Available online at www.sciencedirect.com

SCIENCE DIRECT®

Fuzzy Sets and Systems 157 (2006) 1036–1056

FUZZY sets and systems

www.elsevier.com/locate/fss

# A self-adaptive neural fuzzy network with group-based symbiotic evolution and its prediction applications

Cheng-Jian Lin*, Yong-Ji Xu

*Department of Computer Science and Information Engineering, Chaoyang University of Technology, 168 Gifeng E. Rd., Wufeng, Taichung County, 413 Taiwan, ROC*

Received 30 March 2004; received in revised form 27 June 2005; accepted 2 September 2005
Available online 30 September 2005

**Abstract**

In this paper, we propose a self-adaptive neural fuzzy network with group-based symbiotic evolution (SANFN-GSE) method. A self-adaptive learning algorithm consists of two major components. First, a self-clustering algorithm (SCA) identifies a parsimonious internal structure. An internal structure is said to be parsimonious in the sense that the number of clusters (fuzzy rules) is equal to the true number of clusters in a given training data set. The proposed SCA is an online method and is a distance-based connectionist clustering method. Unlike the traditional cluster techniques that only consider the total variation to updates the only one mean and deviation. The proposed SCA method considers the variation of each dimension for the input data. Second, a group-based symbiotic evolution learning (GSE) method is used to adjust the parameters for the desired outputs. The GSE method is different from traditional GAs (genetic algorithms), with each chromosome in the GSE method representing a fuzzy system. Moreover, in the GSE method, there are several groups in the population. Each group represents a set of the chromosomes that belong to a cluster computing by the SCA. In this paper we used numerical time series examples (one-step-ahead prediction, Mackey-Glass chaotic time series, and sunspot number forecasting) to evaluate the proposed SANFN-GSE model. The performance of the SANFN-GSE model compares excellently with other existing models in our time series simulations.
© 2005 Elsevier B.V. All rights reserved.

*Keywords:* Genetic algorithms; Cluster; Input partition; Symbiotic evolution; Neural fuzzy network

## 1. Introduction

In recent years, neural fuzzy networks [19,10] have become a popular research topic [3,10,19]. They are widely applied in fields such as time series prediction [14], control problem [18], and pattern recognition [4]. The reason is that neural fuzzy networks combine the semantic transparency of rule-based fuzzy systems with the learning capability of neural networks. The main advantage of the neural fuzzy network is that the black box nature of the neural network paradigm is resolved, as the connectionist structure of a neural fuzzy network essentially defines the IF-THEN rules. Moreover, a neural fuzzy network can adjust the parameter of the fuzzy rules using neural-network-based learning algorithms.

* Corresponding author. Fax: +886 43742375.
*E-mail address:* cjlin@mail.cyut.edu.tw (C.-J. Lin).

0165-0114/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.fss.2005.09.001

Several approaches that are used to generate the fuzzy IF-THEN rules from numerical data have been proposed [3,4,10,14,18,19]. These methods were developed for supervised learning; that is, the desired output is clearly given for each input pattern in order to guide the network's learning.

The training of the parameter is the problem in designing a neural fuzzy system. To solve this problem, backpropagation (BP) training is widely used. It is a powerful training technique that can be applied to networks with a forward structure. Since the steepest descent technique is used in BP training to minimize the error function, the algorithms may reach the local minima very fast and never find the global solution. In addition, the performance of BP training depends on the initial values of the system parameter, and for different network topologies one has to derive new mathematical expressions for each network layer.

Considering the disadvantages mentioned above, one may be faced with suboptimal performances, even for a suitable neural fuzzy network topology. Hence, techniques capable of training the system parameters and finding a global solution while optimizing the overall structure are needed. In that respect, genetic algorithms (GAs) appear to be better candidates than backpropagation algorithm and several GA-based approaches have appeared in the literature [9,11,13,16]. Karr [13] used a GA to generate membership functions for a fuzzy system. In Karr's work, a user needs to declare an exhaustive rule set and then use a GA to design only the membership functions. In [9], a fuzzy controller design method that used GAs to find the membership functions and the rule sets simultaneously was proposed. In [13] and [9], the input space was partitioned into a grid. The number of fuzzy rules (i.e., the length of each chromosome in the GA) increased exponentially as the dimension of the input space increased. In [9], a GA was used to tune the consequent parameters of TSK-type fuzzy rules [3], as well as the membership functions in the precondition parts. Juang [11] proposed a TSK-type recurrent fuzzy network with a genetic algorithm for control problems.

However, these approaches encountered one or more of the following major problems: (1) all the fuzzy rules were encoded into one chromosome; (2) the initial values in the population were generated randomly according to a fixed range and cannot evaluate each fuzzy rule locally; (3) the initial numbers of fuzzy rule were obtained by trial and error; (4) the input space was partitioned into a grid; and (5) the mutational value was always generated according to the constant range.

In this paper, we propose a self-adaptive neural fuzzy network with group-based symbiotic evolution (SANFN-GSE) for solving the above problems. Moreover, in this paper, we used time series problems to evaluate the proposed SANFN-GSE model. The idea of prediction topic has been extensively explored for many years under the title of time series analysis [2,19,26]. Traditionally, the prediction is based on the statistical model that is either linear or nonlinear [17]. Recently, there are several researches using evolution based neural fuzzy network for predicting time series [14,6,21]. These researchers have discussed that the network paradigm is a very useful model for predicting the time series problems and is especially for predicting nonlinear time series. About this, in this paper we used numerical time series examples (one-step-ahead prediction, Mackey-Glass chaotic time series, and sunspot number forecasting) to evaluate the proposed SANFN-GSE model.

The proposed SANFN-GSE consists of two major components. First, a structure learning scheme is used to determine proper input space partitioning and to find the mean and deviation of each cluster. Second, a supervised learning scheme is used to adjust the parameters for the desired outputs. The proposed structure learning algorithm uses the self-clustering algorithm (SCA) to perform structure learning and the new symbiotic evolution algorithm called group-based symbiotic evolution (GSE) uses to perform parameter learning.

With the proposed self-clustering algorithm (SCA) method, a flexible partitioning of the input space is achieved, and the number of rules is relatively small compared to the grid partition. Moreover, the proposed group-based symbiotic evolution method (GSE) is different from traditional GAs, with each chromosome in GAs representing a fuzzy system. In the GSE, each chromosome represents only one fuzzy rule, and an $n$-rule fuzzy system is constructed by selecting and combining $n$ chromosomes from a given population. The GSE method, promotes both cooperation and specialization, which ensures diversity and prevents a population from converging to suboptimal solutions. Compared with the normal symbiotic evolution, there are several groups in the population in the proposed GSE method. Each group represents a set of the chromosomes that belong to a cluster. Each group will perform the reproduction and crossover operations in each generation. The advantages of the proposed SANFN-GSE model are summarized as follows: (1) the proposed SCA method can online cluster the input partitions and considers the variation of each dimension for the input data; (2) the GSE method used group-based population to evaluate the fuzzy rule locally; (3) the initial and mutational value was generated according to each clusters' mean and deviation computing by the SCA; and (4) the SANFN-GSE model converges more quickly than existing evolution methods.

This paper is organized as follows. Section 2 introduces the general neural fuzzy network. A self-adaptive learning algorithm is proposed in Section 3. In Section 4, the proposed SANFN-GSE model is evaluated using three different prediction cases, and its performances are benchmarked against other structures. Finally, conclusions on the proposed algorithm are summarized in the last section.

## 2. The general neural fuzzy network

In this section, the general neural fuzzy network is introduced. A neural fuzzy network consists of a set of fuzzy IF-THEN rules that describe the input–output mapping relationship of the network. The antecedents of fuzzy rules partition the input space into a number of linguistic term sets while the consequent constituent can be chosen as a fuzzy membership function (Mamdani model) [19], a singleton value [20,24], or a function of a linear combination of input variables (TSK model) [3]. No matter which type of neural fuzzy networks is chosen, different consequent constituents result in different types of fuzzy models.

For simplicity, the singleton consequent of fuzzy rules is adopted in this paper. The fuzzy rule with singleton consequent can be given in the following form:

$$\begin{aligned} &R_j\text{: } \mathit{IF}\ x_1\ \mathit{is}\ A_{1j}\ \mathit{and}\ x_2\ \mathit{is}\ A_{2j} \ldots \mathit{and}\ x_n\ \mathit{is}\ A_{nj} \\ &\quad \mathit{THEN}\ y' = b_j, \end{aligned} \tag{1}$$

where $x_i$ is the input variable, $y'$ is the output variable, $A_{ij}$ is the linguistic term of the precondition part, $b_j$ is the constant consequent part, and $n$ is the number of input variables.

The structure of a general neural fuzzy network is shown in Fig. 1, where $n$ and $R$ are, respectively, the number of input variables and the number of rules. The network is called a self-adaptive neural fuzzy network with group-based symbiotic evolution (SANFN-GSE) in this paper. It is a four-layer network structure. The functions of the nodes in each layer are described as follows:

*Layer* 1: No function is performed in this layer. The node only transmits input values to layer 2.

$$u_i^{(1)} = x_i. \tag{2}$$

*Layer* 2: Nodes in this layer correspond to one linguistic label of the input variables in layer 1; that is, the membership value specifying the degree to which an input value belongs to a fuzzy set is calculated in this layer. For an external input $x_i$, the following Gaussian membership function is used:

$$u_{ij}^{(2)} = \exp\left(-\frac{[u_i^{(1)} - m_{ij}]^2}{\sigma_{ij}^2}\right), \tag{3}$$

where $m_{ij}$ and $\sigma_{ij}$ are, respectively, the center and the width of the Gaussian membership function of the *j*th term of the *i*th input variable $x_i$.

*Layer* 3: The output of each node in this layer is determined by the fuzzy AND operation. Here, the product operation is utilized to determine the firing strength of each rule. The function of each rule is

$$u_j^{(3)} = \prod_i u_{ij}^{(2)}. \tag{4}$$

*Layer* 4: Nodes in this layer perform defuzzification. The mathematical function of each node is:

$$u^{(4)} = \frac{\sum_j u_j^{(3)} w_j}{\sum_j u_j^{(3)}}, \tag{5}$$

where the weight $w_j$ is the output action strength associated with the *j*th rule and $u^{(4)}$ is the output of the network.

In the next section, we shall discuss our development of an efficient learning algorithm for establishing the internal structure of the SANFN-GSE model.

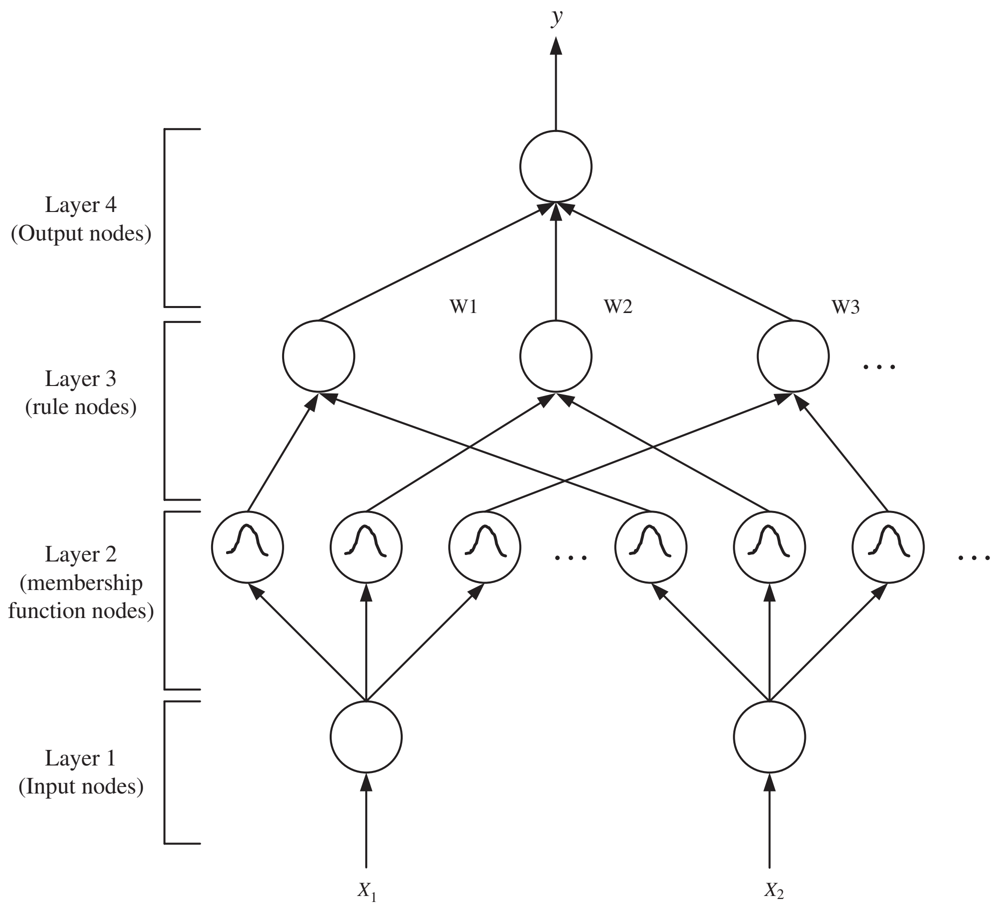

Fig. 1. Structure of a general neural fuzzy network.

## 3. A self-adaptive learning algorithm for the SANFN-GSE model

A self-adaptive learning algorithm consists of two major components. First, there is a self-clustering algorithm (SCA) which identifies a parsimonious internal structure. The internal structure is said to be parsimonious in the sense that the number of clusters (fuzzy rules) is equal to the true number of clusters in a given training data set. Second, there is a group-based symbiotic evolution learning method which is used to adjust the parameters for the desired outputs. The group-based symbiotic evolution method is different from the traditional symbiotic evolution, with each population in the GSE method was divided to several groups. Each group represents a set of the chromosomes that belong to a cluster computing by the SCA. Each group will perform the reproduction and crossover operations in each generation. A detailed self-adaptive learning algorithm of the SANFN-GSE model is presented in Fig. 2.

### *3.1. The self-clustering algorithm*

When the parsimonious SANFN-GSE is being established, a number of fuzzy rules and approximate estimates of the corresponding parameters, such as means and deviations, describing the fuzzy term sets in the antecedent parts need to be extracted from given input–output training pairs. Thus, the choice of clustering technique in neural fuzzy networks is an important consideration. This is due to the use of *partition-based* clustering techniques, such as fuzzy *C*-means (FCM) [8], linear vector quantization (LVQ) [7], fuzzy Kohonen partitioning (FKP), and pseudo FKP [1], to perform cluster analysis. However, such clustering techniques require *prior* knowledge such as the number of clusters present in a data set. To solve the above problem, online-based cluster techniques were proposed [14,27]. But there still is a

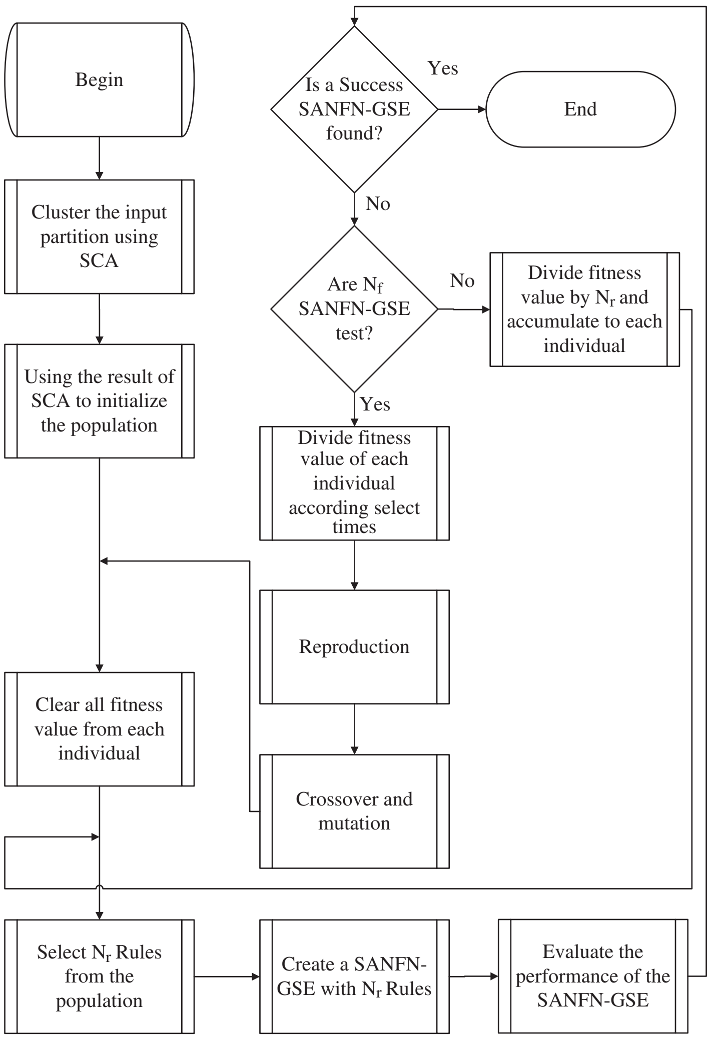
Begin
Cluster the input partition using SCA
Using the result of SCA to initialize the population
Clear all fitness value from each individual
Select Nr Rules from the population
Create a SANFN-GSE with Nr Rules
Evaluate the performance of the SANFN-GSE
Is a Success SANFN-GSE found?
Yes
End
No
Are Nf SANFN-GSE test?
No
Divide fitness value by Nr and accumulate to each individual
Yes
Divide fitness value of each individual according select times
Reproduction
Crossover and mutation

Fig. 2. The self-adaptive learning algorithm for the SANFN-GSE model.

problem with these methods; namely, the clustering methods [14,27] only consider the total variations of the mean and deviation in all dimensions of the input data. This is because the cluster numbers increase quickly.

In this paper, we use a self-clustering algorithm (SCA) to partition the input space to create fuzzy rules. The proposed SCA is an online and distance-based clustering method which is unlike the traditional clustering techniques [1,7,8,14,27]. The tendency of the traditional clustering techniques is to consider the total variations in all dimensions of the input data that will cause clusters to extend too fast. According to the above-mentioned problems, the proposed SCA method considers the variation of each dimension for the input data.

The main advantage of the proposed method is that the SCA is a one-pass algorithm that dynamically estimates the number of clusters in a set of data and finds their current means in the input data space. For the above reason, the SCA can cluster the input space quickly. The details of the SCA algorithm are described in the following steps:

*Step* 0: Create the first cluster $C_1$ by simply taking the position of the first input data as the first cluster mean $Cc_{i\_1}$ where $i$ means the $i$th input variables. And setting a dimension distance value to 0 (see Fig. 3(a)).

*Step* 1: If all of the training input data have been processed, the algorithm is finished. Otherwise, the current input example, $x_i[k]$, is taken and the distances between this input example and all already created cluster mean $Cc_{i\_j}$

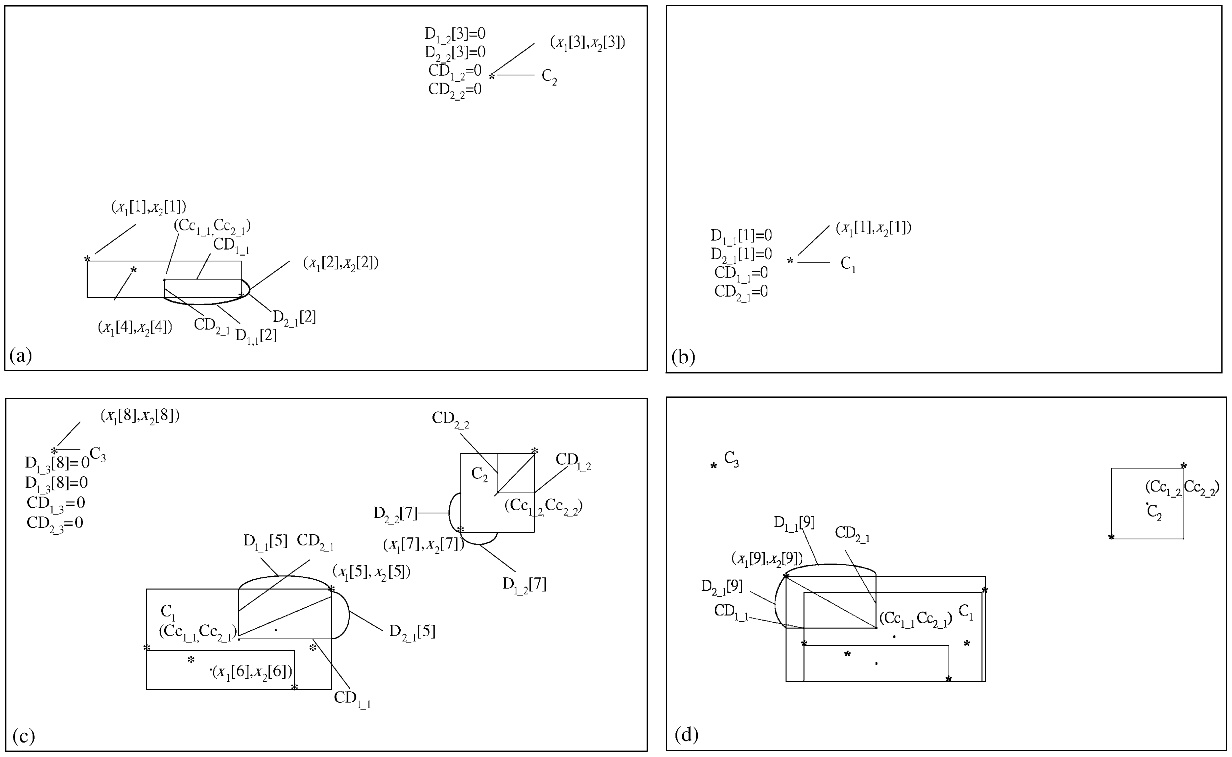

Fig. 3. A brief clustering process using SCA with samples in 2-D space.

are calculated:

$$D_{i\_j}[k] = \|x_i[k] - Cc_{i\_j}\|, \tag{6}$$

where $j = 1, 2, \ldots, R$ denotes the $j$th cluster, $k = 1, 2, 3, \ldots, N$ represents the $k$th input, and $i = 1, 2, \ldots, n$ represents the $i$th dimension.

*Step* 2: If the distance calculation in Eq. (6) is equal to or less than all of the dimension distances $CD_{i\_j}$ that represent the $i$th dimension distance in the $j$th cluster (set to 0 initially), then the current input example belongs to a cluster with the minimum distance:

$$D\min_{j}[k] = \min\left(\sum_{i=1}^{n} \|x_i[k] - Cc_{i\_j}\|\right), \tag{7}$$

$$D\min_{i\_j}[k] = \|x_i[k] - Cc_{i\_j}\|, \tag{8}$$

where $j$ in Eq. (8) represents the $j$th cluster that is computed using Eq. (7). The use of Eqs. (7) and (8) is to find the minimum sum of all the dimension distances in a cluster with the $k$th input data. The constraint is described as follows:

$$D\min_{i\_j}[k] \leqslant CD_{i\_j}. \tag{9}$$

If no new clusters are created or no existing clusters are updated (the cases of $(x_1[4], x_2[4])$ and $(x_1[6], x_2[6])$ in Fig. 3(b)), the algorithm returns to Step 1. Otherwise, the algorithm goes to the next step.

*Step* 3: Find a cluster from all existing cluster centers by calculating $S_{i\_j}[k] = D_{i\_j}[k] + CD_{i\_j}$, $j = 1, 2, \ldots, R$, and then choosing the cluster center with the minimum value:

$$S\min_{i\_j}[k] = D\min_{i\_j}[k] + CD_{i\_j}, \quad \text{where } j = 1, 2, \ldots, R. \tag{10}$$

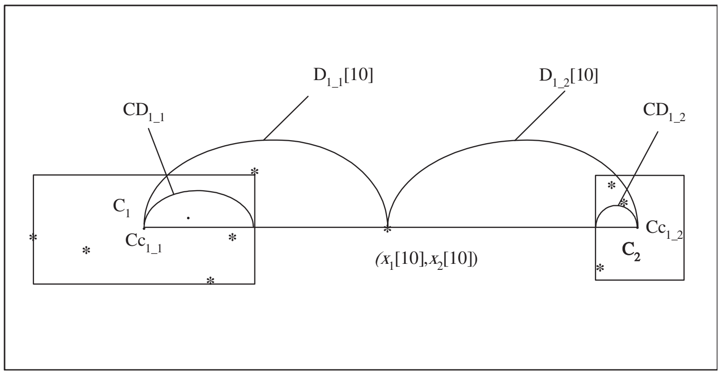

Fig. 4. The special case of SCA.

In Eqs. (8) and (9), the minimum distance from any cluster mean to the examples that belong to this cluster is not greater than the threshold $D_{thr}$, though the algorithm does not keep any information of passed examples. However, we find that the formulation only considers the distance between the input data and the cluster mean in Eq. (10). But the special situation [4] shows that the distances between the given point $x_i[10]$ and both cluster means $Cc_{i\_1}$ and $Cc_{i\_2}$ are the same as in Fig. 4. In the aforementioned technique, the cluster $C_2$, which has small dimension distances $CD_{i\_2}$, will be selected to expand according to Eq. (10). However this causes a problem in that the cluster numbers increase quickly. To avoid this problem, we state a condition, as follows:

*If there are two* $D\min_j[10]$ *compute in Eq.* (9) *that* $D\min_1[10] = D\min_2[10]$ *and* $(CD_{1\_1} + CD_{2\_1} > CD_{1\_2} + CD_{2\_2})$

$$\textit{Then } D\min_{1\_1}[10] = D_{1\_1}[10], \tag{11}$$

$$D\min_{2\_1}[10] = D_{2\_1}[10], \tag{12}$$

where $D\min_1[10]$ represents the minimum distance between the 10th input data and the mean of the 1st cluster that is calculated by Eq. (9); $D\min_{2\_1}[10]$ represents dimension distance between the 2nd dimension of the 10th input data and the 2nd dimension mean of the 1st cluster that is calculated by Eq. (10); $D_{2\_1}[10]$ represents dimension distance between the 2nd dimension of the 10th input data and the 2nd dimension mean of the 1st cluster that is calculated by Eq. (8). In Eqs. (11) and (12), we find that when the distances between the input data and both clusters are the same, the formulation will choose the cluster that has the large dimension distance $CD_{1\_1}$ and $CD_{2\_1}$.

*Step* 4: If $S\min_{i\_j}[k]$ in Eq. (10) is greater than $D_{\text{thr}}$, the input example $x_i[k]$ does not belong to any existing cluster. A new cluster is created in the same way as described in Step 0 (the cases of $(x_1[3], x_2[3])$ and $(x_1[8], x_2[8])$ in Fig. 3(c)), and the algorithm returns to Step 1.

*Step* 5: If $S\min_{i\_j}[k]$ is not greater than $D_{\text{thr}}$, the cluster is updated by moving its mean, $Cc_{i\_j}$, and increasing the value of its dimension distances. The new mean is moved to the point on the line connecting the input data, and the distance from the new mean to the point is equal to its dimension distance (the cases of $(x_1[5], x_2[5])$ and $(x_1[9], x_2[9])$ in Fig. 3(d)). The details for updating the equations are as follows:

$$\textit{If } CD_{i\_j} < (x_i[k] - Cc_{i\_j} + CD_{i\_j})/2$$

$$\textit{Then } CD_{i\_j} = (x_i[k] - Cc_{i\_j} + CD_{i\_j})/2, \tag{13}$$

$$Cc_{i\_j} = x_i[k] - CD_{i\_j} \quad \textit{if } Cc_{i\_j} >= x_i[k], \tag{14}$$

$$Cc_{i\_j} = x_i[k] + CD_{i\_j} \quad \textit{if } Cc_{i\_j} < x_i[k], \tag{15}$$

where $k = 1, 2, 3, \ldots, N$ represents the $k$th input, $j$ represents the $j$th cluster that has a minimum distance in Eq. (7), $x$ represents the input data, and $i$ represents the $i$th dimension. After this step is performed, the algorithm returns to step 1.

The threshold parameter $D_{\mathrm{thr}}$ is an important parameter in the input space partition scheme. A low threshold value leads to the learning of fine clusters (such that many fuzzy rules are generated), whereas a high threshold value leads to the learning of coarse clusters (such that fewer fuzzy rules are generated). Therefore, the selection of the threshold value $D_{\mathrm{thr}}$ critically affects the simulation results, and the threshold value is determined by practical experimentation or trial-and-error tests.

## 3.2. Genetic algorithm through group-based symbiotic evolution

### 3.2.1. Symbiotic evolution and fuzzy systems

The idea of symbiotic evolution was first proposed in an implicit fitness sharing algorithm that was used in an immune system model [25]. The authors developed artificial antibodies to identify artificial antigens. Because each antibody can match only one antigen, a different population of antibodies was required to effectively defend against a variety of antigens.

Unlike traditional GAs that use each individual in a population as a full solution to a problem, symbiotic evolution assumes that each individual in a population represents only a partial solution to a problem; complete solutions combine several individuals in the population. In a normal evolution algorithm, a single individual is responsible for the overall performance, with a fitness value assigned to that individual according to its performance. In symbiotic evolution, the fitness of an individual (a partial solution) is calculated by summing up the fitness values of all possible combinations of that individual with other current individuals (partial solutions) and dividing the sum by the total number of combinations.

As shown in [12,22], partial solutions can be characterized as *specializations*. The specialization property ensures diversity, which prevents a population from converging to suboptimal solutions. A single partial solution cannot "take over" a population since there must be other specializations present. Unlike the standard evolutionary approach, which always causes a given population to converge, hopefully at the global optimum, but often at a local one, the symbiotic evolution finds solutions in different, unconverted populations [12,22].

The basic idea of symbiotic evolution is that an individual (i.e., a chromosome) is used to represent a single fuzzy rule. A fuzzy system is formed when several individuals, which are randomly selected from a population, are combined. With the fitness assignment performed by symbiotic evolution, and with the local property of a fuzzy rule, symbiotic evolution and the fuzzy system design can complement each other. If a normal GA evolution scheme is adopted for fuzzy system design, only the overall performance of the fuzzy system is known, not the performance of each fuzzy rule. The best method to replace the unsuitable fuzzy rules that degrade the overall performance of a fuzzy system is to use crossover operations, followed by observing the performance of the offspring.

### 3.2.2. A new symbiotic evolution for the SANFN-GSE model

After the initial SANFN-GSE structure has been identified according to the SCA clustering algorithm, the SANFN-GSE then performs parameter identification. The parameter identification is used to tune the parameters of the existing structure optimally in the sense that the best fit of the SANFN-GSE output to the training patterns is found. In this paper, we proposed the group-based symbiotic evolution method (GSE) to tune the free parameters. Unlike the normal symbiotic evolution, in the GSE method, there are several groups in the population. Each group represents a set of the chromosomes that belong to a cluster. Each group will perform the reproduction and crossover operations in each generation. In the GSE method, we restrict the top half of the population to reproducing the next generation. Moreover, we also require that chromosomes from different groups cannot be selected for performing the crossover operation. In the GSE method, each chromosome in the group must be selected at least one time for the fuzzy model to be constructed. The advantages of the GSE method are as follows: (1) the GSE method used group-based population to evaluate the fuzzy rule locally; and (2) the initial and mutational value was generated according to each cluster's mean and deviation computing by the SCA.

Like the traditional symbiotic evolution, the proposed GSE method consists of three major operators: reproduction, crossover, and fitness assignment. Before the details of these three operators are introduced, the issues of clustering, coding, initialization, and mutation are discussed. The clustering step uses the SCA technique to cluster the input space. The coding step is concerned with the membership functions and fuzzy rules of a fuzzy system that represent chromosomes suitable for group based symbiotic evolution. The initialization step assigns the population values before

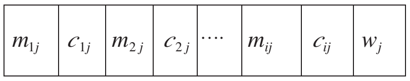

Fig. 5. Coding a fuzzy rule into a chromosome in symbiotic evolution.

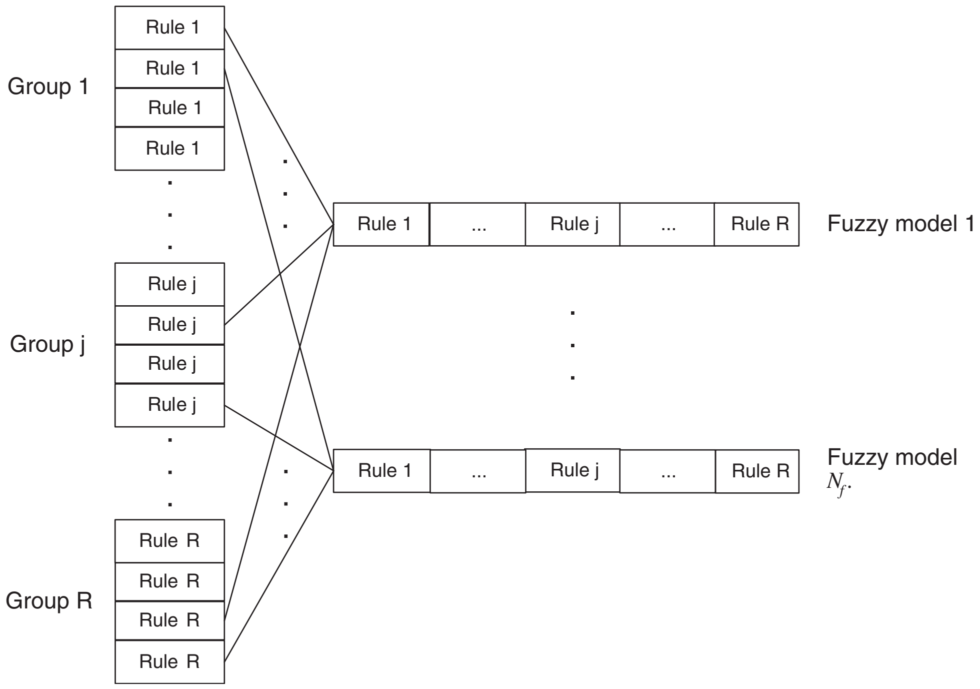

Fig. 6. The structure of the chromosome in the GSE.

the evolution process begins. The fitness assignment gives a suitable fitness value to each fuzzy rule during the evolution process. The whole learning process is described step by step as follows (see Fig. 2):

(a) *Clustering step*: The SCA clustering method, which was described in Section 3.1, is used to determine the number of fuzzy rules and to estimate the corresponding parameters.

(b) *Coding step*: The first step in GAs is the coding of a fuzzy rule into a chromosome. Fig. 5 shows an example of a fuzzy rule coded into a chromosome where *i* and *j* represents *i*th dimension and *j*th rule. In this paper, a Gaussian membership function is used with variables representing the mean and deviation of the membership function. Fig. 5 describes a fuzzy rule that according to Eq. (1), where $m_{ij}$ and $\sigma_{ij}$ represent a Gaussian membership function with mean and deviation, respectively, and $w_j$ represents the weight connected to the *j*th rule node. For each chromosome, a gene is represented by a real number.

(c) *Initialization step*: Before the SANFN-GSE model is designed, individuals forming an initial population should be generated. In the traditional symbiotic evolution, an initial population is generated randomly according to a fixed range. In this study, the initial population was generated randomly according to the range of the mean and dimension distance of every input dimension, which were computed by the SCA clustering method. This will reduce the generations for searching the solutions on the fixed range. Assume that there are *R* fuzzy rules (i.e., clusters) generated by using the SCA. In this study, a group represents a set of chromosomes that belong to a fuzzy rule. Therefore, there are *R* groups in a population. About this, the GSE method can evaluate the fuzzy rule locally. The structure of the chromosomes in the GSE method is shown in Fig. 6. The following formulations show how to generate the initial population:

$$\begin{aligned}&\mathit{Mean}\colon\ Chr_{c\_j}[p] = Cc_{i\_j} + random[CD_{i\_j}, -CD_{i\_j}],\\ &\quad \mathit{where}\ i = 1, 2, \ldots, n\ \mathit{and}\ p = 2^{*}i - 1.\end{aligned} \tag{16}$$

$$\textit{Deviation}: \; Ch_{c\_j}[p] = random[0, |Cc_{i\_j} - Cc_{i+1\_j}|], \quad \text{where } p = 2^{*}i. \tag{17}$$

$$\textit{Weight}: \; Chr_{c\_j}[p] = random[-2, 2], \quad \text{where } p = 2^{*}n + 1, \tag{18}$$

where $Chr_{c\_j}$ represents *c*th chromosome from the *j*th group; *p* represents the *p*th gene in the *c*th chromosome; *i* and *j* represent the *i*th input dimension and the *j*th cluster; and $Cc_{i\_j}$ and $D_{i\_j}$ represent the *i*th dimension mean and the *i*th dimension distance in the *j*th cluster that are computed by Eqs. (15)–(17), respectively.

The size of the population depends on the complexity of a problem. Besides the population size, some other parameters need to be set. These are the number of fuzzy systems to be formed and evaluated in every generation and the mutation rate and crossover rate and theses parameters also depend on the complexity of a problem.

(d) *Fitness assignment step*: As previously stated, for the GSE method, the fitness value of a rule (an individual) is calculated by summing up the fitness values of all the possible combinations in the chromosomes that are selected randomly from *R* groups. The details for assigning the fitness value are described step by step as follows:

*Step* 1: Randomly choose *R* fuzzy rules from the *R* groups in the population with size $N_f$.

*Step* 2: Evaluate every fuzzy system that is generated from step 1 to obtain a fitness value.

*Step* 3: Divide the fitness value by *R* and accumulate the divided fitness value to the selected rules with their fitness value records that were set to zero initially.

*Step* 4: Repeat the above steps until each rule (individual) in a population has been selected a sufficient number of times, and record the number of fuzzy systems in which each individual has participated.

*Step* 5: Divide the accumulated fitness value of each individual by the number of times it has been selected. The average fitness value represents the performance of a rule. In this paper, the fitness value is designed according the follow formulation:

$$\text{Fitness Value} = 1/(1 + E(y, \bar{y})), \tag{19}$$

$$\text{where } E(y, \bar{y}) = (y_i - \bar{y}_i)^2 \quad \text{for } i = 1, 2, \ldots, N, \tag{20}$$

where $y_i$ represents the true value of the *i*th output, $\bar{y}_i$ represents the predicted value, $E(y, \bar{y})$ is a error function and *N* represents a numbers of the training data of each generation. The average fitness value represents the performance of a rule. The flowchart of the major operations in the GSE is presented in Fig. 7.

(e) *Reproduction step*: Reproduction is a process in which individual strings are copied according to their fitness value. This operator is an artificial version of neural selection. A fitness value is assigned to each individual using a fitness assignment method in which high numbers denote a good fit. In this study, we use the roulette-wheel selection method [5]—a simulated roulette is spun—for this reproduction process (see Fig. 8). In Fig. 8, the intermediate population is $P'$, which is generated from identical copies of a chromosome sampled by spinning the roulette wheel a sufficient number of times. The best performing individuals in the top half of the population [12] advances to the next generation. The other half is generated to perform crossover operations on individuals in the top half of the parent generation. In the reproduction step, we must keep the same number of chromosomes in the top half of the population for each group.

(f) *Crossover step*: Reproduction directs the search toward the best existing individuals but does not create any new individuals. In nature, an offspring has two parents and inherits genes from both. The main operator working on the parents is the crossover operator, the operation of which occurred for a selected pair with a crossover rate that was set to 0.5 in this study. The first step is to select the individuals from the population for the crossover. Tournament selection [5] is used to select the top-half of the best performing individuals shown in Fig. 9. In the GSE method, we require that the individuals from different groups cannot be selected for performing the crossover operation. The individuals are crossed and separated using a two-point crossover operation, as shown in Fig. 10. In Fig. 10, new individuals are created by exchanging the site's values between the selected sites of parents' individual. In the crossover step, we also keep the same number of chromosomes in the other half of the population for each group. After this operation, the individuals with poor performances are replaced by the newly produced offspring.

(g) *Mutation step*: Although reproduction and crossover will produce many new strings, they do not introduce any new information to the population at the site of an individual. Mutation is an operator that randomly alters the allele of a gene. The mutation operation of an individual is shown in Fig. 11. Unlike the traditional symbiotic evolution that

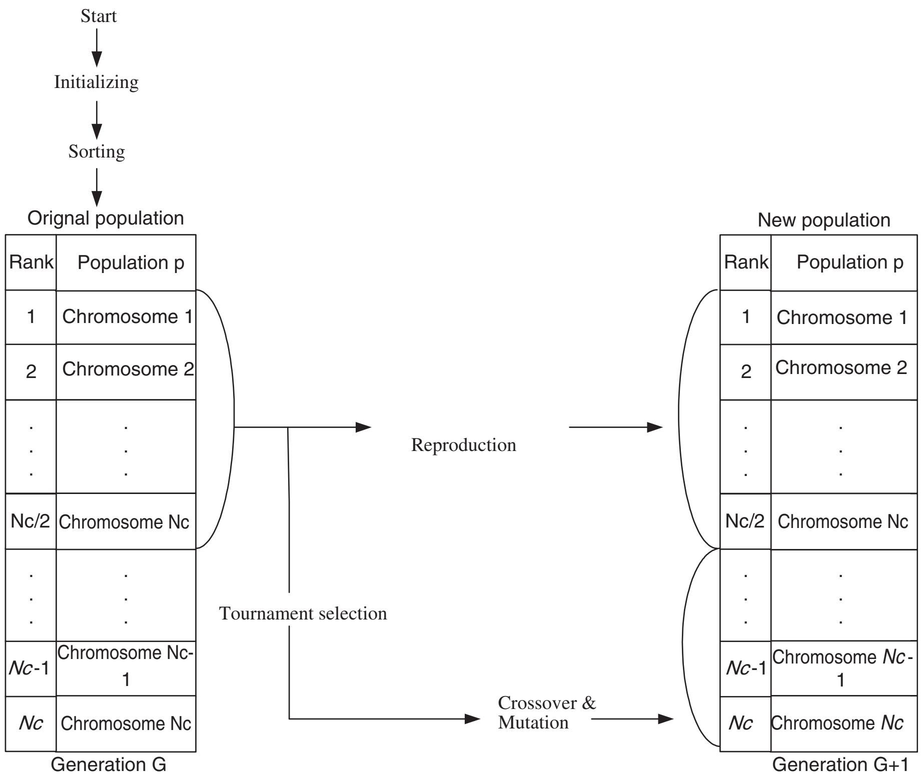

Fig. 7. The flowchart of the major operations in the GSE.

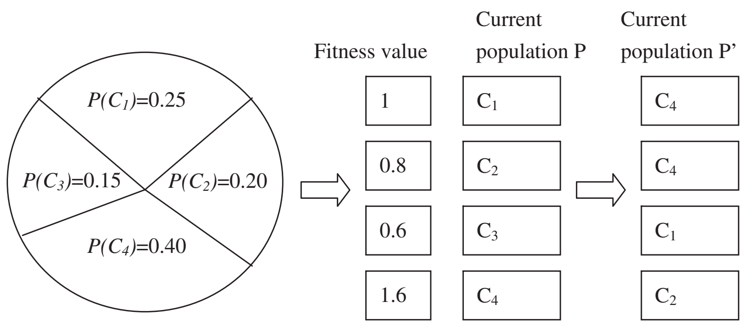

Fig. 8. The roulette wheel selection.

generated the mutation value according to the constant range. In the GSE, the mutation value is generated according to Eqs. (16)–(18). With mutation, new genetic materials can be introduced into the population. Mutation should be used sparingly because it is a random search operator. In the following simulations, a mutation rate was set to 0.3.

The aforementioned steps are done repeatedly and stopped when the predetermined condition is achieved.

Recently, several researchers [3,10,19] try to emphasize the readability and human-understandable of the neuro-fuzzy network. In [3], the authors proposed a neuro-fuzzy network to generated human-understandable fuzzy knowledge based. In order to obtain readable knowledge from data, they proposed a new neuro-fuzzy model and its learning algorithm that works in a parameter space with reduced dimensionality. The dimensionality of the new parameter space is necessary and sufficient to generate human-understandable fuzzy rules, in the sense formally defined by a set of properties. The learning procedure is based on a gradient descent technique and the proposed model is general enough to be applied to other neuro-fuzzy architectures. In this paper, we use the SCA method to determine the number of

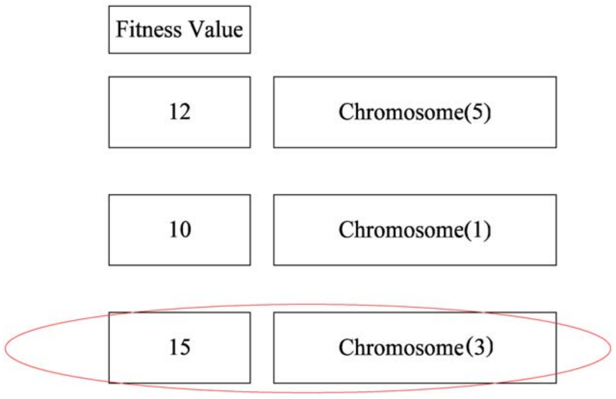

Fig. 9. The tournament selection.

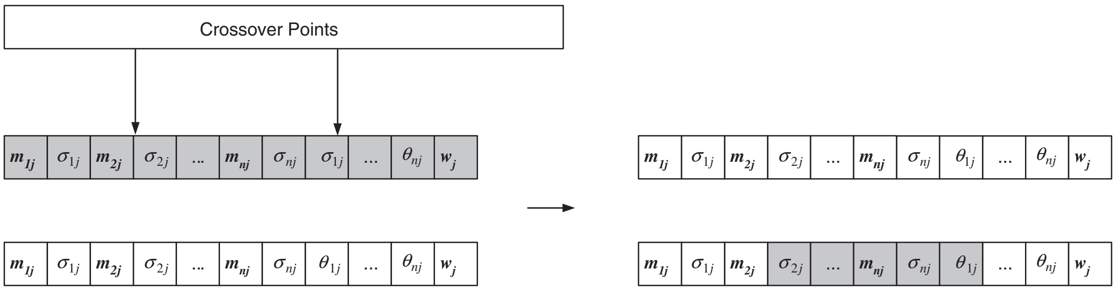
Crossover Points
$m_{1j}$ $\sigma_{1j}$ $m_{2j}$ $\sigma_{2j}$ ... $m_{nj}$ $\sigma_{nj}$ $\theta_{1j}$ ... $\theta_{nj}$ $w_j$

Fig. 10. Two-point crossover operation.

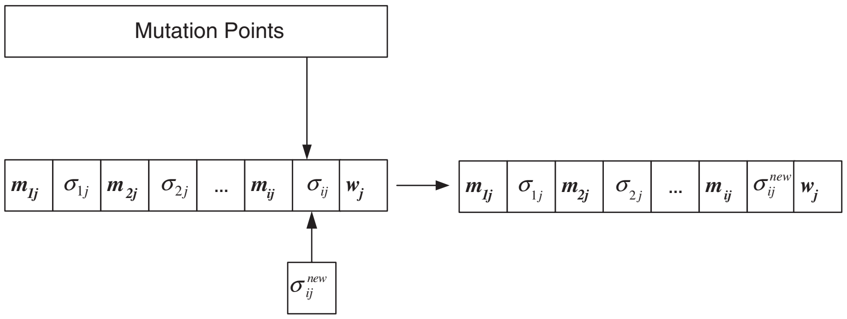
Mutation Points
$m_{1j}$ $\sigma_{1j}$ $m_{2j}$ $\sigma_{2j}$ ... $m_{ij}$ $\sigma_{ij}$ $w_j$
$\sigma_{ij}^{new}$

Fig. 11. Mutation operation.

fuzzy rules and the GSE method to adjust the free parameters of fuzzy rules.

## 4. Illustrative examples

This section discusses three different examples that were used to evaluate the SANFN-GSE model. The first example was performed to predict a chaotic signal that is described in [19], the second example was run to predict the chaotic time series [6], and the third example was a sunspot number forecasting [21]. For the three computer simulations, the initial parameters are given in Table 1 before training.

Table 1
The initial parameters before training

| Parameters | Value |
|---|---|
| Population size | 200 |
| Fuzzy systems | 200 |
| Crossover rate | 0.5 |
| Mutation rate | 0.3 |
| Coding type | Real number |

Table 2
Means and dimension distances after SCA clustering

| Cluster | $Cc_{1\_j}$ | $CD_{1\_j}$ |
|---|---|---|
| 1 | 0.0274 | 0.0264 |
| 2 | 0.235 | 0.0538 |
| 3 | 0.5614 | 0.1288 |
| 4 | 0.90689 | 0.0429 |
| 5 | 0.77894 | 0.0623 |
| 6 | 0.32243 | 0.0084 |

## 4.1. Example 1: Prediction of a chaotic signal

In this example, the proposed SANFN-GSE model was used to predict a chaotic signal. The classical time series prediction problem is a one-step-ahead prediction that is described in [19]. The following equations describe the logistic function:

$$x(k+1) = ax(k)(1 - x(k)), \tag{21}$$

The behavior of the time series generated by this equation depends critically upon the value of the parameter $a$. If $a < 1$, the system has a single fixed point at the origin, and from a random initial value between [0, 1] the time series collapses to a constant value. For $a > 3$, the system generates a periodic attractor. Beyond the value $a = 3.6$, the system becomes chaotic. In this study, we set the parameter value $a$ to 3.8. The first 60 pairs (from $x(1)$ to $x(60)$), with the initial value $x(1) = 0.001$, were the training data set, while the remaining 100 pairs (from $x(1)$ to $x(100)$), with initial value $x(1) = 0.9$, were the testing data set used for validating the proposed method.

The proposed prediction is that we will find several chromosomes that form a SANFN-GSE to maximize the fitness value. In this example we set the threshold value in SCA to 0.2. Table 2 shows the results after the SCA clustering method was used, where $Cc_{i\_j}$ and $D_{i\_j}$ represent the mean and dimension distance with $i$th input dimension and $j$th cluster.

The learning stage entered parameter learning through GSE method. The evolution learning processed for 30 generations and was repeated 50 times. After 30 generations, the final average rms error of the predicted output approximates 0.00081. After 30 generations of training, the final six fuzzy rules were generated. They are given as follows:

*Rule* 1: *IF* $x_1$ *is* $\mu(-0.167857, 0.210462)$ *THEN* $y' = -0.541397$
*Rule* 2: *IF* $x_1$ *is* $\mu(0.645200, 0.141451)$ *THEN* $y' = 0.071760$
*Rule* 3: *IF* $x_1$ *is* $\mu(1.011675, 0.134370)$ *THEN* $y' = -0.321057$
*Rule* 4: *IF* $x_1$ *is* $\mu(0.390224, 0.294088)$ *THEN* $y' = 0.319729$
*Rule* 5: *IF* $x_1$ *is* $\mu(0.322841, 0.012615)$ *THEN* $y' = -0.011077$
*Rule* 6: *IF* $x_1$ *is* $\mu(0.581367, 0.572351)$ *THEN* $y' = 0.657562$,

where $\mu(m_{ij}, \sigma_{ij})$ represents a Gaussian membership function with mean $m_{ij}$ and deviation $\sigma_{ij}$ in $i$th input dimension and $j$th rule. Fig. 12(a) shows the prediction results of the desired output and the model output through the learning process of 30 generations. The solid line represents the desired output of the time series, and the notation "*" represents the output of the SANFN-GSE model. The prediction errors of the proposed method are shown in Fig. 12(b). The experimental results demonstrate the perfect prediction capability of the SANFN-GSE model.

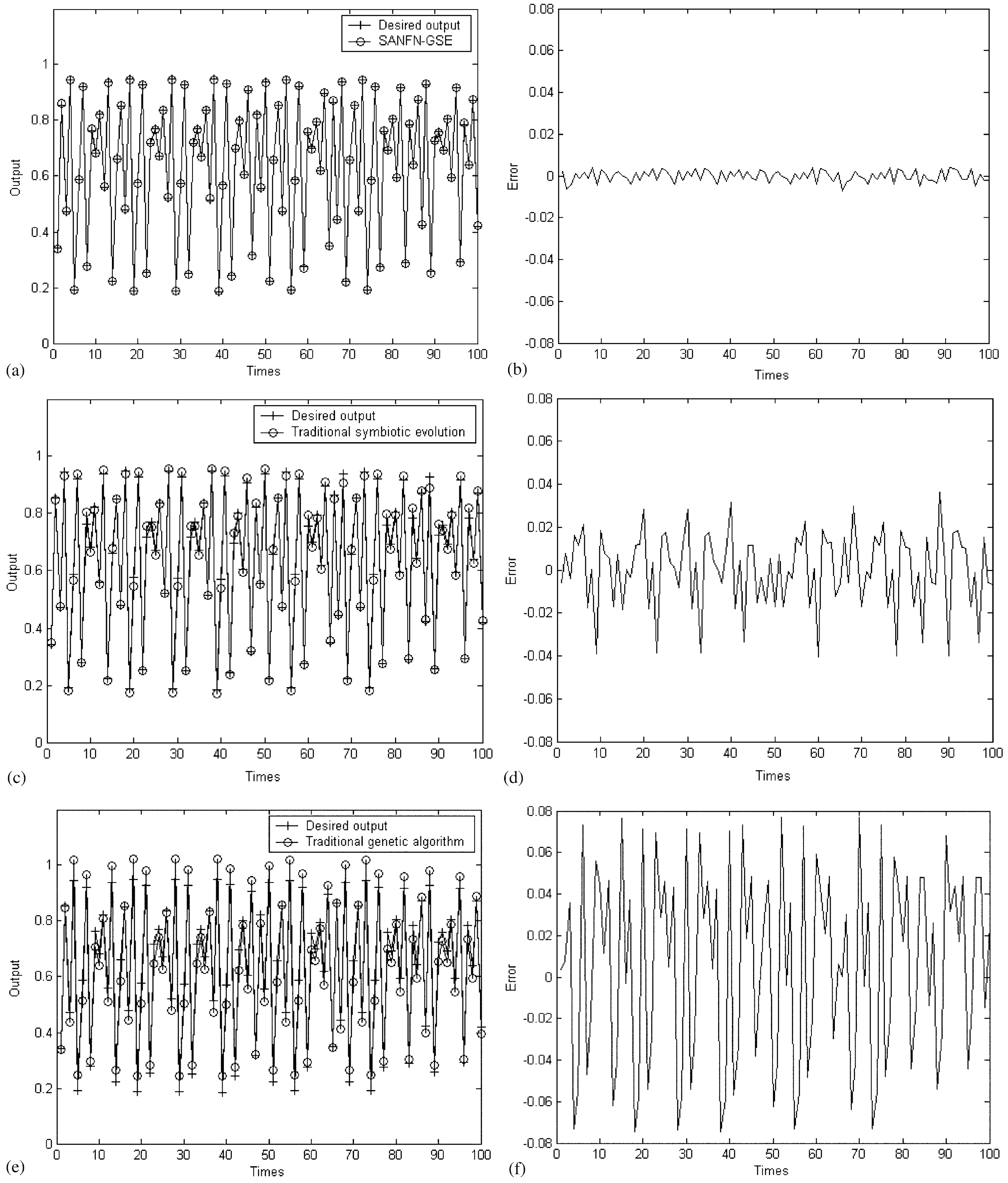

Fig. 12. (a) The prediction results of the proposed method. (b) The prediction errors of the proposed method. (c) The prediction results of the traditional symbiotic evolution model [12]. (d) The prediction errors of the traditional symbiotic evolution model [12]. (e) The prediction results of the traditional genetic algorithm [21]. (f) The prediction errors of the traditional genetic algorithm [21].

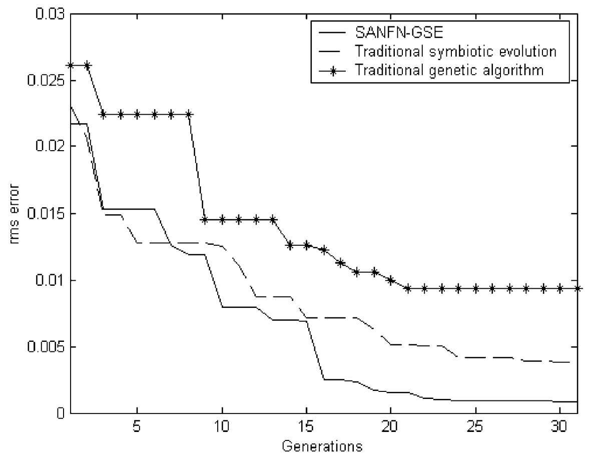

Fig. 13. The learning curves of the proposed method, the traditional symbiotic evolution model [12], and the traditional genetic algorithm [21].

In this example, in order to show the effectiveness and efficiency of the proposed SANFN-GSE model, we applied the traditional symbiotic evolution (TSE) [12] and traditional genetic algorithm (TGA) [21] to the same problem. In the TSE and TGA, the population sizes were set to 200, and the crossover and mutation probabilities were set to 0.5 and 0.3, respectively. We set six rules to construct the fuzzy model. In the TSE [12] and TGA [1], the evolution learning processed for 30 generations and was repeated 50 times.

We now discuss the performance of the SANFN-GSE model compared with the performance of other methods [12,21]. First, we compared the performance of the SANFN-GSE model with that of the traditional symbiotic evolution model [12]. Fig. 12(c) shows the prediction results of the traditional symbiotic evolution model. The prediction errors of the traditional symbiotic evolution model are shown in Fig. 12(d). Second, the traditional genetic algorithm [21] that adopts binary code is used for the prediction problem. Figs. 12(e) and (f) show the prediction results and the prediction errors of the traditional genetic algorithm. As shown in Fig. 13, the prediction results of the SANFN-GSE model are better than those of other methods.

Fig. 12 shows the learning curves of the SANFN-GSE model, the TSE [12], and the TGA [21]. In this figure, we find that the proposed method converges quickly and obtains a lower rms error than other models. Computer simulations demonstrated that the proposed method performs better than other existing models.

## 4.2. *Example 2: Prediction of the chaotic time series*

The Mackey-Glass chaotic time series $x(t)$ in consideration here was generated from the following delay differential equation:

$$\frac{\mathrm{d}x(t)}{\mathrm{d}t} = \frac{0.2x(t-\tau)}{1+x^{10}(t-\tau)} - 0.1x(t). \tag{22}$$

Crowder [6] extracted 1000 input–output data pairs $\{x, y^d\}$ which consist of four past values of $x(t)$, i.e.

$$[x(t-18), x(t-12), x(t-6), x(t); x(t+6)], \tag{23}$$

where $\tau = 17$ and $x(0) = 1.2$. There were four inputs to the SANFN-GSE model, corresponding to these values of $x(t)$, and one output representing the value $x(t+\Delta t)$, where $\Delta t$ is a time prediction into the future. The first 500 pairs (from $x(1)$ to $x(500)$) were the training data set, while the remaining 500 pairs (from $x(501)$ to $x(1000)$) were the testing data set used for validating the proposed method. In this example, we set the threshold value in SCA to 0.4. Table 3 shows the mean and deviation values after the SCA clustering method was used, where $Cc_{i\_j}$ and $D_{i\_j}$ represent the mean and dimension distance with *i*th input dimension and *j*th cluster.

Table 3
Means and dimension distances after clustering by SCA

| Cluster | $Cc_{1\_i}$ | $Cc_{2\_j}$ | $Cc_{3\_j}$ | $Cc_{4\_j}$ | $CD_{1\_j}$ | $CD_{2\_j}$ | $CD_{3\_j}$ | $CD_{4\_j}$ |
|---|---|---|---|---|---|---|---|---|
| 1 | 1.001 | 0.9601 | 0.9187 | 0.8793 | 0.0904 | 0.0907 | 0.0855 | 0.0770 |
| 2 | 0.7840 | 0.6323 | 0.7124 | 0.7475 | 0.1167 | 0.1045 | 0.0912 | 0.1165 |
| 3 | 0.8603 | 0.9112 | 0.9592 | 1.002 | 0.1103 | 0.1031 | 0.0929 | 0.0815 |
| 4 | 1.1365 | 1.1592 | 1.1679 | 1.1581 | 0.122 | 0.1070 | 0.0910 | 0.0778 |
| 5 | 1.1125 | 1.1034 | 1.1334 | 0.9919 | 0.1208 | 0.0762 | 0.0675 | 0.1087 |
| 6 | 0.9318 | 0.8663 | 0.8160 | 0.7832 | 0.0721 | 0.0603 | 0.0607 | 0.0759 |
| 7 | 0.7043 | 0.6654 | 0.6233 | 0.6149 | 0.0967 | 0.0994 | 0.0958 | 0.1219 |
| 8 | 0.6794 | 0.7629 | 0.8121 | 0.8234 | 0.0771 | 0.0587 | 0.0654 | 0.0995 |
| 9 | 1.2400 | 1.2619 | 1.2427 | 1.1932 | 0.0558 | 0.0428 | 0.0586 | 0.0891 |
| 10 | 0.5321 | 0.4853 | 0.5243 | 0.5582 | 0.0822 | 0.0509 | 0.0938 | 0.1163 |
| 11 | 0.5817 | 0.6185 | 0.6656 | 0.7647 | 0.0843 | 0.0696 | 0.0517 | 0.0792 |

The learning phase entered parameter learning through GSE method. The evolution learning processed for 500 generations and was repeated 50 times. After 500 generations of training, the final fuzzy rules were obtained. They are described as follows:

*Rule* 1: *IF* $x_1$ *is* $\mu(1.147840, 0.039380)$ *and* $x_2$ *is* $\mu(1.030099, 1.158860)$ *and* $x_3$ *is* $\mu(1.183048, 0.377265)$ *and* $x_4$ *is* $\mu(1.135862, 0.695881)$ *THEN* $y' = -0.004662$

*Rule* 2: *IF* $x_1$ *is* $\mu(0.988884, 0.131503)$ *and* $x_2$ *is* $\mu(0.617742, 0.094698)$ *and* $x_3$ *is* $\mu(0.723764, 0.946031)$ *and* $x_4$ *is* $\mu(0.674479, 0.297900)$ *THEN* $y' = -1.909980$

*Rule* 3: *IF* $x_1$ *is* $\mu(0.493016, 0.505969)$ *and* $x_2$ *is* $\mu(0.603224,0.301429)$ *and* $x_3$ *is* $\mu(0.797574, 1.036234)$ *and* $x_4$ *is* $\mu(0.963893, 0.818812)$ *THEN* $y' = 0.121602$

*Rule* 4: *IF* $x_1$ *is* $\mu(1.254618, 1.188000)$ *and* $x_2$ *is* $\mu(1.292633, 1.424605)$ *and* $x_3$ *is* $\mu(1.332230, 1.184361)$ *and* $x_4$ *is* $\mu(1.392626, 0.412278)$ *THEN* $y' = 0.565143$

*Rule* 5: *IF* $x_1$ *is* $\mu(1.092013, 0.857488)$ *and* $x_2$ *is* $\mu(0.826633, 1.359844)$ *and* $x_3$ *is* $\mu(1.016145, 1.289026)$ *and* $x_4$ *is* $\mu(1.150432, 0.785982)$ *THEN* $y' = 0.216318$

*Rule* 6: *IF* $x_1$ *is* $\mu(1.080345, 0.294261)$ *and* $x_2$ *is* $\mu(0.999115,0.172435)$ *and* $x_3$ *is* $\mu(0.696495, 0.271626)$ *and* $x_4$ *is* $\mu(0.993496, 0.186253)$ *THEN* $y' = 0.210649$

*Rule* 7: *IF* $x_1$ *is* $\mu(0.707328, 1.035837)$ *and* $x_2$ *is* $\mu(0.712613, 0.379692)$ *and* $x_3$ *is* $\mu(0.858039, 0.944639)$ *and* $x_4$ *is* $\mu(0.881292, 0.911557)$ *THEN* $y' = 0.214920$

*Rule* 8: *IF* $x_1$ *is* $\mu(0.611153, 0.518064)$ *and* $x_2$ *is* $\mu(0.721121, 1.046751)$ *and* $x_3$ *is* $\mu(0.763731, 0.229884)$ *and* $x_4$ *is* $\mu(0.900132, 0.636204)$ *THEN* $y' = 0.136141$

*Rule* 9: *IF* $x_1$ *is* $\mu(1.113252, 1.348987)$ *and* $x_2$ *is* $\mu(1.187067, 0.784869)$ *and* $x_3$ *is* $\mu(1.229577, 1.414364)$ *and* $x_4$ *is* $\mu(1.297621, 0.593058)$ *THEN* $y' = 0.560923$

*Rule*10: *IF* $x_1$ *is* $\mu(0.397857, 0.177363)$ *and* $x_2$ *is* $\mu(0.400806, 0.469635)$ *and* $x_3$ *is* $\mu(0.501227, 0.208031)$ *and* $x_4$ *is* $\mu(0.882006, 0.711170)$ *THEN* $y' = 0.311094$

*Rule*11: *IF* $x_1$ *is* $\mu(0.353253, 0.470454)$ *and* $x_2$ *is* $\mu(0.468391, 0.938213)$ *and* $x_3$ *is* $\mu(0.685677, 0.707227)$ *and* $x_4$ *is* $\mu(1.038509, 0.912712)$ *THEN* $y' = 0.168185$,

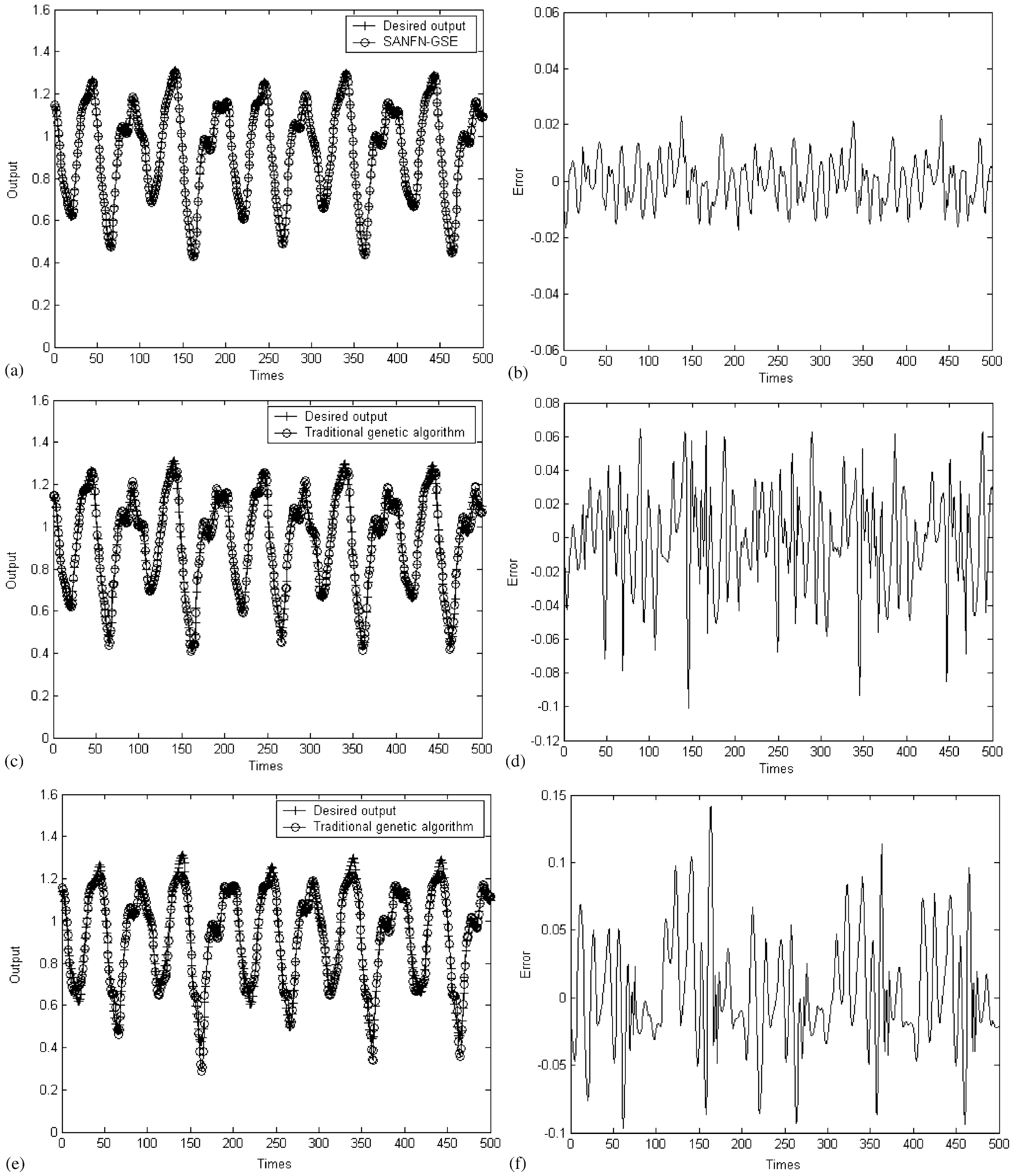

Fig. 14. (a) The prediction results of the proposed method. (b) The prediction errors of the proposed method. (c) The prediction results of the traditional symbiotic evolution model [12]. (d) The prediction errors of the traditional symbiotic evolution model [12]. (e) The prediction results of the traditional genetic algorithm [21], (f) The prediction errors of the traditional genetic algorithm [21].

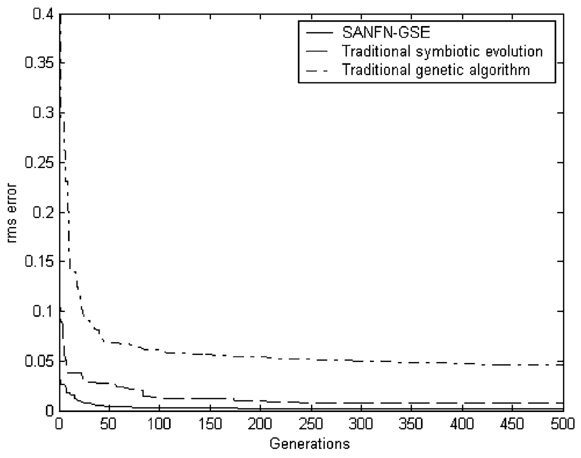

Fig. 15. The learning curves of the proposed method, the traditional symbiotic evolution model [12], and the traditional genetic algorithm [21].

where $\mu(m_{ij}, \sigma_{ij})$ represents a Gaussian membership function with mean $m_{ij}$ and deviation $\sigma_{ij}$ in $i$th input dimension and $j$th rule. The final rms error of the prediction output approximates 0.0015. Fig. 14(a) shows the prediction outputs of the chaotic time series from $x(501)$ to $x(1000)$, when 500 training data from $x(1)$ to $x(500)$ were used. The solid line represents the desired output while the dotted line represents the model output. The prediction errors between the proposed model and the desired output are shown in Fig. 14(b).

In this example, as with example 1, we also compared the performance of the SANFN-GSE model with other methods [12,21]. In [21] and [12], the parameters are the same with example 1. We set eleven rules to construct the fuzzy model. The evolution learning processed for 500 generations and was repeated 50 times. Figs. 14(c) and (d) show the prediction results and the prediction errors of the traditional symbiotic evolution model [12]. Figs. 14(e) and (f) show the prediction results and the prediction errors of the traditional genetic algorithm [21]. The learning curves of the proposed SANFN-GSE model, the traditional symbiotic evolution model, and the traditional genetic algorithm are shown in Fig. 15. We find that the proposed method obtains better prediction results than other existing models.

For comparative analysis, we shall use the normalized root mean square error (NRMSE) [23], which is defined as the RMSE divided by the standard deviation of the desired output:

$$\mathrm{NRMSE} = \frac{1}{\sigma_t}\left[\frac{1}{N_t}\sum_{l=1}^{N_t}(Y_l(t+6) - Y_l^d(t+6))^2\right]^{1/2}, \tag{24}$$

where $\sigma_t^2$ is the estimated variance of the data, $N_t$ the number of the training data, $Y^d(t+6) = x(t+6)$ is the desired value, and $Y(t+6)$ is the predicted value by the model with four inputs and one output. Table 4 lists the generalization capabilities of other methods [6,15]. The generalization capabilities were measured by using each model to predict 500 points immediately following the training data set. Clearly, simulation results show that the proposed model can obtain smaller NRMSE than other methods.

### 4.3. Example 3: Forecast of the sunspot number

The sunspot numbers from 1700 to 2004 exhibit nonlinear, nonstationary, and non-Gaussian cycles that are difficult to predict [21]. In this example, we use the proposed SANFN-GSE model for forecasting the sunspot number. The inputs $x_i$ of the proposed SANFN-GSE model are defined as $x_1(t) = y_1^d(t-1)$, $x_2(t) = y_1^d(t-2)$, and, $x_3(t) = y_1^d(t-3)$ where $t$ represents the year and $y_1^d(t)$ is the sunspot numbers at the $t$ year. In this example, the first 180 years (from 1705 to 1884) of the sunspot numbers were used to train the proposed SANFN-GSE model while the remaining 119 years (from 1885 to 2004) of the sunspot numbers were the used to test the proposed SANFN-GSE model.

Table 4
Performance comparison of various existing models

| Method | Training cases | $\text{NRMSE}_{\text{train}}$ |
|---|---|---|
| SANFN-GSE | 500 | 0.008 |
| Back-propagation NN | 500 | 0.02 |
| Traditional symbiotic evolution | 500 | 0.032 |
| Six-order polynomial | 500 | 0.04 |
| Cascaded-correlation | 500 | 0.06 |
| Traditional genetic algorithm | 500 | 0.26 |
| Linear predictive | 2000 | 0.55 |

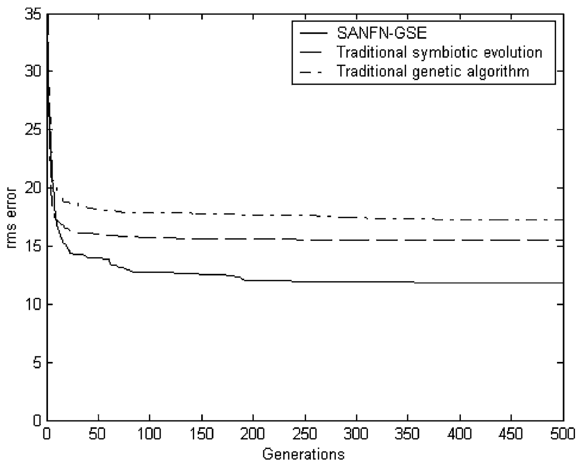

Fig. 16. The learning curves of the proposed method, the traditional symbiotic evolution model [12], and the traditional genetic algorithm [21].

Table 5
Performance comparison of various existing models

| Method | Training cases | RMSE | Training error | Forecasting error |
|---|---|---|---|---|
| SANFN-GSE | 500 | 12.12 | 8.75 | 12.35 |
| Symbiotic evolution [12] | 500 | 15.26 | 10.05 | 15.05 |
| Genetic algorithm [21] | 500 | 17.47 | 12.27 | 19.81 |

In this example, we set the threshold value in SCA to 70. After clustering by SCA, the nine clusters (fuzzy rules) were obtained. The learning phase entered parameter learning through GSE method. The evolution learning processed for 500 generations and was repeated 50 times. After 500 generations of training, the final average rms error of the prediction output approximates 12.12.

In this example, as with examples 1 and 2, we also compared the performance of the SANFN-GSE model with the TGA [21] and TSE [12] method. In [21] and [12], the parameters are the same with examples 1 and 2. We set nine rules to construct the fuzzy model. The evolution learning processed for 500 generations and was repeated 50 times. The learning curves of the proposed SANFN-GSE model, the TSE, and the TGA are shown in Fig. 16.

Table 5 tabulated the rms error (governed by Eq. (19)), the training error (governed by $\sum_{t=1705}^{1884} |y_1^d(t) - y_1(t)|/180$), and the forecasting error (governed by $\sum_{t=1885}^{2004} |y_1^d(t) - y_1(t)|/119$). As shown in Table 5, the proposed SANFN-GSE model performs a better performance than other models.

## 5. Conclusion

In this paper, a novel hybrid learning algorithm was proposed. First, a structure learning scheme is used in the algorithm to determine proper input space partitioning and to find the mean and deviation of each cluster. With the proposed self-clustering algorithm (SCA) method, a flexible partitioning of the input space is achieved. The number of rules is relatively small compared to the grid partition. Second, a supervised learning scheme called group-based symbiotic evolution algorithm (GSE) is proposed to adjust the parameters for the desired outputs. In the GSE method, there are several groups in the population. Each group represents a set of the chromosomes that belong to a cluster computing by SCA. The advantages of the proposed SANFN-GSE model are summarized as follows: (1) the proposed SCA method can online cluster the input partitions and considers the variation of each dimension for the input data; (2) the GSE method used group-based population to evaluate the fuzzy rule locally; (3) the initial and mutational value was generated according to each clusters' mean and deviation computing by the SCA; (4) the SANFN-GSE model converges more quickly than existing evolution methods. In our simulations, the proposed SANFN-GSE model can obtain a smaller rms error and a quicker convergence than other methods for time series prediction problems.

## Acknowledgements

This research is supported by the National Science Council of R.O.C. under grant NSC 93-2213-E-324-008.

## References

[1] K.K. Ang, C. Quek, M. Pasquier, POPFNN-CRI(S): Pseudo outer product based fuzzy neural network using the compositional rule of inference and singleton fuzzifier, IEEE Trans. Systems Man Cybern. B 33 (6) (2003) 838–849.
[2] J.E. Box, G.M. Jenkins, Time Series Analysis, Forecasting and Control, Holden Day, 1970.
[3] G. Castellano, A.M. Fanelli, C. Mencar, A neuro-fuzzy network to generate human-understandable knowledge from data, Cognitive Systems Res. J. 3 (2) (2002) 125–144.
[4] J.H. Chiang, P.Y. Hao, A new kernel-based fuzzy clustering approach: support vector clustering with cell growing, IEEE Trans. Fuzzy Systems 11 (4) (2003) 518–527.
[5] O. Cordon, F. Herrera, F. Hoffmann, L. Magdalena, Genetic fuzzy systems evolutionary tuning and learning of fuzzy knowledge bases, in: L.A. Zadeh (Honorary Ed.), Advances in Fuzzy Systems—Applications and Theory, vol. 19, World Scientific Publishing, NJ, 2001.
[6] R.S. Cowder III, in: D. Touretzky, G. Hinton, T. Sejnowski (Eds.), Predicting the Mackey-glass Time Series with Cascade-Correlation Learning, 1990, pp. 117–123.
[7] J. He, L. Liu, G. Palm, Speaker identification using hybrid LVQ-SLP networks, in: Proceedings of the IEEE International Conference on Neural Networks, vol. 4, December 1995, pp. 2052–2055.
[8] M.C. Hung, D.L. Yang, The efficient fuzzy c-means clustering technique, in: Proceedings of the IEEE International Conference on Data Mining, December 2001, pp. 225–232.
[9] A. Homaifar, E. McCormick, Simultaneous design of membership functions and rule sets for fuzzy controllers using genetic algorithms, IEEE Trans. Fuzzy Systems 3 (9) (1995) 129–139.
[10] J.-S.R. Jang, ANFIS: Adaptive-network-based fuzzy inference system, IEEE Trans. Systems Man Cybern. 23 (1993) 665–685.
[11] C.F. Juang, A TSK-type recurrent fuzzy network for dynamic systems processing by neural network and genetic algorithms, IEEE Trans. Fuzzy Systems 10 (2) (2002) 155–170.
[12] C.F. Juang, J.Y. Lin, C.T. Lin, Genetic reinforcement learning through symbiotic evolution for fuzzy controller design, IEEE Trans. Systems Man Cybern. B 30 (2) (2000) 290–302.
[13] C.L. Karr, Design of an adaptive fuzzy logic controller using a genetic algorithm, in: Proceedings of the Fourth Conference on Genetic Algorithms, 1991, pp. 450–457.
[14] N.K. Kasabov, Q. Song, DENFIS: Dynamic evolving neural-fuzzy inference system and its application for time-series prediction, IEEE Trans. Fuzzy Systems 10 (2) (2002) 144–154.
[15] A.S. Lapeds, R. Farber, Nonlinear signal processing using neural network: prediction and system modelling, Tech Rep. LA-UR-87-2662, Los Alamos Nat. Lab., Los Alamos, New Mexico, 1987.
[16] M.A. Lee, H. Takagi, Integrating design stages of fuzzy systems using genetic algorithms, in: Proceedings of the IEEE International Conference on Fuzzy Systems, New York, vol. 1, April 1993, pp. 612–617.
[17] M. Li, K. Mehrotra, C. Mohan, S. Ranka, Sunspot numbers forecasting using neural networks, in: Proceedings of the IEEE International Conference on Intelligent Control, vol. 1, September 1990, pp. 524–529.
[18] C.J. Lin, C.H. Chen, Nonlinear system control using compensatory neuro-fuzzy networks, IEICE Trans. Fundamentals E86-A (9) (2003) 2309–2316.
[19] C.T. Lin, C.S.G. Lee, Neural Fuzzy Systems: A Neural-fuzzy Synergism to Intelligent Systems, Prentice-Hall, Englewood Cliffs, NJ, May 1996.

[20] F.J. Lin, C.H. Lin, P.H. Shen, Self-constructing fuzzy neural network speed controller for permanent-magnet synchronous motor drive, IEEE Trans. Fuzzy Systems 9 (5) (2001) 751–759.
[21] S.H. Ling, H. Frank, F. Leung, H.K. Lam, Y.-S. Lee, P.K.S. Tam, A novel genetic-algorithm-based neural network for short-term load forecasting, IEEE Trans. Industrial Electornic. 50 (4) (2003) 793–799.
[22] D.E. Moriarty, R. Miikkulainen, Efficient reinforcement learning through symbiotic evolution, Mach. Learn. 22 (1996) 11–32.
[23] J.H. Nie, T.H. Lee, Rule-based modeling: Fast construction and optimal manipulation, IEEE Trans. Systems Man Cybern. A 26 (6) (1996) 728–738.
[24] M. Segeno, On stability of fuzzy systems expressed by fuzzy rules with singleton consequents, IEEE Trans. Fuzzy Systems 7 (2) (1999) 201–224.
[25] R.E. Smith, S. Forrest, A.S. Perelson, Searching for diverse, cooperative populations with genetic algorithms, Evol. Comput. 1 (2) (1993) 127–149.
[26] H. Tong, Non-linear Time Series: A Dynamical System Approach, Oxford University Press, Oxford, 1990.
[27] W.L. Tung, C. Quek, GenSoFNN: A generic self-organizing fuzzy neural network, IEEE Trans. Neural Networks 13 (5) (2002) 1075–1086.