers for nonlinear systems subject to parameter uncertainties using genetic algorithm," in *Proc. 10th IEEE Int. Conf. Fuzzy Syst. (FUZZ-IEEE 2001)*, Melbourne, Australia, 2-5 December 2001, pp. 908-911.

[8] M. Li, K. Mechrotra, C. Mohan, and S. Ranka, "Sunspot numbers forecasting using neural network," in *Proc. 5th IEEE Int. Symp. Intelligent Control, 1990*, pp.524-528.

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

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

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

[12] 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.

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

[15] 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.

[16] M. Srinivas and L.M. Patnaik, "Genetic algorithms: a survey," *IEEE Computer*, vol. 27, issue 6, pp. 17-26, June 1994.

[17] Lawrence Davis, *Handbook of genetic algorithms.* New York: Van Nostrand Reinhold, 1991.

[18] J.D. Schaffer, D. Whitley, and L.J. Eshelman "Combinations of genetic algorithms and neural networks: a survey of the state of the art," in *Proc. Int. Workshop Combinations of Genetic Algorithms and Neural Networks, (COGANN-92)*, 1992, pp 1-37.

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

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

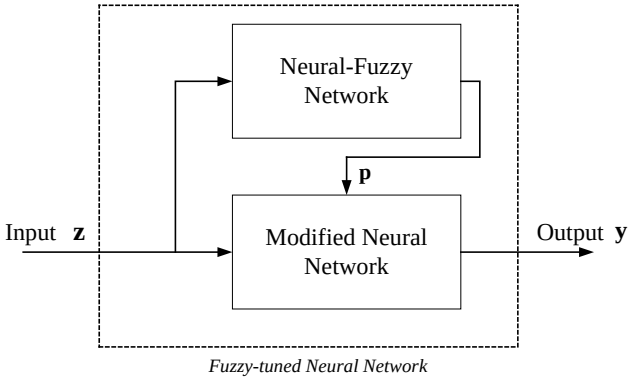


Fig. 1. Block diagram of the proposed fuzzy-tuned neural network.

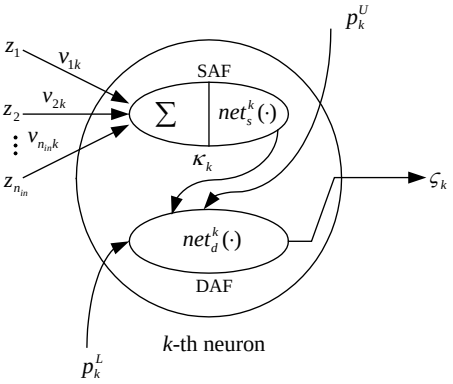


Fig. 2. Proposed modified neuron.

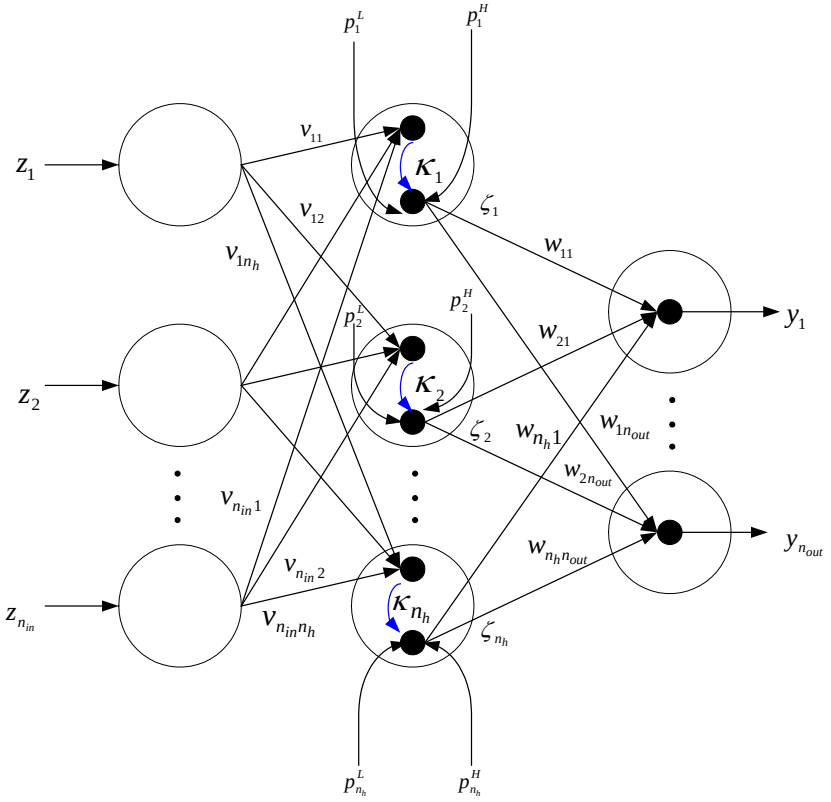
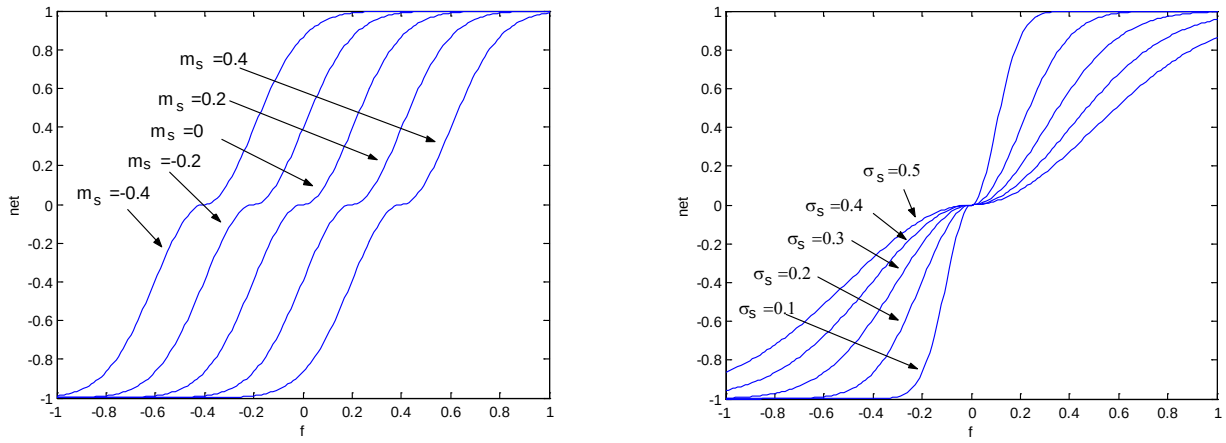
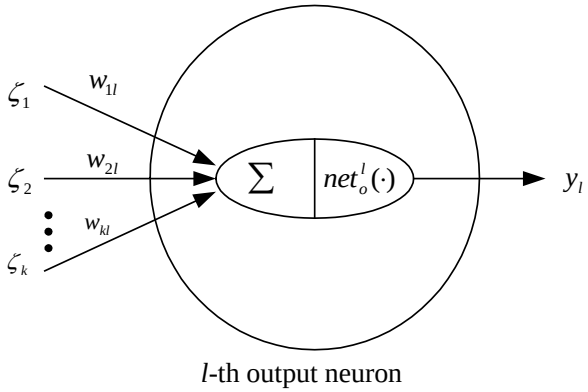


Fig. 3. Connection of the modified neural network.



(a) (b)

Fig. 4. Sample activation functions of the proposed neuron: (a) σ_s = 0.2, (b) m_s = 0.



l-th output neuron

Fig. 5. Model of the proposed output neuron.

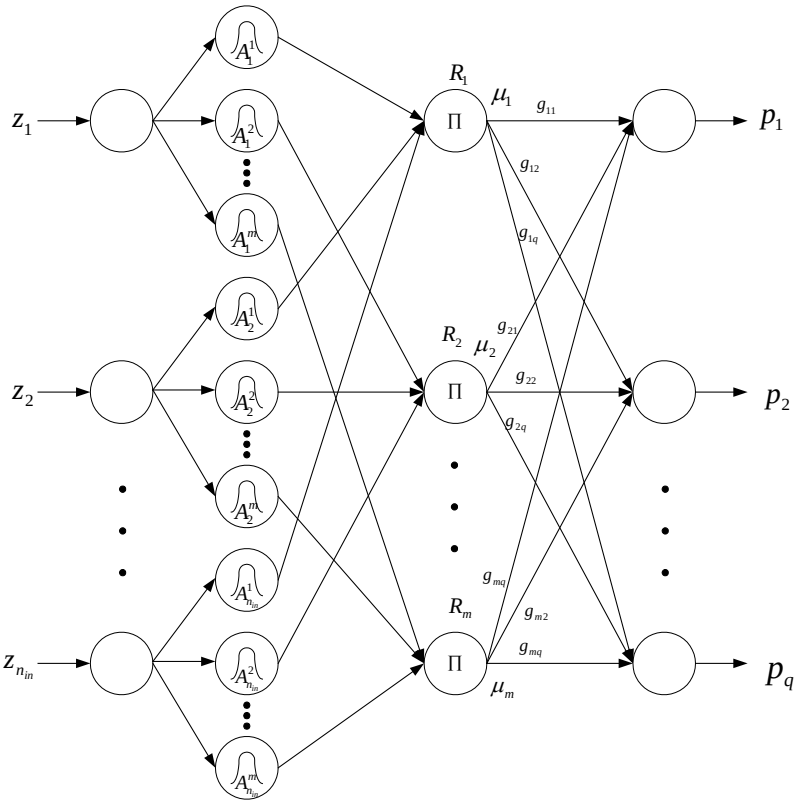


Fig. 6. Neural-fuzzy network model.

Procedure of the improved GA
begin
$\tau \rightarrow 0$ // τ: iteration number
initialize $P(\tau)$ // $P(\tau)$: population for iteration τ
evaluate $f(P(\tau))$ // $f(P(\tau))$:fitness function
while (not termination condition) **do**
begin
$\tau \rightarrow \tau+1$
select 2 parents $\mathbf{p}_1$ and $\mathbf{p}_2$ from $P(\tau-1)$
perform *crossover* operation according to equations (19) to (25)
offspring $\mathbf{o}_s$ obtained by crossover operation
perform *fuzzy mutation* operation according to equations (26) to (34) with p_m // p_m :probability of mutation
offspring $\mathbf{o}_s$ obtained by fuzzy mutation operation
// reproduce a new $P(\tau)$
if random number < p_a // p_a: probability of acceptance
$\mathbf{o}_s$ replaces the chromosome with the smallest value
else if $f(\mathbf{o}_s)$ > smallest fitness value in the $P(\tau-1)$
$\mathbf{o}_s$ replaces the chromosome with the smallest value
end
evaluate $P(\tau)$
end
end

Fig. 7. Improved GA process.

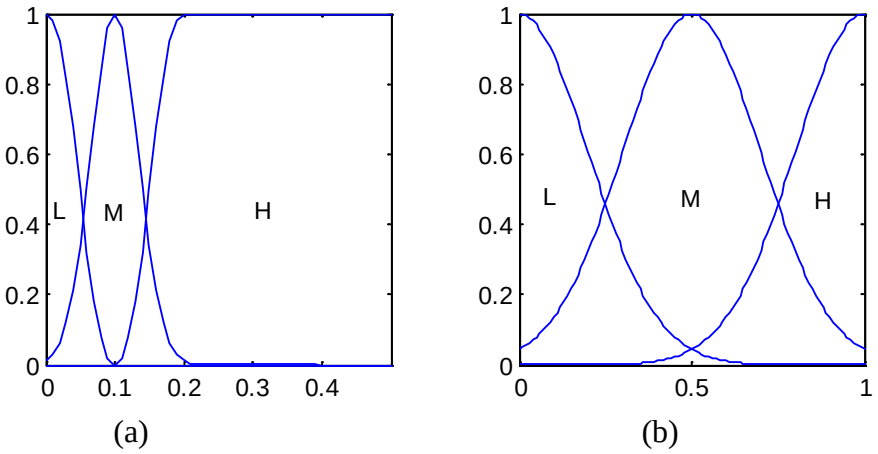
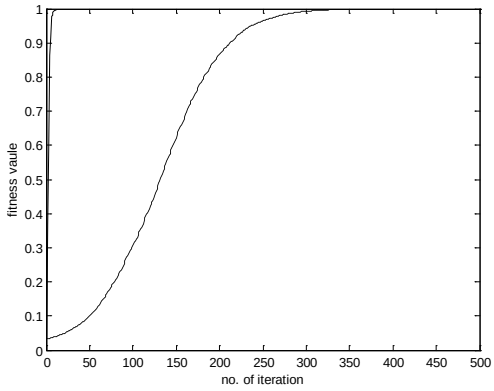


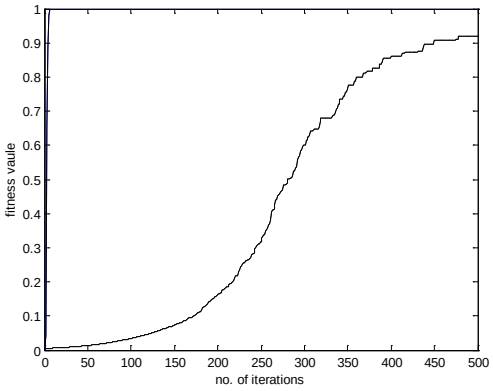
Fig. 8. Membership functions for fuzzy mutation operation: (a) *x*-axis: $\left|\frac{\partial f(\mathbf{o}_s)}{\partial o_{s_k}}\right|$, *y*-axis: $\mu_{N_1}\left(\left|\frac{\partial f(\mathbf{o}_s)}{\partial o_{s_k}}\right|\right)$ and (b) *x*-axis: $\frac{\tau}{T}$, *y*-axis: $\mu_{N_2}\left(\frac{\tau}{T}\right)$.

w_s : $\left\|\frac{\partial f(\mathbf{o}_s)}{\partial o_{s_k}}\right\|$ / $\frac{\tau}{T}$	L	M	H
L	H	H	H
M	M	H	H
H	L	M	H

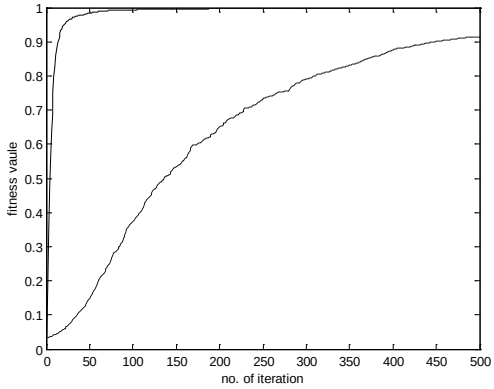
Fig. 9. Rule tables for fuzzy mutation.



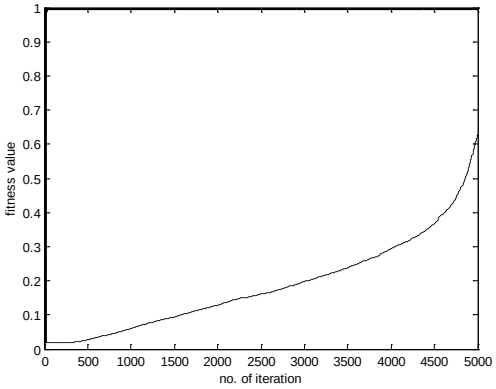
(a). f_1 .



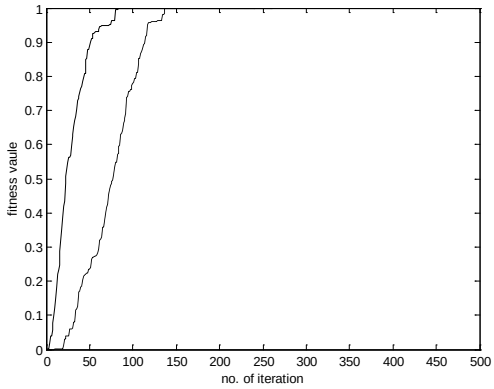
(b). f_2 .



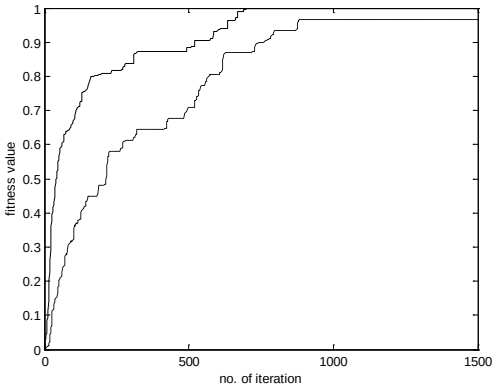
(c). f_3 .



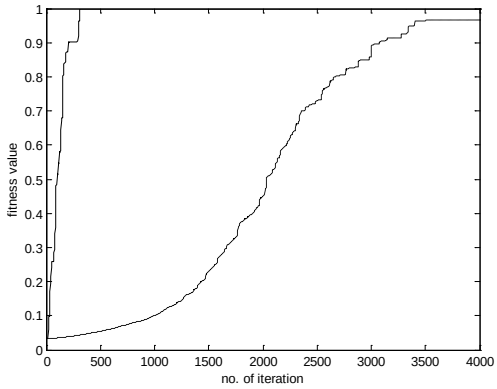
(d). f_4 .



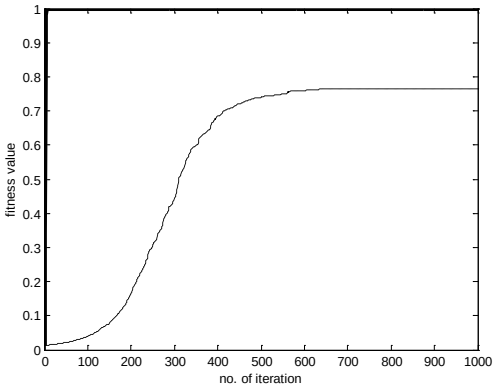
(e). f_5 .



(f). f_6 .



(g). f_7 .



(h). f_8 .

Fig.10. Averaged fitness value of the test functions f_1 to f_8 obtained by the improved GA (solid line) and traditional GA (dotted line).

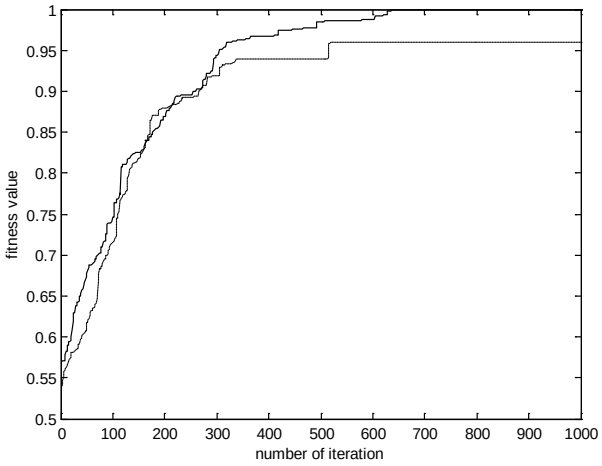


Fig. 11. The training results of the proposed network trained by the improved GA (solid line) and the traditional GA (dotted line) on realizing the XOR function.

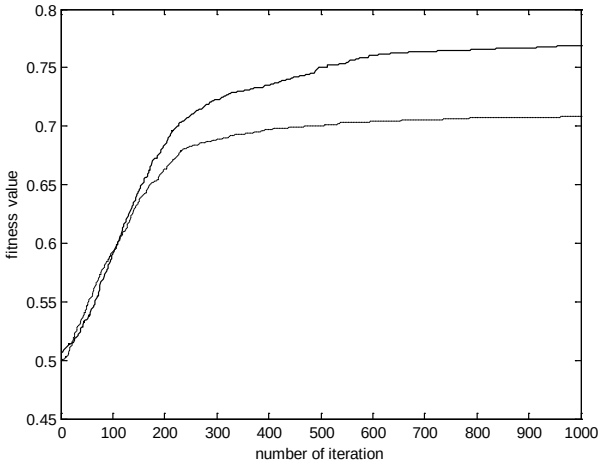


Fig. 12. The training results of the traditional neural network trained by the improved GA (solid line) and the traditional GA (dotted line) on realizing the XOR function.

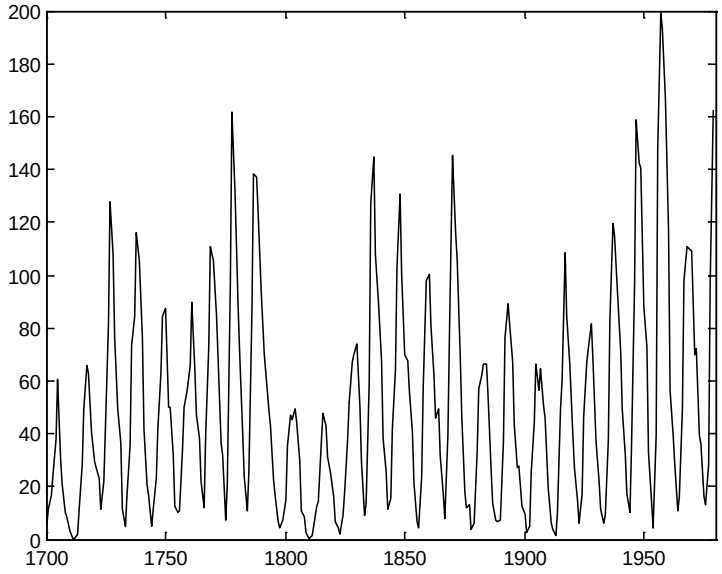


Fig. 13. Sunspot numbers from year 1700 to 1980.

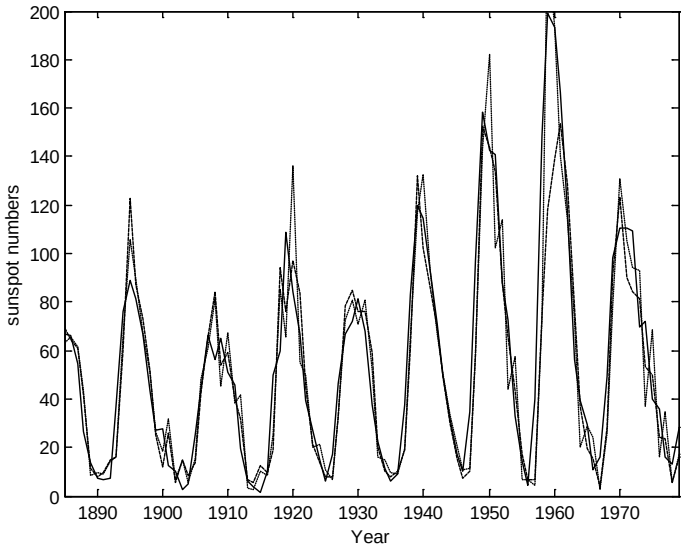
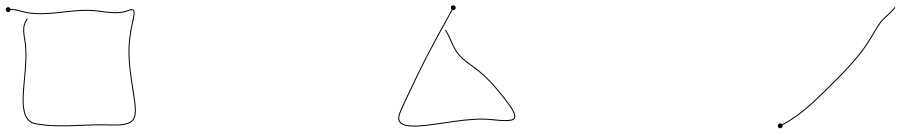


Fig. 14. Actual sunspots numbers (solid line), and forecasting results using the proposed network with the improved GA (dashed line) and the traditional network with the traditional GA (dotted line) for the year 1885-1979.



(a). Square. (b). Triangle. (c). Straight line.

Fig. 15. Samples of the hand-written patterns.

		Improved GA				Traditional GA			
	No. of iters.	Ave. Fitness	Max. Fitness	Min. Fitness	Standard Deviation	Ave. Fitness	Max. Fitness	Min. Fitness	Standard Deviation
f_1	500	1.0000	1.0000	1.0000	0.0000	1.0000	1.0000	1.0000	0.0000
f_2	500	0.9994	1.0000	0.9972	0.0006	0.9300	1.0000	0.5000	0.1753
f_3	500	0.9995	1.0000	0.9982	0.0004	0.9177	0.9630	0.8412	0.0252
f_4	5000	1.0000	1.0000	1.0000	0.0000	0.6264	0.7563	0.5421	0.0546
f_5	500	1.0000	1.0000	1.0000	0.0000	1.0000	1.0000	1.0000	0.0000
f_6	1500	1.0000	1.0000	1.0000	0.0000	0.9671	1.0000	0.0122	0.1803
f_7	4000	1.0000	1.0000	1.0000	0.0000	0.9667	1.0000	0.5013	0.1265
f_8	1000	1.0000	1.0000	1.0000	0.0000	0.7637	1.0000	0.2585	0.2569

Table. I. Results for f_1 to f_8 based on the improved GA and the traditional GA.

	Improved GA			Traditional GA		
	w	p_m	r_s	b	p_c	p_m
f_1	0.1	0.1	0.25	5	0.7	0.5
f_2	0.1	0.2	1	0.1	0.7	0.2
f_3	0.5	0.2	0.25	1	0.8	0.5
f_4	0.1	0.3	0.25	1	0.8	0.4
f_5	0.7	0.8	0.25	5	0.8	0.35
f_6	0.1	0.9	0.25	1	0.8	0.9
f_7	0.1	0.1	0.25	1	0.8	0.1
f_8	0.1	0.1	0.25	5	0.8	0.1

Table. II. Parameter settings for f_1 to f_8 based on the improved GA and the traditional GA (b is the shape parameter of non-uniform mutation, p_c is the probability of crossover).

	Improved GA			Traditional GA		
	w	p_m	r_s	b	p_c	p_m
Proposed Network	0.1	0.2	0.25	1	0.8	0.2
Traditional Network	0.1	0.25	0.25	1	0.8	0.3

Table. III. Parameter settings for the XOR problem.

Fitness value	Proposed network with improved GA	Proposed network with traditional GA	Traditional network with improved GA	Traditional network with traditional GA
Max.:	1.0000	1.0000	0.7751	0.7743
Min.:	1.0000	0.8000	0.7359	0.6165
Average:	1.0000	0.9600	0.7687	0.7082
Standard Deviation:	0.0000	0.0843	0.0134	0.0705

Table. IV. Results of the proposed and traditional neural networks trained by improved and traditional GAs for the XOR problem.

Input			Output		
x_1	x_2	x_3	y^d	y (Proposed approach)	y (Traditional approach)
-1	-1	-1	-1	-1.0000	-0.9867
-1	-1	+1	+1	+1.0000	+0.9227
-1	+1	-1	+1	+1.0000	-0.0607
-1	+1	+1	-1	-1.0000	-0.9848
+1	-1	-1	+1	+1.0000	+0.9204
+1	-1	+1	-1	-1.0000	-0.0044
+1	+1	-1	-1	-1.0000	-0.9901
+1	+1	+1	+1	+1.0000	+0.9196

Table. V. Output of the proposed and traditional approaches for the XOR problem.

	Improved GA			Traditional GA		
	w	p_m	r_s	b	p_c	p_m
Proposed Network	0.1	0.25	0.25	1	0.8	0.2
Traditional Network	0.1	0.25	0.4	1	0.8	0.25

Table. VI. Parameter settings for forecasting the sunspot number.

	Proposed Network with Improved GA	Proposed Network with Traditional GA	Traditional Network with Improved GA	Traditional Network with Traditional GA
Average fitness value	0.9551	0.9536	0.9453	0.9432
Number. of parameter	44	44	46	46
Average training error (MAE)	9.40	9.56	11.57	12.04
Average forecasting error (MAE)	12.34	13.03	13.21	13.93

Table. VII. Results of the proposed and traditional neural networks trained by improved and traditional GAs on forecasting the sunspot number.

	Improved GA			Traditional GA		
	w	p_m	r_s	b	p_c	p_m
Proposed Network	0.5	0.05	0.25	1	0.8	0.05
Traditional Network	0.9	0.08	0.4	1	0.8	0.06

Table. VIII. Parameter settings for the pattern recognition.

	Proposed Network with Improved GA	Proposed Network with Traditional GA	Traditional Network with Improved GA	Traditional Network with Traditional GA
Average fitness value	0.9873	0.9837	0.9728	0.9717
Number of parameter	224	224	227	227
Average training error (MSE)	0.0129	0.0166	0.0280	0.0291
Average forecasting error (MSE)	0.0185	0.0211	0.0396	0.0411
Recognition accuracy	96.50%	96.17%	93.67%	93.33%

Table IX. Results of the proposed neural network and the traditional neural network for hand-written pattern recognition.