Expert Systems with Applications 37 (2010) 5320–5330

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Multi groups cooperation based symbiotic evolution for TSK-type neuro-fuzzy systems design

Yung-Chi Hsu, Sheng-Fuu Lin *, Yi-Chang Cheng

Department of Electrical and Control Engineering, National Chiao-Tung University, 1001 Ta Hsueh Road, Hsinchu 300, Taiwan, ROC

ARTICLE INFO

Keywords:
Genetic algorithms
Symbiotic evolution
Chaotic time series
Neural fuzzy system

ABSTRACT

In this paper, a TSK-type neuro-fuzzy system with multi groups cooperation based symbiotic evolution method (TNFS-MGCSE) is proposed. The TNFS-MGCSE is developed from symbiotic evolution. The symbiotic evolution is different from traditional GAs (genetic algorithms) that each chromosome in symbiotic evolution represents a rule of fuzzy model. The MGCSE is different from the traditional symbiotic evolution; with a population in MGCSE is divided into several groups. Each group formed by a set of chromosomes represents a fuzzy rule and cooperate with other groups to generate the better chromosomes by using the proposed cooperation based crossover strategy (CCS). In this paper, the proposed TNFS-MGCSE is used to evaluate by numerical examples (Mackey-Glass chaotic time series and sunspot number forecasting). The performance of the TNFS-MGCSE achieves excellently with other existing models in the simulations.

1. Introduction

In recent years, a fuzzy system uses for several problems has become a popular research topic (Jang, 1993; Juang & Lin, 1998; Lin & Lee, 1996; Lin & Lin, 1997; Lin, Lin, & Shen, 2001; Mizutani & Jang, 1995; Takagi & Sugeno, 1985; Takagi, Suzuki, Koda, & Kojima, 1992; Towell & Shavlik, 1993; Wang & Mendel, 1992). The reason is that classical control theory usually requires a mathematical model for designing controllers. Inaccurate mathematical modeling of plants usually degrades the performance of the controllers, especially for nonlinear and complex problems (Juang & Lin, 1999; Lin & Chin, 2004; Mastorocostas & Theocharis, 2002; Narendra & Parthasarathy, 1990). A fuzzy system consists of a set of fuzzy if-then rules. Conventionally, the selection of fuzzy if-then rules often relies on a substantial amount of heuristic observations to express the knowledge of proper strategies. Obviously, it is difficult for human experts to examine all the input–output data from a complex system to find proper rules for a fuzzy system. To cope with this difficulty, several approaches for generating if-then rules from numerical data have been proposed (Lin & Lin, 1997; Juang & Lin, 1998; Towell & Shavlik, 1993). These methods are developed for supervised learning; that is, the correct "target" output values are given for each input pattern to guide the network's learning.

The most well-known supervised learning algorithm is backpropagation (BP) (Jang, 1993; Juang & Lin, 1998; Lin & Lin, 1997; Lin et al., 2001; Mizutani & Jang, 1995; Takagi et al., 1992). It is a powerful training technique that can be applied to networks. Since the steepest descent technique is used in BP training to minimize the error function, the algorithm may reach the local minima but never find the global solution. In addition, the performance of BP training depends on the initial values of the system parameters, and for different network topologies one has to derive new mathematical expressions for each network layer.

Considering the disadvantages mention above, one may face 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, evolutionary algorithms appear to be better candidates than backpropagation algorithm.

Several evolutionary algorithms, such as genetic algorithm (GA) (Goldberg, 1989), genetic programming (Koza, 1992), evolutionary programming (Fogel, 1994), and evolution strategies (Rechenberg, 1994), have been proposed. They are parallel and global search techniques. Because they simultaneously evaluate many points in the search space, they are more likely to converge toward the global solution. For this reason, an evolutionary method using for training the fuzzy model has become an important field.

The evolutionary fuzzy model generates a fuzzy system automatically by incorporating evolutionary learning procedures (Bandyopadhyay, Murthy, & Pal, 2000; Belarbi & Titel, 2000; Carse, Fogarty, & Munro, 1996; Homaifar & McCormick, 1995; Juang, 2004; Karr, 1991; Lee & Takagi, 1993; Tang, 1996; Yi-Ta Wu, Yoo Jung An, Geller, & Yih-Tyng Wu, 2006), where the well-known

* Corresponding author.
E-mail addresses: ottocheng.ece94g@nctu.edu.tw (Y.-C. Cheng), ericbogi2001@yahoo.com.tw (Y.-C. Hsu), sflin@mail.nctu.edu.tw (S.-F. Lin).

0957-4174/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2010.01.003

procedure is the genetic algorithms (GAs). Several genetic fuzzy models, that is, fuzzy models augment by a learning process based on GAs, have been proposed (Belarbi & Titel, 2000; Homaifar & McCormick, 1995; Juang, 2004; Lee & Takagi, 1993; Karr, 1991; Tang, 1996). Karr (1991) applied GAs to the design of the membership functions of a fuzzy controller, with the fuzzy rule set assigned in advance. Since the membership functions and rule sets are co-dependent, simultaneous design of these two approaches will be a more appropriate methodology. Based on this concept, many researchers have applied GAs to optimize both the parameters of the membership functions and the rule sets (Belarbi & Titel, 2000; Lee & Takagi, 1993; Juang, 2004; Lee & Takagi, 1993). Tang proposed a hierarchical genetic algorithm (HGA) (Tang, 1996). Carse et al.used the genetic algorithm to evolve fuzzy rule based controllers (Carse et al., 1996). The hierarchical genetic algorithm enables the optimization of the fuzzy system design for a particular application. Bandyopadhyay et al. used the variable-length genetic algorithm (VGA) that let the different lengths of the chromosomes in the population (Bandyopadhyay et al., 2000). Wu et al. proposed a data mining based GA algorithm to efficiently improve the Traditional GA by using analyzing support and confidence parameters (Wu et al., 2006).

However, these approaches encounter one or more of the following major problems: (1) all the fuzzy rules are encoded into one chromosome and (2) the population cannot evaluate each fuzzy rule locally.

Recently, Gomez and Schmidhuber proposed lots of work to solve above problems (Gomez, 2003, 2005). The proposed enforced sub-populations (ESP) used sub-populations of neurons for the fitness evaluation and overall control. As shown in Gomez's and Schmidhuber's work, the sub-populations that use to evaluate the solution locally can obtain better performance compared to systems of only one population is used to evaluate the solution.

Same with ESP, in this paper, a TSK-type neuro-fuzzy system with multi groups cooperation based symbiotic evolution (TNFS-MGCSE) is proposed for solving the problems that mention above. In TNFS-MGCSE, each chromosome represents only one fuzzy rule and an n-rules fuzzy system is constructed by selecting and combining n chromosomes from several groups. The TNFS-MGCSE is developed from symbiotic evolution. The symbiotic evolution is different from traditional GAs (genetic algorithms) that each chromosome in symbiotic evolution represents a rule of fuzzy model. In TNFS-MGCSE, compared with normal symbiotic evolution, there are several groups in the population. Each group formed by a set of chromosomes represents a fuzzy rule. Compare with the ESP, for allowing the well-performing groups of individuals cooperate to generate better generation, a cooperation based crossover strategy (CCS) is proposed in this paper. In CCS, each group will cooperate to perform the crossover steps. Therefore, the better chromosomes of each group will be selected to perform crossover in the next generation. The TNFS-MGCSE promotes both cooperation and specialization, ensures diversity and prevents a population from converging to suboptimal solutions.

The advantages of the proposed TNFS-MGCSE are summarized as follows: (1) the TNFS-MGCSE uses multi groups in a population to evaluate the fuzzy rule locally. (2) The TNFS-MGCSE uses CCS to let the better solutions form different groups can cooperate with each other for generating better solutions. (3) It indeed can obtain better performance and converge more quickly than some traditional genetic methods.

This paper is organized as follows. The TSK-type neuro-fuzzy system (TNFS) is introduced in Section 2. The group cooperation based symbiotic evolution (GCSE) is described in Section 3. The simulation results are presented in Section 4. The conclusions are summarized in the last section.

2. Structure of TSK-type neuro-fuzzy system (TNFS)

A TSK-type neuro-fuzzy system (TNFS) (Lin & Lee, 1996) employs different implication and aggregation methods from a standard Mamdani fuzzy system. Instead of using fuzzy sets, the conclusion part of a rule is a linear combination of the crisp inputs.

$$\textbf{IF } x_1 \textbf{ is } A_{1j}(m_{1j},\sigma_{1j}) \textbf{ and } x_2 \textbf{ is } A_{2j}(m_{2j},\ \sigma_{2j}) \ldots \textbf{ and } x_n \textbf{ is } A_{nj}(m_{nj},\sigma_{nj})$$
$$\textbf{THEN } y' = w_{0j} + w_{1j}x_1 + \cdots + w_{nj}x_n \quad (1)$$

The structure of a TNFS is shown in Fig. 1, where n and R are the number of input dimensions and the number of rules, respectively. It is a five-layer network structure. In the proposed TNFS, the firing strength of a fuzzy rule is calculated by performing the following "AND" operation on the truth values of each variable to its corresponding fuzzy sets by:

$$u_{ij}^{(3)} = \prod_{i=1}^{n} \exp\left(-\frac{\left[u_i^{(1)} - m_{ij}\right]^2}{\sigma_{ij}^2}\right) \quad (2)$$

where $u_i^{(1)} = x_i$ and $u_{ij}^{(3)}$ are the output of 1th and 3th layers ; m_{ij} and σ_{ij} are the center and the width of the Gaussian membership function of the jth term of the ith input variable x_i, respectively.

The output of a fuzzy system is computed by:

$$y = u^{(5)} = \frac{\sum_{j=1}^{R} u_j^{(4)}}{\sum_{j=1}^{R} u_j^{(3)}} = \frac{\sum_{j=1}^{R} u_j^{(3)}(w_{0j} + \sum_{i=1}^{n} w_{ij}x_i)}{\sum_{j=1}^{R} u_j^{(3)}} \quad (3)$$

where $u^{(5)}$ is the output of 5th layer ; w_{ij} is the weighting value with ith dimension and jth rule node; M is the number of fuzzy rule.

3. Multi groups cooperation based symbiotic evolution

The proposed multi groups cooperation based symbiotic evolution (MGCSE) will be introduced in this section. The MGCSE is proposed for improving the symbiotic GA (Moriarty & Miikkulainen, 1996). In the proposed MGCSE, the algorithm is developed from symbiotic evolution. The idea of symbiotic evolution was first proposed in an implicit fitness-sharing algorithm that is used in an immune system model (Moriarty & Miikkulainen, 1996). The authors developed artificial antibodies to identify artificial antigens. Because each antibody can match only one antigen, a different population of antibodies is required to effectively defend against a variety of antigens. As shown in symbiotic evolution, 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 find solutions in different, unconverted populations (Juang, Lin, & Lin, 2000; Moriarty & Miikkulainen, 1996). In MGCSE, compared with normal symbiotic evolution, there are several groups in the population. Each group formed by a set of chromosomes represents a fuzzy rule.

In MGCSE, each group represents a set of chromosomes that belong to a fuzzy rule. The structure of the chromosome in MGCSE is shown in Fig. 2.

In MGCSE, the coding structure of the chromosomes must be suitable for the concept of each chromosome represents only one fuzzy rule. A fuzzy rule with the form introduced in Eq. (1) is described in Fig. 3. As shown in this figure m_{ij} and σ_{ij} represent a Gaussian membership function with mean and deviation with ith dimension and jth rule node. The coding type of MGCSE is real-value code.

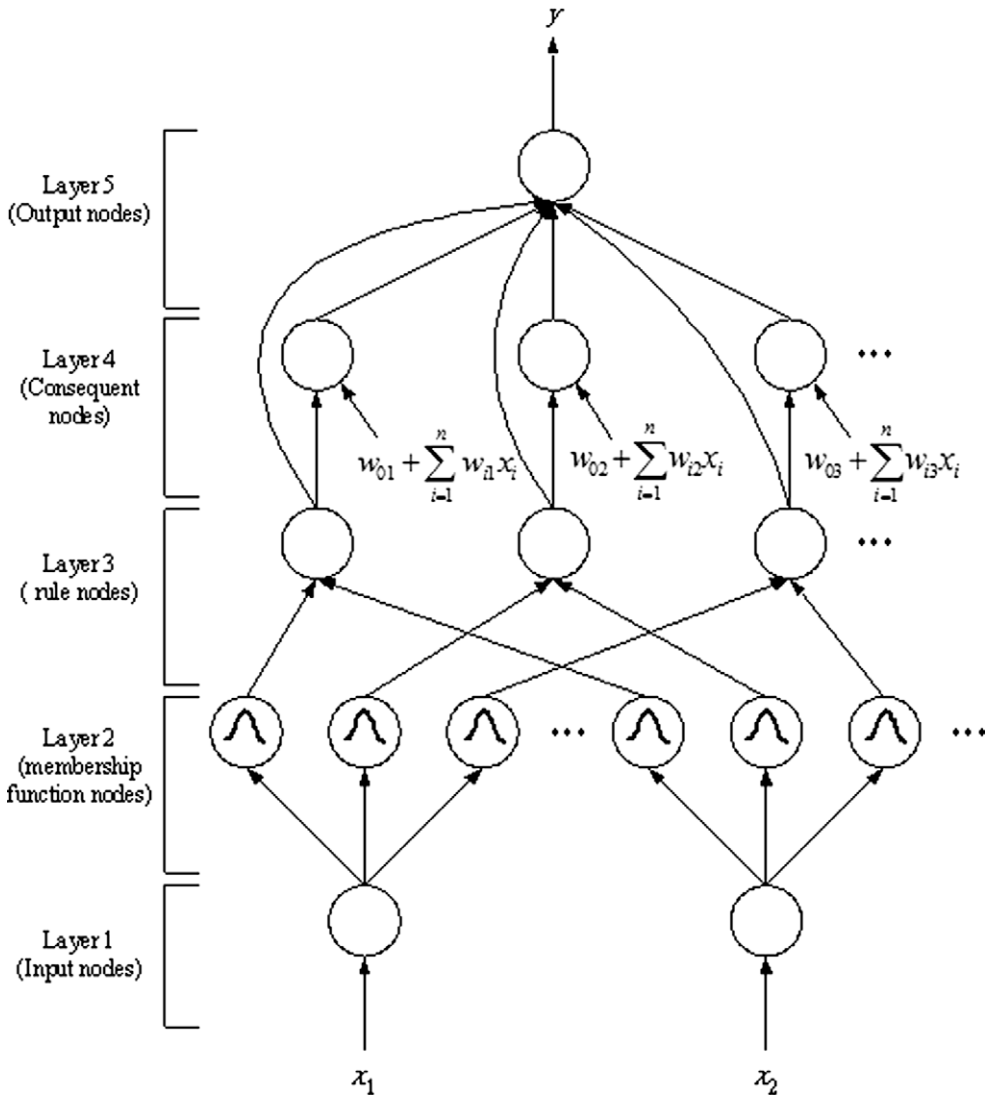

Fig. 1. Structure of the TSK-type neuro-fuzzy system.

The learning process of MGCSE in each group involves five major operators: initialization, fitness assignment, elite-based reproduction strategy (ERS), cooperation based crossover strategy (CCS), and mutation strategy. The whole learning process is described step-by-step as follows:

(a) *Initialization step*:

Before the TNFS-MGCSE is designed, individuals forming several initial groups should be generated. The initial groups of MGCSE are generated randomly within a predefined range. The following formulations show how to generate the initial chromosomes in each group:

$$\text{Deviation}: \ Chr_{g,c}[p] = random[\sigma_{\min}, \sigma_{\max}]$$
$$\text{where } p = 2,4,\ldots,2n; \quad g = 1,2,\ldots,R; \quad c = 1,2,\ldots,N_C. \tag{4}$$
$$\text{Mean}: \ Chr_{g,c}[p] = random[m_{\min}, m_{\max}]$$
$$\text{where } p = 1,3,\ldots,2n-1. \tag{5}$$
$$\text{Weight}: \ Chr_{g,c}[p] = random[w_{\min}, w_{\max}]$$
$$\text{where } p = 2n+1, 2n+2, \ldots, 2n+(1+n). \tag{6}$$

where $Chr_{g,c}$ represents cth chromosome in gth group; R represents total number of groups and N_C is the total number of chromosomes in each group; p represents the pth gene in a $Chr_{g,c}$; and $[\sigma_{\min}, \sigma_{\max}]$, $[m_{\min}, m_{\max}]$, and $[w_{\min}, w_{\max}]$ represent the predefined range.

(b) *Fitness assignment step*:

As previously state, in MGCSE, 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 N_f TNFS systems with R fuzzy rules from the R groups with size N_C.
- *Step 2*: Evaluate every TFNS that is generated from step1 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 each group has been selected a sufficient number of times, and record the number of TNFS systems in which each individual has participated.

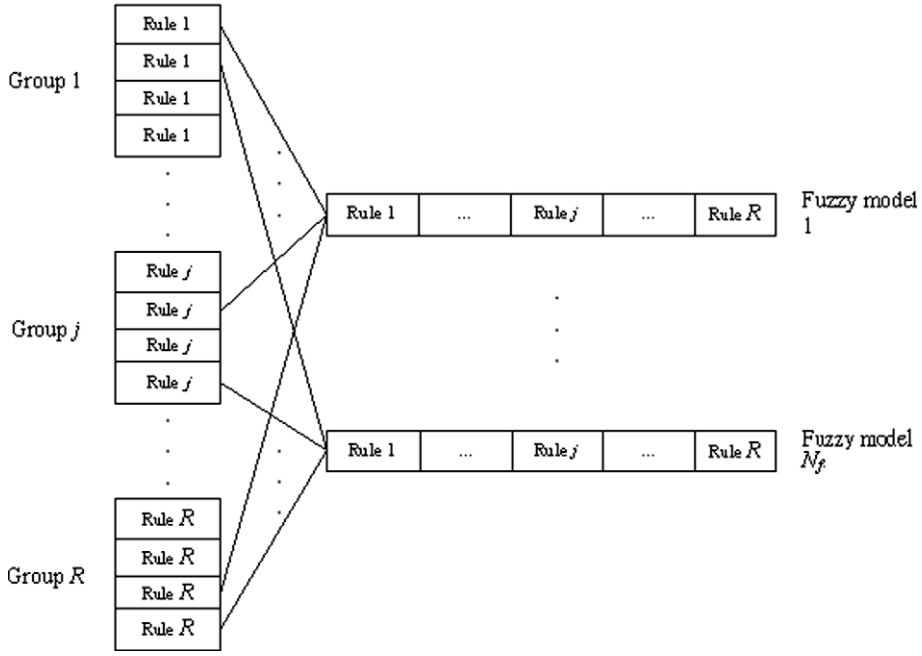

Fig. 2. The structure of the chromosome in the MGCSE.

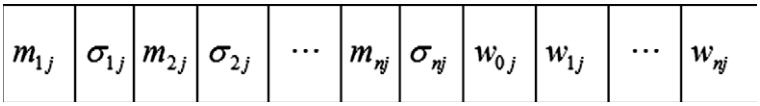

Fig. 3. Coding a rule of a TNFS into a chromosome in MGCSE.

- *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:

$$\text{FitnessValue} = 1/(1 + E(y, \bar{y})), \tag{7}$$

$$\text{where } E(y, \bar{y}) = \sum_{i=1}^{N} (y_i - \bar{y}_i)^2 \tag{8}$$

where y_i represents the desired value of the ith 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 (individual).

(c) *Elites-based Reproduction Strategy (ERS)*:

Reproduction is a process in which individuals are copied according to their fitness values. A fitness value is assigned to each chromosome according to a fitness assignment step in which high values denote a good fit. The goal of the MGCSE is to maximize the fitness value. For keeping the stability, this study proposes an elite-based reproduction strategy (ERS) to allow the best combination of chromosomes can be kept in the next generation. In MGCSE, the chromosome with the best fitness value may not be in the best combination. Therefore, every chromosome in the best combination must be kept by applying ERS. Other chromosomes in each group are selected under roulette-wheel selection method (Cordon, Herrera, Hoffmann, & Magdalena, 2001) – a simulated roulette is spun. The best performing chromosomes in the top half of each group advance to the next generation (Juang, Lin, & Lin, 2000). The other half is generated by applying crossover and mutation operations on chromosomes in the top half of the parent generation. In the reproduction step, the top half of each group must be kept the same number of chromosomes.

(d) *Cooperation based crossover strategy (CCS)*:

Although the ERS operation can search for the best existing individuals, it 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 occurs for a selected pair under a crossover rate. In this paper, a cooperation based crossover strategy (CCS) is proposed for allowing groups cooperates with each other. The CCS mimics the cooperation phenomenon in society, in which individuals become more suited to the environment as they acquire and share more knowledge of their surroundings. In CCS, the best performing individuals in the top half of each group called elites are used to select the parents for performing CCS. Details of CCS are shown below.

- *Step 1*: The first parent used to perform the crossover operation is selected from the original group by using the following equations:

$$Fitness\_Ratio_{g,t} = \frac{\sum_{u=1}^{t} fitness_{g,u}}{\sum_{c=1}^{Nc} fitness_{g,c}}, \quad \text{where} \quad t = 1, 2, \ldots, Nc. \tag{9}$$

$$Rand\_Value[g] = Random[0, 1], \quad \text{where } g = 1, 2, \cdots, R. \tag{10}$$

$$Parent\_SiteA[g] = t, \quad \text{if} \quad Fitness\_Ratio_{g,t-1} < Rand\_Value[g] \leqslant Fitness\_Ratio_{g,t} \tag{11}$$

where $Fitness\_Ratio_{g,t}$ is a fitness ratio of tth chromosome in the gth group; $Rand\_Value[g] \in [0,1]$ is the random values of gth group; $Parent\_SiteA[g]$ is the site of first parent. According to Eq. (11), if the $Rand\_Value[g]$ is greater than the fitness ratio at $(t-1)$th chromosome in gth group and equal to or smaller than the fitness ratio at tth chromosome in gth group, the site of the first parent of gth group is assigned to t.

- *Step 2*: After determining the first parent, the best performing elites in every group is used to determine the second parent. In this step, the total fitness ratio of every group is computed according to the following equations:

$$Total\_Fitness_g = \sum_{c=1}^{Nc} fitness_{g,c}, \quad \text{where } g = 1,2,\ldots,R. \tag{12}$$

$$Total\_Fitness\_Ratio_w = \frac{\sum_{u=1}^{w} Total\_Fitness_u}{\sum_{g=1}^{R} Total\_Fitness_g}, \quad \text{where } w = 1,2,\ldots,R \tag{13}$$

where $Total\_Fitness_g$ represents the summation of all chromosomes' fitness value in gth group; $Total\_Fitness\_Ratio_w$ is a total fitness ratio of wth group.

- *Step 3*: Determine the second parental group for applying crossover with the $Parent_SiteA[g]$th chromosome in gth group according to the following equations:

$$Group\_Rand\_Value[g] = Random[0,1] \quad \text{where } g = 1,2,\ldots,R \tag{14}$$

$$Parent\_Group\_SiteB[g] = w, \quad \text{if} \tag{15}$$

$$Total\_Fitness\_Ratio_{w-1} < Group\_Rand\_Value[g] \leqslant Total\_Fitness\_Ratio_w$$

where $Group\_Rand\_Value[g] \in [0,1]$ is a random values of gth group; $Parent\_Group\_SiteB[g]$ represents the site of the group where the second parent is selected from.

- *Step 4*: After the $Parent_Group_SiteB[g]$th group is selected, the CCS determines the other present in the selected $Parent_Group_SiteB[g]$th group according to the following equations:

$$Fitness\_Ratio_{Selected\_g,t} = \frac{\sum_{u=1}^{t} fitness_{Selected\_g,u}}{\sum_{c=1}^{Nc} fitness_{Selected\_g,c}} \tag{16}$$

$$\text{where } t = 1,2,\ldots,Nc; Selected\_g = Parent\_Group\_SiteB[g]$$

$$Rand\_Value[g] = Random[0,1], \quad \text{where } g = 1,2,\ldots,R \tag{17}$$

$$Parent\_SiteB[g] = l, \text{if}$$

$$Fitness\_Ratio_{Selected\_g,l-1} < Rand\_Value[g] \leqslant Fitness\_Ratio_{Selected\_g,l} \tag{18}$$

where $Fitness\_Ratio_{Selected\_g,t}$ is a fitness ratio of tth chromosome in the $Parent_Group_SiteB[g]$th group; and $Parent\_SiteB[g]$ is the site of the second parent. The pseudo code of CCS is listed in Fig. 4.After selecting the parents from the gth group and $Parent_Group_SiteB[g]$th group by CCS, the individuals ($Parent_SiteA[g]$th chromosome and the $Parent_SiteB[g]$th chromosome) are crossed and separated by using a two-point crossover (Cordon et al., 2001) in the gth group, as shown in Fig. 5. In this figure, exchanging the site's values between the selected sites of parents' individual create new individuals. After this operation, the individuals with poor performances are replaced by the newly produced offspring.

(e) *Mutation strategy*:

Although ERS and CCS would produce many new strings, they do not introduce any new information to the population at the site of an individual. Mutation can randomly alter the allele of a gene. In this paper, to emphasize the capability of the CCS, the proposed MGCSE tries to simplify the mutation operation. Therefore, a uniform mutation (Cordon et al., 2001) is adopted, and the mutated gene is generated randomly from the domain of the corresponding variable.

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

4. Illustrative examples

Four examples are discussed in this section. The first example was run to predict the chaotic time series (Cowder, 1990) and the second example was a sunspot number forecasting (Ling, Leung, Lam, Lee, & Tam, 2003). For the two computer simulations, the initial parameters are given in Table 1 before training. The initial parameters are determined by practical experimentation or trial-and-error tests.

4.1. Prediction of the chaotic time series

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

$$\frac{dx(t)}{dt} = \frac{0.2x(t-\tau)}{1+x^{10}(t-\tau)} - 0.1x(t) \tag{19}$$

Cowder (1990) extracted 1000 input–output data pairs $\{x, y^d\}$ which consisted of four past values of $x(t)$, i.e.

$$[x(t-18), x(t-12), x(t-6), x(t); x(t+6)] \tag{20}$$

where $\tau = 17$ and $x(0) = 1.2$. There are four inputs to the RWNFS-CEGSE, corresponding to these values of $x(t)$, and one output representing the value $x(t+\Delta t)$, where Δt is a time prediction into the future. The first 500 pairs (from $x(1)$ to $x(500)$) are the training data set, while the remaining 500 pairs (from $x(501)$ to $x(1000)$) are the testing data set that is used for validating the proposed method. The values are floating-point numbers assigned using the MGCSE initially. The fitness function in this example is defined in Eqs. (7) and (8) to train TNFS. There are six fuzzy rules used to construct TNFS. The evolution learning processed for 500 generations and is repeated 50 times. For comparative analysis, this paper uses the normalized root mean square error (NRMSE) [37], which is defined as the RMSE divides by the standard deviation of the desired output:

$$\text{NRMSE} = \frac{1}{\sigma_t}\left[\frac{1}{N_t}\sum_{l=1}^{N_t}(Y_l(t+6) - Y_l^d(t+6))^2\right]^{1/2} \tag{21}$$

where σ_t^2 is the estimated variance of the data, N_t is 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. After 50 runs, the final average NRMS error of the predicted output approximates 0.0051. The outputs of TNFS-MGCSE in one of 50 runs are shown in Fig. 6(a). The notation "o" represents the desired output of the time series, and the notation "*" represents the output of the six models.

In this example, in order to demonstrate the effectiveness and efficiency of the proposed R-GCSE, the symbiotic evolution (SE) (Moriarty & Miikkulainen, 1996), genetic algorithm (GA) (Karr, 1991), and enforce sub-population (ESP) (Gomez, 2003) are applied to the same problem. There are six rules to construct the fuzzy model. The parameters set for three methods are as follows: (1) the numbers of fuzzy rules are all set for 6; (2) the population sizes of SE and GA are 100 and 50, respectively; (4) the N_c of the SE and ESP are both set for 50; (3) the crossover rates of SE, ESP, and GA are 0.55, 0.34, and 0.6, respectively; (3) the mutation rate of SE, ESP, and GA are 0.08, 0.14, and 0.12, respectively. The evolution learning processes for 500 generations and is repeated 50 times. After 50 runs, the final average NRMS error of the SE, ESP, and GA outputs approximate 0.021, 0.0084, and 0.032. The outputs of the three models in one of 50 runs are shown in Figs. 6(b)–(d).

```
Procedure of Cooperation based Crossover Strategy
Begin
  Let g=0,c=0,q=0;
  // Decide the parents used for performing the crossover
  Repeat
    q=q+1;
    //Decide the first parent
    Repeat
        g=g+1;
        Compute Parent_SiteA[g]  by (9) to (11);
    Until g=R;
    //Compute the Group fitness Ratio
    Let w=0;
    Repeat
        w=w+1;
        Compute  Total_Fitness_Ratio_w  by (12) and (13);
    Until w=R;
    //Compute the group that the second parent is selected from
    g=0;
    Repeat
        g=g+1;
        Compute  Parent_Group_SiteB[g]  by (14) and (15);
    Until g=R
    // Decide the Second parent;
    g=0;
    Repeat
        g=g+1;
        Compute  Parent_SiteB[g]  by (16) to (18);
    Until g=R;
  Until q=Nc/2;
End
```

Fig. 4. The pseudo code of CCS.

Fig. 7(a)–(d) illustrates the error between the desired and four methods' outputs. As shown in Fig. 6(a)–(d) and Fig. 7(a)–(d), the performance of the TNFS-MGCSE is better than those of others. The learning curves of the four methods in one of 50 runs are shown in Fig. 8. In Fig. 8, the proposed TNFS-MGCSE converges quickly and obtains a lower rms error than others.

Table 2 lists the generalization capabilities and CPU times of proposed method and other methods (Bandyopadhyay et al., 2000; Gomez, 2003; Karr, 1991; Moriarty & Miikkulainen, 1996; Tang, 1996). This experiment uses a Pentium III chip with a 400 MHz CPU, a 512 MB memory, and the visual C++ 6.0 simulation software. A total of thirty runs were performed. Clearly, Table 2

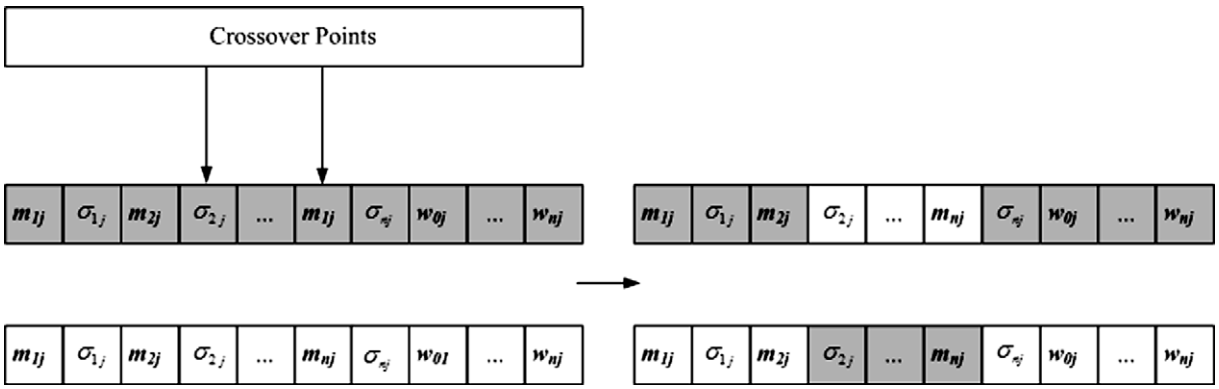

Fig. 5. Two-point crossover.

Table 1
The initial parameters before training.

Parameters	Value	Parameters	Value
N_f	20	$[\sigma_{\min}, \sigma_{\max}]$	[0,2]
N_c	50	$[m_{\min}, m_{\max}]$	[0,2]
Crossover rate	0.5	$[w_{\min}, w_{\max}]$	[−10,10]
Mutation rate	0.3		

shows that the proposed model can obtain shorter CPU time and NRMSE than other methods.

For demonstrating the efficiency of the proposed MGCSE, in this example, three different methods: the proposed MGCSE without ECCS (Type I), the SE method (Type II), and the proposed MGCSE (Type III) are used. In the Type I method, each group performs the two-point crossover strategy independently. In the Type II

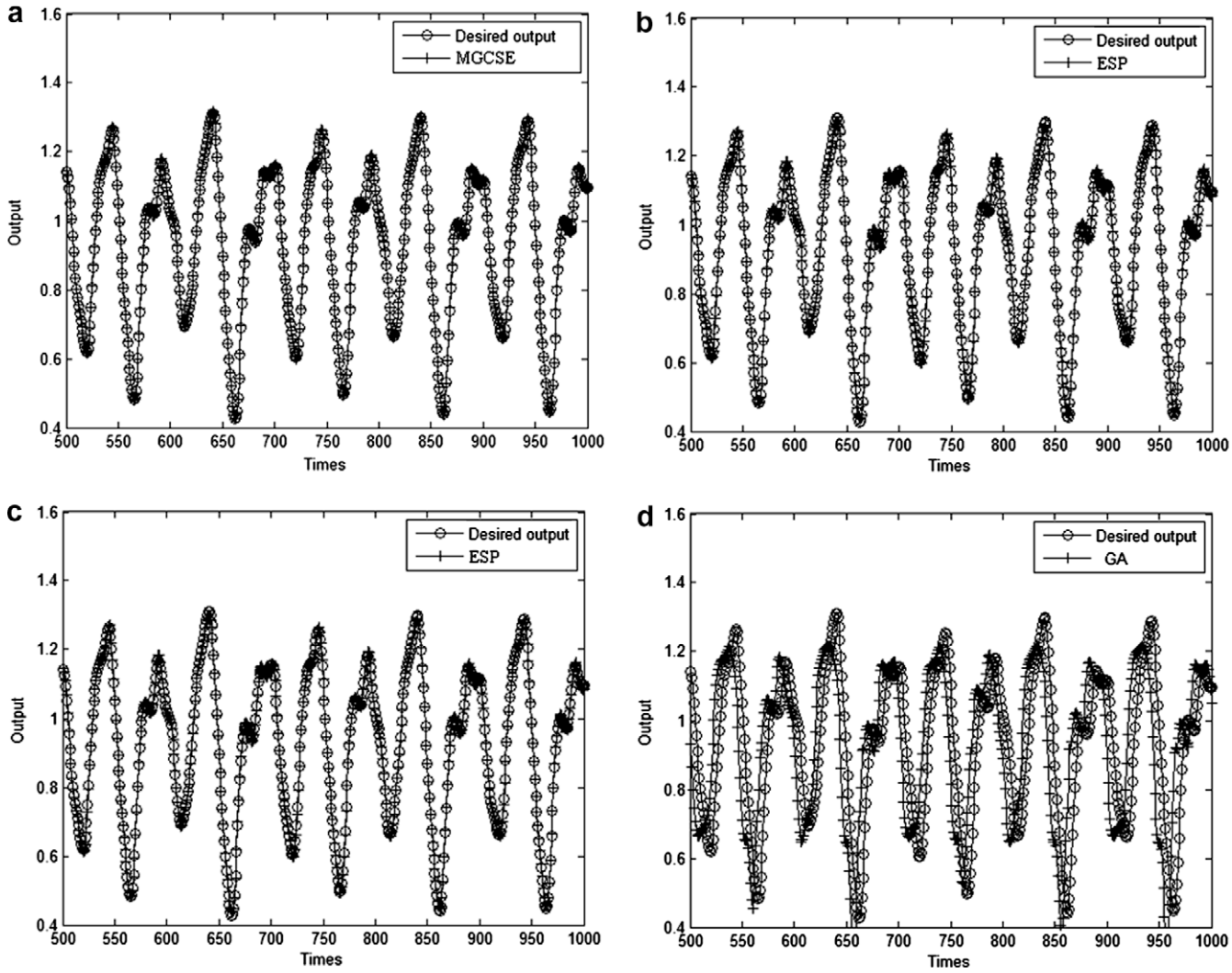

Fig. 6. The prediction results of the (a) MGCSE, (b) ESP, (c) SE, and (d) GA methods.

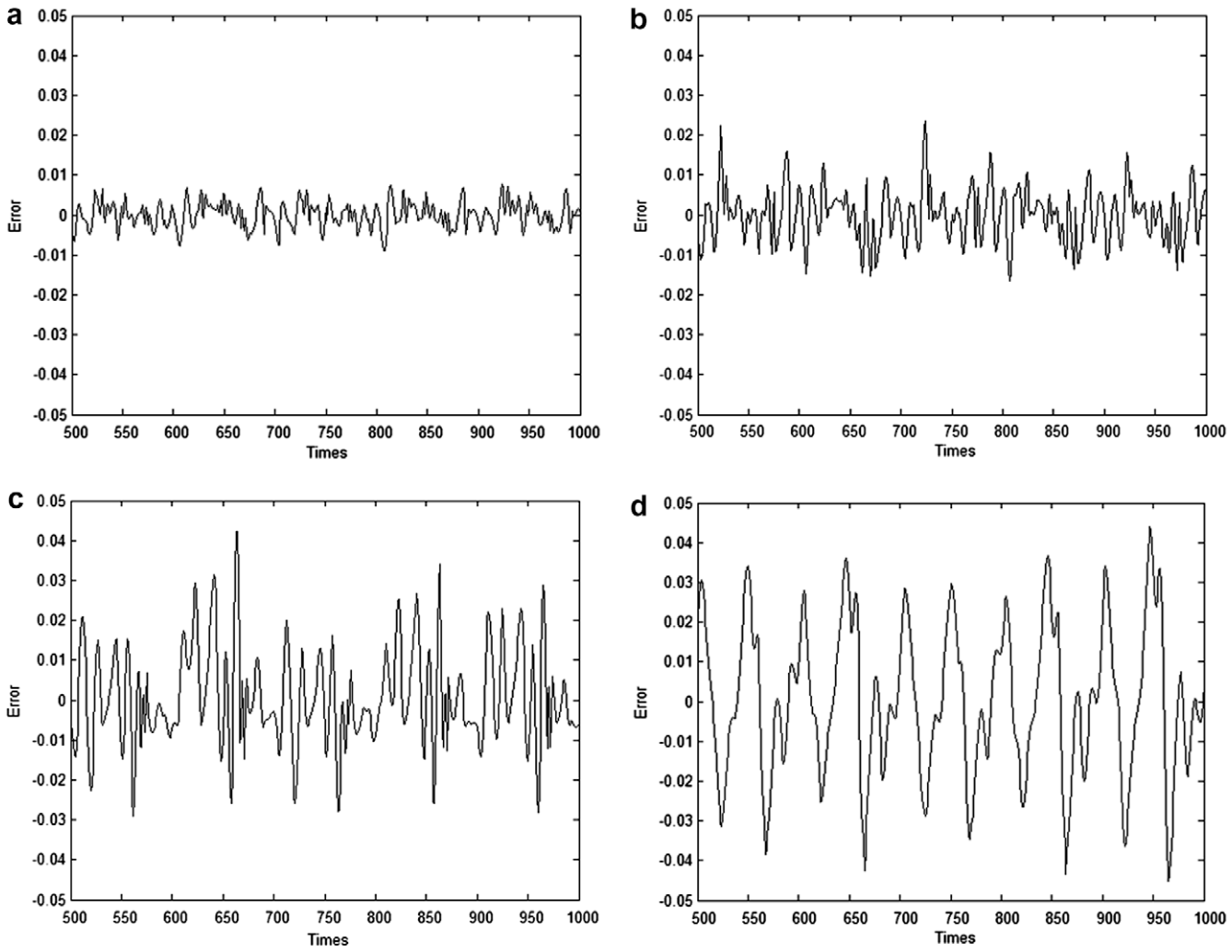

Fig. 7. The prediction errors of the (a) MGCSE, (b) ESP, (c) SE, and (d) GA methods.

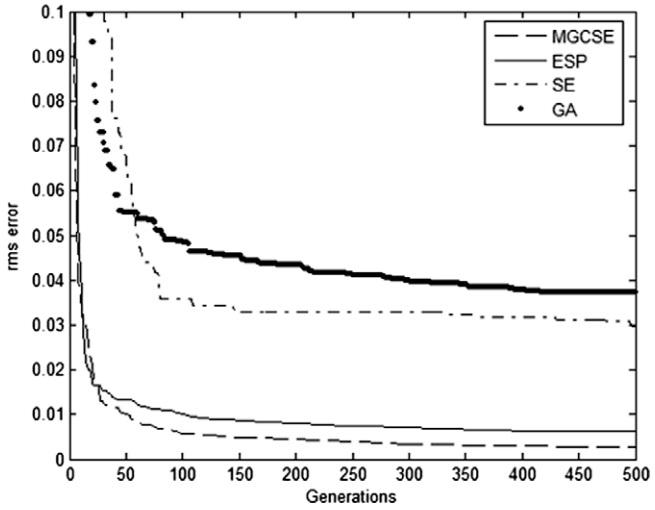

Fig. 8. The learning curves of the proposed method. ESP, SE, and GA.

method, the SE (Moriarty & Miikkulainen, 1996) is adopted. In the Type III method, the MGCSE uses the proposed CCS to perform crossover strategy. The performance (generalization capabilities and CPU time) of the three types of methods is shown in Table 3.

Table 2
Performance comparison of various existing models.

Method	Training cases	NRMSE			CPU time
		Best	Mean	Worst	
MGCSE	500	0.0038	0.0051	0.009	308.45
ESP[28]	500	0.0063	0.0084	0.014	306.48
VGA[26]	500	0.0095	0.0124	0.021	376.1
HGA[24]	500	0.0101	0.0185	0.029	394.66
SE[30]	500	0.0114	0.021	0.048	775.37
GA[19]	500	0.023	0.032	0.062	797.39

The proposed MGCSE (Type III) performs better than other types of methods. Comparing Type III with Type I method, it is observed that CCS improves the performance and reduces CPU time.

4.2. Forecasting the sunspot number

The sunspot numbers from 1700 to 2004 exhibit nonlinear, nonstationary, and non-Gaussian cycles that are difficult to predict (Ling et al., 2003). In this example, TNFS-MGCSE is used for forecasting the sunspot number. The inputs x_i of the proposed TNFS-MGCSE 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

Table 3
Comparison of performance for different methods.

Method	Training cases	NRMSE			CPU time
		Best	Mean	Worst	
Type I	500	0.0058	0.0091	0.013	305.17
Type II	500	0.0114	0.021	0.048	775.37
Type III	500	0.0038	0.0051	0.009	308.45

sunspot numbers at the tth year. In this example, the first 180 years (from 1705 to 1884) of the sunspot numbers are used to train TNFS-MGCSE while the remaining 119 years (from 1885 to 2004) of the sunspot numbers are used to test TNFS-MGCSE. The values are floating-point numbers assigned using the MGCSE initially. The fitness function in this example is defined in Eqs. (7) and (8) to train the TNFS. There are five fuzzy rules used to construct the TNFS. The evolution learning processes for 500 generations and is repeated 50 times. After 50 runs, the final average rms error of the prediction output approximates 8.21.

In this example, same with examples 1, the TNFS-MGCSE are also compared the performance with ESP (Gomez, 2003), SE (Moriarty & Miikkulainen, 1996), and GA (Karr, 1991). The parameters set for three methods are as follows: (1) the numbers of fuzzy rules are all set for 5; (2) the population sizes of SE and GA are 100 and 50, respectively; (4) the N_c of the SE and ESP are both set for 50; (3) the crossover rates of SE, ESP, and GA are 0.4, 0.25, and 0.35, respectively; (3) the mutation rate of SE, ESP, and GA are 0.06, 0.16, and 0.18, respectively. The evolution learning processes for 500 generations and is repeated 50 times. After 50 runs, the final average rms error of the SE, ESP, and GA outputs approximate 15.26, 12.12, and 17.47. The outputs of the four methods (MGCSE, ESP, SE, and GA) are shown in Fig. 9(a)–(d). The notation "o" represents the desired output of the time series, and the notation "*" represents the output of the six models. The errors between the desired and four methods' outputs are shown in Fig. 10(a)–(d). As shown in Figs. 9 and 10, the TNFS-MGCSE is better than those of others (Gomez, 2003; Karr, 1991; Moriarty & Miikkulainen, 1996). The learning curves of the four methods are shown in Fig. 11.

Table 4 lists the generalization capabilities and CPU time of other methods (Bandyopadhyay et al., 2000; Gomez, 2003; Karr, 1991; Moriarty & Miikkulainen, 1996; Tang, 1996). Table 5 tabulates the training error (governed by $\sum_{t=1705}^{1884} \frac{|y_1^d(t)-y_1^{(t)}|}{180}$), and the forecasting error (governed by $\sum_{t=1885}^{2004} \frac{|y_1^d(t)-y_1^{(t)}|}{119}$). Clearly, Tables 4 and 5 show that the proposed method can obtain better CPU time, rms error, training error, and forecasting error than other methods.

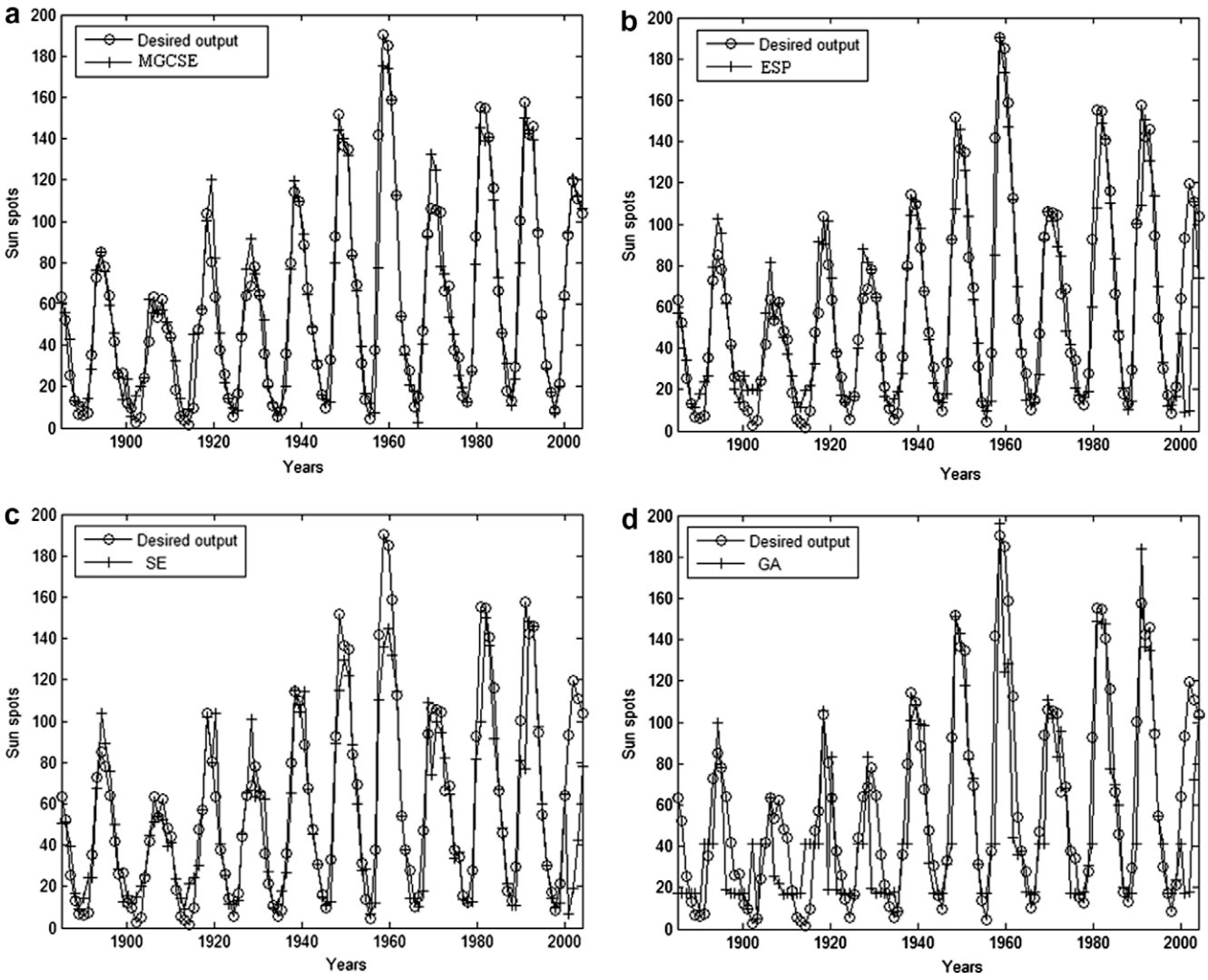

Fig. 9. The prediction results of the (a) MGCSE, (b) ESR, (c) SE, and (d) GA methods.

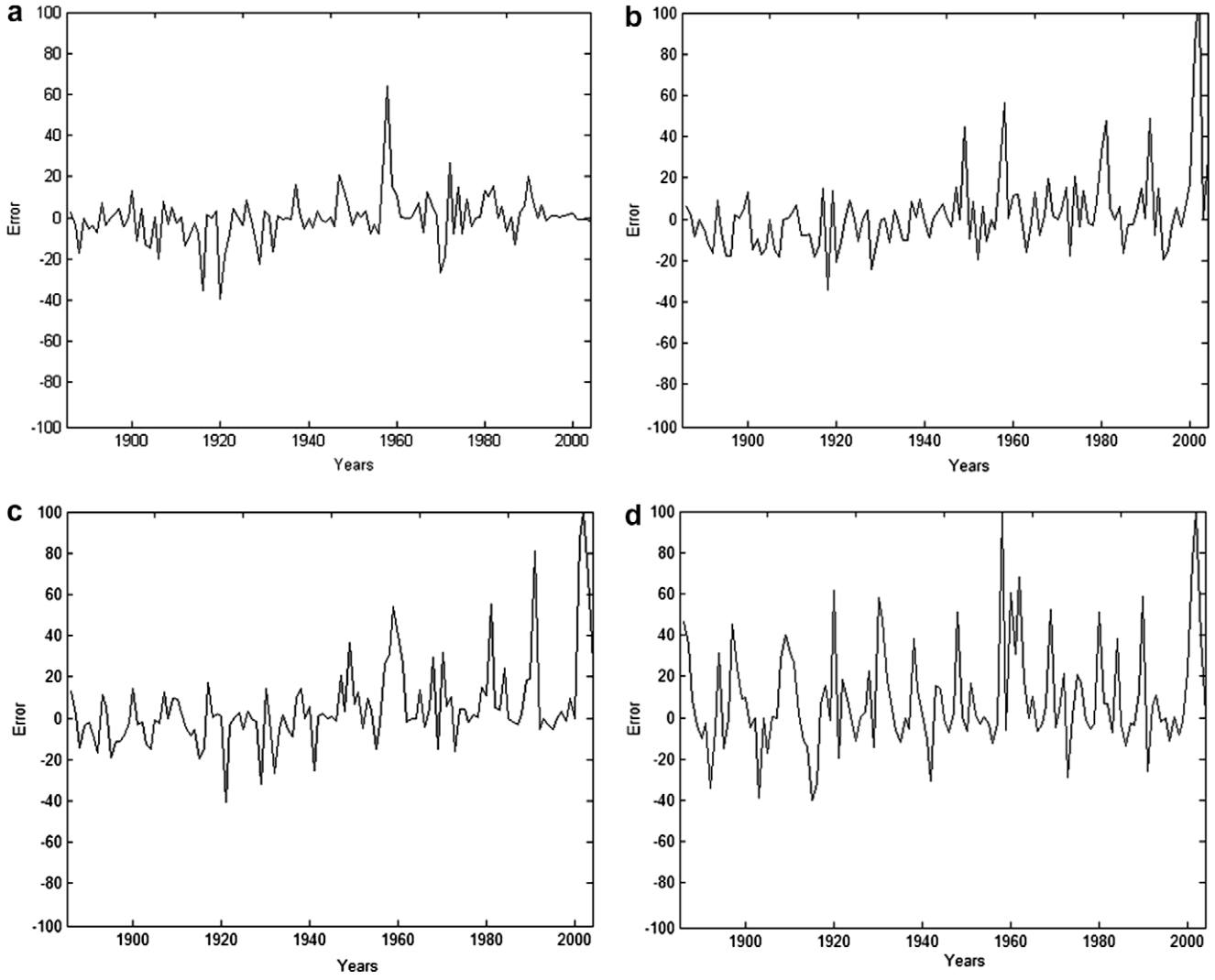

Fig. 10. The prediction errors of the (a) proposed method, (b) ESP (c) SE and (d) GA methods.

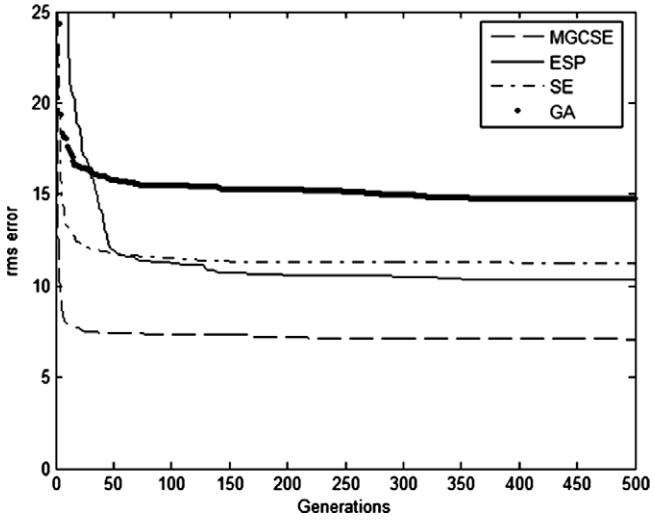

Fig. 11. The learning curves of MGCSK, ESP, SE, and GA.

Table 4
Performance comparison of various existing models in example 2.

Method	Training cases	RMSE			CPU time
		Best	Mean	Worst	
MGCSE	500	7.51	8.21	12.98	202.36
ESP	500	10.33	12.12	10.33	200.98
VGA	500	11.25	13.05	14.65	249.7
HGA	500	12.93	13.68	19.21	285.2
SE	500	13.67	15.26	20.43	354.77
GA	500	14.98	17.47	24.76	402.46

Table 5
Training and forecasting error comparison of various existing models in example 2.

Method	Training error			Forecasting error		
	Best	Mean	Worst	Best	Mean	Worst
MGCSE	5.23	6.75	10.32	6.24	8.31	12.44
ESP	7.35	8.75	12.51	11.25	12.35	16.02
VGA	8.02	9.01	12.98	12.85	14.28	17.76
HGA	8.43	9.32	13.34	13.12	14.92	18.42
SE	8.55	10.05	13.97	14.21	15.05	18.91
GA	10.11	12.27	18.34	16.31	19.81	25.22

5. Conclusion

In this paper, a TSK-type neuro-fuzzy system with a multi groups cooperation based symbiotic evolution method (TNFS-MGCSE) is proposed. The TNFS-MGCSE is developed from symbiotic evolution. The TNFS-MGCSE can evaluate the fuzzy rule locally and make groups cooperate with each other to generate the better chromosomes by using a cooperation based crossover strategy (CCS). The advantages of TNFS-MGCSE are summarized as follows: (1) the TNFS-MGCSE uses multi groups in a population to evaluate the fuzzy rule locally. (2) The TNFS-MGCSE uses CCS to let the better solutions form different groups can cooperate with each other for generating better solutions. (3) It indeed can obtain better performance and converge more quickly than some traditional genetic methods. Computer simulations have shown that the TNFS-MGCSE has a better performance than the other methods.

Although TNFS-MGCSE can perform better than other methods, there still has a limitation. The initial parameters are determined by practical experimentation or trial-and-error tests. There is not a systematic method to determine the initial parameters. In the future work, how to find a well-defined method to define such parameters can be tried.

Acknowledgement

This work is supported in part by the National Science Council, Taiwan, R.O.C. under Grant NSC 95-2221-E-009-214.

References

Bandyopadhyay, S., Murthy, C. A., & Pal, S. K. (2000). VGA-classfifer: Design and applications. *IEEE Transactions on Systems, Man, and Cybernatics, Part B, 30*, 890–895.

Belarbi, K., & Titel, F. (2000). Genetic algorithm for the design of a class of fuzzy controllers: An alternative approach. *IEEE Transactions on Fuzzy Systems, 8*(4), 398–405.

Carse, B., Fogarty, T. C., & Munro, A. (1996). Evolving fuzzy rule based controllers using genetic algorithms. *Fuzzy Sets and Systems, 80*(3), 273–293.

Cordon, O., Herrera, F., Hoffmann, F., & Magdalena, L. (2001). *Genetic fuzzy systems evolutionary tuning and learning of fuzzy knowledge bases. Advances in Fuzzy Systems –Applications and Theory* (Vol. 19). NJ: World Scientific Publishing.

Cowder, R. S. III (1990). In D. Touretzky, G. Hinton, T. Sejnowski (Eds.), *Predicting the Mackey–Glass time series with cascade-correlation learning* (pp. 117–123).

Fogel, L. J. (1994). Evolutionary programming in perspective: The top-down view. In J. M. Zurada, R. J. Marks, II, & C. Goldberg (Eds.), *Computational intelligence: Imitating life*. Piscataway, NJ: IEEE Press.

Goldberg, D. E. (1989). *Genetic algorithms in search optimization and machine learning*. Reading, MA: Addison-Wesley.

Gomez, F. J. (2003). *Robust non-linear control through neuroevolution*. Ph.D. dissertation, The University of Texas at Austin.

Gomez, F., & Schmidhuber, J. (2005). Co-evolving recurrent neurons learn deep memory POMDPs. In: *Proceeding of conference on genetic and evolutionary computation* (pp. 491–498).

Homaifar, A., & McCormick, E. (1995). Simultaneous design of membership functions and rule sets for fuzzy controllers using genetic algorithms. *IEEE Transactions on Fuzzy Systems, 3*(2), 129–139.

Jang, J.-S. R. (1993). ANFIS: Adaptive-network-based fuzzy inference system. *IEEE Transactions on Systems, Man and Cybernatics, 23*, 665–685.

Juang, C. F. (2004). A hybrid of genetic algorithm and particle swarm optimization for recurrent network design. *IEEE Transactions on Systems, Man, and Cybernatics, Part B, 34*(2), 997–1006.

Juang, C. F., & Lin, C. T. (1998). An on-line self-constructing neural fuzzy inference network and its applications. *IEEE Transactions on Fuzzy Systems, 6*(1), 12–31.

Juang, C. F., & Lin, C. T. (1999). A recurrent self-organizing neural fuzzy inference network. *IEEE Transactions on Neural Networks, 10*(4), 828–845.

Juang, C. F., Lin, J. Y., & Lin, C. T. (2000). Genetic reinforcement learning through symbiotic evolution for fuzzy controller design. *IEEE Transactions on Systems, Man, and Cybernatics, Part B, 30*(2), 290–302.

Karr C. L. (1991). Design of an adaptive fuzzy logic controller using a genetic algorithm. In: *Proceeding of the Fourth international conference on genetic algorithms* (pp. 450–457).

Koza, J. K. (1992). *Genetic programming: On the programming of computers by means of natural selection*. Cambridge, MA: MIT Press.

Lee, M., & Takagi, H. (1993). Integrating design stages of fuzzy systems using genetic algorithms. In: *Proceeding of the IEEE international conference on fuzzy systems* (pp. 612–617).

Lin, C.-J., & Chin, C.-C. (2004). Prediction and identification using wavelet-based recurrent fuzzy neural networks. *IEEE Transactions on Systems, Man and Cybernatics, Part B, 34*(5), 2144–2154.

Ling, S. H., Leung, Frank H. F., Lam, H. K., Lee, Yim-Shu, & Tam, Peter K. S. (2003). A novel genetic-algorithm-based neural network for short-term load forecasting. *IEEE Transactions on Industrial Electronic, 50*(4), 793–799.

Lin, C. T., & Lee, C. S. G. (1996). *Neural fuzzy systems: A neuro-fuzzy synergism to intelligent system*. Englewood Cliffs, NJ: Prentice-Hall.

Lin, C. J., & Lin, C. T. (1997). An ART-based fuzzy adaptive learning control network. *IEEE Transaction on Fuzzy Systems, 5*(4), 477–496.

Lin, F. J., Lin, C. H., & Shen, P. H. (2001). Self-constructing fuzzy neural network speed controller for permanent-magnet synchronous motor drive. *IEEE Transactions on Fuzzy Systems, 9*(5), 751–759.

Mastorocostas, P. A., & Theocharis, J. B. (2002). A recurrent fuzzy-neural model for dynamic system identification. *IEEE Transactions on Systems, Man, and Cybernatics, 32*(2), 176–190.

Mizutani, E., & Jang J-S R. (1995). Coactive neural fuzzy modeling. In: *Proceeding of IEEE international conference on neural networks* (pp. 760-765).

Moriarty, D. E., & Miikkulainen, R. (1996). Efficient reinforcement learning through symbiotic evolution. *Machine Learning, 22*, 11–32.

Narendra, K. S., & Parthasarathy, K. (1990). Identification and control of dynamical systems using neural networks. *IEEE Transactions on Neural Networks, 1*, 4–27.

Rechenberg, I. (1994). Evolution strategy. In J. M. Zurada, R. J. Marks, II, & C. Goldberg (Eds.), *Computational intelligence: Imitating life*. Piscataway, NJ: IEEE Press.

Takagi, T., & Sugeno, M. (1985). Fuzzy identification of systems and its applications to modeling and control. *IEEE Transactions on Systems, Man, and Cybernatics, 15*, 116–132.

Takagi, H., Suzuki, N., Koda, T., & Kojima, Y. (1992). Neural networks designed on approximated reasoning architecture and their application. *IEEE Transactions on Neural Networks, 3*(5), 752–759.

Tang, K. S. (1996). *Genetic algorithms in modeling and optimization*. Ph.D. dissertation, Dep. Electron. Eng., City Univ. Hong Kong, Hong Kong.

Towell, G. G., & Shavlik, J. W. (1993). Extracting refined rules from knowledge-based neural networks. *Machine Learning, 13*, 71–101.

Wang, L. X., & Mendel, J. M. (1992). Generating fuzzy rules by learning from examples. *IEEE Transactions on Systems, Man and Cybernatics, 22*(6), 1414–1427.

Yi-Ta Wu, Yoo Jung An, Geller, J., & Yih-Tyng Wu (2006). A data mining based genetic algorithm. In: *Proceeding of IEEE workshop SEUS-WCCIA* (pp. 27–28).