Tuning of the Structure and Parameters of Neural Network using an Improved Genetic Algorithm [1]

H. K. Lam
Centre for Multimedia Signal Processing, Department of Electronic and Information Engineering, The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong
hkl@eie.polyu.edu.hk

S. H. Ling
Centre for Multimedia Signal Processing, Department of Electronic and Information Engineering, The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong
ensteve@eie.polyu.edu.hk

F. H. F. Leung
Centre for Multimedia Signal Processing, Department of Electronic and Information Engineering, The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong
enfrank@polyu.edu.hk

P. K. S. Tam
Department of Electronic and Information Engineering, The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong
enptam@hkpucc.polyu.edu.hk

Abstract — This paper presents the tuning of the structure and parameters of a neural network using an improved genetic algorithm (GA). The improved GA is implemented by floating-point number. The processing time of the improved GA is faster than that of the GA implemented by binary number as coding and decoding are not necessary. By introducing new genetic operators to the improved GA, it will also be shown that the improved GA performs better than the traditional GA based on some benchmark test functions. A neural network with switches introduced to links is proposed. By doing this, the proposed neural network can learn both the input-output relationships of an application and the network structure. Using the improved GA, the structure and the parameters of the neural network can be tuned. An application example on sunspot forecasting is given to show the merits of the improved GA and the proposed neural network.

I. Introduction

GA is a directed random search technique invented by Holland [1] in 1975, which is a widely applied in optimization problems [1-2, 5]. This is especially useful for complex optimization problems where the number of parameters is large and the analytical solutions are difficult to obtain. GA can help to find out the optimal solution globally over a domain [1-2, 5]. It has been applied in different areas such as fuzzy control [9-11, 15], path planning [12], greenhouse climate control [13], modeling and classification [14] etc..

Neural network was proved to be a universal approximator [16]. A 3-layer feed-forward neural network can approximate any nonlinear continuous function to an arbitrary accuracy. Neural networks are widely applied in areas such as prediction [7], system modeling and control [16]. Owing to its particular structure, a neural network is very good in learning [2] using some learning algorithms such as GA [1] and back propagation algorithm [2]. In general, the learning steps of the neural network are as follows. First, a network structure is defined with fixed numbers of inputs, hidden nodes and outputs. Second, a learning algorithm is chosen to realize the learning process. It can be seen that the fixed structure may not provide the optimal performance within a defined training period. Sometimes, the chosen neural network is large enough to provide a performance better than it should have. When this happens, the training period may have to be longer (as the performance is not met) and the implementation cost will also be increased.

The contributions of this paper are six-fold. First, new genetic operators are introduced to the improved GA. It will be shown that the improved GA performs more efficiently than the traditional GA [1-2, 5] based on some benchmark De Jong's test functions [3-4, 6, 17]. Second, an improved GA is proposed. This improved GA is implemented in floating-point number; hence, the processing time is much faster than that of the traditional GA [1-2, 5]. Third, the improved GA has only one parameter (population size), instead of three parameters, to be chosen by the user. This makes the improved GA simple and easy to use, especially for the users who do not have too much knowledge on tuning. Fourth, a three-layer neural network with switches introduced in some links is proposed to facilitate the tuning of the network structure. As a result, for a given fully connected neural network, it may no longer be a fully connected network after learning. This implies that the cost of implementing the proposed neural network, in terms of hardware implementation, processing time and simulation time can be reduced. Fifth, the improved GA is able to help tuning the structure as well as the parameters of the proposed neural network. Sixth, as an application example, the proposed neural network tuned by the improved GA is used to estimate the numbers of sunspot [7-8]. It will be shown that a better performance can be obtained as compared with that from a traditional feed-forward neural network [2] tuned by the traditional GA [1-2, 5].

[1] The work described in this paper was substantially supported by a Research Grant of The Hong Kong Polytechnic University (project numbers B-Q394 and G-V954).

II. Improved Genetic Algorithm

Genetic algorithms (GAs) are powerful searching algorithms. 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. However, a superior offspring is not guaranteed to produce in each reproduction process. In this paper, the traditional GA is modified and new genetic operators are introduced to improve its performance. Our improved GA is implemented by floating-point numbers, and the processing time is shorter than the GA implemented by binary numbers as the coding and decoding processes are not needed [1-2, 5]. Two parameters, the probabilities of crossover and mutation, in the traditional GA are no longer needed. Only the population size is needed to be defined. 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.

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

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

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

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; $para^j_{\min}$ and $para^j_{\max}$ 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,

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

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 can be done by assigning a probability q_i to the chromosome $\mathbf{p}_i$:

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

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

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

The selection process starts by randomly generating a nonzero floating-point number, $d \in [0 \;\; 1]$, for each chromosome. 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 observed from this selection process 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 copies, the average will stay and the worst will die off. In the selection process, only 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 average 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 instance, if the two selected chromosomes are $\mathbf{p}_1$ and $\mathbf{p}_2$, the offspring generated by the averaging process is given by,

$$\mathbf{os} = [os_1 \;\; os_2 \;\; \cdots \;\; os_{no\_vars}] = \frac{\mathbf{p}_1 + \mathbf{p}_2}{2} \tag{7}$$

This offspring (7) will then undergo the mutation operation. The mutation operation is to change 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,

$$\mathbf{nos}_j = [os^j_1 \;\; os^j_2 \;\; \cdots \;\; os^j_{no\_vars}] + [b_1\Delta nos_1 \;\; b_2\Delta nos_2 \;\; \cdots \;\; b_{no\_vars}\Delta nos_{no\_vars}], \; j = 1, 2, 3 \tag{8}$$

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 numbers such that $para^i_{\min} \le os^j_i + \Delta nos_i \le para^i_{\max}$. 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 zeros. The second new offspring is obtained according to (8) with that some b_i chosen randomly are set to be 1 and others are zero. 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. For the traditional GA process depicted in Fig. 1, 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 improved GA process, however, if $f_l < f_s$, the previous population is used again in the next genetic cycle.

III. Benchmark Test Functions

De Jong's Test Functions [3-4, 6, 17] are used as the benchmark test functions to examine the applicability and efficiency of the improved GA. Five test functions, $f_i(\mathbf{x})$, $i = 1, 2, 3, 4, 5$, will be used, where $\mathbf{x} = [x_1 \;\; x_1 \;\; \cdots \;\; x_n]$. n is an integer denoting the dimension of the vector $\mathbf{x}$. The five test functions are defined as follows,

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

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

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

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

$$f_3(\mathbf{x}) = 6 \times n + \sum_{i=1}^{n} floor(x_i), \; -5.12 \le x_i \le 5.12 \tag{11}$$

where $n = 5$ and the minimum point is at $f_3([5.12, 5], \ldots, [5.12, 5]) = 0$. The floor function, $floor(\cdot)$, is to round down the argument to an integer.

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

where n = 30 and the minimum point is at $f_4(0, \ldots, 0) = 0$. *Gauss*(0, 1) is a function to generate uniformly a floating-point number between 0 and 1 inclusively.

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

$$-65.356 \le x_i \le 65.356 \tag{13}$$

where

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

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,

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

and the fitness function for f_5 is defined as,

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

The improved GA goes through these five test functions. The results are compared with those obtained by the traditional GA [5]. For each test function, the simulation takes 500 iterations and the population size is 20. Each parameter of the traditional GA is encoded into a 40-bit chromosome and the probabilities of crossover and mutation are 0.25 and 0.03 respectively. The initial values of $\mathbf{x}$ in the population for a test function are set to be the same. For tests 1 to 5, the initial value are $[1 \;\; 1 \;\; 1]$, $[0.5 \;\; 0.5]$, $[1 \;\; \cdots \;\; 1]$, $[0.5 \;\; \cdots \;\; 0.5]$ and $[10 \;\; \cdots \;\; 10]$ respectively. The results of the average fitness values over the 30 times of simulations of the improved and traditional GAs are tabulated in Table I. It can be seen from Table I that the performance of the improved GA is better than that of the traditional GA. From Table I, the processing time of the improved GA is much shorter than that of the traditional GA.

IV. Neural Network with Link Switches and Tuning using the Improved GA

In this section, a neural network with link switches is presented. By introducing a switch to a link, not only the parameters but also the structure of the neural network can be tuned using the improved GA.

A. Neural Network with Link Switches

Neural networks [5] for tuning usually have a fixed structure. The number of connections must be large enough to fit a given application. This may cause the neural network structure to be unnecessarily complex, and increase the implementation cost. In this section, a multiple-input-multiple-output three-layer neural network is proposed as shown in Fig. 3. The main different point is that a unit step function is introduced to each link. Such a unit step function is defined as,

$$\delta(\alpha) = \begin{cases} 0 \text{ if } \alpha < 0 \\ 1 \text{ if } \alpha \ge 0 \end{cases}, \; \alpha \in \Re \tag{16}$$

This is equivalent to adding a switch to each link of the neural network. Referring to Fig. 3, the input-output relationship of the proposed multiple-input multiple-output three-layer neural network is as follows,

$$y_k(t) = \sum_{j=1}^{n_h} \delta(s_{jk}^2) w_{jk} logsig\left[\sum_{i=1}^{n_{in}} \left(\delta(s_{ij}^1) v_{ij} z_i(t) - \delta(s_j^1) b_j^1\right)\right] - \delta(s_k^2) b_k^2$$

$$, k = 1, 2, \ldots, n_{out} \tag{17}$$

$z_i(t)$, $i = 1, 2, \ldots, n_{in}$, are the inputs which are functions of a variable t; n_{in} denotes the number of inputs; v_{ij}, $i = 1, 2, \ldots, n_{in}$; $j = 1, 2, \ldots, n_h$, denote the weight of the link between the i-th input and the j-th hidden node; n_h denotes the number of the hidden nodes; s_{ij}^1, $i = 1, 2, \ldots, n_{in}$; $j = 1, 2, \ldots, n_h$, denotes the parameter of the link switch from the i-th input to the j-th hidden node; s_{jk}^2, $j = 1, 2, \ldots, n_h$; $k = 1, 2, \ldots, n_{out}$, denotes the parameter of the link switch from the j-th hidden node to the k-th output; n_{out} denotes the number of outputs of the proposed neural network; b_j^1 and b_k^2 denote the biases for the hidden nodes and output nodes respectively; s_j^1 and s_k^2 denote the parameters of the link switches of the biases to the hidden and output layers respectively; *logsig*(·) denotes the logarithmic sigmoid function:

$$logsig(\alpha) = \frac{1}{1 + e^{-\alpha}}, \; \alpha \in \Re \tag{18}$$

$y_k(t)$, $k = 1, 2, \ldots, n_{out}$, is the k-th output of the proposed neural network. By introducing the switches, the weights v_{ij} and the switch states can be tuned. It can be seen that the weights of the links govern the input-output relationship of the neural network while the switches of the links govern the structure of the neural network.

B. Tuning of the Parameters and Structure

In this section, the proposed neural network is employed to learn the input-output relationship of an application using the improved GA. The input-output relationship is described by,

$$\mathbf{y}^d(t) = \mathbf{g}\left(\mathbf{z}^d(t)\right), \; t = 1, 2, \ldots, n_d \tag{19}$$

where $\mathbf{y}^d(t) = \left[y_1^d(t) \;\; y_2^d(t) \;\; \cdots \;\; y_{n_{out}}^d(t)\right]$ and $\mathbf{z}^d(t) = \left[z_1^d(t) \;\; z_2^d(t) \;\; \cdots \;\; z_{n_{in}}^d(t)\right]$ 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,

$$fitness = \frac{1}{1 + err} \tag{20}$$

$$err = \sum_{k=1}^{n_{out}} \frac{n_k \sum_{t=1}^{n_d} \left| y_k^d(t) - y_k(t) \right|}{n_d} \tag{21}$$

The objective is to maximize the fitness value of (20) using the improved GA by setting the chromosome to be $\left[s_{jk}^2 \;\; w_{jk} \;\; s_{ij}^1 \;\; v_{ij} \;\; s_j^1 \;\; b_j^1 \;\; s_k^2 \;\; b_k^2\right]$ for all i, j, k.

Referring to (21), the maximum fitness value sometimes will be dominated by the error value of certain outputs that may not be of great interest. To avoid this, n_k, $k = 1, 2, \ldots, n_d$, are chosen to reduce the effects of the error from these dominated outputs. It can be seen from (20) and (21) that a larger fitness value implies a smaller error value.

V. Application Example

An application example on forecasting of the sunspot number [7-8] will be given in this section. The sunspot cycles from 1700 to 1980 are shown in Fig. 4. The cycles generated are non-linear, non-stationary, and non-Gaussian which are difficult to model and predict. We use the proposed 3-layer neural network (3-input-single-output) with link switches for the sunspot number forecasting. The inputs, z_i, of the purposed neural network are defined as $z_1(t) = y_1^d(t-1)$, $z_2(t) = y_1^d(t-2)$ and $z_3(t) = y_1^d(t-3)$ where t denotes the year and $y_1^d(t)$ is the sunspot numbers 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. Refer to (17), the proposed neural network used for the sunspot forecasting is governed by,

$$y_1(t) = \sum_{j=1}^{n_h} \delta(s_{j1}^2) w_{j1} logsig\left[\sum_{i=1}^{3}\left(\delta(s_{ij}^1) v_{ij} z_i(t) - \delta(s_j^1) b_j^1\right)\right] - \delta(s_1^2) b_1^2 \quad (22)$$

The value of n_h are changed from 3 to 7 to test the learning performance. The fitness function is defined as follows,

$$fitness = \frac{1}{1+err} \quad (23)$$

$$err = \sum_{t=1705}^{1884} \frac{\left|y_1^d(t) - y_1(t)\right|}{180} \quad (24)$$

The improved GA is employed to tune the parameters and structure of the neural network of (22). The objective is to maximize the fitness function of (23). The larger value of the fitness function indicates the smaller value of *err* of (24). The best fitness value is 1 and the worst one is 0. The population size used for the improved GA is 20. The lower and the upper bounds of the link weights are defined as $\frac{-3}{\sqrt{n_h}} \ge v_{ij}, w_{jk}, b_j^1, b_1^2 \ge \frac{3}{\sqrt{n_h}}$ and, $-1 \ge s_{j1}^2, s_{ij}^1, s_j^1, s_1^2 \ge 1$, $i = 1, 2, \ldots, 3; j = 1, 2, \ldots, n_h, k = 1$. The chromosomes used for the improved GA are $\left[s_{j1}^2 \quad w_{jk} \quad s_{ij}^1 \quad v_{ij} \quad s_j^1 \quad b_j^1 \quad s_k^2 \quad b_1^2\right]$. The initial values of the link weights are randomly generated. For comparison purpose, a fully connected 3-layer feed-forward neural network (3-input-single-output) [2] trained by the traditional GA [1-2, 5] is used for the sunspot number forecasting . The working conditions are exactly the same as those mentioned above. Additionally, a bit length of 10 is used for the parameter coding. The probabilities of the crossover and mutation are 0.25 and 0.03 respectively. In both approaches, the learning processes are carried out by a personal computer with PIII 500Hz CPU. The number of iterations to train the neural network is 2000 for both approaches.

The tuned neural networks are used to forecast the sunspot number during the years 1885-1980. Fig. 6 shows the simulation results of the forecasting of the sunspot number during the years 1885-1980 using the proposed neural network trained with improved GA (dashed lines), and the traditional feed-forwards trained with traditional GA (dotted lines). The actual sunspot numbers are represented in solid lines. The number of hidden nodes n_h changes from 3 to 7. The simulation results are tabulated in Table II and Table III. From the Table II, it is observed that the proposed neural network trained with the improved GA provides better results than those of traditional feed-forward neural network trained with traditional GA in terms of accuracy (fitness values), number of links and learning time. The training error (governed by (24)) and the forecasting error (governed by $\sum_{t=1885}^{1980} \frac{\left|y_1^d(t) - y_1(t)\right|}{96}$), represented in mean absolute error (MAE), are tabulated in Table III. It can be observed in Table III that our approach performs better than the traditional approach. Refer to Table III, the best result is obtained when the number of hidden node is 4 and the number of iterations for learning process is 2000. The number of connected link is 16 after learning (the number of fully connected links is 21 which includes the bias links). It is about 24% reduction of the links after learning. The training error and the forecasting error in term of mean absolute error (MAE) are 11.8869 and 13.0378 respectively.

VI. Conclusion

An improved GA has been proposed in this paper. The improved GA is implemented by floating-point numbers. As no coding and encoding of the chromosomes are necessary, the process time for learning using the improved GA is faster. New genetic operators have been introduced to the improved GA. By using the benchmark De Jong's test functions, it has been shown that the improved GA performs more efficiently than the traditional GA. Besides, by introducing a switch to each link, a neural network that facilitates the tuning of its structure has been proposed. Using the improved GA, the proposed neural network is able to learn both the input-output relationship of an application and the network structure. As a result, a given fully connected neural network can be reduced to a partly connected network after learning. This implies that the cost of implementation of the neural network can be reduced. An application example on forecasting the sunspot numbers using the proposed neural network trained with the improved GA has been given. The simulation results have been compared with those obtained by a traditional feed-forward networks trained by the traditional GA. It has been shown that our proposed network trained by the improved GA can perform better.

References

[1] J. H. Holland, *Adaptation in natural and artificial systems*, University of Michigan Press, Ann Arbor, MI, 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," *IEEE SMC '99 Conference Proceedings of IEEE International Conference on Systems, Man, and Cybernetics*, vol. 1, 1999, pp. 643-648.

[4] De Jong, K. A., "An analysis of the behavior of a class of genetic adaptive systems," *Ph.D. Thesis*, University of Michigan, Ann Arbor, MI., 1975.

[5] Z. Michalewicz, *Genetic Algorithm + Data Structures = Evolution Programs*, second, extended edition, Springer-Verlag, 1994.

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

[7] M. Li, K. Mechrotra, C. Mohan, S. Ranka, "Sunspot Numbers Forecasting Using Neural Network," *5th IEEE International Symposium on Intelligent Control, 1990. Proceedings*, pp.524-528, 1990.

[8] T. J. Cholewo, J. M. Zurada, "Sequential network construction for time series prediction," *International Conference on Neural Networks*, vol.4, pp.2034-2038, 1997.

[9] 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, Feb. 2001, pp. 32-53.

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

[11] 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, April, 2000, pp. 290–302.

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

[13] 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, Dec. 2000, pp. 753-760.

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

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

[16] M. Brown and C. Harris, *Neuralfuzzy adaptive modeling and control*, Prentice Hall, 1994.

[17] S. Amin and J.L. Fernandez-Villacanas, "Dynamic Local Search," *Second International Conference On Genetic Algorithms in Engineering Systems: Innovations and Applications*, pp129-132, 1997.

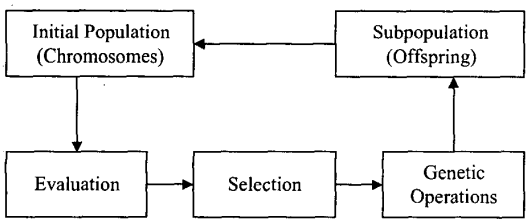


Fig. 1. Traditional GA.

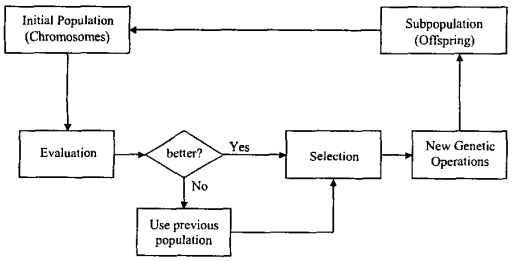


Fig. 2. Improved GA.

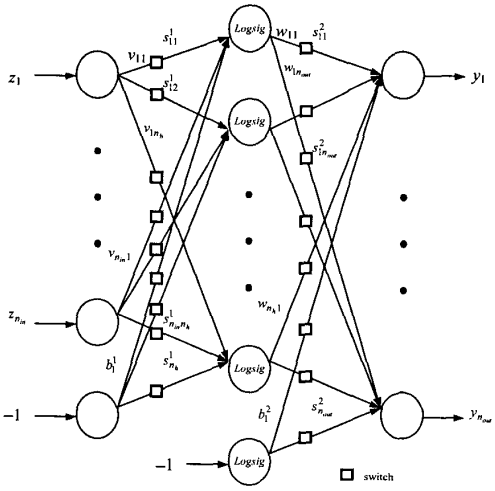


Fig. 3. The proposed 3-layer neural network.

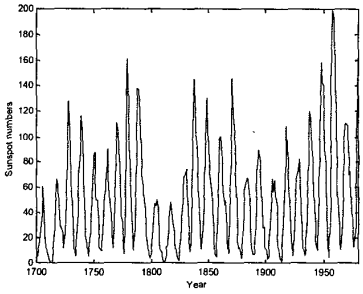
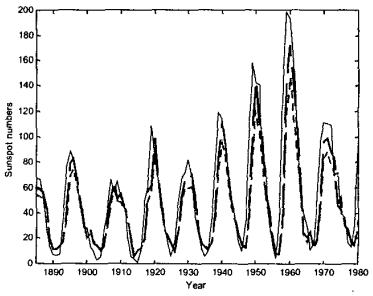
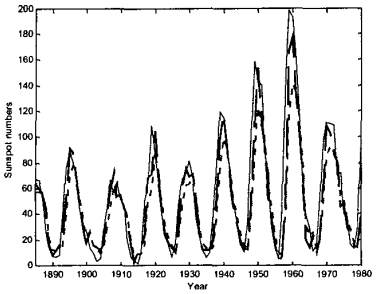


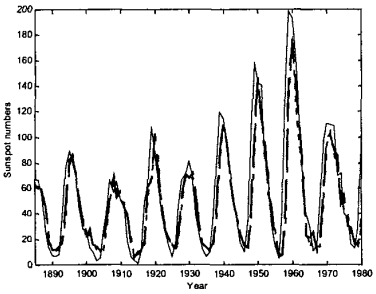
Fig. 4. Sunspot cycles from year 1700 to 1980



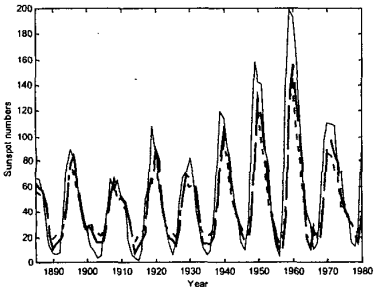
(a) Number of hidden nodes (n_h) = 3



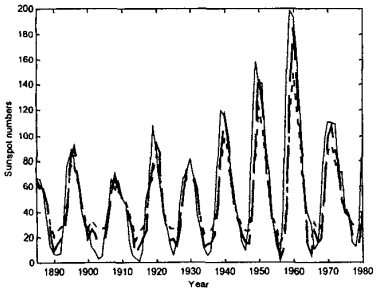
(b). Number of hidden nodes (n_h) = 4



(c). Number of hidden nodes (n_h) = 5



(d). Number of hidden nodes (n_h) = 6



(e). Number of hidden nodes (n_h) = 7

Fig. 6. Simulation results of a 96-year prediction using the proposed neural network with the proposed GA (dashed line) and the traditional network with the traditional GA (dotted line), compared with actual sunspot numbers (solid line) for the years 1885-1980.

	Improved GA		Traditional GA	
Test Functions	Fitness Values	Searching Time (s)	Fitness Values	Searching Time (s)
$f_1(\mathbf{x})$	0.999955	1.21	0.999382	8.35
$f_2(\mathbf{x})$	0.984039	1.22	0.810813	8.36
$f_3(\mathbf{x})$	0.583333	1.09	0.520833	10.93
$f_4(\mathbf{x})$	0.737526	2.69	0.14211	74.21
$f_5(\mathbf{x})$	0.995509	6.15	0.982912	21.13

Table I. Simulation results of the improved and the traditional GAs.

	Our Approach			Traditional Approach		
n_h	Fitness Values	Learning Time (s)	Number of Links	Fitness Values	Learning Time (s)	Number of Links
3	0.9398	2438.6	11	0.9316	3558.9	16
4	0.9439	2799.6	16	0.9342	6177.9	21
5	0.9414	3198.4	19	0.9338	6875.1	26
6	0.9356	3524.2	27	0.9294	7541.3	31
7	0.9351	3950.8	29	0.9287	8364.1	36

Table II. Simulation results of the application example on forecasting the sunspot number after 2000 iterations of learning.

	Our Approach	
n_h	Training error	Forecasting error
3	12.8112	14.6383
4	11.8869	13.0378
5	12.4495	14.3853
6	13.7666	15.4244
7	13.8809	13.9724
	Traditional Approach	
n_h	Training error	Forecasting error
3	14.6844	17.5568
4	14.0869	17.2024
5	14.1786	15.5637
6	15.1926	18.0312
7	15.3548	17.1553

Table III. Training error and forecasting error represented in mean absolute error (MAE).