Reinforcement Hybrid Evolutionary Learning for Recurrent Wavelet-Based Neurofuzzy Systems

Cheng-Jian Lin, *Member, IEEE*, and Yung-Chi Hsu

***Abstract*—This paper proposes a recurrent wavelet-based neurofuzzy system (RWNFS) with the reinforcement hybrid evolutionary learning algorithm (R-HELA) for solving various control problems. The proposed R-HELA combines the compact genetic algorithm (CGA), and the modified variable-length genetic algorithm (MVGA) performs the structure/parameter learning for dynamically constructing the RWNFS. That is, both the number of rules and the adjustment of parameters in the RWNFS are designed concurrently by the R-HELA. In the R-HELA, individuals of the same length constitute the same group. There are multiple groups in a population. The evolution of a population consists of three major operations: group reproduction using the compact genetic algorithm, variable two-part crossover, and variable two-part mutation. Illustrative examples were conducted to show the performance and applicability of the proposed R-HELA method.**

***Index Terms*—Control, genetic algorithms, neurofuzzy system, recurrent network, reinforcement learning.**

I. Introduction

In recent years, a fuzzy system used for control problems has become a popular research topic [1]–[10]. 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 [11]–[14]. A fuzzy system consists of a set of fuzzy if-then rules. By convention, the selection of fuzzy if-then rules often relies on a substantial amount of heuristic observations to express knowledge of proper strategies. Obviously, it is difficult for human experts to examine all the input–output data from a complex system to find the proper rules for a fuzzy system. To cope with this difficulty, several approaches to generating if-then rules from numerical data have been proposed [6], [8], [11], [54].These methods were developed for supervised learning; that is, the correct "target" output values are given for each input pattern to guide the network's learning. Lin and Chin [11] used mechanisms of rules/neurons update based on errors, while evolving fuzzy rule-based (eR) models [54] used the informative potential of the new data sample as a trigger to update the rule-base.

Manuscript received April 14, 2005; revised February 16, 2006 and May 16, 2006. This work was supported by the National Science Council, R.O.C., under Grant NSC95-2221-E-324-028.

C.-J. Lin is with the Department of Computer Science and Information Engineering, Chaoyang University of Technology, Taichung County 41349, Taiwan, R.O.C. (e-mail: cjlin@mail.cyut.edu.tw).

Y.-C. Hsu is with the Department of Electrical and Control Engineering, National Chiao-Tung University, Taiwan, R.O.C. (e-mail: ericbogi2001@yahoo.com.tw).

Digital Object Identifier 10.1109/TFUZZ.2006.889920

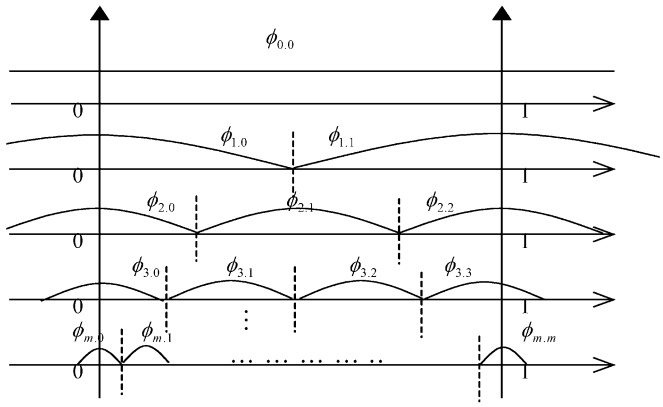


Fig. 1. Wavelet bases are overcompleted and compactly supported.

The most well-known supervised learning algorithm is backpropagation (BP) [3], [6]–[8]. 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 very fast and never find the global solution. In addition, the performance of BP training depends on the initial values of the system parameter. For different network topologies, one has to derive new mathematical expressions for each network layer. If precise training data can be easily obtained, the supervised learning algorithm may be efficient in many applications. For some real-world applications, precise training data are usually difficult and expensive to obtain. For this reason, there has been a growing interest in reinforcement learning problems [15]–[17]. For the reinforcement learning problems, training data are very rough and coarse, and they are only "evaluative" when compared with the "instructive" feedback in the supervised learning problem.

Recently, many evolutionary algorithms, programming types, and strategies, such as the genetic algorithm (GA) [18], genetic programming [19], evolutionary programming [20], and evolution strategies [21], have been proposed. Since they are heuristic and stochastic, they are less likely to get stuck at the local minimum. They are based on populations made up of individuals with specific behaviors similar to certain biological phenomena. These common characteristics have led to the development of evolutionary computation as an increasingly important field.

The evolutionary fuzzy model generates a fuzzy system automatically by incorporating evolutionary learning procedures [22]–[32], such as the GA, which is a well-known procedure. Several genetic fuzzy models, that is, fuzzy models augmented by a learning process based on GAs, have been proposed [22]–[29]. In [22], Karr applied GAs to the design of the

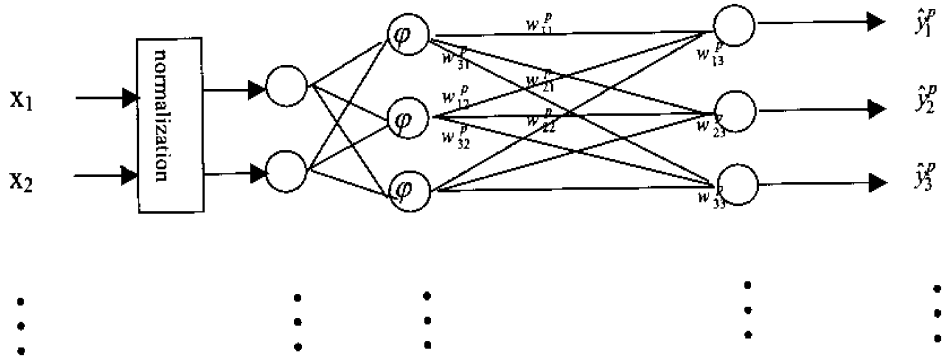


Fig. 2. Schematic diagram of the WNN.

membership functions of a fuzzy controller, with the fuzzy rule set assigned in advance. Since the membership functions and rule sets are codependent, simultaneous design of these two approaches would 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 [23]–[25], [53]. Ishibuchi *et al.* [53] proposed a genetic-algorithm-based method for selecting a small number of significant fuzzy if-then rules to construct a compact fuzzy classification system with high classification power. The rule selection problem is formulated as a combinatorial optimization problem with two objectives: to maximize the number of correctly classified patterns and to minimize the number of fuzzy if-then rules. Lin and Jou [27] proposed GA-based fuzzy reinforcement learning to control magnetic bearing systems. In [28], Juang *et al.* proposed using genetic reinforcement learning in the design of fuzzy controllers. The GA adopted in [28] was based upon traditional symbiotic evolution, which, when applied to fuzzy controller design, complements the local mapping property of a fuzzy rule. However, the aforementioned approaches may require one or both of the following: 1) the numbers of fuzzy rules have to be assigned in advance and 2) the lengths of the chromosomes in the population must be the same.

Recently, several researchers proposed new genetic algorithms for solving the above-mentioned problems. In [29], Bandyopadhyay *et al.* used the variable-length genetic algorithm (VGA) that allows for different lengths of chromosomes in the population. Carse *et al.* [30] used the genetic algorithm [29] to evolve fuzzy rule based controllers. In [31], Tang proposed a hierarchical genetic algorithm, which enables the optimization of a fuzzy system design for a particular application. Juang [32] proposed the CQGAF to simultaneously design the number of fuzzy rules and free parameters in a fuzzy system.

In this paper, we proposed a new hybrid evolutionary learning algorithm to enhance the VGA [29]. The performance of the number of fuzzy rules in the VGA has not been evaluated, so that the best group that has the same length of chromosomes cannot be reproduced many times for each generation. In this paper, we use the elite-based reproduction strategy to keep the best group that has chromosomes of the same length. Therefore, the best group can be reproduced many times for each generation. The elite-based reproduction strategy is similar to the maturing phenomenon in society, where individuals become more suited to the environment as they acquire more knowledge of their surroundings.

In this paper, we present a recurrent wavelet-based neurofuzzy system (RWNFS) with the reinforcement hybrid evolutionary learning algorithm (R-HELA). The proposed R-HELA automatically determines the number of fuzzy rules and processes the variable-length chromosomes. The length of each individual denotes the total number of genes in that individual. The initial length of one individual may be different from another individual, depending on the total number of rules encoded in it. Individuals with an equal number of rules constitute the same group. Thus, initially there are several groups in a population. We use the elite-based reproduction strategy to keep the best group. Therefore, the best group can be reproduced many times for each generation. The reinforcement signal from the environment is used as a fitness function for the R-HELA. That is, we formulate the number of time steps before failure occurs as the fitness function. In this way, the R-HELA can evaluate the candidate solutions for the parameters of the RWNFS model.

The advantages of the proposed R-HELA method are summarized as follows.

1) It determines the number of fuzzy rules and tunes the free parameters of the RWNFS model in a highly autonomous way. Thus, users need not give it any a priori knowledge or even any initial information on these parameters.
2) It is applicable to chromosomes of different lengths.
3) It does not require precise training data for setting the parameters of the RWNFS model.
4) It performs better and converges more quickly than some traditional genetic methods.

This paper is organized as follows. Section II introduces the RWNFS. The proposed HELA is described in Section III. Section IV introduces the reinforcement hybrid evolution learning algorithms used for constructing the RWNFS model. Section V presents the simulation results. The conclusions are given in the last section.

II. Structure of a Recurrent Wavelet-Based Neurofuzzy System

This section introduces the structure of an RWNFS model. For traditional TSK-type fuzzy systems [5]–[7], the consequence of each rule is a function of the input linguistic variable.

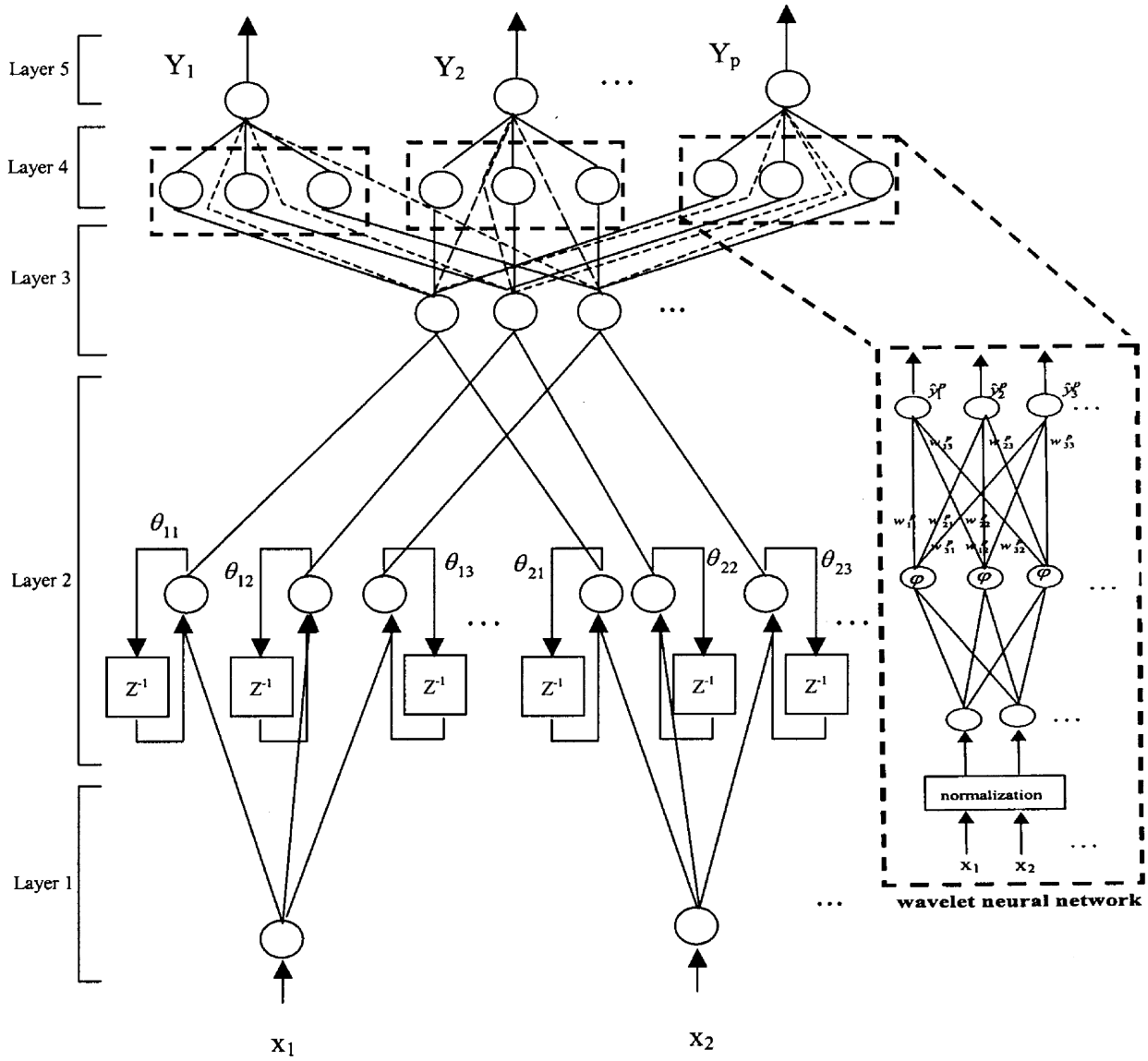


Fig. 3. Schematic diagram of RWNFS model.

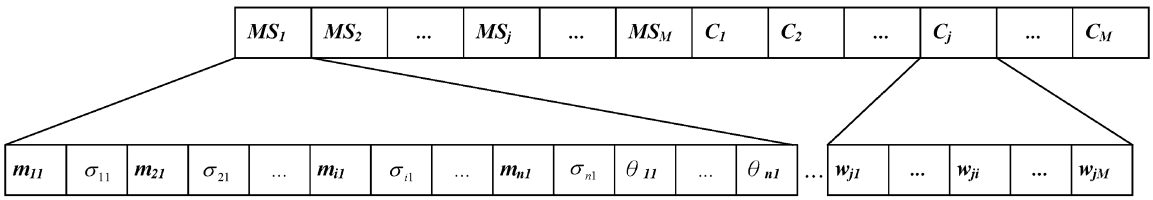


Fig. 4. Coding the adjustable parameters of a RWNFS into a chromosome in the MVGA.

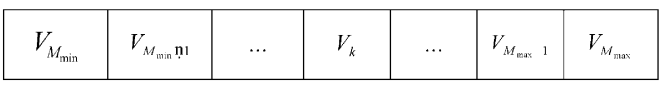


Fig. 5. Coding the probability vector into the building blocks in the CGA.

A widely adopted function is a linear combination of input variables plus a constant term. This paper adopts a nonlinear combination of input variables [i.e., wavelet neural network (WNN)]. Each fuzzy rule corresponds to a sub-WNN consisting of single-scaling wavelets [33]. We adopt the nonorthogonal and compact wavelet functions as the node function (wavelet bases).

A. Description of Wavelet Bases and Wavelet Neural Networks

A set of wavelet bases is a suitable tool for representing nonlinearity effectively. These orthogonal wavelets are infinite, continuous, and differentiable. The support of these wavelets is $-\infty < x < +\infty$. Daubechies [34] presented wavelet bases, which are compactly supported but not infinitely supported.

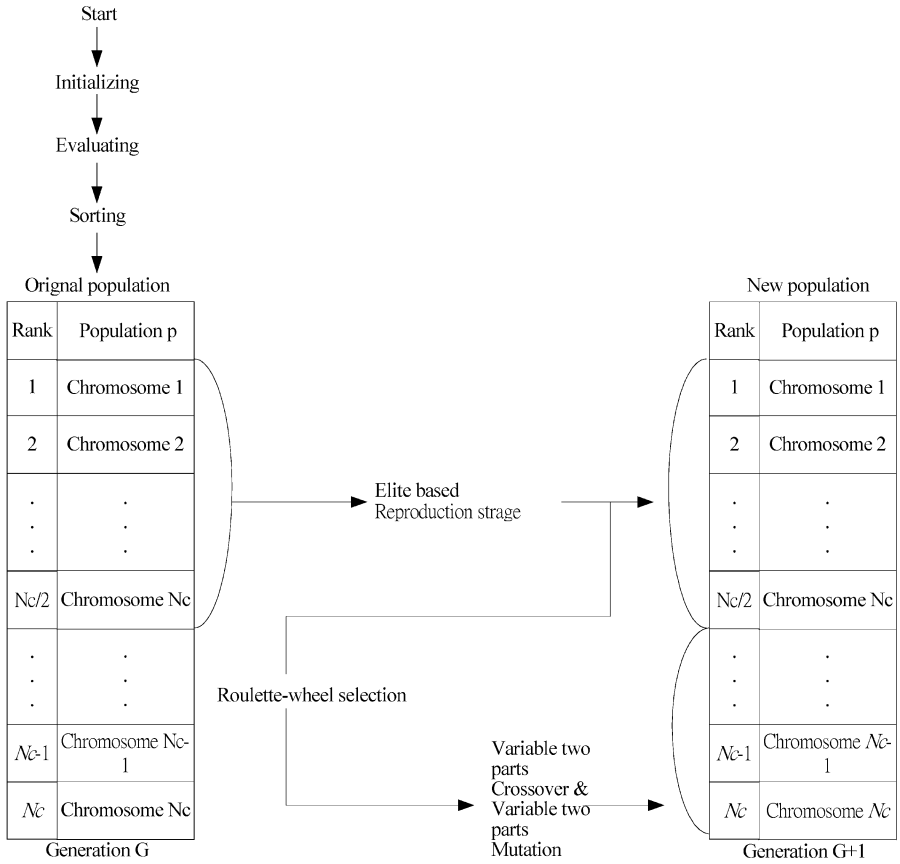


Fig. 6. The flowchart of the parameter learning in the HELA.

Daubechies proposed using a simple wavelet neural network, which exhibits a much higher ability to generalize and a much shorter learning time, rather than a three-layered feedforward neural network. This study adopts the nonorthogonal and compactly supported functions in the finite range as wavelet bases. Fig. 1 shows the shape and position of the wavelet bases. All the wavelet bases are allocated over the normalized range [0, 1] in the variable space.

Neural networks employing wavelet neurons are referred to as *wavelet neural networks*. Fig. 2 shows a novel type of wavelet neural network model [35]. Consider n input vectors $\{x_1,\ldots,x_i,\ldots x_n\} \in \Re^n$ and p output vectors $\{Y_1,\ldots,Y_s,\ldots,Y_p\} \in \Re^p$. This model is obtained by replacing a sigmoidal activation function with single-scaling wavelets [35]. The wavelet neural networks are characterized by weights and wavelet bases. Each linear synaptic weight of the wavelet bases is adjustable by learning. Notably, the ordinary wavelet neural network model applications are often useful for normalizing the input vectors into the interval [0, 1]. The $\phi_{a.b}(x_i)$ functions that are used to input vectors to fire up the wavelet interval are then calculated. The value $\phi_{a.b}$ is obtained as shown in the equation at the bottom of the page,

$$\begin{cases} \phi(x_i) = \cos(x_i) & -0.5 \le x_i \le 0.5 \\ 0 & (\text{otherwise}) \end{cases}, \quad \phi_{a.b}(x_i) = \cos(ax_i - b)$$

where $a = 1,\ldots,m, b = 1,\ldots,a$

$$M = \sum_{a=1}^{m}\sum_{b=1}^{a} 1. \tag{1}$$

The above equation formulates the nonorthogonal wavelets in a finite range, where b denotes a shifting parameter, the maximum value of which equals the corresponding scaling parameter a; and M denotes the number of wavelet bases, which equals the number of existing fuzzy rules in the RWNFS model. In the RWNFS model, wavelet bases do not exist in the initial state. The amount of wavelet bases generated by the online learning algorithm is consistent between wavelet bases and fuzzy rules. The online learning algorithm is detailed in Section III. A crisp value $\phi_{a.b}$ can be obtained as follows:

$$\varphi_{a.b} = \frac{\sum_{i=1}^{n} \phi_{a.b}(x_i)}{X} \tag{2}$$

where X is the number of input dimensions. The final output of the wavelet neural networks is

$$\hat{y}_j^s = \sum_{k=1}^{M} w_{jk}^s \varphi_{a.b} \tag{3}$$

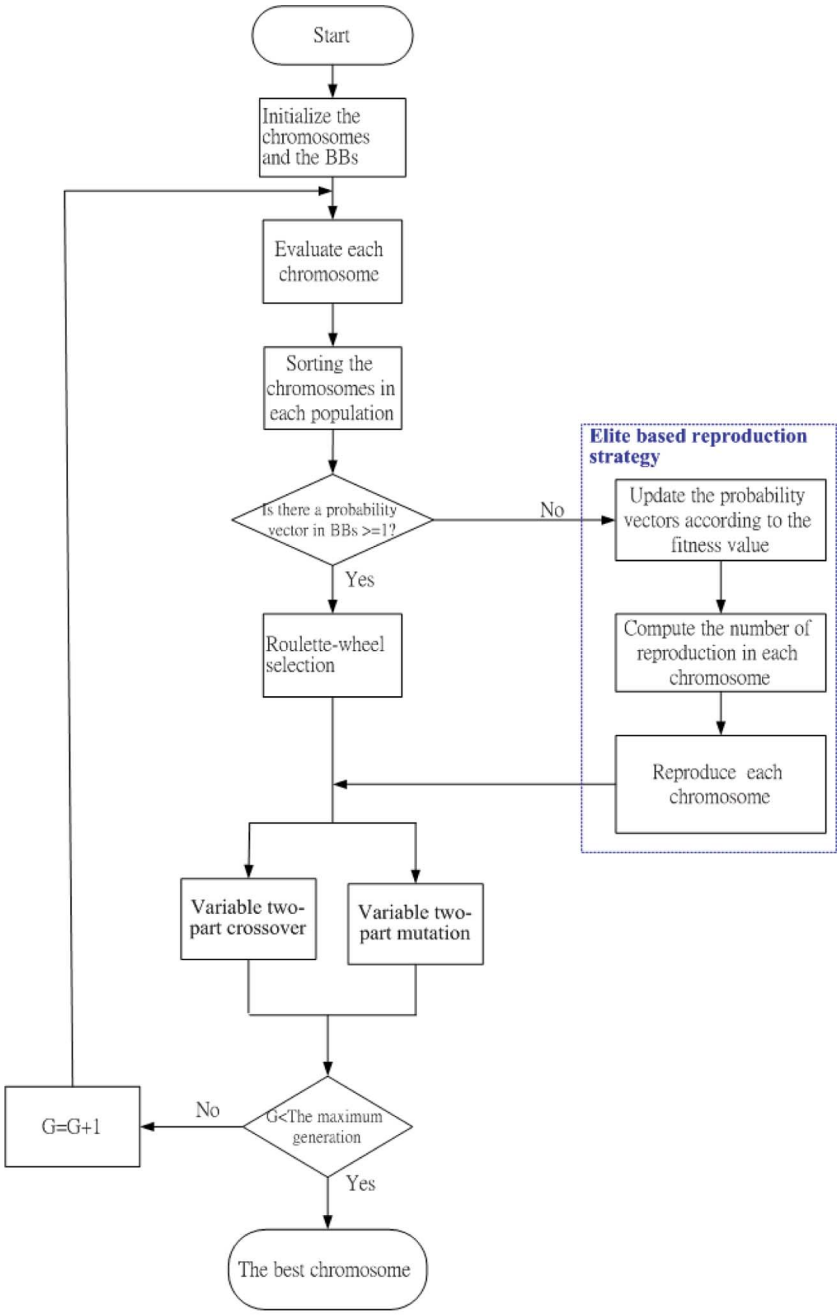


Fig. 7. The learning diagram of the proposed ERS method.

where $\hat{y}_j^s$ denotes the local output of the WNN for output Y_s and the jth rule; and the link weight w_{jk}^s is the output action strength associated with the sth output, jth rule, and kth $\phi_{a.b}$.

B. Structure of the RWNFS Model

This section introduces the structure of the RWNFS model shown in Fig. 3. For TSK-type fuzzy networks [5]–[7], the consequence of each rule is a linear function of input linguistic variables. 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., WNN). Each fuzzy rule corresponds to a sub-WNN consisting of single-scaling wavelets [33]. A novel RWNFS model is composed of fuzzy rules that can be presented in the following general form:

$$\begin{aligned} R^j&: \text{If } I_{1j} \text{ is } A_{1j} \text{ and } \ldots\ I_{ij} \text{ is } A_{ij} \text{ and } \ldots \text{ and } I_{nj} \text{ is } A_{nj} \\ &\text{Then } \hat{y}_j^1 = \sum_{k=1}^{M} w_{jk}^1 \varphi_{a.b} = w_{j1}^1 \varphi_{0.0} + w_{j2}^1 \varphi_{1.0} \\ &\quad + w_{j3}^1 \varphi_{1.1} \ldots \\ &\quad \text{and } \hat{y}_j^2 = \sum_{k=1}^{M} w_{jk}^2 \varphi_{a.b} = w_{j1}^2 \varphi_{0.0} + w_{j2}^2 \varphi_{1.0} \\ &\quad + w_{j3}^2 \varphi_{1.1} \ldots \\ &\quad\ \vdots \end{aligned} \tag{4}$$

Procedure of elite-based reproduction strategy

```
Begin
    Let k= R_min,Temp=0;
    Repeat
        Update V_k by(10)to(13);
        If V_k≧1 then
            Temp=k1; Break;
        End if
        k=k+1;
    Until k= R_max;
    If  Temp!=0 then
        Rep_Temp=Psize/2;
        Exit Procedure;
    else
        k= R_min;
        Repeat
            Compute Rep_k by (14 ) to (15)
            k=k+1;
        Until k= R_max;
    End if
End
```

Fig. 8. The pseudocode of the ERS.

where R^j denotes the jth 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 $A^j = \{A_{1j},\ldots,A_{ij},\ldots,A_{nj}\}$; and the local WNN model outputs $\hat{y}_j^1$ and $\hat{y}_j^2$ are calculated for outputs Y_1 and Y_2 and 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 ith input of a node in the hth layer and $O_i^{(h)}$ denotes the ith node output in layer h.

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

$$O_i^{(1)} = I_i^{(1)} \tag{5}$$

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

$$O_{ij}^{(2)} = \exp\left(-\frac{(I_{ij}^{(2)} - m_{ij})^2}{\sigma_{ij}^2}\right) \tag{6}$$

where m_{ij} and σ_{ij} are the mean and standard deviation, respectively. Additionally, the input of this layer for the discrete time s can be denoted by

$$I_{ij}^{(2)}(t) = O_i^{(1)}(t) + O_{ij}^{(f)}(t), \quad O_{ij}^{(f)}(t) = O_{ij}^{(2)}(t-1)\cdot\theta_{ij} \tag{7}$$

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. This is the apparent difference between the WFNN [11] and RWNFS models.

In the proposed RWNFS model, 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 [36], we need to know the order of both the control input and the network output to participate in the autoregressive with exogenous model. We solve this problem by feeding back the output of each membership function. Only the current control input and system state are fed to the network input. The past values can be memorized by using feedback structure. In [37], a global feedback structure is adopted, and the outputs of all rule nodes, the firing strengths, are fed back and summed. In this case, the TRFN model [37] needs more adjustable parameters. However, we will show by simulation that the proposed RWNFS model achieves better performance and requires a smaller number of tuning parameters than the model in [37].

In layer 3, defining the number and the locations of the membership functions leads to the partition of the premise 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 multidimensional fuzzy set whose membership function is determined by

$$O_j^{(3)} = \prod_{i=1}^{n} I_j^{(3)} = \prod_{i=1}^{n} \exp\left(-\frac{\left(I_{ij}^{(2)} - m_{ij}\right)^2}{\sigma_{ij}^2}\right) \tag{8}$$

where n is the number of external dimensions.

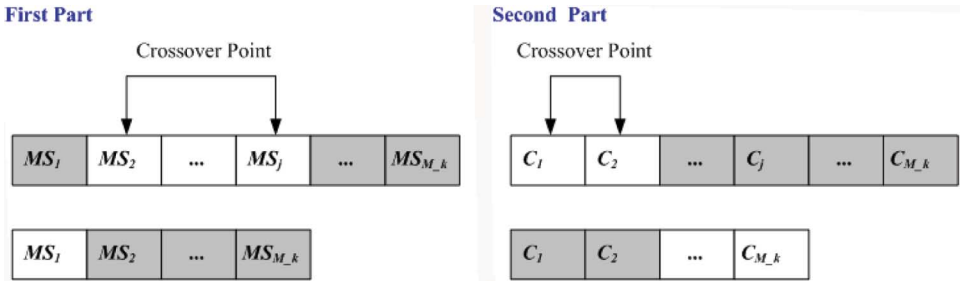


Fig. 9. The variable two-part crossover operation in the HELA.

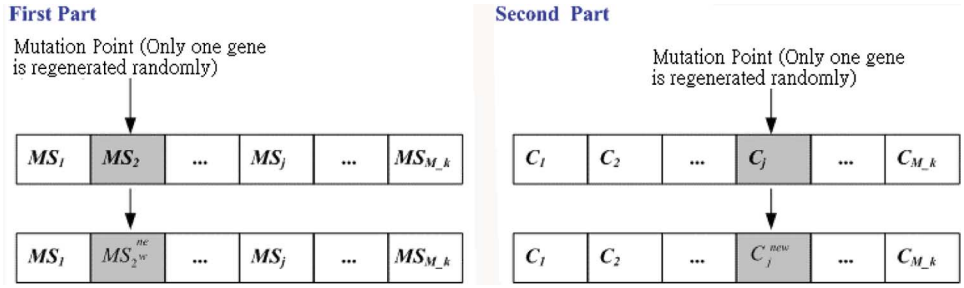


Fig. 10. The variable two-part mutation operation in the HELA.

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

$$\hat{y}_j^s = O_{sj}^{(4)} = \sum_{\mathrm{k}=1}^{M} \mathrm{w}_{\mathrm{jk}}^{\mathrm{s}} \varphi_{a,b}. \tag{9}$$

The final output of the model $\{Y_1, \ldots, Y_s, \ldots, Y_p\}$ is calculated in layer 5. The output node together with related links acts as a defuzzifier. The mathematical function is shown in

$$\begin{aligned} Y_s = O_s^{(5)} &= \frac{\sum_{j=1}^{M} I_{sj}^{(5)} I_j^{(5)}}{\sum_{j=1}^{M} I_j^{(5)}} \\ &= \frac{\sum_{j=1}^{M} \left(w_{j1}^s \phi_{0.0} + \cdots + w_{jk}^s \phi_{a.b} \cdots + w_{jM}^s \phi_{m.m} \right) \cdot I_j^{(5)}}{\sum_{j=1}^{M} I_j^{(5)}} \end{aligned} \tag{10}$$

where $I_{sj}^{(5)} = O_{sj}^{(4)}$ denotes the output of the local model of the WNN model 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 the RWNFS model.

III. A Hybrid Evolutionary Learning Algorithm (HELA)

This section introduces the proposed HELA. Recently, many efforts to enhance the traditional GAs have been made [38]. Among them, one category focuses on modifying the structure of a population or the role an individual plays in it [39]–[41], such as the distributed GA [39], the cellular GA [40], and the symbiotic GA [41].

In a traditional evolution algorithm, the number of rules in a model must be predefined. Our proposed HELA combines the compact genetic algorithm (CGA) and the modified variable-length genetic algorithm (MVGA). In the MVGA, the initial length of each individual may be different from each other, depending on the total number of rules encoded in it. Thus, we do not need to predefine the number of rules. In this paper, individuals with an equal number of rules constitute the same group. Initially, there are several groups in a population. Not following the traditional VGA notation, Bandyopadhyay *et al.* [29] used "#" to mean "does not care." In this paper, we adopt the variable two-part crossover (VTC) and the variable two-part mutation (VTM) to make the traditional crossover and mutation operators applicable to different lengths of chromosomes. Therefore, we do not use "#" to mean "does not care" in the VTC and the VTM.

In this paper, we divide a chromosome into two parts. The first part of the chromosome gives the antecedent parameters of a RWNFS model, while the second part of the chromosome gives the consequent parameters of a RWNFS model. Each part of a chromosome can be performed using the VTC on the overlapping genes of two chromosomes. In the traditional VGA, Bandyopadhyay *et al.* [29] only evaluated the performance of each chromosome in a population. The performance of the number of rules was not evaluated in [29].

In this paper, we use the elite-based reproduction strategy to keep the best group that has chromosomes of the same length. Therefore, the best group can be reproduced many times for each generation. The elite-based reproduction strategy is similar to the maturing phenomenon in society, in which individuals become more suited to the environment as they acquire more knowledge of their surroundings.

In the proposed HELA method, we adopt the CGA [42] to carry out the elite-based reproduction strategy. The CGA represents a population as a probability distribution over the set of solutions and is operationally equivalent to the order-one

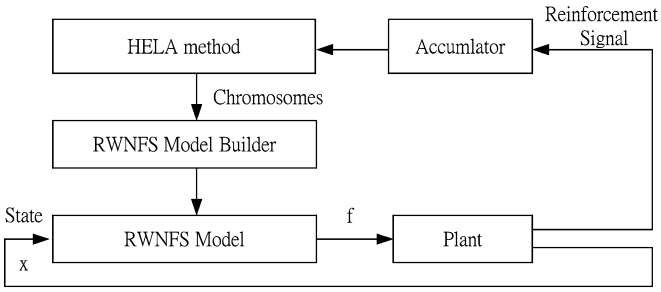


Fig. 11. Schematic diagram of the R-HELA for the RWNFS model.

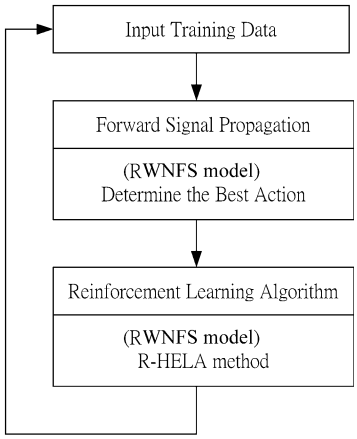


Fig. 12. Flowchart of the R-HELA method.

behavior of the simple GA [43]. The advantage of the CGA is that it processes each gene independently and requires less memory than the normal GA. The building blocks (BBs) in the CGA represent the suitable lengths of the chromosomes, and the CGA reproduces the chromosomes according to the BBs.

The coding scheme consists of the coding done by the MVGA and the CGA. The MVGA codes the adjustable parameters of a RWNFS model into a chromosome, as shown in Fig. 4, where MS_j represents the parameters of the antecedent of the jth rule in the RENFN and C_j represents the parameters of the consequent of the jth rule. In Fig. 5, the CGA codes the probability vector into the BBs, where each probability vector represents the suitability of the rules of a RWNFS model. In the CGA, we must predefine the maximum number of rules $(\mathrm{M}_{\max})$ and the minimum number of rules $(\mathrm{M}_{\min})$ to prevent generating the number of fuzzy rules beyond a certain bound (i.e., $[\mathrm{M}_{\max}, \mathrm{M}_{\min}]$).

The learning process of the HELA involves six major operators: initializing, evaluating, sorting, elite-based reproduction strategy, variable two-part crossover, and variable two-part mutation. Fig. 6 shows the flowchart of the learning process. The whole learning process is described step-by-step as follows.

a) *Initializing:* The initializing step sets the initial values in the MVGA and the CGA. In the MVGA, individuals are initially randomly generated to construct a population. In order to keep the same number of rules in an RWNFS model, the number of rules for each chromosome needs to be generated η chromosomes. That is, we predefine the number of chromosomes generated for each group (η). Therefore, the population size is set to $\eta*(M_{\max}-M_{\min}+1)$. In the CGA, the probability vectors of the BBs are set to 0.5 initially.

b) *Evaluating:* The evaluating step evaluates each chromosome in a population. The goal of the R-HELA method is to maximize the fitness value. The higher a fitness value, the better the fitness. The fitness function is used by a reinforcement signal in (16) that we will introduce in next section.

c) *Sorting:* After the evaluating step, we sort the chromosomes in the population. After the whole population is sorted, we sort the chromosomes in each group in the top half of population. The sorting step can help us to perform the reproduction step because we can keep the best chromosome in each group. After sorting the chromosomes in the population, the algorithm goes to next step.

d) *Elite-Based Reproduction Strategy (ERS):* Reproduction is a process in which individual strings are copied according to their fitness value. A fitness value is assigned to each individual using (16). The goal of the R-HELA method is to maximize the fitness value. The higher a fitness value, the better the fitness. In this paper, we use an ERS to mimic the maturation phenomenon in society, in

TABLE I
THE INITIAL PARAMETERS BEFORE TRAINING

Parameters	*Value*
Population Size	***54***
Crossover Rate	***0.5***
Mutation Rate	***0.3***
$[\sigma_{\min}, \sigma_{\max}]$	***[0,2]***
$[m_{\min}, m_{\max}]$	***[0,2]***
$[w_{\min}, w_{\max}]$	***[-20,20]***
$M_{\max}$	***12***
$M_{\min}$	***3***
λ	***0.01***
η	***6***

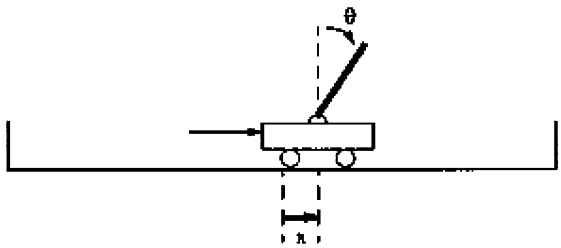
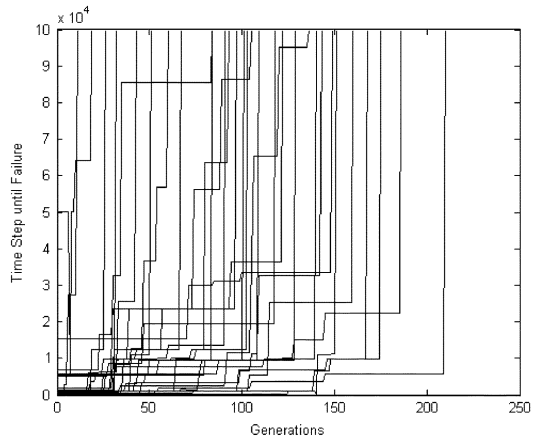
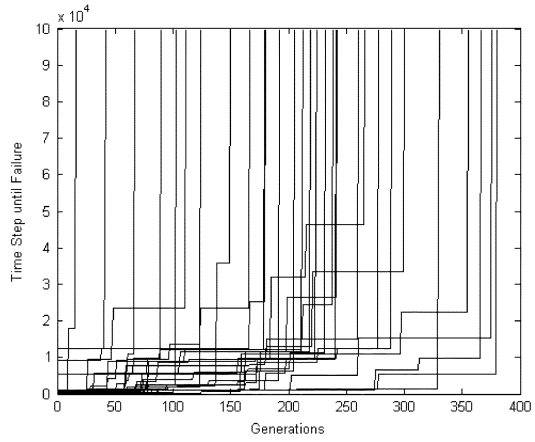


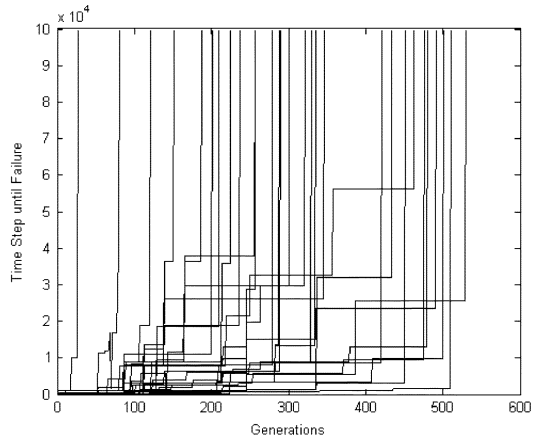
Fig. 13. The cart-pole balancing system.



(a)



(b)



(c)

Fig. 14. The performance of (a) the R-HELA method, (b) the R-SE method [28], and (c) the R-GA method [22] on the cart-pole balancing system.

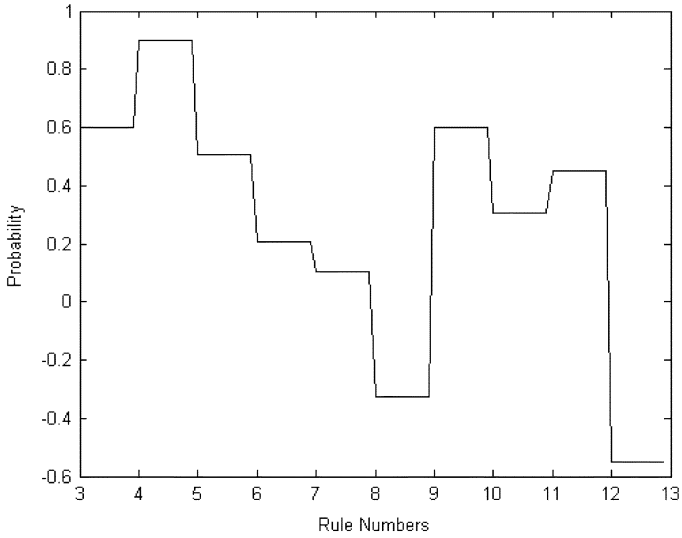


Fig. 15. The probability vectors of the ERS step in the proposed R-HELA.

which individuals become more suited to the environment as they acquire more knowledge of their surroundings. The CGA is used here to carry out the ERS. The CGA represents the population as a probability distribution over the set of solutions and is operationally equivalent to the order-one behavior of the simple GA. The CGA uses the BBs to represent the suitable lengths of the chromosomes and reproduces the chromosomes according to the probability vector in the BBs. The best performing individuals in the top half of each population are used to perform the ERS. According to the results of the ERS, using the crossover and the mutation operations generates the other half of the individuals. The learning diagram of the proposed ERS method is shown in Fig. 7. After the ERS, the suitable length of chromosomes will be preserved and the unsuitable length of chromosomes will be removed. Details of the ERS are shown below.

Step 1) Update the probability vectors of the BBs according to the following equations:

$$\begin{cases} V_k = V_k + (\text{Upt\_value}_k * \lambda), & \text{if Avg} \le \text{Max\_fit}_k \\ V_k = V_k - (\text{Upt\_value}_k * \lambda), & \text{otherwise} \end{cases} \tag{10}$$

$$\text{where } k = [R_{\max}, R_{\min}]$$

$$\text{Avg} = \sum_{p=1}^{Nc} \text{fit}_p / Nc \tag{11}$$

$$\text{Upt\_value}_k = \text{Total\_fit}_k \Big/ \sum_{p=1}^{Nc} \text{fit}_p \tag{12}$$

$$\text{Total\_fit}_k = \sum_{p=1}^{N_k} \text{fit}_p \tag{13}$$

where V_k is the probability vector in the BBs and represents the suitable chromosome in the group with k rules in a population; λ is a threshold value we predefine; Avg represents the average fitness value in the whole population; *Nc* is the population size; N_k is the kth group size; fit_p is the fitness value of the pth chromosome in all *Nc* populations; fit_{kp} is the fitness value of the pth chromosome in the kth group; and Max_fit_k is the best fitness value [maximum value of (16)] in the kth group. As shown in (10), if $\text{Max\_fit}_k \ge \text{Avg}$, then the suitable chromosomes in the kth group should be increased. On the other hand,

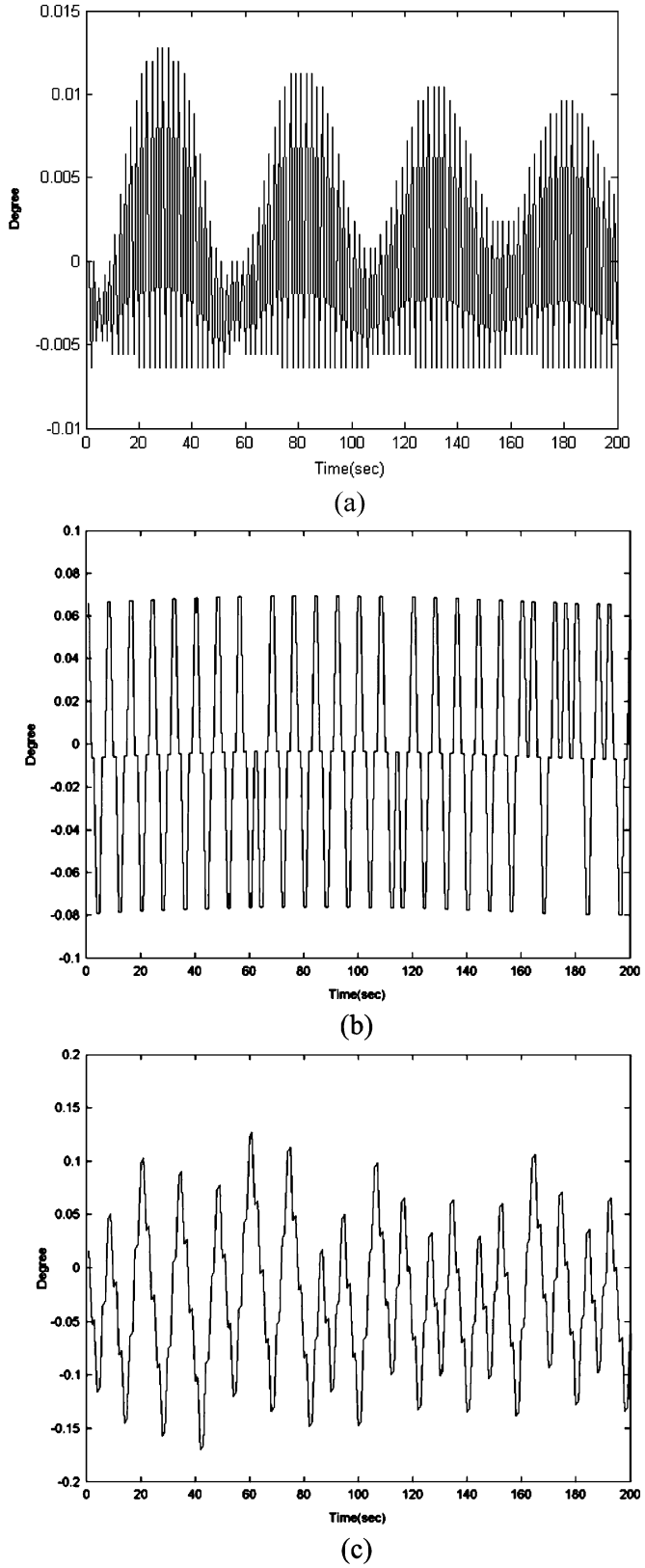


Fig. 16. Angular deviation of the pole by a trained (a) R-HELA method, (b) R-SE method [28], and (c) R-GA method [22].

if $\mathrm{Max\_fit}_k < \mathrm{Avg}$, then the suitable chromosomes in the kth group should be decreased. Equation (13) represents the sum of the fitness values of the chromosomes in the kth group.

Step 2) Determine the reproduction number according to the probability vectors of the BBs as follows:

$$\mathrm{Rep}_k = (\mathrm{P_{size}}/2) * (V_k/\mathrm{Total\_Velocy}) \quad \text{where } k = [R_{\max}, R_{\min}] \tag{14}$$

$$\mathrm{Total\_Velocy} = \sum_{k=R_{\min}}^{R_{\max}} V_k \tag{15}$$

where $\mathrm{P_{size}}$ represents the population size, Rep_k is the recorder, and a chromosome has k rules for constructing an RWNFS.

Step 3) After Step 2), the reproduction number of each group in the top half of a population is obtained. Then we generate Rep_k chromosomes in each group using the roulette-wheel selection method [44].

Step 4) If any probability vector in BBs reaches 1, then stop the ERS and set the probability vector to 1 for all groups with the same number of rules, according to Step 2). The lacks of the chromosomes are generated randomly. To replace the ERS step, we use the roulette-wheel selection method [44]—a simulated roulette is spun—for this reproduction process. The pseudocode for the ERS is shown in Fig. 8.

e) *Variable Two-Part Crossover:* 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 with a crossover rate.

In this paper, we propose using the VTC to perform the crossover operation. In the VTC, the parents are selected from the enhanced elites using the roulette-wheel selection method [44]. The two parents may be selected from the same or different groups. Performing crossover on the selected parents creates the offspring. Since the parents may be of different lengths, we must avoid misalignment of individuals in the crossover operation. Therefore, a variable two-part crossover is proposed to solve this problem. The first part of the chromosome gives the antecedent parameters of an RWNFS model while the second part of the chromosome gives the consequent parameters of an RWNFS model. The two-point crossover is adopted in each part of the chromosome. Thus, new individuals are created by exchanging the site's values between the selected sites of the parents' individuals. To avoid the misalignment of individuals in the crossover operation, in the VTC, the selection of the crossover points in each part will not exceed the shortest length chromosome of two parents. Two individuals of different lengths resulting from the use of the variable two-part crossover operation are shown in Fig. 9. MS_j represents the parameters of the antecedent part of the jth rule in the RENFN; W_j represents the parameters of the consequent of the jth rule in the RENFN; and M_k is the number of fuzzy rules

in the kth chromosome. After the VTC operation, the individuals with poor performances are replaced by the new offspring.

f) *Variable Two-Part Mutation:* Although the ERS and the VTC produce many new strings, these strings do not provide any new information to every population at the site of an individual. Mutation can randomly alter the allele of a gene. In this paper, we propose using the VTM to perform the mutation operation. The proposed VTM is different from the traditional mutation and is applicable to chromosomes of different lengths. The first and second parts of the chromosome are the same as the crossover operation. In each part of a chromosome, uniform mutation is adopted, and the mutated gene is drawn randomly from the domain of the corresponding variable. The VTM operation for each individual is shown in Fig. 10.

After the above-mentioned operations are carried out, the problem of how groups are to be constituted by the most suitable number of rules will be solved. The number of elites in other groups will decrease, and most of them will become zero (in most cases, there will be no elites). That is, our method indeed can eliminate unsuitable groups and rules.

IV. Reinforcement Learning for an RWNFS Model

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, available for learning. In the extreme case, there is only a single bit of information to indicate whether the output is right or wrong. Fig. 11 shows the R-HELA. Its training environment interacts with reinforcement learning problems. In this paper, the reinforcement signal indicates whether a success or a failure occurs.

As shown in Fig. 11, the proposed R-HELA consists of a RWNFS model, which acts as the control network that determines the proper action to take according to the current input vector (environment state). The structure of the proposed R-HELA is different from the actor-critic architecture of Barto *et al.* [15], which consists of a control network and a critic network. The input to the RWNFS model is the state of a 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 model only when a failure occurs.

An accumulator plays a role which is a relative performance measure, as shown in Fig. 11. It accumulates the number of time steps before a failure occurs. In this paper, the feedback takes the form of an accumulator that determines how long the experiment is still a "success"; this is used as a relative measure of the fitness of the proposed R-HELA method. That is, the accumulator will indicate the "fitness" of the current RWNFS model. The key to the R-HELA is formulating a number of time steps before failure occurs and using this formulation as the fitness function for the R-HELA method. It will be observed that the advantage of the proposed R-HELA method is that it can meet global optimization capability.

TABLE II
Performance Comparison of Various Existing Models in Example 1

Method	Mean	Best	Worst
GENITOR [46]	3268	415	18743
SANE [41]	1984	46	5865
R-GA [22]	324	26	550
R-SE [28]	214	15	380
TDGAR [27]	186	18	310
CQGAF [32]	133	12	288
R-HELA	**104**	**10**	**210**

Fig. 12 shows the flowchart of the R-HELA method. The proposed R-HELA method runs in a feed-forward fashion to control the environment (plant) until a failure occurs. Our relative measure of the fitness function takes the form of an accumulator that determines how long the experiment is a "success." In this way, according to a defined fitness function, a fitness value is assigned to each string in the population where a high fitness value means a good fit. In this paper, we use a number of time steps before failure occurs to define the fitness function. The goal of the R-HELA method is to maximize the fitness value. The fitness function is defined by

$$\text{Fitness Value}(i) = \text{TIME-STEP}(i) \tag{16}$$

where TIME-STEP(i) represents how long the experiment is a "success" with the ith population. Equation (16) reflects the fact that long-time steps before failure occurs (to keep the desired control goal longer) means a higher fitness of the R-HELA method.

V. Illustrative Examples

In this section, we compare the performance of the RWNFS model using the R-HELA method with some existing models for two applications. The first simulation was performed to balance the cart-pole system that was described in [45]. The second simulation was performed to balance the ball and beam system that was described in [47]. The initial parameters for the two simulations are given in Table I. The initial parameters were determined by practical experimentation or trial-and-error tests.

Example 1: Control of a Cart-Pole Balancing System: In this example, we apply the R-HELA method to the classic control problem of a cart-pole balancing. This problem is often used as an example of inherently unstable and dynamic systems to demonstrate both modern and the classic control techniques [45]–[47], or the reinforcement learning schemes [27], [28], [48], and is now used as a control benchmark. As shown in Fig. 13, the 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

TABLE III
THE COMPARISON OF CPU TIME FOR VARIOUS EXISTING MODELS IN EXAMPLE 1

Method	Mean	Best	Worst
GENITOR [46]	70.95	33.34	246.36
SANE [41]	43.56	16.54	156.84
R-GA [22]	47.59	11.85	102.63
R-SE [28]	38.85	8.53	90.78
TDGAR [27]	30.23	7.97	76.25
CQGAF [32]	28.15	6.39	61.37
R-HELA	**23.12**	**5.42**	**50.71**

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 the cart to move it toward left or right. The system fails when the pole falls past a certain angle (± 12 is used here) or 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 that we used are

$$\theta(t+1) = \theta(t) + \Delta\dot{\theta}(t) \tag{17}$$

$$\dot{\theta}(t+1) = \dot{\theta}(t) + \Delta\frac{(m+m_p)g\sin\theta(t)}{(4/3)(m+m_p)l - m_p l\cos^2\theta(t)} - \frac{\cos\theta(t)[f(t)+m_p l\dot{\theta}(t)^2\sin\theta(t) - \mu_c \mathrm{sgn}(\dot{x}(t))]}{(4/3)(m+m_p)l - m_p l\cos^2\theta(t)} - \frac{\frac{\mu_p(m+m_p)\dot{\theta}(t)}{m_p l}}{(4/3)(m+m_p)l - m_p l\cos^2\theta(t)} \tag{18}$$

$$x(t+1) = x(t) + \Delta\dot{x}(t) \tag{19}$$

$$x(t+1) = \dot{x}(t) + \Delta\frac{f(t) + m_p l\left[\dot{\theta}(t)^2\sin\theta(t) - \ddot{\theta}(t)\cos\theta(t)\right]}{(m+m_p)} - \frac{\mu_c sgn(\dot{x}(t))}{(m+m_p)} \tag{20}$$

where

$$\begin{aligned} l &= 0.5\ \mathrm{m}, \text{ the length of the pole;} \\ m &= 1.1\ \mathrm{kg}, \text{ combined mass of the pole and the cart;} \\ m_p &= 0.1\ \mathrm{kg}, \text{ mass of the pole;} \\ g &= 9.8\ \mathrm{m/s}, \text{ acceleration due to the gravity} \end{aligned} \tag{21}$$

TABLE IV
PERFORMANCE COMPARISON OF THREE DIFFERENT METHODS

Method	Mean	Best	Worst
Type I method (the proposed R-HELA method)	**104**	**10**	**210**
Type II method (the proposed R-HELA method without ERS)	115	15	229
Type III method (the fixed length genetic algorithm)	210	21	380

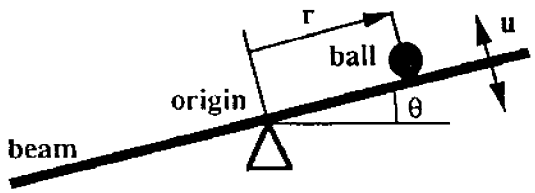


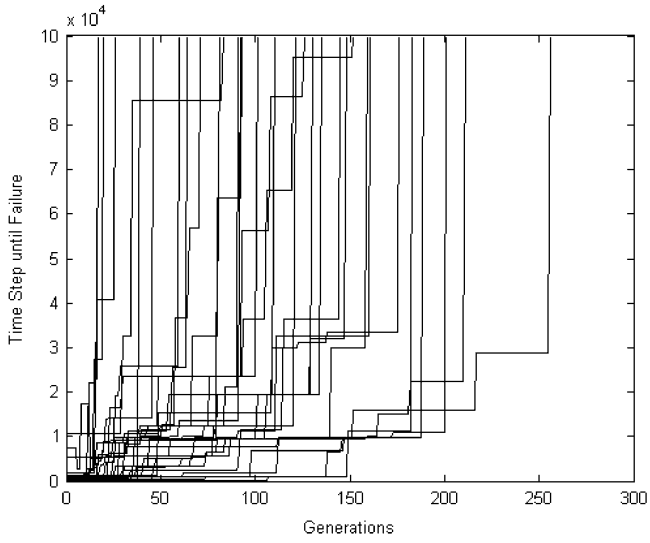
Fig. 17. The ball and beam system.

where $\mu_c = 0.0005$ is the coefficient of friction of the cart on the track,

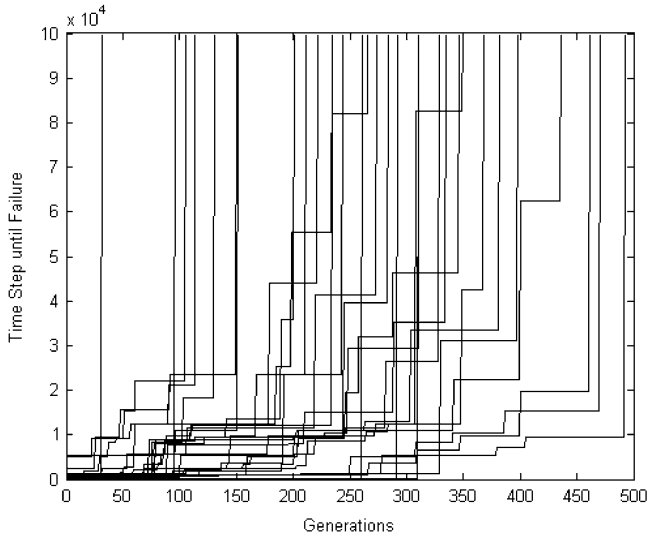
$\mu_p = 0.000002$ is the coefficient of friction of the pole on the cart, and $\Delta = 0.02$ (s) is the sampling interval.

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

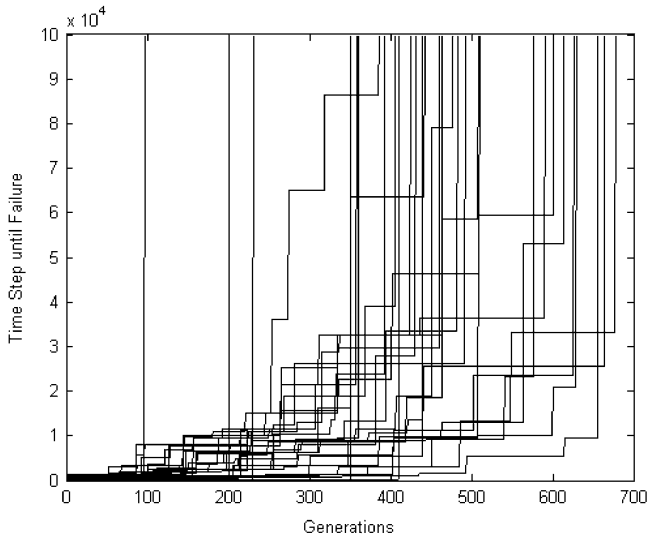
The four input variables $(\theta, \dot{\theta}, x, \dot{x})$ and the output f_t are normalized between zero and one over the following ranges: θ:$[-12, 12]$, $\dot{\theta}$: $[-60, 60]$, x:$[-2.4, 2.4]$, $\dot{x}$: $[-3, 3]$, f_t : $[-10, 10]$. The four normalized state variables are used as inputs to the proposed RWNFS model. The coding of a rule in a chromosome is the form in Fig. 4. The values are floating-point numbers assigned using the R-HELA initially. The fitness function in this example is defined in (16) to train the RWNFS model, where (16) represents how long the cart-pole balancing system fails and receives a penalty signal of -1 when the beam deviates beyond a certain angle ($|\theta| > 12\ ^\circ$C) and the cart runs into the bounds of its track ($|x| > 2.4$ m). In this experiment,



(a)



(b)



(c)

Fig. 18. The performance of (a) the R-HELA method, (b) the R-SE method [28], and (c) the R-GA method [22] on the ball and beam balancing system.

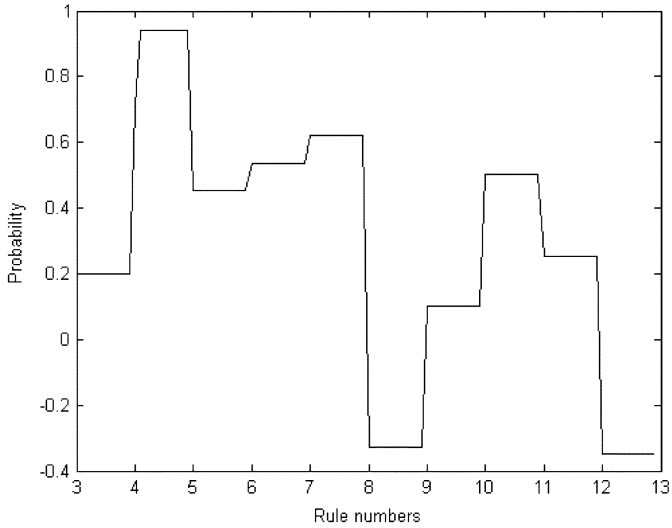


Fig. 19. The probability vectors of the ERS step in the proposed HELA.

the initial values were set to (0, 0, 0, 0). A total of 30 runs were performed. Each run started in the same initial state. Fig. 14(a) shows that the RWNFS model learned on average to balance the pole at the fifty-fourth generation. In this figure, each run represents that largest fitness value in the current generation being selected before the cart-pole balancing system fails. When the R-HELA method was stopped, we chose the best strings in the population in the final generation and tested them on the cart-pole balancing system. Fig. 15 shows the results of the probability vectors in CGA. In this figure, the final average optima number of rules is four. The obtained fuzzy rules of the RWNFS using the R-HELA method are shown as follows:

R^1: If I_{11} is $A_{1,1}(0.019, 0.44)$ and I_{12} is $A_{2,1}(0.44, 1.36)$ and I_{13} is $A_{3,1}(0.97, 0.56)$ and I_{14} is $A_{4,1}(0.29, 0.46)$
Then $\hat{y}_1^1 = 1.067\varphi_{0.0} + 1.066\varphi_{1.0} - 1.092\varphi_{1.1} - 0.223\varphi_{2.0}$

R^2: If I_{21} is $A_{1,2}(0.96, 0.84)$ and I_{22} is $A_{2,2}(0.94, 0.21)$ and I_{23} is $A_{3,2}(0.28, 0.18)$ and I_{24} is $A_{4,2}(0.43, 1.043)$
Then $\hat{y}_2^1 = -0.27\varphi_{0.0} + 0.31\varphi_{1.0} - 0.024\varphi_{1.1} + 0.772\varphi_{2.0}$

R^3: If I_{31} is $A_{1,3}(1.00, 0.20)$ and I_{32} is $A_{2,3}(0.66, 0.18)$ and I_{33} is $A_{3,3}(0.97, 0.56)$ and I_{34} is $A_{4,3}(0.29, 0.46)$
Then $\hat{y}_2^1 = 0.049\varphi_{0.0} + 0.10\varphi_{1.0} - 0.24\varphi_{1.1} - 0.22\varphi_{2.0}$

R^4: If I_{41} is $A_{1,4}(0.56, 0.63)$ and I_{42} is $A_{2,4}(0.05, 0.52)$ and I_{43} is $A_{3,4}(0.52, 0.49)$ and I_{44} is $A_{4,4}(0.89, 0.50)$
Then $\hat{y}_4^1 = -0.001\varphi_{0.0} - 0.053\varphi_{1.0} - 0.24\varphi_{1.1} - 0.22\varphi_{2.0}$.

Fig. 16(a) shows the angular deviation of the pole when the cart-pole balancing system was controlled by a well-trained RWNFS model starting in the initial state: $x(0) = 0, \dot{x}(0) = 0, \theta(0) = 0, \dot{\theta}(0) = 0$. The average angular

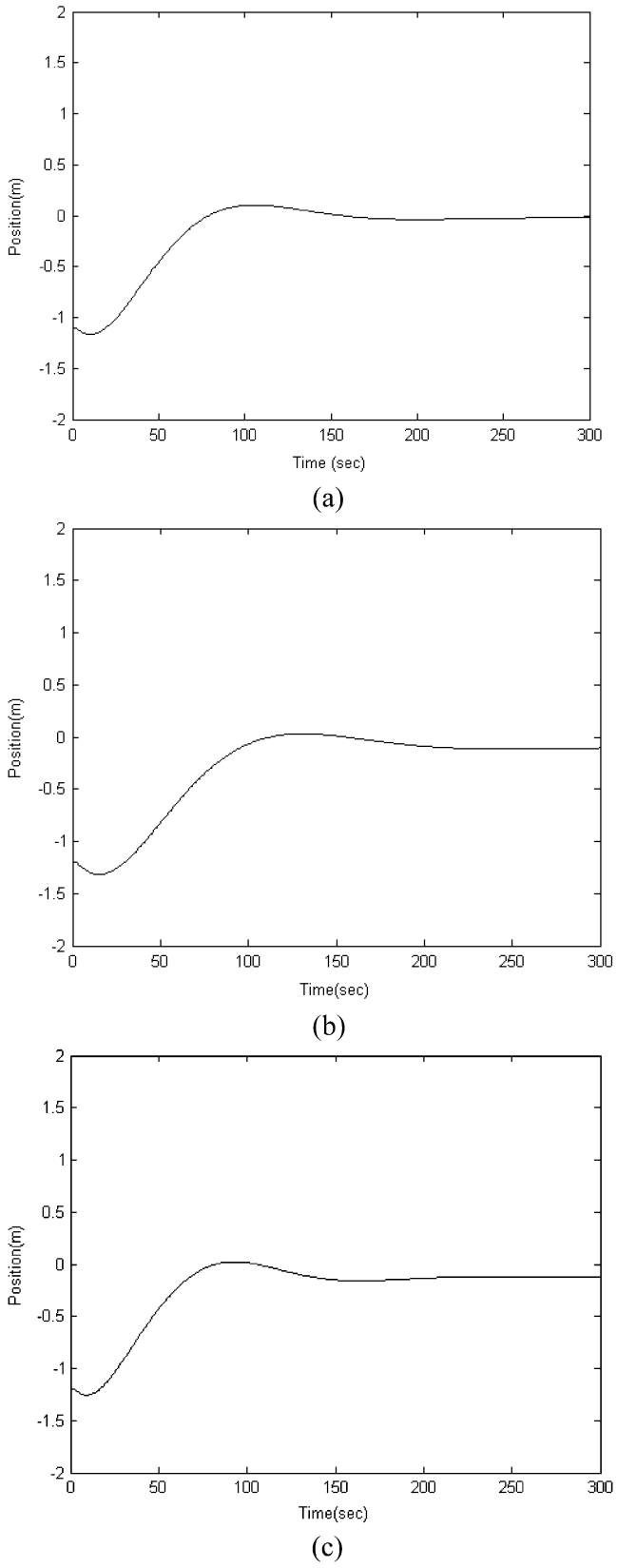


Fig. 20. Position deviation of the ball by a trained (a) R-HELA method, (b) R-SE method [28], and (c) R-GA method [22].

deviation was 0.01°. The results show that the trained RWNFS model had good control in the cart-pole balancing system.

We also compared the performance of our system with the reinforcement symbiotic evolution (R-SE) [28] and the reinforcement genetic algorithm (R-GA) [22] when they were applied to the same problem. In the R-GA and the R-SE, the population size was set to 200 and the crossover and mutation probabilities were set to 0.5 and 0.3, respectively. Fig. 14(b) and (c) shows that the R-SE and the R-GA methods learned to balance the pole on average at the eightieth and one hundred forty-ninth generations. Fig. 16(b) and (c) shows the angular deviation of the pole when the cart-pole balancing system was controlled by [28] and [22]. The average angular deviation of [28] and [22] models was 0.06° and 0.1°. As shown in Figs. 13 and 15, the control capabilities of the trained RWNFS model using the R-HELA are better than [22] and [28] in the cart-pole balancing system.

GENITOR [46], SANE [41], TDGAR [27], and CQGAF [32] have been applied to the same control problem; the simulation results are listed in Table II. Table II shows the number of pole-balancing trials (which reflects the number of training episodes required). In GENITOR [46], the normal evolution algorithm was used to evolve the weights in a fully connected two-layer neural network, with additional connections from each input unit to the output layer. The network consists of five input units, five hidden units, and one output unit. In SANE [41], the symbiotic evolution algorithm is used to evolve a two-layer neural network with five input units, eight hidden units, and two output units. An individual in the SANE represents a hidden unit with five specified connections to input and output units. The TDGAR [27] learning scheme is a new hybrid GA, which integrates the TD prediction method and the GA to fulfill the reinforcement learning task. The CQGAF [32] fulfills the GA-based fuzzy system design in a reinforcement learning environment where only weak reinforcement signals such as "success" and "failure" are available. As shown in Table II, the proposed R-HELA is feasible and effective. We also compared the CPU times with those of other existing methods [20], [22], [27], [28], [32], [41], and [46]. The results are shown in Table III. In this experiment, we used a Pentium 4 chip with a 1.5 GHz CPU, a 512 MB memory, and the visual C++ 6.0 simulation software. The comparison in Table III shows that our proposed HELA method obtains smaller CPU times than those of other existing models [20], [22], [27], [28], [32], [41], and [46].

To demonstrate the efficiency of the proposed ERS method, three different methods were used in this example: the proposed R-HELA method (Type I), the proposed R-HELA method without ERS (Type II), and the fixed-length genetic algorithm (Type III). In Type I, the proposed R-HELA method combines the MVGA and the ERS methods. In Type II, the probability vectors are not used to determine the number of fuzzy rules. That is, only the MVGA method is used. In Type III, the number of fuzzy rules is determined by executing a genetic algorithm with a fixed string length for each specification of the number of fuzzy rules, and then the average of the generations is computed. Table IV shows the performance comparison of the three methods. As shown in Table IV, our proposed HELA with the ERS method performs better than the other methods.

Example 2: Control of a Ball and Beam System: A ball and beam system [47] is shown in Fig. 17. The beam is made to

TABLE V
PERFORMANCE COMPARISON OF VARIOUS EXISTING MODELS IN EXAMPLE 2

Method	Mean	Best	Worst
GENITOR [46]	4982	551	19853
SANE [41]	2287	150	6217
R-GA [22]	466	97	678
R-SE [28]	274	32	492
TDGAR [27]	210	30	324
CQGAF [32]	187	23	298
R-HELA	**115**	**17**	**256**

TABLE VI
THE COMPARISON OF CPU TIME FOR VARIOUS EXISTING MODELS IN EXAMPLE 2

Method	Mean	Best	Worst
GENITOR [46]	113.15	73.34	297.62
SANE [41]	70.26	51.54	197.61
R-GA [22]	60.79	46.35	122.93
R-SE [28]	42.25	18.23	101.43
TDGAR [27]	39.58	11.35	75.76
CQGAF [32]	32.51	8.27	62.21
R-HELA	**25.32**	**6.82**	**58.14**

rotate in a vertical plane when a torque is applied at the center of rotation. The ball is free to roll along the beam. We require that the ball remain in contact with the beam.

The ball and beam system can be written in state space form as

$$\begin{bmatrix}\dot{x}_1\\ \dot{x}_2\\ \dot{x}_3\\ \dot{x}_4\end{bmatrix} = \begin{bmatrix}x_2\\ B(x_1x_4^2 - G\sin x_3)\\ x_4\\ 0\end{bmatrix} + \begin{bmatrix}0\\ 0\\ 0\\ 1\end{bmatrix} u, \quad y = x_1 \tag{22}$$

where $x = (x_1, x_2, x_3, x_4)^T \equiv (r, \dot{r}, \theta, \dot{\theta})^T$ 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 $G = 9.81$ are 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 [47], the control law $u(x)$ is determined as follows: for state x, compute $v(x) = -\alpha_3\phi_4(x) - \alpha_2\phi_3(x) - \alpha_1\phi_2(x) - \alpha_0\phi_1(x)$, 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 are chosen so that $s^4 + \alpha_3 s^3 + \alpha_2 s^2 + \alpha_1 s + \alpha_0$ is Hurwitz polynomial. Compute $a(x) = -BG\cos x_3$ and $b(x) = BGx_4^2 \sin x_3$; then $u(x) = [v(x) - b(x)]/a(x)$.

The four input variables $(r, \dot{r}, \theta, \dot{\theta})$ and the output $u(k)$ are normalized between zero and one over the following ranges: r: $[-5, 5], \dot{r}$: $[-3, 3], \theta$: $[-1, 1], \dot{\theta}$: $[-2, 2], u$: $[-70, 70]$. The values are floating-point numbers initially assigned using the R-HELA. In the proposed R-HELA method, the fitness function in this example is also defined in (16) to train the RWNFS model, where (16) states how long the ball and beam system fails and receives a penalty signal of -1 when the beam deviates beyond a certain angle ($|\theta| > 12$ °C) and the ball reaches the end of the beam ($|r| > 2$ m). A total of 30 runs were performed. Each run started in the same initial state. Fig. 18(a) shows that the RWNFS model learned on average to balance the ball in the sixty–sixth generation. In this figure, each run results in the largest fitness value in the current generation being selected before the ball and beam system fails. After the learning process was stopped, we chose the best string in the population in the final generation and tested it on the ball and beam system. Fig. 19 shows the results of the probability vectors in the ERS. As shown in Fig. 19, the final average optimal number of rules is four. The obtained fuzzy rules of the RWNFS using the R-HELA method are shown as follows:

R^1: If I_{11} is $A_{1,1}(0.48, 0.43)$ and I_{12} is $A_{2,1}(0.29, 0.22)$
and I_{13} is $A_{3,1}(0.96, 0.77)$ and I_{14} is $A_{4,1}(0.58, 0.34)$
Then $\hat{y}_1^1 = 0.99\varphi_{0.0} - 0.31\varphi_{1.0} + 1.98\varphi_{1.1} + 0.56\varphi_{2.0}$

R^2: If I_{21} is $A_{1,2}(0.73, 1.14)$ and I_{22} is $A_{2,2}(0.59, 0.18)$
and I_{23} is $A_{3,2}(0.50, 0.85)$ and I_{24} is $A_{4,2}(0.50, 0.074)$
Then $\hat{y}_2^1 = -0.099\varphi_{0.0} + 0.29\varphi_{1.0} - 0.50\varphi_{1.1} - 0.06\varphi_{2.0}$

R^3: If I_{31} is $A_{1,3}(1.01, 0.59)$ and I_{32} is $A_{2,3}(0.30, 0.56)$
and I_{33} is $A_{3,3}(0.39, 0.35)$ and I_{34} is $A_{4,3}(0.02, 0.53)$
Then $\hat{y}_2^1 = 0.11\varphi_{0.0} - 0.19\varphi_{1.0} + 0.25\varphi_{1.1} + 0.06\varphi_{2.0}$

R^4: If I_{41} is $A_{1,4}(0.66, 1.31)$ and I_{42} is $A_{2,4}(0.28, 0.49)$
and I_{43} is $A_{3,4}(0.22, 0.072)$ and I_{44} is $A_{4,4}(1.061, 0.61)$
Then $\hat{y}_4^1 = 0.022\varphi_{0.0} - 0.25\varphi_{1.0} + 0.25\varphi_{1.1} + 0.15\varphi_{2.0}$.

Fig. 20(a) shows the position deviation of the ball when the ball and beam system was controlled by the well-trained RWNFS model starting at the initial state: $x(0) = -1.2, \dot{x}(0) = -0.01, \theta(0) = 0.58, \dot{\theta}(0) = 0.58$. In this figure, the position

of the ball decays to zero gradually. The results show that the trained RWNFS model has good control in the ball and beam balancing system.

In this example, as with example 1, we also compared the performance of our method with the R-SE method [28] and the R-GA method [22]. The R-GA and the R-SE methods used the same as those used in example 1. Figs. 18(b) and (c) shows that the R-SE method and the R-GA method learned, on average, to balance the ball in the one hundred ninety-fourth generation and two hundred sixty-eighth generation. Fig. 20(b) and (c) shows the position deviation of the ball when the ball and beam system was controlled by the R-SE and the R-GA methods starting in the initial state $r(0) = -1.2, \dot{r}(0) = -0.01, \theta(0) = 0.58, \dot{\theta}(0) = 0.58$. As shown in Figs. 17 and 19, the control capabilities of the trained RWNFS model using the R-HELA are also better than those in [22] and [28] in the ball and beam balancing system. Table V shows a performance comparison of various existing models [20], [22], [27], [28], [32], [41], [46] in Example 2. As shown in Table V, the performance indexes (i.e., mean, best, and worst generations) of the proposed learning method are better than for the methods in [20], [22], [27], [28], [32], [41], and [46]. In addition to comparing the performance of the seven models, as shown in Table VI, we also compared the CPU times of the existing models [22], [27], [28], [32], [41], [46]. From Table VI, we can see that the proposed HELA method obtains smaller CPU times than the existing models.

VI. Conclusion

In this paper, an RWNFS with an R-HELA was proposed for dynamic control problems. The proposed R-HELA has structure-and-parameter learning ability. That is, it can determine the average optimal number of fuzzy rules and tune the free parameters in the RWNFS model. The proposed learning method also processes variable lengths of chromosomes in a population. Computer simulations have shown that the proposed R-HELA performs better than the other methods. In addition to being used to solve the problems given in this paper, the proposed R-HELA method was also used in our laboratory to solve practical control problems in magnetic levitation systems.

Although the R-HELA method can perform better than other methods, there are limitations to the proposed R-HELA method. In this paper, a systematic method was not used to determine the initial parameters. The initial parameters are determined by practical experimentation or by trial and error. In future works, we will find a well-defined method to determine the initial parameters.

In addition, multiobjective algorithm has become an important topic in the field of genetic fuzzy systems [49]–[52]. The algorithm uses evolutionary multiobjective optimization algorithms to search for a number of Pareto-optimal fuzzy rule-based systems with respect to their accuracy and their complexity. The multiobjective algorithm can overcome problems like overfitting/overlearning faced by single objective algorithms. This is also an interesting future research topic that we will consider when we extend the proposed R-HELA method to evolutionary multiobjective optimization problems.

Acknowledgment

The authors would like to thank the reviewers for their helpful suggestions in improving the quality of the final manuscript.

References

[1] C. T. Lin and C. S. G. Lee, *Neural Fuzzy Systems: A Neuro-Fuzzy Synergism to Intelligent System*. Englewood Cliffs, NJ: Prentice-Hall, 1996.

[2] G. G. Towell and J. W. Shavlik, "Extracting refined rules from knowledge-based neural networks," *Machine Learn.*, vol. 13, pp. 71–101, 1993.

[3] C. J. Lin and C. T. Lin, "An ART-based fuzzy adaptive learning control network," *IEEE Trans. Fuzzy Syst.*, vol. 5, no. 4, pp. 477–496, Nov. 1997.

[4] L. X. Wang and J. M. Mendel, "Generating fuzzy rules by learning from examples," *IEEE Trans. Syst., Man, Cybern.*, vol. 22, no. 6, pp. 1414–1427, Nov./Dec. 1992.

[5] T. Takagi and M. Sugeno, "Fuzzy identification of systems and its applications to modeling and control," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-15, pp. 116–132, 1985.

[6] C. F. Juang and C. T. Lin, "An on-line self-constructing neural fuzzy inference network and its applications," *IEEE Trans. Fuzzy Syst.*, vol. 6, no. 1, pp. 12–31, Feb. 1998.

[7] J.-S. R. Jang, "ANFIS: Adaptive-network-based fuzzy inference system," *IEEE Trans. Syst., Man, Cybern.*, vol. 23, pp. 665–685, 1993.

[8] F. J. Lin, C. H. Lin, and P. H. Shen, "Self-constructing fuzzy neural network speed controller for permanent-magnet synchronous motor drive," *IEEE Trans. Fuzzy Syst.*, vol. 9, pp. 751–759, Oct. 2001.

[9] H. Takagi, N. Suzuki, T. Koda, and Y. Kojima, "Neural networks designed on approximate reasoning architecture and their application," *IEEE Trans. Neural Netw.*, vol. 3, no. 5, pp. 752–759, 1992.

[10] E. Mizutani and J.-S. R. Jang, "Coactive neural fuzzy modeling," in *Proc. Int. Conf. Neural Networks*, 1995, pp. 760–765.

[11] C.-J. Lin and C.-C. Chin, "Prediction and identification using wavelet-based recurrent fuzzy neural networks," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 34, pp. 2144–2154, Oct. 2004.

[12] K. S. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks," *IEEE Trans. Neural Netw.*, vol. 1, pp. 4–27, 1990.

[13] C. F. Juang and C. T. Lin, "A recurrent self-organizing neural fuzzy inference network," *IEEE Trans. Neural Netw.*, vol. 10, pp. 828–845, Jul. 1999.

[14] P. A. Mastorocostas and J. B. Theocharis, "A recurrent fuzzy-neural model for dynamic system identification," *IEEE Trans. Syst., Man, Cybern.*, vol. 32, pp. 176–190, Apr. 2002.

[15] A. G. Barto, R. S. Sutton, and C. W. Anderson, "Neuron like adaptive elements that can solve difficult learning control problem," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-13, no. 5, pp. 834–847, 1983.

[16] H. R. Berenji and P. Khedkar, "Learning and tuning fuzzy logic controllers through reinforcements," *IEEE Trans. Neural Netw.*, vol. 3, no. 5, pp. 724–740, 1992.

[17] C. J. Lin, "A GA-based neural network with supervised and reinforcement learning," *J. Chin. Inst. Electr. Eng.*, vol. 9, no. 1, pp. 11–25, 2002.

[18] D. E. Goldberg, *Genetic Algorithms in Search Optimization and Machine Learning*. Reading, MA: Addison-Wesley, 1989.

[19] J. K. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge, MA: MIT Press, 1992.

[20] L. J. Fogel, "Evolutionary programming in perspective: The top-down view," in *Computational Intelligence: Imitating Life*, J. M. Zurada, R. J. Marks II, and C. Goldberg, Eds. Piscataway, NJ: IEEE Press, 1994.

[21] I. Rechenberg, "Evolution strategy," in *Computational Intelligence: Imitating Life*, J. M. Zurada, R. J. Marks II, and C. Goldberg, Eds. Piscataway, NJ: IEEE Press, 1994.

[22] C. L. Karr, "Design of an adaptive fuzzy logic controller using a genetic algorithm," in *Proc. 4th Int. Conf. Genetic Algorithms*, 1991, pp. 450–457.

[23] A. Homaifar and E. McCormick, "Simultaneous design of membership functions and rule sets for fuzzy controllers using genetic algorithms," *IEEE Trans. Fuzzy Syst.*, vol. 3, no. 2, pp. 129–139, 1995.

[24] M. Lee and H. Takagi, "Integrating design stages of fuzzy systems using genetic algorithms," in *Proc. 2nd IEEE Int. Conf. Fuzzy Syst.*, San Francisco, CA, 1993, pp. 612–617.

[25] K. Belarbi and F. Titel, "Genetic algorithm for the design of a class of fuzzy controllers: An alternative approach," *IEEE Trans. Fuzzy Syst.*, vol. 8, no. 4, pp. 398–405, 2000.
[26] C. F. Juang, "A hybrid of genetic algorithm and particle swarm optimization for recurrent network design," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 34, no. 2, pp. 997–1006, 2004.
[27] C. T. Lin and C. P. Jou, "GA-based fuzzy reinforcement learning for control of a magnetic bearing system," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 30, pp. 276–289, Apr. 2000.
[28] C. F. Juang, J. Y. Lin, and C. T. Lin, "Genetic reinforcement learning through symbiotic evolution for fuzzy controller design," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 30, pp. 290–302, Apr. 2000.
[29] S. Bandyopadhyay, C. A. Murthy, and S. K. Pal, "VGA-classfifer: Design and applications," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 30, pp. 890–895, Dec. 2000.
[30] B. Carse, T. C. Fogarty, and A. Munro, "Evolving fuzzy rule based controllers using genetic algorithms," *Fuzzy Sets Syst.*, vol. 80, no. 3, pp. 273–293, Jun. 24, 1996.
[31] K. S. Tang, "Genetic algorithms in modeling and optimization," Ph.D. dissertation, Dept. Electron. Eng., City Univ. Hong Kong, Hong Kong, China, 1996.
[32] C. F. Juang, "Combination of online clustering and Q-value based GA for reinforcement fuzzy system design," *IEEE Trans. Fuzzy Syst.*, vol. 13, no. 3, pp. 289–302, Jun. 2005.
[33] D. W. C. Ho, P. A. Zhang, and J. Xu, "Fuzzy wavelet networks for function learning," *IEEE Trans. Fuzzy Syst.*, vol. 9, no. 1, pp. 200–211, 2001.
[34] I. Daubechies, "Orthonormal bases of compactly supported wavelets," *Commun. Pur. Appl. Math.*, vol. 41, 1998.
[35] T. Yamakawa, E. Uchino, and T. Samatsu, "Wavelet neural networks employing over-complete number of compactly supported non-orthogonal wavelets and their applications," in *Proc. IEEE Conf. Neural Networks*, 1994, vol. 3, pp. 1391–1396.
[36] J. Zhang and A. J. Morris, "Recurrent neuro-fuzzy networks for nonlinear process modeling," *IEEE Trans. Neural Netw.*, vol. 10, no. 2, pp. 313–326, 1999.
[37] S. F. Su and F. Y. Yang, "On the dynamical modeling with neural fuzzy networks," *IEEE Trans. Neural Netw.*, vol. 13, pp. 1548–1553, Nov. 2002.
[38] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*. New York: Springer-Verlag, 1999.
[39] R. Tanese, "Distributed genetic algorithm," in *Proc. Int. Conf. Genetic Algorithms*, 1989, pp. 434–439.
[40] J. Arabas, Z. Michalewicz, and J. Mulawka, "GAVaPS—A genetic algorithm with varying population size," in *Proc. IEEE Int. Conf. Evol. Comput.*, Orlando, FL, 1994, pp. 73–78.
[41] D. E. Moriarty and R. Miikkulainen, "Efficient reinforcement learning through symbiotic evolution," *Machine Learn.*, vol. 22, pp. 11–32, 1996.
[42] G. R. Harik, F. G. Lobo, and D. E. Goldberg, "The compact genetic algorithm," *IEEE Trans. Evol. Comput.*, vol. 3, pp. 287–297, Nov. 1999.
[43] K. Y. Lee, B. Xiaomin, and P. Young-Moon, "Optimization method for reactive power planning by using a modified simple genetic algorithm," *IEEE Trans. Power Syst.*, vol. 10, pp. 1843–1850, Nov. 1995.
[44] O. Cordon, F. Herrera, F. Hoffmann, and L. Magdalena, *Genetic Fuzzy Systems Evolutionary Tuning and Learning of Fuzzy Knowledge Bases* Transl.:Advances in Fuzzy Systems—Applications and Theory. Singapore: World Scientific, 2001, vol. 19.
[45] K. C. Cheok and N. K. Loh, "A ball-balancing demonstration of optimal and disturbance-accommodating control," *IEEE Contr. Syst. Mag.*, pp. 54–57, 1987.
[46] D. Whitley, S. Dominic, R. Das, and C. W. Anderson, "Genetic reinforcement learning for neuro control problems," *Mach. Learn.*, vol. 13, pp. 259–284, 1993.
[47] J. Hauser, S. Sastry, and P. Kokotovic, "Nonolinear control via approximate input-output linearization: The ball and beam example," *IEEE Trans. Autom. Control*, vol. 37, pp. 392–398, Mar. 1992.
[48] C. T. Lin and C. S. G. Lee, "Reinforcement structure/parameter learning for neural-network-based fuzzy logic control systems," *IEEE Trans. Fuzzy Syst.*, vol. 2, pp. 46–63, Feb. 1994.
[49] H. Wang, S. Kwong, Y. Jin, W. Wei, and K.-F. Man, "Agent-based evolutionary approach for interpretable rule-based knowledge extraction," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 35, pp. 143–155, May 2005.
[50] H. Wang, S. Kwong, Y. Jin, W. Wei, and K.-F. Man, "Multi-objective hierarchical genetic algorithm for interpretable fuzzy rule-based knowledge extraction," *Fuzzy Sets Syst.*, vol. 149, pp. 149–186, 2005.
[51] H. Ishibuchi and T. Yamamoto, "Fuzzy rule selection by multi-objective genetic local search algorithms and rule evaluation measures in data mining," *Fuzzy Sets Syst.*, vol. 141, pp. 59–88, 2004.
[52] H. Ishibuchi, T. Murata, and I. B. Tiirksen, "Single-objective and two-objective genetic algorithms for selecting linguistic rules for pattern classification problems," *Fuzzy Sets Syst.*, vol. 89, pp. 135–150, 1997.
[53] H. Ishibuchi, K. Nozaki, N. Yamamoto, and H. Tanaka, "Selecting fuzzy if-then rules for classification problems using genetic algorithms," *IEEE Trans. Fuzzy Syst.*, vol. 3, no. 3, pp. 260–270, 1995.
[54] P. Angelov and R. Buswell, "Identification of evolving fuzzy rule-based models," *IEEE Trans. Fuzzy Syst.*, vol. 10, pp. 667–677, Oct. 2002.

Cheng-Jian Lin (M'95) received the B.S. degree in electrical engineering from Ta-Tung University, Taiwan, R.O.C., in 1986 and the M.S. and Ph.D. degrees in electrical and control engineering from the National Chiao-Tung University, Taiwan, in 1991 and 1996, respectively.

From 1996 to 1999, he was an Associate Professor in the Department of Electronic Engineering, Nan-Kai College, Nantou, Taiwan. Since 1999, he has been with the Department of Computer Science and Information Engineering, Chaoyang University of Technology. Currently, he is a Professor in the Computer Science and Information Engineering Department, Chaoyang University of Technology, Taichung, Taiwan. His current research interests are neural networks, fuzzy systems, pattern recognition, intelligence control, information retrieval, e-learning, and field-programmable gate-array design. He has published more than 100 papers in referred journals and conference proceedings. He is an Associate Editor of the *International Journal of Applied Science and Engineering*.

Dr. Lin is a member of Phi Tau Phi, the Chinese Fuzzy Systems Association, the Chinese Automation Association, the Taiwanese Association for Artificial Intelligence, and the IEEE Computational Intelligence Society. He is an Executive Committee Member of the Taiwanese Association for Artificial Intelligence.

Yung-Chi Hsu received the B.S. degree in information management from Ming-Hsin University of Science and Technology, Taiwan, R.O.C., in 2002 and the M.S. degree in computer science and information engineering from Chaoyang University of Technology, Taiwan. He is currently pursuing the Ph.D. degree in the Department of Electrical and Control Engineering, National Chiao-Tung University, Taiwan. His research interests include neural networks, fuzzy systems, and genetic algorithms.

Mr. Hsu is a member of Phi Tau Phi.