Contents lists available at ScienceDirect

Neurocomputing

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

Reinforcement group cooperation-based symbiotic evolution for recurrent wavelet-based neuro-fuzzy systems

Yung-Chi Hsu, Sheng-Fuu Lin *

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

A R T I C L E I N F O

Article history:
Received 9 August 2007
Received in revised form
26 June 2008
Accepted 8 December 2008
Communicated by T. Heskes
Available online 23 January 2009

Keywords:
Neuro-fuzzy system
Symbiotic evolution
Control
Reinforcement learning
Recurrent network

A B S T R A C T

This paper proposes a recurrent wavelet-based neuro-fuzzy system (RWNFS) with a reinforcement group cooperation-based symbiotic evolution (R-GCSE) for solving various control problems. The R-GCSE is different from the traditional symbiotic evolution. In the R-GCSE method, a population is divided to several groups. Each group formed by a set of chromosomes represents a fuzzy rule and cooperates with other groups to generate better chromosomes by using the proposed elite-based compensation crossover strategy (ECCS). In this paper, the proposed R-GCSE is used to evaluate numerical control problems. The performance of the R-GCSE in the simulations is excellent compared with other existing models.

© 2009 Elsevier B.V. All rights reserved.

1. Introduction

In recent years, fuzzy logic or artificial neural networks used to solve control problems have become a popular research topic [1–10]. The reason is that classical control theory usually requires a mathematical model for designing controllers. The inaccuracy of mathematical modeling of plants usually degrades the performance of the controllers, especially for nonlinear and complex control problems [11–14]. Fuzzy logic has the ability to express the ambiguity of human thinking and to translate expert knowledge into computable numerical data.

A fuzzy system consists of a set of fuzzy IF–THEN rules that describe the input–output mapping relationship of networks. 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 used to generate the fuzzy IF–THEN rules from numerical data have been proposed [2,3,6]. These methods were developed for supervised learning; i.e., the correct "target" output values are given for each input pattern to guide the learning of the network. However, most of the supervised learning algorithms for neural fuzzy networks require precise training data in order to tune the networks for various applications. For some real world applications, precise training data are usually difficult and expensive, if not impossible, to obtain. For this reason, there has been a growing interest in reinforcement learning algorithms for neural controller [15–18] or fuzzy [19–21] design.

In designing a fuzzy controller, adjusting the required parameters is important. To do this, back-propagation (BP) training was used in [3,6–8]. 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. To solve these problems, several evolutionary algorithms, such as genetic algorithm (GA) [22], genetic programming [23], evolutionary programming [24], and evolution strategies [25], have recently 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, evolutionary methods, which are used for training fuzzy models, have become an important field.

The evolutionary fuzzy model generates a fuzzy system automatically by incorporating evolutionary learning procedures [26–33]. The most well-known evolutionary learning procedure is GAs. Several genetic fuzzy models have been proposed [26–31]. In [26], Karr applied GAs to design the membership functions of a fuzzy controller with its fuzzy rule set being assigned in advance. Since the membership functions and rule sets are co-dependent, simultaneous design of these two approaches is a more appropriate methodology.

Based on this concept, many researchers have applied GAs to optimize both the parameters of the membership functions and

* Corresponding author.
E-mail address: sflin@mail.nctu.edu.tw (S.-F. Lin).

the rule sets [27–29]. Lin and Jou [30] proposed GA-based fuzzy reinforcement learning to control magnetic bearing systems. Juang et al. [31] proposed using genetic reinforcement learning to design fuzzy controllers. The GA adopted in [31] was based on traditional symbiotic evolution which, when applied to fuzzy controller design, complements the local mapping property of a fuzzy rule. In [32] Tang proposed a hierarchical genetic algorithm. The hierarchical GA enables the optimization of the fuzzy system design for a particular application. Juang [33] proposed the combination of online clustering and Q-value based GA for reinforcement fuzzy system (CQGAF) to simultaneously design the number of fuzzy rules and the free parameters in a fuzzy system.

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

Recently, Gomez and Schmidhuber [34,35] proposed solutions for these problems. The proposed enforced sub-populations (ESP) used sub-populations of neurons for the fitness evaluation and overall control. As shown in [34,35], the sub-populations that are used to evaluate the solution locally can obtain better performance compared to systems of only one population which are used to evaluate the solution.

As with [34,35], in this paper, a RWNFS with a reinforcement group cooperation-based symbiotic evolution (R-GCSE) is proposed for solving the problems mentioned above. In the proposed R-GCSE, each chromosome represents only one fuzzy rule, and the *n*-rules fuzzy system is constructed by selecting and combining *n* chromosomes from several groups. The R-GCSE, which promotes both cooperation and specialization, ensures diversity and prevents a population from converging to suboptimal solutions. In the R-GCSE, compared with normal symbiotic evolution, several groups are in the population. Each group formed by a set of chromosomes represents a fuzzy rule. Compared with [34,35] to let the well-performing groups of individuals cooperate to create better generations, an elite-based compensation crossover strategy (ECCS) is proposed in this paper. In the ECCS, each group cooperates to perform the crossover steps. Therefore, the better chromosomes of each group will be selected to perform the crossover steps in the next generation.

The advantages of the R-GCSE are summarized as follows: (1) the R-GCSE uses group-based populations to evaluate the fuzzy rule locally; (2) the R-GCSE uses the ECCS to allow better solutions from different groups to cooperate in order to generate better solutions in the next generation; (3) it indeed performs better performance and converges more quickly than some traditional genetic methods.

This paper is organized as follows. In Section 2, the RWNFS is introduced. In Section 3, the proposed group cooperation-based symbiotic evolution (GCSE) is described. In Section 4, the reinforcement group cooperation-based symbiotic evolution (R-GCSE) using for constructing the RWNFS model is introduced. In Section 5, the simulation results are presented. The conclusions are summarized in the last section.

2. Structure of a RWNFS

In this section, the structure of RWNFS shown in Fig. 1 will be introduced. For TSK-type fuzzy networks [1,5], the consequence of each rule is a function input linguistic variable. A widely adopted function is a linear combination of input variables plus a constant term. This study adopts a nonlinear combination of input variables (i.e., wavelet neural network (WNN)). The advantages of the WNN are as follows: (1) its ability to find "universal

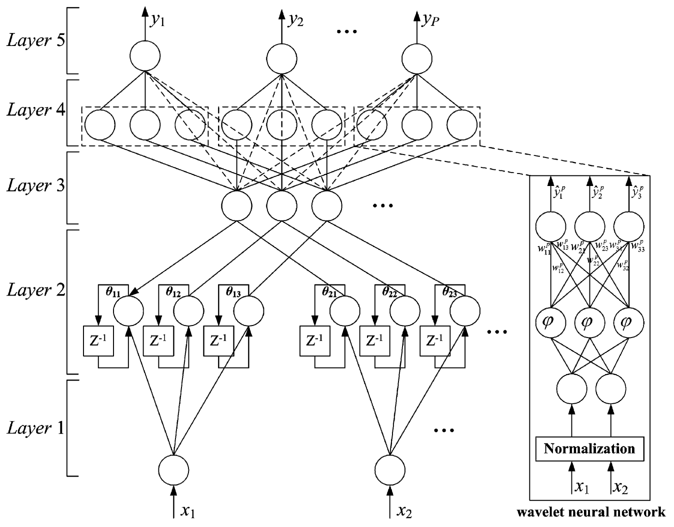

Fig. 1. Schematic diagram of RWNFS model.

approximation"; (2) an explicit link between the wavelet transform and the network coefficient is completed, and an initial guess of network parameters can be derived by the decomposition of a wavelet formula; (3) it probably obtains the same approximation performance as a smaller size network; in addition, wavelet networks are optimal approximators since the smallest number of bits are required to obtain an arbitrary precision [36].

In RWNFS, each fuzzy rule corresponds to a sub-WNN which consists of single-scaling wavelets [37]. The non-orthogonal and compact wavelet functions used as the node function (wavelet bases) are adopted in this paper. The purpose of introducing a fuzzy model into WNN is to improve the accuracy of function approximation based on the dilation and translation parameters of wavelets while not increasing the number of wavelet bases. A RWNFS is composed of fuzzy rules that can be presented in the following general form:

$$\begin{aligned}
R^j:\ & \text{If } I_1 \text{ is } A_{1j} \text{ and} \ldots I_{ij} \text{ is } A_{ij} \text{ and } \ldots \text{and } I_{nj} \text{ is } A_{nj} \\
& \text{Then } \hat{y}_j^1 = \sum_{k=1}^{M} w_{jk}^1 \varphi_{a.b} = w_{j1}^1\varphi_{0.0} + w_{j2}^1\varphi_{1.0} + w_{j3}^1\varphi_{1.1}\cdots \\
& \text{and } \hat{y}_j^2 = \sum_{k=1}^{M} w_{jk}^2 \varphi_{a.b} = w_{j1}^2\varphi_{0.0} + w_{j2}^2\varphi_{1.0} + w_{j3}^2\varphi_{1.1}\cdots \\
& \quad\vdots
\end{aligned} \tag{1}$$

where R^j denotes the *j*th rule; $(I_{1j},\ldots, I_{ij},\ldots,I_{nj})$ is the network input pattern $(x_1,\ldots, x_i,\ldots,x_n)$ plus the temporal term for the linguistic term of the precondition part $A^j = (A_{1j},\ldots,A_{ij},\ldots,A_{nj})$; the local WNN model's outputs $\hat{y}_j^1$ and $\hat{y}_j^2$ are calculated for outputs y_1 and y_2 of rule R^j.

Next, the signal propagation is indicated, along with the operation functions of the nodes in each layer. In the following description, $I_i^{(h)}$ denotes the *i*th node's input in the *h*th layer, and $O_i^{(h)}$ denotes the *i*th node's output in layer *h*.

In layer 1, nodes just transmit input signals to the next layer directly, that is,

$$O_i^{(1)} = I_i^{(1)} \tag{2}$$

where $I_i^{(1)} = (x_1,\ldots,x_i,\ldots,x_n)$. Each precondition part of the *j*th rule $A^j = (A_{1j},\ldots,A_{ij},\ldots,A_{nj})$ (a group of fuzzy sets) is described here by a Gaussian-type membership function; that is, the membership value specifying the degree of how an input value belongs to a fuzzy set is determined in layer 2. The Gaussian function is defined by

$$O_{ij}^{(2)} = \exp\left(-\frac{(I_{ij}^{(2)} - m_{ij})^2}{\sigma_{ij}^2}\right) \tag{3}$$

where m_{ij} and σ_{ij} are the mean and standard deviation with the ith dimension and jth rule node, respectively. Additionally, the input of this layer for the discrete time can be denoted by

$$I_{ij}^{(2)}(t) = O_i^{(1)}(t) + O_{ij}^{(f)}(t), \quad O_{ij}^{(f)}(t) = O_{ij}^{(2)}(t-1)\theta_{ij} \tag{4}$$

where θ_{ij} is the feedback weight. Clearly, the input of this layer contains the memory terms $O_{ij}^{(2)}(t-1)$, which store the past information of the network. In RWNFS, the recurrent property is achieved by feeding the output of each membership function back to itself so that each membership value is influenced by its previous value.

Although some recurrent neural fuzzy networks have been proposed and applied to dynamic system identification and control, there are still disadvantages to these network structures. In [38], the order of both the control input and the network output in the Auto Regressive with eXogenous (ARX) model needs to be known. This problem can be solved by feeding back the output of each membership function in the recurrent property of RWNFS. Only the current control input and system state are fed to the network input. The past values can be memorized by using the feedback structure. In [39], a global feedback structure is adopted, and the outputs of all the rule nodes, the firing strengths, are fed back and summed. In this case, the TRFN [39] needs more adjustable parameters.

In layer 3, defining the number and the locations of the membership functions leads to the partition of the space $D = D_1 \times \cdots \times D_n$. The collection of fuzzy sets $A^j = (A_{1j}, \ldots, A_{ij}, \ldots, A_{nj})$ pertaining to the premise part of R^j formulates a fuzzy region in D that can be regarded as a multi-dimensional fuzzy set whose membership function is determined by

$$O_j^{(3)} = \prod_{i=1}^{n} I_j^{(3)} = \prod_{i=1}^{n} \exp\left(-\frac{(I_{ij}^{(2)} - m_{ij})^2}{\sigma_{ij}^2}\right) \tag{5}$$

where n is the number of external dimensions.

Layer 4 only receives the signal $\hat{y}_j^s$ from the output of the WNN for an output Y_s and the jth rule. The mathematical function of each node j is

$$\hat{y}_j^s = O_{sj}^{(4)} = \sum_{k=1}^{M} w_{jk}^s \varphi_{a.b}. \tag{6}$$

The crisp $\varphi_{a.b}$ can be obtained as follows:

$$\varphi_{a.b} = \frac{\sum_{i=1}^{n} \phi_{a.b}(x_i)}{|X|} \tag{7}$$

where $|X|$ is the number of input dimensions. The $\phi_{a.b}(x_i)$ functions which are used to input vectors to fire up the wavelet interval are calculated as follows:

$$\begin{cases} \phi(x_i) = \cos(x_i) & -0.5 \leqslant x_i \leqslant 0.5 \\ 0, & \text{otherwise} \end{cases}, \quad \phi_{a.b}(x_i) = \cos(ax_i - b)$$
$$\text{where } a = 1, \ldots, m; \; b = 1, \ldots, a. \tag{8}$$

The above equation formulates the non-orthogonal wavelets in a finite range, where b denotes a shifting parameter with its maximum value equal to the corresponding scaling parameter a.

The final output of the model ($y_1, \ldots, y_s, \ldots, y_p$) is calculated in layer 5, and the node's output together with related links acts as a defuzzifier. The mathematical function is

$$\begin{aligned} y_s = O_s^{(5)} &= \frac{\sum_{j=1}^{M} I_{sj}^{(5)} I_j^{(5)}}{\sum_{j=1}^{M} I_j^{(5)}} \\ &= \frac{\sum_{j=1}^{M} (w_{j1}^s \phi_{0.0} + \cdots + w_{jk}^s \phi_{a.b} \cdots + w_{jM}^s \phi_{m.m}) I_j^{(5)}}{\sum_{j=1}^{M} I_j^{(5)}} \end{aligned} \tag{9}$$

where $I_{sj}^{(5)} = O_{sj}^{(4)}$ denotes the output of the local model of WNN for an output Y_s and the jth rule, $I_j^{(5)} = O_j^{(3)}$ is the output of layer 3, and y_s is the sth output of RWNFS.

3. A group cooperation-based symbiotic evolution (GCSE)

In this section, the proposed GCSE method will be discussed. Recently, there have been many studies which have tried to enhance the traditional GAs [40–43]. One category of these studies tries to modify the structure of a population. Examples in this category include the distributed GA [41], the cellular GA [42], and the symbiotic GA [43].

This study proposes using the GCSE to improve the symbiotic GA [43]. In the GCSE, 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 [44]. 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 [31,43], partial solutions can be characterized as *specializations*. The specialization property ensures diversity and prevents a population from converging to suboptimal solutions. A single partial solution cannot "take over" a population since it must correspond with other specializations. Unlike the standard evolutionary approach which always causes a given population to converge, hopefully at the global optimum, the symbiotic evolution finds solutions in different, unconverted populations [31,43]. In the GCSE, compared with normal symbiotic evolution, several groups are in the population. Each group formed by a set of chromosomes represents a fuzzy rule.

In the GCSE, the structure of the population consists of several groups. The structure of the chromosome in the GCSE is shown in Fig. 2. However, to allow groups to cooperate with each other in order to generate better solutions, the GCSE proposes an ECCS.

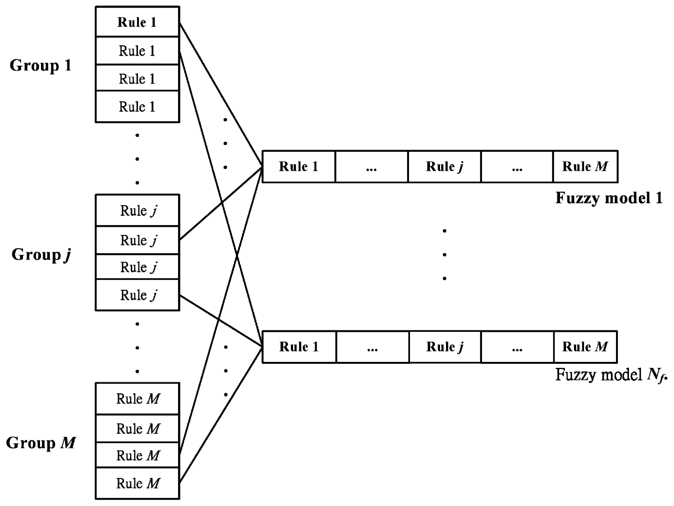

Fig. 2. The structure of chromosomes in GCSE.

In the GCSE, the coding structure of the chromosomes must be suitable for each chromosome to represent only one fuzzy rule. A fuzzy rule with the form introduced in Eq. (1) is shown in Fig. 3.

The learning process of the GCSE in each group involves five major steps: initialization, fitness assignment, elite-based reproduction strategy (ERS), ECCS, and mutation strategy. The flowchart of the learning process is shown in Fig. 4. The learning process is described step-by-step as follows:

m_{1j}	σ_{1j}	m_{2j}	σ_{2j}	$\cdots$	m_{nj}	σ_{nj}	θ_{1j}	$\cdots$	θ_{nj}	w_{j1}	w_{j2}	$\cdots$	w_{jM}

Fig. 3. Coding a rule of a RWNFS into a chromosome in GCSE.

3.1. Initialization

Before the GCSE is designed, individuals forming several initial groups should be generated. The following formulations show how to generate the initial chromosomes in each group:

$$\textit{Deviation}: \ Chr_{g,c}[p] = random[\sigma_{\min}, \sigma_{\max}]$$
$$\text{where } p = 2,4,\ldots,2n; \quad g = 1,2,\ldots,M; \quad c = 1,2,\ldots,N_C \quad (10)$$

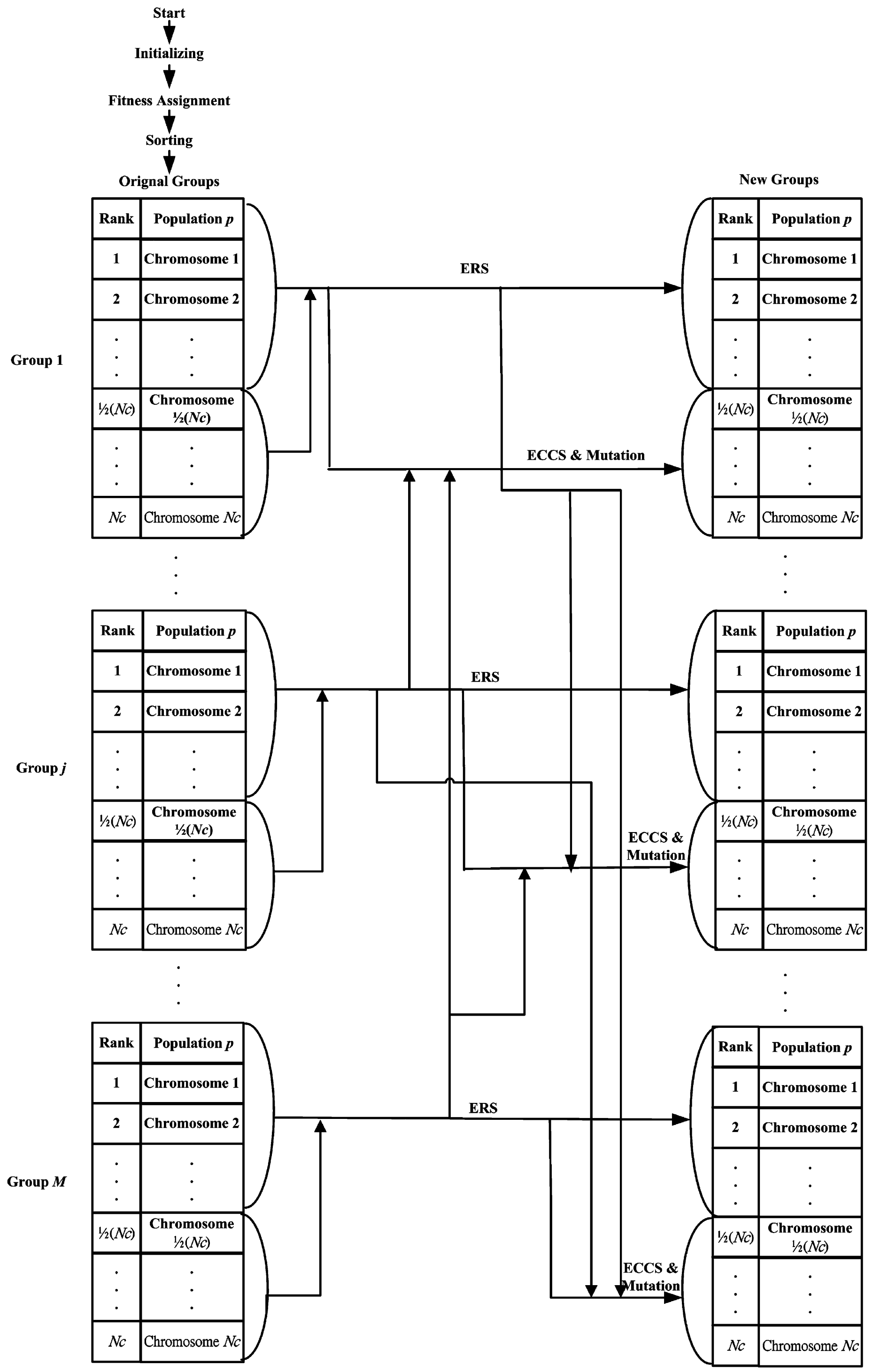

Fig. 4. The learning process of GCSE.

$$\text{Mean}: \ Chr_{g,c}[p] = random[m_{\min}, m_{\max}] \quad \text{where } p = 1, 3, \ldots, 2n-1 \tag{11}$$

$$\text{Theta}: \ Chr_{g,c}[p] = random[\theta_{\min}, \theta_{\max}] \quad \text{where } p = 2n+1, 2n+2, \ldots 2n+1 \tag{12}$$

$$\text{Weight}: \ Chr_{g,c}[p] = random[w_{\min}, w_{\max}] \quad \text{where } p = 2(n+1)+1, 2(n+1)+2, \ldots, 2(n+1)+M \tag{13}$$

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

3.2. Fitness assignment

As discussed previously, in the GCSE, the fitness value of a single rule (an individual) is calculated by summing up the fitness values of all the possible combinations which contain that single rule. The details for assigning the fitness value are described step-by-step as follows:

- *Step* 1. Randomly choose M fuzzy rules from the M groups with size N_C.
- *Step* 2. Evaluate every RWNFS, which is generated from step 1, to obtain a fitness value.
- *Step* 3. Divide the fitness value by M and accumulate the divided fitness values to the selected rules with their fitness value records initially set to zero.
- *Step* 4. Repeat the above steps until each rule (an individual) in each group has been selected a sufficiently large number of times, and record the number of RWNFS models in which each individual has participated.
- *Step* 5. Divide the accumulated fitness value of each chromosome by the number of times it has been selected. The average fitness value represents the performance of a single rule.

3.3. Elite-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 GCSE is to maximize the fitness value. For stability, this study proposes an ERS to allow the best combination of chromosomes to be kept in the next generation. In the GCSE, 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 by the roulette-wheel selection method [45]—a simulated roulette is spun—in this study. The best performing chromosomes in the top half of each group [31] advance to the next generation. The other half is generated by applying the crossover and mutation operations on the chromosomes in the top half of the parent generation. In the reproduction step, the top half of each group must keep the same number of chromosomes.

3.4. Elite-based compensation crossover strategy (ECCS)

Although the ERS 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 step which works on the parents is the crossover step, which occurs on a selected pair under a crossover rate. In this paper, an ECCS is proposed to improve the crossover operation. The ECCS mimics the cooperation phenomenon in society, in which individuals become more suitable for the environment as they acquire and share more knowledge of their surroundings. The best performing individuals in the top half of each group that are called elites are used to select the parents so that the ECCS can be applied. Details of the ECCS are shown below.

Step 1. The first of the parents that is used in the crossover operation is selected from the original group by using the following equations:

$$Fitness\_Ratio_{g,t} = \frac{\sum_{u=1}^{t} fitness_{g,u}}{\sum_{c=1}^{N_c} fitness_{g,u}}, \quad \text{where } t = 1, 2, \ldots, N_c \tag{14}$$

$$Rand\_Value[g] = Random[0,1], \quad \text{where } g = 1, 2, \ldots, M; \tag{15}$$

$$Parent\_SiteA[g] = t, \quad \text{if} \quad Fitness\_Ratio_{g,t-1} < Rand\_Value[g] \leqslant Fitness\_Ratio_{g,t}, \tag{16}$$

where $Fitness\_Ratio_{g,t}$ is the fitness ratio of the tth chromosome in the gth group; $Rand\_Value[g] \in [0,1]$ is a random value in the gth group; and $Parent\_SiteA[g]$ is the site of the first parent. According to Eq. (16), if the $Rand\_Value[g]$ is greater than the fitness ratio at the $(t-1)$th chromosome in the gth group and equal to or smaller than the fitness ratio at the tth chromosome in the gth group, the site of the first parent of the gth group is assigned to t.

Step 2. After the first parent is determined, the best performing elites in every group are used to determine the other parent. In this step, the total fitness ratio of every group is computed as follows:

$$Total\_Fitness_g = \sum_{c=1}^{Nc} fitness_{g,c}, \quad \text{where } g = 1, 2, \ldots, M; \tag{17}$$

$$Total\_Fitness\_Ratio_w = \frac{\sum_{u=1}^{w} Total\_Fitness_u}{\sum_{g=1}^{M} Total\_Fitness_g} \quad where \ \ w = 1, 2, \ldots, M; \tag{18}$$

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

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

$$Group\_Rand\_Value[g] = Random[0,1] \quad \text{where } g = 1, 2, \ldots, M; \tag{19}$$

$$Parent\_Group\_SiteB[g] = w, \quad \text{if } Total\_Fitness\_Ratio_{w-1} < Group\_Rand\_Value[g] \leqslant Total\_Fitness\_Ratio_w. \tag{20}$$

where $Group\_Rand\_Value[g] \in [0,1]$ is a random value in the gth group and $Parent\_Group\_SiteB[g]$ 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 other parent which is selected from the $Parent_Group_SiteB[g]$th group is determined by the ECCS according to the following equations:

$$Fitness\_Ratio_{Selected\_g,t} = \frac{\sum_{u=1}^{t} fitness_{Selected\_g,u}}{\sum_{c=1}^{Nc} fitness_{Selected\_g,c}}, \tag{21}$$

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

$$Rand\_Value[g] = Random[0,1], \quad \text{where } g = 1, 2, \ldots, M; \tag{22}$$

$$Parent\_SiteB[g] = l, \quad \text{if } Fitness\_Ratio_{Selected\_g,l-1} < Rand\_Value[g] \leqslant Fitness\_Ratio_{Selected\_g,l}, \tag{23}$$

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 the ECCS is listed in Fig. 5.

After the parents from the gth group and the $Parent_Group_SiteB[g]$th group are selected using ECCS, the individuals (the $Parent_SiteA[g]$th chromosome and the $Parent_SiteB[g]$th chromosome) are crossed and separated by using a two-point crossover in the gth group, as shown in Fig. 6. In Fig. 6, exchanging the site's values between the selected sites of the parents' individuals creates new individuals. After this operation, the individuals with poor performances are replaced by the newly produced offspring.

```
Procedure Elite-based Compensatory of Crossover Strategy
Begin
  Let $g=0, c=0, q=0$;
  // Decide the parents that using for performing the crossover to
     generate the other half of each group.
  Repeat
      $q=q+1$;
      //Decide the first parent
      Repeat
          $g=g+1$;
          Compute $Parent\_SiteA[g]$ by (14) to (16);
     Until $g=M$;
      //Compute the Group fitness Ratio
      Let $w=0$;
      Repeat
          $w=w+1$;
          Compute $Total\_Fitness\_Ratio_w$ by (17) and (18);
      Until $w=M$;
      //Compute the group that the second parent is selected from
      $g=0$;
      Repeat
          $g=g+1$;
          Compute $Parent\_Group\_SiteB[g]$ by (19) and (20);
      Until $g=M$
      // Decide the Second parent;
      $g=0$;
      Repeat
          $g=g+1$;
          Compute $Parent\_SiteB[g]$ by (21) to (23);
      Until $g=M$;
  Until $q=Nc/2$;
End
```

Fig. 5. The pseudo code of the ECCS method.

3.5. Mutation strategy

Although the ERS and ECCS 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 ECCS, the GCSE tries to simplify the mutation operation. Therefore, a uniform mutation [45] 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. Reinforcement learning for a RWNFS

Unlike the supervised learning problem, in which the correct "target" output values are given for each input pattern, the reinforcement learning problem has only very simple "evaluative" or "critical" information rather than "instructive" information. In the extreme case, there is only a single bit of information to indicate whether the output is right or wrong. The training environment of reinforcement group cooperation-based symbiotic evolution (R-GCSE), which interacts with reinforcement learning problems, is shown in Fig. 7. In this paper, the reinforcement signal indicates whether a success or a failure occurs.

As shown in Fig. 7, the R-GCSE consists of a RWNFS in order to determine a proper action according to the current input vector (environment state). The structure of the R-GCSE is different from Barto and his colleagues' actor-critic architecture [17], which consists of a control network and a critic network. The input of the RWNFS is the state of the plant, and the output is a control action of the state denoted by f. The only available feedback is a reinforcement signal that notifies the RWNFS only when a failure occurs. An accumulator plays the role of a relative performance measure. It is shown in Fig. 7. It accumulates the number of time steps before a failure occurs. In this paper, the feedback is decided by an accumulator that determines how long the experiment is still a "success." The accumulator is used as a relative measure of the fitness in the R-GCSE. The key to the R-GCSE is formulating a number of time steps before a failure occurs and using this formulation as the fitness function of the R-GCSE. It will be observed that the advantage of the R-GCSE is that it can meet global optimization capability.

A flowchart of the R-GCSE is shown in Fig. 8. The R-GCSE runs in a feed forward fashion to control the environment (plant) until a failure occurs. In this paper, the fitness function is defined as a number of time steps before a failure occurs. The goal of the R-GCSE is to maximize the fitness value. The fitness function is defined by:

$$Fitness\ Value = TIME\text{-}STEP \tag{24}$$

where $TIME\text{-}STEP$ represents how long the experiment is still a "success." Eq. (24) indicates that long-time steps before a failure

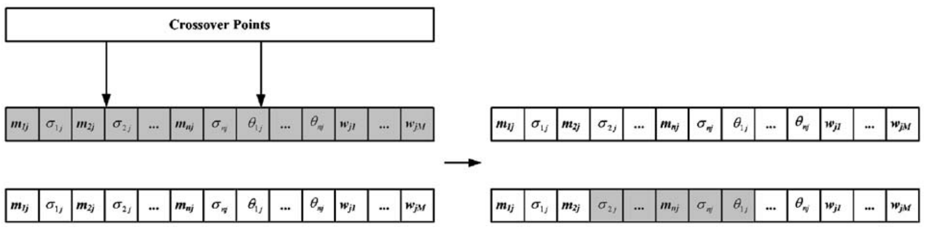
Crossover Points
m_{1j} σ_{1j} m_{2j} σ_{2j} … m_{nj} σ_{nj} θ_{1j} … θ_{nj} w_{j1} … w_{jM}

Fig. 6. Two-point crossover.

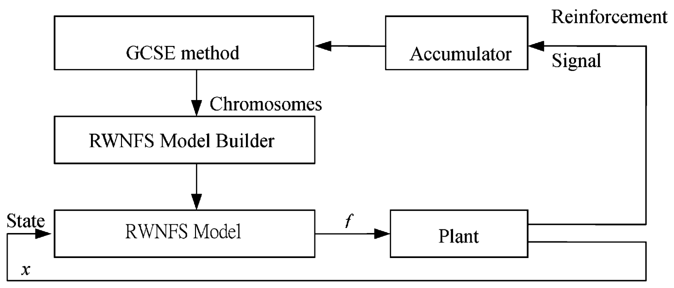

Fig. 7. Schematic diagram of the R-GCSE for the RWNFS model.

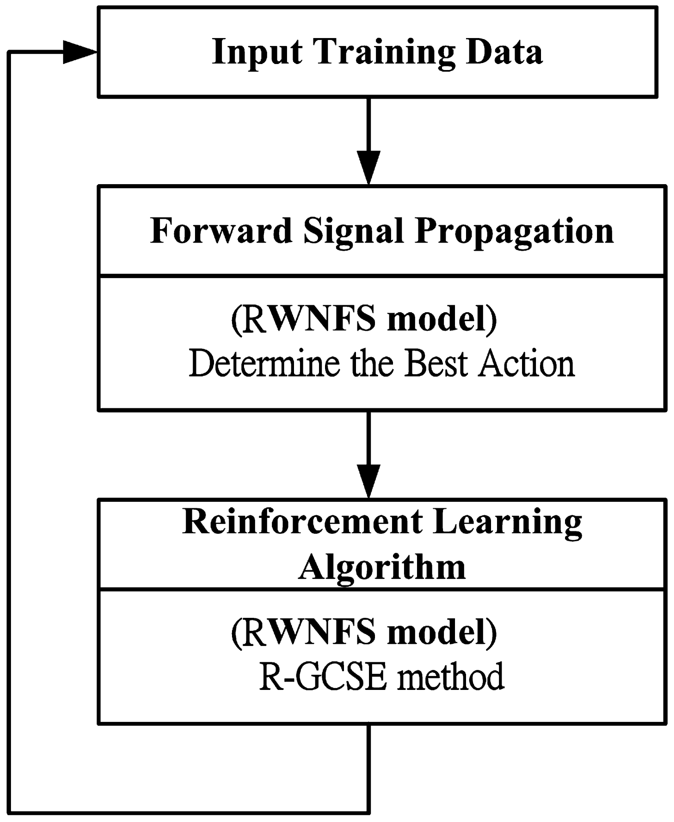

Fig. 8. Flowchart of the R-GCSE.

occurs (to keep the desired control goal longer) means a higher fitness of the R-GCSE.

5. Illustrative examples

Two applications are discussed in this section. The first simulation simulated balance a cart-pole system that was described in [46–48]. The second simulation simulated the balancing of a ball and beam system that was described in [49,50]. The initial parameters for these two examples are given in Table 1. The initial parameters were determined by practical experimentation or trial-and-error tests.

5.1. Example 1: Control of a cart-pole balancing system

In this example, the R-GCSE was applied to the classic control problem of a cart-pole balancing system. This problem is often used as an example of inherently unstable and dynamic systems to demonstrate both modern and classic control techniques [46–48] or

Table 1
The initial parameters before training.

Parameters	Value
$[\sigma_{\min}, \sigma_{\max}]$	[0, 2]
$[m_{\min}, m_{\max}]$	[0, 2]
$[\theta_{\min}, \theta_{\max}]$	[−2, 2]
$[w_{\min}, w_{\max}]$	[−20, 20]

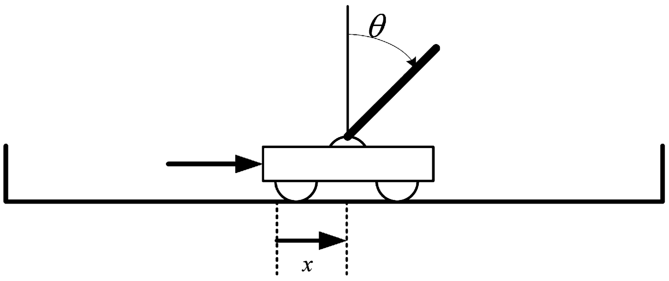

Fig. 9. The cart-pole balancing system.

reinforcement learning schemes [15–21], and is now used as a control benchmark. As shown in Fig. 9, a cart-pole balancing problem is the problem of learning how to balance an upright pole. The bottom of the pole is hinged to a cart that travels along a finite-length track to its right or left. Both the cart and the pole can move only in the vertical plane; that is, each has only one degree of freedom.

There are four state variables in the system: θ, the angle of the pole from an upright position (in degrees); $\dot{\theta}$, the angular velocity of the pole (in degrees/seconds); x, the horizontal position of the cart's center (in meters); and $\dot{x}$, the velocity of the cart (in meters/seconds). The only control action is f, which is the amount of force (in Newtons) applied to a cart to move it left or right. The system fails when the pole falls past a certain angle ($\pm 12^\circ$ is used here) or when the cart runs into the bounds of its track (the distance is 2.4 m from the center to each bound of the track). The goal of this control problem is to determine a sequence of forces that is applied to the cart to balance the pole upright. The equations of motion are as follows:

$$\theta(t+1) = \theta(t) + \Delta\dot{\theta}(t), \tag{25}$$

$$\begin{aligned}\dot{\theta}(t+1) = {} & \dot{\theta}(t) + \Delta((m+m_p)g\sin\theta(t))/((4/3)(m+m_p)l - m_p l\cos^2\theta(t)) \\ & - \frac{\cos\theta(t)\left[f(t) + m_p l\dot{\theta}(t)^2\sin\theta(t) - \mu_c\,\mathrm{sgn}(\dot{x}(t))\right]}{(4/3)(m+m_p)l - m_p l\cos^2\theta(t)} \\ & - \frac{(\mu_p(m+m_p)\dot{\theta}(t)/m_p l)}{(4/3)(m+m_p)l - m_p l\cos^2\theta(t)},\end{aligned} \tag{26}$$

$$x(t+1) = x(t) + \Delta\dot{x}(t), \tag{27}$$

$$\begin{aligned}\dot{x}(t+1) = {} & \dot{x}(t) + \Delta\frac{f(t) + m_p l[\dot{\theta}(t)^2\sin\theta(t) - \ddot{\theta}(t)\cos\theta(t)]}{(m+m_p)} \\ & - \frac{\mu_c\,\mathrm{sgn}(\dot{x}(t))}{(m+m_p)},\end{aligned} \tag{28}$$

where

$$\begin{aligned}& l = 0.5\,\text{m, the length of the pole;} \\ & m = 1.1\,\text{kg, combined mass of the pole and the cart;} \\ & m_p = 0.1\,\text{kg, mass of the pole;} \\ & g = 9.8\,\text{m/s, acceleration due to the gravity;} \\ & \mu_c = 0.0005,\ \text{coefficient of friction of the cart on the track,} \\ & \mu_p = 0.000002,\ \text{coefficient of friction of the pole on the cart,} \\ & \Delta = 0.02(\text{s}),\ \text{sampling interval.}\end{aligned} \tag{29}$$

The constraints on the variables were $-12^\circ \leqslant \theta \leqslant 12^\circ$, $-2.4\,\mathrm{m} \leqslant x \leqslant 2.4\,\mathrm{m}$, and $-10\,\mathrm{N} \leqslant f \leqslant 10\,\mathrm{N}$. A control strategy is deemed successful if it can balance a pole for 100,000 time steps.

The four input variables $(\theta, \dot{\theta}, x, \dot{x})$ and the output $f(t)$ were normalized between 0 and 1 over the following ranges: θ: [−12,12], $\dot{\theta}$: [−240,240],: [−2.4, 2.4], $\dot{x}$: [−2.4, 2.4], and $f(t)$: [−10,10]. The ranges of $\dot{\theta}$ and $\dot{x}$ were calculated by experiments with extreme boundary conditions. The car was placed at the location of 2.4 m (or −2.4 m) with the pole angle set at -12° (or 12°), respectively. Then the maximum force of −10N (or 10N) was applied to the cart. When the system failed, the observed $\dot{\theta}$ and $\dot{x}$ were the boundaries.

The four normalized state variables were used as inputs to the RWNFS. The coding of a rule in a chromosome is the form shown in Fig. 5. The values are floating-point numbers initially assigned to the R-GCSE. The fitness function in this example is defined in Eq. (24) to train the RWNFS and represents how long before the pole falls past a certain angle ($|\theta| > 12^\circ$) or before the cart runs into the bounds of its track ($|x| > 2.4\,\mathrm{m}$).

Table 2
The initial parameters before training.

Parameters	Value
Fuzzy rules	4
Group size	30
Crossover rate	0.4
Mutation rate	0.15

The initial parameters of the R-GCSE were determined by parameter exploration. The first study in parameter exploration was proposed by De Jong [51]. As shown in [51], a small population size is good for the initial performance, and a large population size is good for long-term performance. Moreover, a low mutation rate is good for on-line performance, and a high mutation rate is good for off-line performance. In [52], the author found from his simulation that the best population size and mutation rate were 30 and 0.01, respectively.

In this study, the parameters were found using the method given in [52]. Therefore, the number of fuzzy rules was from 2 to 20 in increments of 1, the group size was from 10 to 100 in increments of 10, the crossover rate was from 0.25 to 1 in increments of 0.05, and the mutation rate was from 0 to 0.3 in exponential increments. The parameters set for the R-GCSE are shown in Table 2.

There were four rules that were used to construct the RWNFS. A total of 30 runs were performed. Each run started at different initial states ($\dot{\theta}$ and $\dot{x}$ were set to 0, and θ and x were set randomly according to the predefined ranges). The learning curve of the R-GCSE after 30 runs is shown in Fig. 10(a). The learning curve represents how long before the cart-pole balancing system failed. As shown in this figure, the RWNFS learned to balance the pole in the 198th generation on average. The standard deviation in Fig. 10(a) is 72.43. When the R-GCSE was stopped, the best combination of strings from the groups in the final generation was

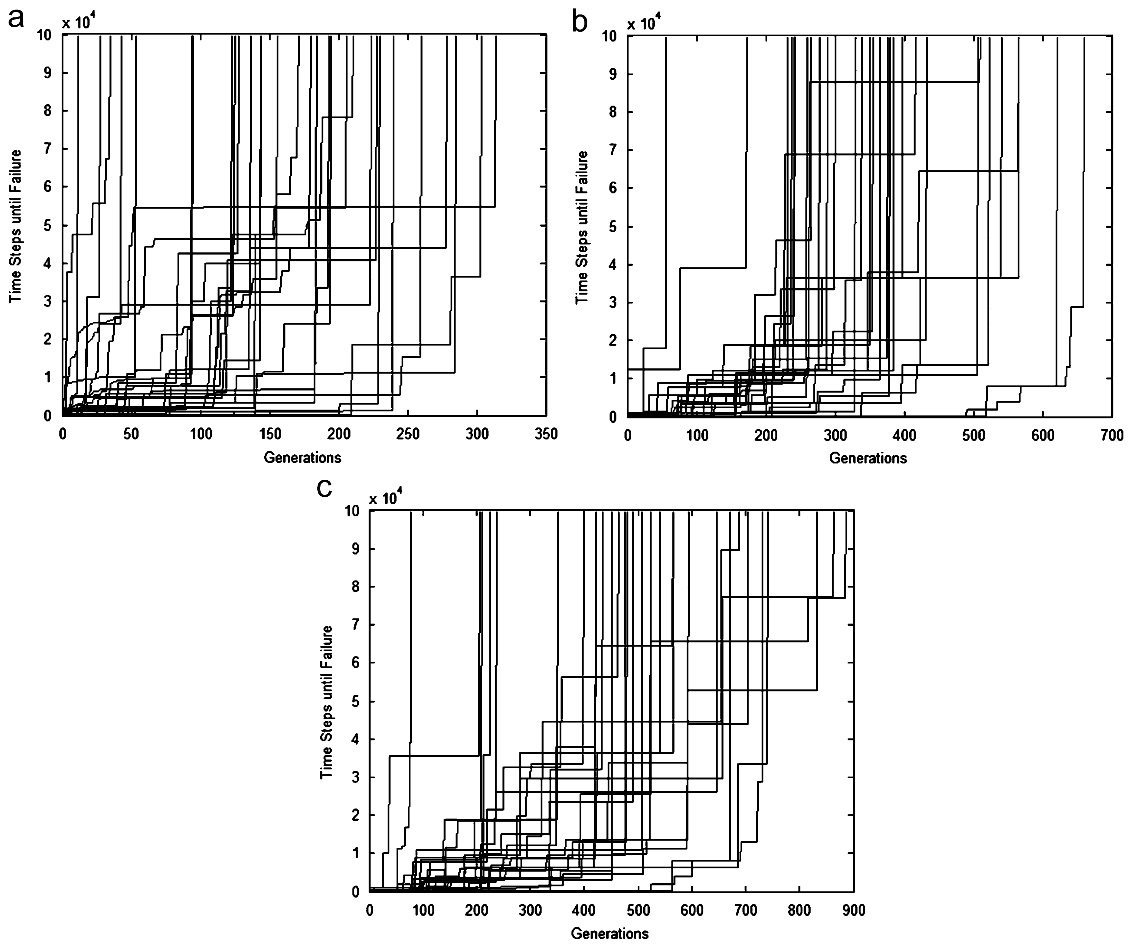

Fig. 10. The performance of (a) the R-GCSE method; (b) R-SE method [49]; (c) R-GA method [26] on the cart pole balancing system.

selected and tested on the cart-pole balancing system. The obtained fuzzy rules of the RWNFS are as follows:

R^1 : If I_{11} is $A_{1,1}(0.134, 0.038)$ and I_{12} is $A_{2,1}(0.61, 1.12)$
and I_{13} is $A_{3,1}(1.02, 0.71)$ and I_{14} is $A_{4,1}(0.83, 0.12)$
Then $\hat{y}_1^1 = 1.533\varphi_{0.0} - 0.147\varphi_{1.0} + 0.011\varphi_{1.1} + 0.147\varphi_{2.0}$

R^2 : If I_{21} is $A_{1,2}(0.43, 0.93)$ and I_{22} is $A_{2,2}(0.28, 0.81)$
and I_{23} is $A_{3,2}(0.32, 0.18)$ and I_{24} is $A_{4,2}(0.61, 0.87)$
Then $\hat{y}_2^1 = -0.45\varphi_{0.0} + 0.42\varphi_{1.0} + 0.013\varphi_{1.1} - 0.783\varphi_{2.0}$

R^3 : If I_{31} is $A_{1,3}(0.96, 0.21)$ and I_{32} is $A_{2,3}(0.56, 0.14)$
and I_{33} is $A_{3,3}(1.01, 0.46)$ and I_{34} is $A_{4,3}(0.38, 0.39)$
Then $\hat{y}_2^1 = 0.074\varphi_{0.0} + 0.23\varphi_{1.0} - 0.17\varphi_{1.1} - 0.38\varphi_{2.0}$

R^4 : If I_{41} is $A_{1,4}(0.64, 0.58)$ and I_{42} is $A_{2,4}(0.21, 0.34)$
and I_{43} is $A_{3,4}(0.56, 0.31)$ and I_{44} is $A_{4,4}(0.82, 0.58)$
Then $\hat{y}_4^1 = 0.183\varphi_{0.0} - 0.193\varphi_{1.0} + 0.412\varphi_{1.1} + 0.039\varphi_{2.0}$

The simulation was carried out for 30 runs. The results, which consisted of the pole angle, cart position and controller output, are shown in Fig. 11. Each line in Fig. 11 represents each run with a different initial state. The results shown in this figure are the first 1,000 time steps in the 100,000 control time steps. As shown in Fig. 11, the R-GCSE successfully controlled the cart-pole balancing system in 30 runs.

In this example, in order to demonstrate the effectiveness and efficiency of the R-GCSE, the reinforcement symbiotic evolution (R-SE) [49] and reinforcement genetic algorithm (R-GA) [26] were applied to the same problem. In the R-SE and R-GA, the parameters were set according to [52]. Therefore, the number of fuzzy rules was from 2 to 20 in increments of 1, the population

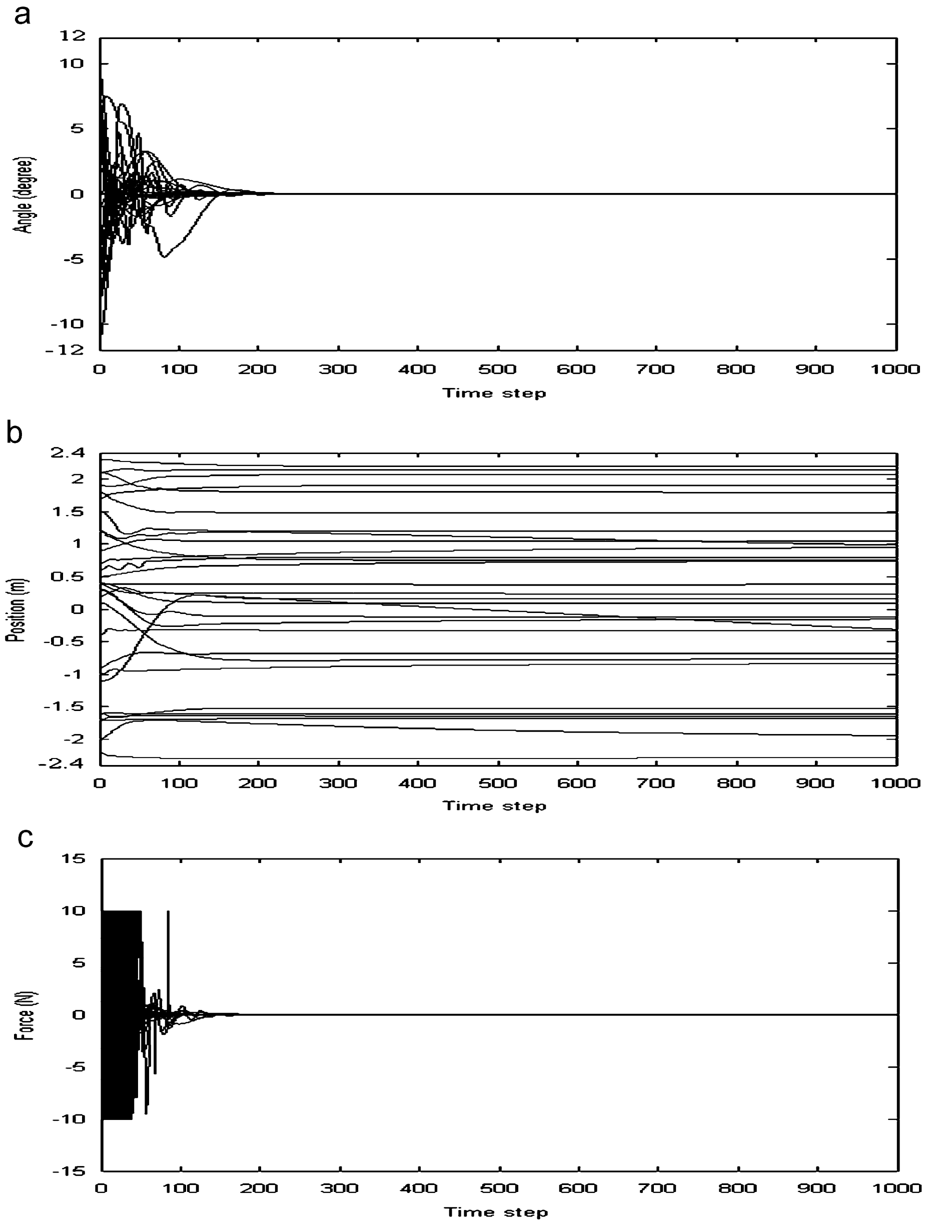

Fig. 11. Control results of the cart and pole balancing system using the R-GCSE in Example 1. (a) Angle of the pole; (b) position of the cart; (c) control face.

size was from 10 to 250 in increments of 10, the crossover rate was from 0.25 to 1 in increments of 0.05, and the mutation rate was from 0 to 0.3 in exponential increments. The parameters set for two methods (the R-SE and R-GA) were as follows: (1) the numbers of fuzzy rules were both set to 4; (2) the population sizes of the R-SE and R-GA were 170 and 70, respectively; (3) the crossover rates of the R-SE and R-GA were 0.55 and 0.6, respectively; (4) the mutation rate of the R-SE and R-GA were 0.08 and 0.12, respectively.

A total of 30 runs were performed. Each run started at different initial states. The fitness is defined in Eq. (24). The learning curves of the R-SE and R-GA after 30 runs are shown in Fig. 10(b) and (c). The R-SE and R-GA learned to balance the pole in the 346th and 514th generations on average. The standard deviations in Fig. 10(b) and (c) are 141.37 and 199.12. The R-GCSE only compares the performance of the fitness value with the R-SE and R-GA. This is because, in the reinforcement learning signal design that is adopted in this study, a well-performing controller is defined as a controller that does not exceed the predefined boundaries. As shown in Fig. 11, the control capabilities of the R-GCSE are better than those of [26,49].

Genetic reinforcement learning for neuro control (GENITOR) [48], symbiotic adaptive neuro-evolution (SANE) [43], temporal difference and genetic algorithm-based reinforcement learning (TDGAR) [30], combination of online clustering and Q-value based GA for reinforcement fuzzy system (CQGAF) [33], and enforce sub-population (ESP) [34] methods were applied to the same control problem. The simulation results are listed in Table 3. The number of pole-balance trials (which reflects the number of training episodes required) and the CPU time are shown in Table 3. This experiment used a Pentium III chip with a 400 MHz CPU, a 512 MB memory, and the visual C++ 6.0 simulation software.

A total of 30 runs were performed. Each run started at different initial states. The initial parameters of these methods [30,33,34,43,48] were determined according to [52]. In [48], the normal evolution algorithm was used to evolve the weights of a fully-connected two-layer neural network, with additional connections from each input unit to the output layer. After trial-and-error tests, the network size was 10 in [48]. In [43], the symbiotic evolution algorithm was used to evolve a two-layer neural network. In [43], the network size was 10.

The TDGAR [30] consists of the critic network and action network to the learning system. The critic network is a standard three-layer feedforward network that uses sigmoid functions in the hidden layer and output layer. The action network is a fuzzy neural network with five layers of nodes, and each layer performs one stage of the fuzzy inference process. There are five hidden nodes and five rules in the critic network and the action network.

In CQGAF [33], the fuzzy controller with Q-value based GA was proposed to solve controller problems. After trial-and-error tests, the final average number of rules in CQGAF from 30 runs was 8 using the on-line clustering algorithm.

In the ESP [34], the author proposed using ESP to evaluate the solution locally. There are five sub-populations in the ESP. The other parameters set for five methods [30,33,34,43,48] were as follows: (1) the population sizes of the five methods were 130, 170, 100, 130 and 40, respectively; (2) the crossover rates of the five methods were 0.45, 0.55, 0.35, 0.45 and 0.5, respectively; (3) the mutation rate of the five methods were 0.21, 0.17, 0.16, 0.24 and 0.18, respectively.

As shown in Table 3, the proposed R-GCSE method is feasible and effective and obtains smaller CPU times than other existing methods.

To demonstrate the efficiency of the RWNFS, two different networks are introduced in this example: the RWNFS and the TSK-type recurrent neuro-fuzzy network (TRFN) [39]. There are four rules that are used to construct the TRFN. The parameters of the R-GCSE used to train the TRFN are the same as the parameters of the R-GCSE used to train the RWNFS. A performance (time steps and CPU time) comparison of the two models is shown in Table 4.

Table 3
Comparison of time steps and CPU time for various existing models in Example 1.

Method	Mean		Best		Worst		Standard deviation	
	Steps	Seconds	Steps	Seconds	Steps	Seconds	Steps	Seconds
GENITOR [48]	1981	69.65	519	20.54	3143	185.51	598.78	62.54
SANE [43]	879	34.25	89	11.15	1541	75.34	337.91	24.97
R-GA [26]	514	25.34	78	8.23	887	64.75	199.12	23.88
R-SE [49]	346	21.37	56	7.87	658	61.39	141.37	23.67
TDGAR [30]	327	31.34	23	10.84	469	69.91	124.77	24.28
ESP [34]	294	18.92	14	3.08	401	34.74	91.56	8.37
CQGAF [33]	264	28.77	15	6.24	376	57.49	95.82	14.67
R-GCSE	198	11.64	12	2.34	314	26.54	72.43	7.29

Table 4
Comparison of time steps and CPU time for two different networks in Example 1.

Method	Mean		Best		Worst		Standard deviation	
	Steps	Seconds	Steps	Seconds	Steps	Seconds	Steps	Seconds
RWNFS	198	12.64	12	2.34	314	28.54	72.43	7.29
RTNFN	232	15.83	14	3.43	331	31.45	81.58	7.93

Table 5
Comparison of time steps and CPU time for two different methods.

Method	Mean		Best		Worst		Standard deviation	
	Steps	Seconds	Steps	Seconds	Steps	Seconds	Steps	Seconds
Type I	257	15.34	13	2.94	371	31.58	89.67	8.19
Type II	346	21.37	56	7.87	658	61.39	141.37	23.67
Type III	198	11.64	12	2.34	314	27.54	72.43	7.29

To demonstrate the efficiency of the proposed GSE and ECCS, in this example, three different methods, the R-GCSE without the ECCS (Type I), the R-SE method (Type II), and the R-GCSE (Type III), were used. In the Type I method, each group performed the two-point crossover strategy independently. In the Type II method, the R-SE [49] was adopted. In the Type III method, the R-GCSE used the ECCS to perform crossover strategy. In the Type I method, the parameters were set according to [52]. The parameters set for Type I method were as follows: (1) the number of fuzzy rules was 4; (2) the population size was 40; (3) the crossover rate was 0.45; (4) the mutation rate was 0.07. The performance (time steps and CPU time) of the three types of methods is shown in Table 5. The R-GCSE (Type III) performs better than the other two types of methods. In Table 5, a comparison of the Type III and Type I methods is given, from which it can be observed that the ECCS can reduce time steps and the CPU time.

Although the R-GCSE performs better than other methods in the cart-pole balancing problem, it is too easy to find solutions quickly for this problem. With regards to this, extensions of a basic cart-pole balancing problem have been used. In [53], the author proposed several variations of the cart-pole balancing problem. The most challenging extension of the cart-pole balancing problem in [53] was a double pole balancing problem, where two poles of different lengths must be balanced synchronously.

Therefore, a double pole balancing problem was used to evaluate the R-GCSE. There are six state variables in the system:

θ_i, the angle of the ith pole; $\dot{\theta}_i$, the angular velocity of the ith pole; x, the position of the cart; and $\dot{x}$, the velocity of the cart. The only control action is f, which is the amount of force applied to the cart to move it left or right. The system fails when the pole falls past a certain angle ($\pm 36^\circ$ was used here) or when the cart runs into the bounds of its track (the distance is 2.4 m from the center to each bound of the track). The equations of motion for N poles balanced on a single cart are as follows:

$$\ddot{x} = \frac{F - \mu_c \,\mathrm{sgn}(\dot{x}) + \sum_{i=1}^{N} \tilde{F}_i}{M + \sum_{i=1}^{N} \tilde{m}_i}, \tag{30}$$

$$\ddot{\theta}_i = -\left(\frac{3}{4l_i}\right)\left(\ddot{x}\cos\theta_i + g\sin\theta_i + \frac{\mu_{p_i}\dot{\theta}_i}{m_i l_i}\right), \tag{31}$$

where $\tilde{F}_i$ is the effective force from ith pole, the equation of $\tilde{F}_i$ if

$$\tilde{F}_i = m_i l_i \dot{\theta}_i^2 \sin\theta_i + \frac{3}{4} m_i \cos\theta_i \left(\frac{\mu_{p_i}\dot{\theta}_i}{m_i l_i} + g\sin\theta_i\right), \tag{32}$$

where $\tilde{m}_i$ is the effective mass ot the ith pole, the equation of $\tilde{m}_i$ is

$$\tilde{m}_i = m_i\left(1 - \tfrac{3}{4}\cos^2\theta_i\right). \tag{33}$$

The parameters used for the double pole problem are shown in Table 6. The parameters set for the R-GCSE are shown in Table 7. A total of 30 runs were performed. Each run started at different initial states.

Table 6
The parameters for the double pole balancing problem.

Parameters	Description	Value
x	Position of the cart	[−2.4,2.4] m
θ	Angle of the pole	[−36,36] deg.
F	Force applied to cart	[−10,10] N
l_1	Half length of 1st pole	0.5 m
l_2	Half length of 2nd pole	0.05 m
M	Mass of cart	1.0 kg
m_1	Mass of the 1st pole	0.1 kg
m_2	Mass of the 2nd pole	0.01 kg
μ_c	Coefficient of friction of cart on track	0.0005
μ_P	Coefficient of friction if ith pole's hinge	0.000002

Table 7
The initial parameters before training.

Parameters	Value
Fuzzy rules	6
Group size	60
Crossover rate	0.5
Mutation rate	0.18

The R-SE [49], R-GA [26], GENITOR [48], SANE [43], TDGAR [30], CQGAF [33], and ESP [34] were also applied to the same problem. In these seven methods, the parameters were set according to [52]. A total of 30 runs were performed. Each run started at different initial states. In [26,49], the numbers of fuzzy rules were both set to 6. In [48], the network size was eighteen. In [43], the network size was sixteen. In TDGAR [30], there were 10 hidden nodes in the critic network and 10 rules in the action network. In the CQGAF [33], the final average number of rules in CQGAF of 30 runs was 13. In the ESP [34], there were eight sub-populations. The parameters set for seven methods [26,30,33,34,43,48,49] were as follows: (1) the population sizes of the seven methods were 210, 120, 180, 240, 160, 200 and 60, respectively; (2) the crossover rates of the seven methods were 0.5, 0.6, 0.40, 0.55, 0.45, 0.35 and 0.45, respectively; (3) the mutation rate of the five methods were 0.12, 0.22, 0.21, 0.15, 0.26, 0.14 and 0.16, respectively.

This paper compares time steps and the CPU time with those of other existing methods [26,30,33,34,43,48,49] in a double pole balancing problem in Table 8. A comparison shows that the R-GCSE is feasible and effective and requires less CPU time than other existing models in the double pole balancing problem.

5.2. Example 2: Control of a ball and beam system

The ball and beam system [49,50] is shown in Fig. 12. The beam is made to rotate in vertical plane by applying a torque at the center of rotation, and the ball is free to roll along the beam. The goal is for the ball to remain in contact with the beam. The ball and beam system can be written in state space form as

$$\begin{bmatrix}\dot{x}_1\\ \dot{x}_2\\ \dot{x}_3\\ \dot{x}_4\end{bmatrix} = \begin{bmatrix} x_2 \\ B(x_1x_4^2 - G\sin x_3) \\ x_4 \\ 0\end{bmatrix} + \begin{bmatrix}0\\0\\0\\1\end{bmatrix}u,$$
$$y = x_1, \tag{34}$$

where $x = (x_1, x_2, x_3, x_4)^{\mathrm{T}} \equiv (r, \dot{r}, \theta, \dot{\theta})^{\mathrm{T}}$ is the state of the system and $y = x_1 \equiv r$ is the output of the system. The control u is the angular acceleration ($\ddot{\theta}$), and the parameters $B = 0.7143$ and $G = 9.81$ were chosen in this system. The purpose of control is to determine $u(x)$ such that the closed-loop system output y will converge to zero from different initial conditions.

According to the input/output-linearization algorithm [54], the control law $u(x)$ is determined as follows: in state x, $v(x) = -\alpha_3\phi_4(x) - \alpha_2\phi_3(x) - \alpha_1\phi_2(x) - \alpha_0\phi_1(x)$, where $\phi_1(x) = x_1$, $\phi_2(x) = x_2$, $\phi_3(x) = -BG\sin x_3$, $\phi_4(x) = -BGx_4\cos x_3$, and the α_i is chosen so that $s^4 + \alpha_3 s^3 + \alpha_2 s^2 + \alpha_1 s + \alpha_0$ is the Hurwitz polynomial. We compute $a(x) = -BG\cos x_3$ and $b(x) = BGx_4^2\sin x_3$; then $u(x) = [v(x) - b(x)]/a(x)$.

Table 8
Comparison of time steps and CPU time in a double pole balancing problem.

Method	Mean		Best		Worst		Standard deviation	
	Steps	Seconds	Steps	Seconds	Steps	Seconds	Steps	Seconds
GENITOR [48]	31.760	412.49	6.560	91.85	60.120	572.54	14892.45	109.69
SANE [43]	14800	225.73	3140	63.87	25.560	276.54	4589.87	60.23
R-GA [26]	12250	218.34	2870	49.56	21.150	251.68	3404.33	52.97
R-SE [49]	9790	192.67	2150	47.49	16.870	241.67	2943.62	47.38
TDGAR [30]	8970	231.45	1987	56.37	15.230	258.74	2314.24	54.05
CQGAF [33]	6790	187.96	1230	39.54	9870	238.95	1691.38	46.32
ESP [34]	4120	123.73	396	26.18	7190	214.51	1186.73	37.89
R-GCSE	3380	98.47	267	21.48	6390	191.78	1056.54	32.45

The four input variables $(r, \dot{r}, \theta, \dot{\theta})$ and the output $u(x)$ were normalized between 0 and 1. The values were floating-point numbers initially assigned to the R-GCSE. In the R-GCSE, the fitness function is also defined in Eq. (24) to train the RWNFS, and represents how long before the beam deviates beyond a certain angle ($|\theta| > 12^\circ$) or before the ball reaches the end of the beam ($|r| > 2\,\mathrm{m}$). In this example, the parameters were set according to [52]. The parameters set for the R-GCSE are shown in Table 9.

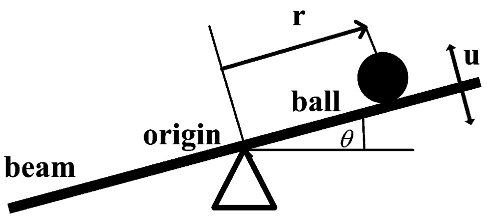

Fig. 12. The ball and beam system.

Table 9
The initial parameters before training.

Parameters	Value
Fuzzy rules	5
Group size	40
Crossover rate	0.45
Mutation rate	0.18

There are five rules that were used to construct the RWNFS. A total of 30 runs were performed. Each run started at the same initial state. The learning curve of the RWNFS is shown in Fig. 13(a). The RWNFS learned to balance the ball in the 121st generation on average. The standard deviation in Fig. 13(a) is 37.21. When the learning process was stopped, the best combination of strings from groups at the final generation was selected and tested on the ball and beam system. The obtained fuzzy rules of the RWNFS are as follows:

$$R^1: \text{If } I_{11} \text{ is } A_{1,1}(0.37, 0.42) \text{ and } I_{12} \text{ is } A_{2,1}(0.51, 0.21) \text{ and } I_{13} \text{ is } A_{3,1}(0.66, 0.18) \text{ and } I_{14} \text{ is } A_{4,1}(0.13, 0.31)$$
$$\text{Then } \hat{y}_1^1 = -1.06\varphi_{0.0} + 2.41\varphi_{1.0} - 0.31\varphi_{1.1} - 5.06\varphi_{2.0} + 0.36\varphi_{2.1}$$

$$R^2: \text{If } I_{21} \text{ is } A_{1,2}(0.81, 0.37) \text{ and } I_{22} \text{ is } A_{2,2}(0.63, 0.63) \text{ and } I_{23} \text{ is } A_{3,2}(0.57, 0.23) \text{ and } I_{24} \text{ is } A_{4,2}(0.33, 0.011)$$
$$\text{Then } \hat{y}_2^1 = -3.11\varphi_{0.0} + 1.01\varphi_{1.0} + 0.07\varphi_{1.1} + 0.29\varphi_{2.0} - 1.33\varphi_{2.1}$$

$$R^3: \text{If } I_{31} \text{ is } A_{1,3}(0.37, 0.61) \text{ and } I_{32} \text{ is } A_{2,3}(0.72, 0.22) \text{ and } I_{33} \text{ is } A_{3,3}(0.93, 0.35) \text{ and } I_{34} \text{ is } A_{4,3}(0.89, 0.17)$$
$$\text{Then } \hat{y}_3^1 = 0.23\varphi_{0.0} - 0.28\varphi_{1.0} - 1.19\varphi_{1.1} + 0.38\varphi_{2.0} - 0.04\varphi_{2.1}$$

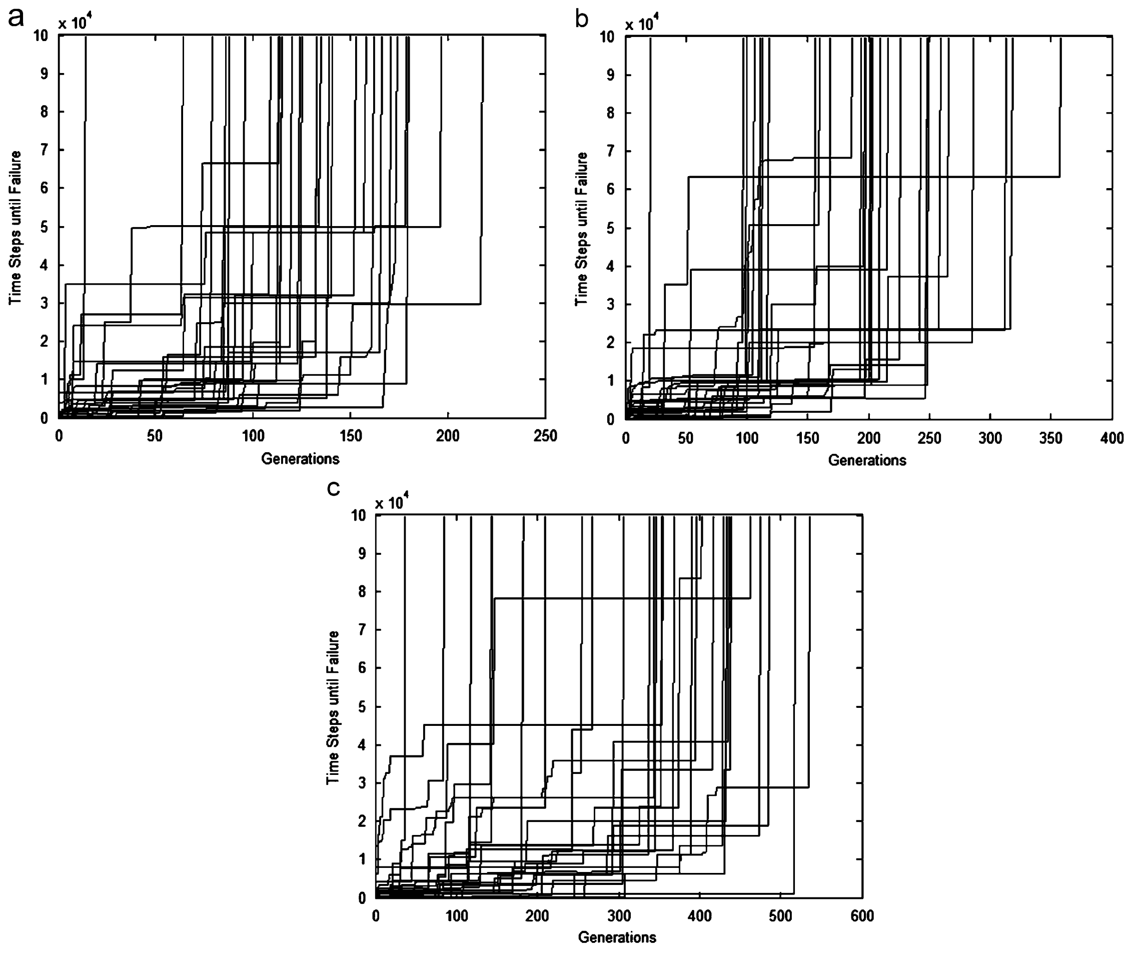

Fig. 13. The performance of (a) The R-GCSE method; (b) R-SE method [49]; and (c) R-GA method [26] on the ball and beam balancing system.

R^4 : If I_{41} is $A_{1,4}(0.46, 0.33)$ and I_{42} is $A_{2,4}(0.12, 0.94)$
and I_{43} is $A_{3,4}(0.39, 0.77)$ and I_{44} is $A_{4,4}(0.07, 0.83)$
Then $\hat{y}_4^1 = 0.06\varphi_{0.0} + 0.17\varphi_{1.0} + 0.35\varphi_{1.1} + 0.15\varphi_{2.0} - 1.46\varphi_{2.1}$

R^5 : If I_{51} is $A_{1,5}(0.68, 0.03)$ and I_{52} is $A_{2,5}(0.33, 0.24)$
and I_{53} is $A_{3,5}(0.87, 0.41)$ and I_{54} is $A_{4,5}(0.94, 0.06)$
Then $\hat{y}_5^1 = -0.74\varphi_{0.0} - 0.63\varphi_{1.0} + 1.24\varphi_{1.1} + 0.91\varphi_{2.0} + 0.04\varphi_{2.1}$

The simulation was run 30 times. The results, which consist of the beam angle, ball position, and controller output, are shown in Fig. 14. The results shown in this figure is from the first 1000 time steps in the 100,000 control time steps. As shown in Fig. 14, the R-GCSE in 30 runs successfully controlled the ball and beam system. The results show that the trained RWNFS has good ability in controlling the ball and beam balancing system.

In this example, as with Example 1, the performance of the R-GCSE was also compared with the performance of other methods (the R-SE [49] and R-GA [26]). In [26,49] the parameters were set according to [52]. The parameters set for the R-SE and R-GA were as follows: (1) the numbers of fuzzy rules were both set to 5; (2) the population sizes of the R-SE and R-GA were 180 and 100, respectively; (3) the crossover rates of the R-SE and R-GA were 0.4 and 0.5, respectively; (4) the mutation rates of the R-SE and R-GA were 0.10 and 0.05, respectively. A total of 30 runs were performed. Each run started at the same initial state. The learning curves of the R-SE and R-GA are shown in Fig. 13(b) and (c). The R-SE [49] and R-GA [26] learned to balance the ball in the 217th generation and 386th generation, on average. The standard deviations in Fig. 13(b) and (c) are 74.21 and 132.68.

The performance (time steps and CPU time) in this example compared with various existing models [26,30,33,34,43,48,49] is shown in Table 10. In [48], the network size was eleven. In [43], the network size was 10. In the TDGAR, there were six hidden nodes in the critic network and six rules in the action network. In

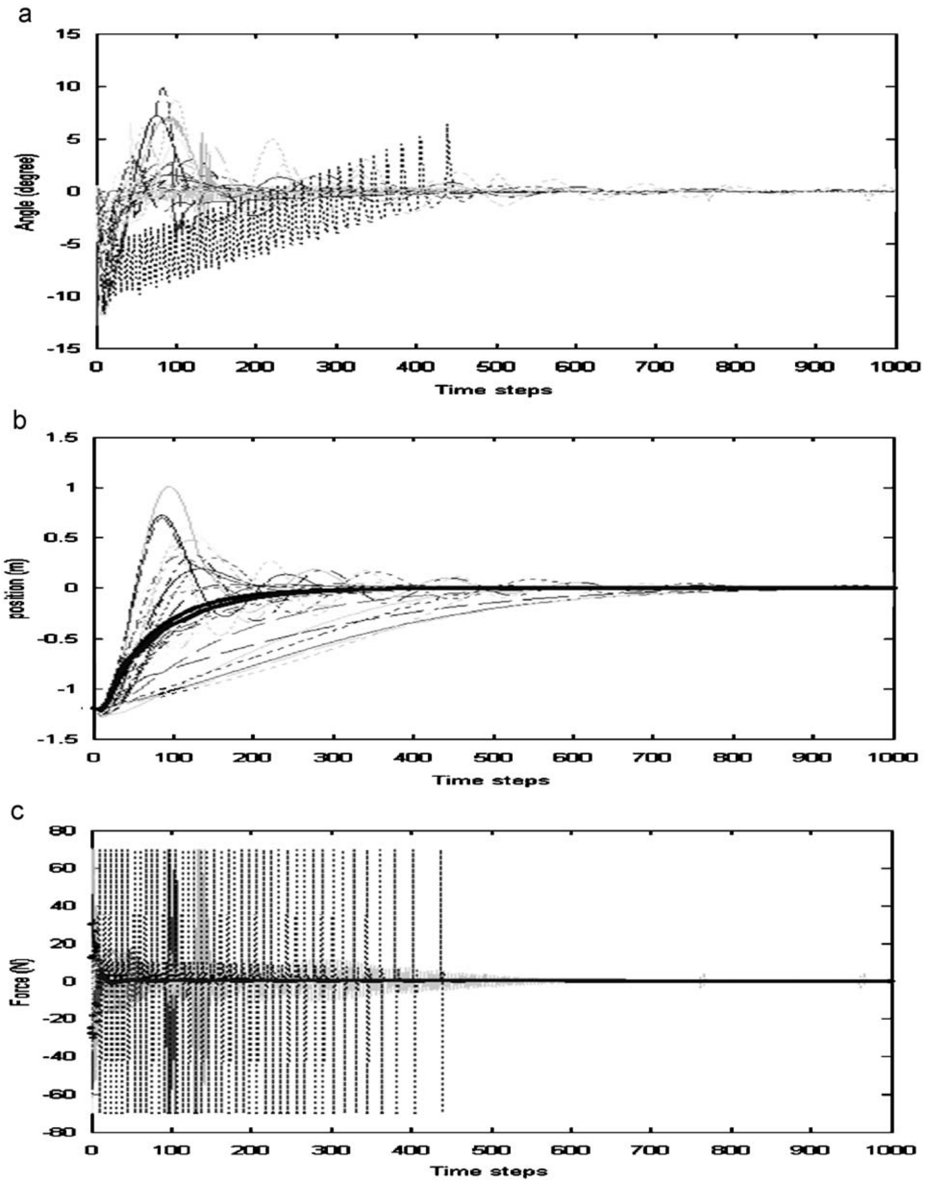

Fig. 14. Control results of the ball and beam balancing system using the R-GCSE in Example 2. (a) Angle of the beam; (b) position of the ball; (c) control force.

Table 10
Comparison of time steps and CPU time for various existing models in Example 2.

Method	Mean		Best		Worst		Standard deviation	
	Steps	Seconds	Steps	Seconds	Steps	Seconds	Steps	Seconds
GENTOR [48]	914	78.43	79	13.04	1531	98.64	334.37	23.80
SANE [43]	697	50.26	52	10.73	912	74.52	218.39	15.19
R-GA [26]	386	30.21	37	8.05	536	42.36	132.68	8.04
R-SE [49]	217	28.51	21	6.87	358	39.91	74.21	7.12
TDGAR [30]	194	31.25	24	9.23	347	46.43	67.36	8.45
ESP [34]	167	17.46	60	3.51	276	31.24	51.91	5.37
CQGAF [33]	148	21.51	18	5.27	264	34.21	48.67	5.91
R-CGSE	121	14.29	14	3.04	218	26.71	37.21	4.64

the CQGAF, the final average number of rules in the CQGAF from 30 runs was 8. In the ESP, there were five sub-populations. The parameters set for five methods [30,33,34,43,48] were as follows: (1) the population sizes of the five methods were 140, 170, 120, 150 and 60, respectively; (2) the crossover rates of the five methods were 0.45, 0.45, 0.35, 0.4 and 0.5, respectively; and (3) the mutation rate of the five methods were 0.16, 0.24, 0.18, 0.12 and 0.21, respectively. A total of 30 runs were performed. Each run started at the same initial state. As shown in Table 10, the R-GCSE has shorter time steps and CPU times than the existing models.

6. Conclusion

In this paper, a recurrent wavelet-based neuro-fuzzy system (RWNFS) with the reinforcement group cooperation-based symbiotic evolution method (R-GCSE) was proposed. The R-GCSE can evaluate fuzzy rules locally and make groups cooperate with each other to generate better chromosomes by using an elite-based compensation crossover strategy (ECCS). The advantages of the R-GCSE are summarized as follows: (1) the R-GCSE uses group-based population to evaluate fuzzy rules locally; (2) the R-GCSE uses the ECCS to let the better solutions from different groups cooperate in order to generate better solutions in the next generation; and (3) the R-GCSE indeed performs better and converges more quickly than some genetic methods. Computer simulations show that the R-GCSE performs better than the other methods.

Acknowledgment

This work is supported in part by the National Science Council, Taiwan. ROC under Grant NSC 95-2221-E-009-214 and NSC 95-2752-E-009-011-PAE.

References

[1] C.T. Lin, C.S.G. Lee, Neural Fuzzy Systems: A Neuro-Fuzzy Synergism to Intelligent System, Prentice-Hall, NJ, 1996.
[2] G.G. Towell, J.W. Shavlik, Extracting refined rules from knowledge-based neural networks, Mach. Learn. 13 (1993) 71–101.
[3] C.J. Lin, C.T. Lin, An ART-based fuzzy adaptive learning control network, IEEE Trans. Fuzzy Syst. 5 (4) (1997) 477–496.
[4] L.X. Wang, J.M. Mendel, Generating fuzzy rules by learning from examples, IEEE Trans. Syst. Man Cybern. 22 (6) (1992) 1414–1427.
[5] T. Takagi, M. Sugeno, Fuzzy identification of systems and its applications to modeling and control, IEEE Trans. Syst. Man Cybern. 15 (1985) 116–132.
[6] C.F. Juang, C.T. Lin, An on-line self-constructing neural fuzzy inference network and its applications, IEEE Trans. Fuzzy Syst. 6 (1) (1998) 12–31.
[7] J.S.R. Jang, ANFIS: adaptive-network-based fuzzy inference system, IEEE Trans. Syst. Man Cybern. 23 (3) (1993) 665–685.
[8] 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 Syst. 9 (5) (2001) 751–759.
[9] H. Takagi, N. Suzuki, T. Koda, Y. Kojima, Neural networks designed on approximate reasoning architecture and their application, IEEE Trans. Neural Networks 3 (5) (1992) 752–759.
[10] E. Mizutani, J.S.R. Jang, Coactive neural fuzzy modeling, in: Proceedings of International Conference on Neural Networks, 1995, pp. 760–765.
[11] C.J. Lin, C.C. Chin, Prediction and identification using wavelet-based recurrent fuzzy neural networks, IEEE Trans. Syst. Man Cybern. Part B 34 (5) (2004) 2144–2154.
[12] K.S. Narendra, K. Parthasarathy, Identification and control of dynamical systems using neural networks, IEEE Trans. Neural Networks 1 (1) (1990) 4–27.
[13] C.F. Juang, C.T. Lin, A recurrent self-organizing neural fuzzy inference network, IEEE Trans. Neural Networks 10 (4) (1999) 828–845.
[14] P.A. Mastorocostas, J.B. Theocharis, A recurrent fuzzy-neural model for dynamic system identification, IEEE Trans. Syst. Man Cybern. Part B 32 (2) (2002) 176–190.
[15] X. Xu, H.G. He, Residual-gradient-based neural reinforcement learning for the optimal control of an acrobat, in: Proceedings of IEEE International Conference on Intelligent Control, 2002, pp. 27–30.
[16] O. Grigore, Reinforcement learning neural network used in control of nonlinear systems, in: Proceedings of IEEE International Conference on Industrial Technology, vol. 1, 2000, pp. 19–22.
[17] A.G. Barto, R.S. Sutton, C.W. Anderson, Neuron like adaptive elements that can solve difficult learning control problem, IEEE Trans. Syst. Man Cybern. 13 (5) (1983) 834–847.
[18] C.J. Lin, A GA-based neural network with supervised and reinforcement learning, J. Chin. Inst. Electr. Eng. 9 (1) (2002) 11–25.
[19] X.W. Yan, Z.D. Deng, Z.Q. Sun, Competitive Takagi–Sugeno fuzzy reinforcement learning, in: Proceedings of IEEE International Conference on Control Applications, 2001, pp. 878–883.
[20] C.T. Lin, C.P. Jou, GA-based fuzzy reinforcement learning for control of a magnetic bearing system, IEEE Trans. Syst. Man Cybern. Part B 30 (2) (2000) 276–289.
[21] H.R. Berenji, P. Khedkar, Learning and tuning fuzzy logic controllers through reinforcements, IEEE Trans. Neural Networks 3 (5) (1992) 724–740.
[22] D.E. Goldberg, Genetic Algorithms in Search Optimization and Machine Learning, Addison-Wesley, Reading, MA, 1989.
[23] J.K. Koza, Genetic Programming: On the Programming of Computers by Means of Natural Selection, MIT Press, Cambridge, MA, 1992.
[24] L.J. Fogel, Evolutionary programming in perspective: the top-down view, in: J.M. Zurada, R.J. Marks II, C. Goldberg (Eds.), Computational Intelligence: Imitating Life, IEEE Press, Piscataway, NJ, 1994.
[25] I. Rechenberg, Evolution strategy, in: J.M. Zurada, R.J. Marks II, C. Goldberg (Eds.), Computational Intelligence: Imitating Life, IEEE Press, Piscataway, NJ, 1994.
[26] C.L. Karr, Design of an adaptive fuzzy logic controller using a genetic algorithm, in: Proceedings of the Fourth International Conference on Genetic Algorithms, 1991, pp. 450–457.
[27] M. Lee, H. Takagi, Integrating design stages of fuzzy systems using genetic algorithms, in: Proceedings of the Second IEEE International Conference on Fuzzy Systems, San Francisco, CA, 1993, pp. 612–617.
[28] K. Belarbi, F. Titel, Genetic algorithm for the design of a class of fuzzy controllers: an alternative approach, IEEE Trans. Fuzzy Syst. 8 (4) (2000) 398–405.
[29] C.F. Juang, A hybrid of genetic algorithm and particle swarm optimization for recurrent network design, IEEE Trans. Syst. Man Cybern. Part B 34 (2) (2004) 997–1006.
[30] C.T. Lin, C.P. Jou, GA-based fuzzy reinforcement learning for control of a magnetic bearing system, IEEE Trans. Syst. Man Cybern. Part B 30 (2) (2000) 276–289.
[31] C.F. Juang, J.Y. Lin, C.T. Lin, Genetic reinforcement learning through symbiotic evolution for fuzzy controller design, IEEE Trans. Syst. Man Cybern. Part B 30 (2) (2000) 290–302.
[32] K.S. Tang, Genetic algorithms in modeling and optimization, Ph.D. Dissertation, Department of Electronic Engineering, City University Hong Kong, Hong Kong, 1996.

[33] C.F. Juang, Combination of online clustering and Q-value based GA for reinforcement fuzzy system design, IEEE Trans. Fuzzy Syst. 13 (3) (2005) 289–302.
[34] F.J. Gomez, Robust non-linear control through neuroevolution, Ph.D. Disseration, The University of Texas at Austin, 2003.
[35] F. Gomez, J. Schmidhuber, Co-evolving recurrent neurons learn deep memory POMDPs, in: Proceedings of Conference on Genetic and Evolutionary Computation, 2005, pp. 491–498.
[36] V. Kreinovich, O. Sirisaengtaksin, S. Cabrera, Wavelet neural networks are asymptotically optimal approximators for functions of one variable, in: Proceedings of IEEE Conference on Neural Networks, vol. 1, 1994, pp. 299–304.
[37] D.W.C. Ho, P.A. Zhang, J. Xu, Fuzzy wavelet networks for function learning, IEEE Trans. Fuzzy Syst. 9 (1) (2001) 200–211.
[38] J. Zhang, A.J. Morris, Recurrent neuro-fuzzy networks for nonlinear process modeling, IEEE Trans. Neural Networks 10 (2) (1999) 313–326.
[39] S.F. Su, F.Y. Yang, On the dynamical modeling with neural fuzzy networks, IEEE Trans. Neural Networks 13 (6) (2002) 1548–1553.
[40] Z. Michalewicz, Genetic Algorithms+Data Structures = Evolution Programs, Springer, New York, 1999.
[41] R. Tanese, Distributed genetic algorithm, in: Proceedings of International Conference on Genetic Algorithms, 1989, pp. 434–439.
[42] J. Arabas, Z. Michalewicz, J. Mulawka, GAVaPS—A genetic algorithm with varying population size, in: Proceedings of IEEE International Conference on Evolutionary Computation, Orlando, 1994, pp. 73–78.
[43] D.E. Moriarty, R. Miikkulainen, Efficient reinforcement learning through symbiotic evolution, Mach. Learn. 22 (1996) 11–32.
[44] R.E. Smith, S. Forrest, A.S. Perelson, Searching for diverse, cooperative populations with genetic algorithms, Evol. Comput. 1 (2) (1993) 127–149.
[45] O. Cordon, F. Herrera, F. Hoffmann, L. Magdalena, Genetic fuzzy systems evolutionary tuning and learning of fuzzy knowledge bases. Advances in Fuzzy Systems—Applications and Theory, vol. 19, World Scientific Publishing, NJ, 2001.
[46] C.J. Lin, Y.J. Xu, The design of TSK-type fuzzy controllers using a new hybrid learning approach, Int. J. Adaptive Control Signal Process. 20 (2006) 1–25.
[47] K.C. Cheok, N.K. Loh, A ball-balancing demonstration of optimal and disturbance-accommodating control, IEEE Control Syst. Mag. (1987) 54–57.
[48] D. Whitley, S. Dominic, R. Das, C.W. Anderson, Genetic reinforcement learning for neuro control problems, Mach. Learn. 13 (1993) 259–284.
[49] C.J. Lin, Y.J. Xu, Efficient reinforcement learning through dynamical symbiotic evolution for TSK-type fuzzy controller design, Int. J. Gen. Syst. 34 (5) (2005) 559–578.
[50] J. Hauser, S. Sastry, P. Kokotovic, Nonlinear control via approximate input–output linearization: the ball and beam example, IEEE Trans. Autom. Control 37 (3) (1992) 392–398.
[51] K.A. De Jong, Analysis of the behavior of a class of genetic adaptive systems, Ph.D. Disseration, The University of Michigan, Ann Arbor, MI, 1975.
[52] J.J. Grefenstette, Optimization of control parameters for genetic algorithms, IEEE Trans. Syst. Man Cybern. 6 (1) (1986) 122–128.
[53] A. Wieland, Evolving neural network controllers for unstable systems, in: Proceedings of IEEE Conference on Neural Networks, vol. 2, 1991, pp. 667–673.

Yung-Chi Hsu received the B.S. degree in Information Management from Ming-Hsin University of Science and Technology, Taiwan, ROC, in 2002 and the M.S. degree in Computer Science and Information Engineering from Chaoyang University of Technology, Taiwan, ROC. He is currently pursuing the Ph.D. degree at the Department of Electrical and Control Engineering from the National Chiao Tung University, Taiwan, ROC. He is a member of the Phi Tau Phi. He is also a member of the Taiwanese Association for Artificial Intelligence (TAAI). His research interests include neural networks, fuzzy systems, and genetic algorithms.

Sheng-Fuu Lin was born in Tainan, the Republic of China, in 1954. He received the B.S. and M.S. degree in Mathematics from National Normal University in 1976 and 1979, respectively, the M.S. degree in Computer Science from the University of Maryland in 1985, and the Ph.D. degree in Electrical Engineering from the University of Illinois, Champaign, in 1988.

Since 1988, he has been on the faculty of the Department of Electrical and Control Engineering at National Chiao Tung University, Hsinchu, Taiwan, where he is currently a professor.

His research interests include fuzzy systems, genetic algorithms, neural networks automatic target recognition, scheduling, image processing, and image recognition.