Available online at www.sciencedirect.com

SCIENCE DIRECT®

Mathematical and Computer Modelling 43 (2006) 563–581

MATHEMATICAL AND COMPUTER MODELLING

www.elsevier.com/locate/mcm

A hybrid evolutionary learning algorithm for TSK-type fuzzy model design

Cheng-Jian Lin*, Yong-Ji Xu

Department of Computer Science and Information Engineering, Chaoyang University of Technology, No. 168, Jifong E. Road, Wufong Township, Taichung County 41349, Taiwan, ROC

Received 14 January 2005; received in revised form 2 August 2005; accepted 4 August 2005

Abstract

In this paper, a TSK-type fuzzy model (TFM) with a hybrid evolutionary learning algorithm (HELA) is proposed. The proposed HELA method combines the compact genetic algorithm (CGA) and the modified variable-length genetic algorithm (MVGA). Both the number of fuzzy rules and the adjustable parameters in the TFM are designed concurrently by the HELA method. In the proposed HELA method, individuals of the same length constitute the same group, and there are multiple groups in a population. Moreover, the proposed HELA adopts the compact genetic algorithm (CGA) 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 behavior of the simple GA. The evolution processes of a population consist of three major operations: group reproduction using the compact genetic algorithm, variable two-part individual crossover, and variable two-part mutation. Computer simulations have demonstrated that the proposed HELA method gives a better performance than some existing methods.

Keywords: Evolutionary algorithms; TSK-type fuzzy model; Variable-length genetic algorithm; Identification; Control

1. Introduction

In recent years, fuzzy models for various applications have become a popular research topic [1–9]. The key advantage of the fuzzy model approach lies in the fact that it does not require a mathematical description of a system when the system is modeled. Two typical types of fuzzy models are the Mamdani-type and TSK-type models. Many researchers have shown that using a TSK-type fuzzy model achieves superior performance in network size and learning accuracy than using the Mamdani-type fuzzy model [6,7].

Several learning algorithms of fuzzy models have been proposed in [1–9]. The backpropagation learning algorithm [1] is a widely used algorithm for training fuzzy models by means of error propagation via variation calculus. However, the backpropagation learning algorithm is a powerful training technique that can be applied in networks with feed-forward structure to transform them into adaptive systems. But the algorithm may reach the local minima and the global solution may never be found because the steepest descent optimization technique is

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

used in the backpropagation learning algorithm to minimize the error function. In addition, the performance of the backpropagation learning algorithm depends on the initial values of the model parameters, and for different network topologies one has to derive new mathematical expressions for each network layer.

The advent of evolutionary computation has inspired new designs and models, such as the optimal design of neural networks and fuzzy models, for optimization problem solving. In contrast to traditional computation systems, which may be good at accurate and exact computation but have brittle operations (i.e. for different topologies one has to derive new mathematical expressions), evolutionary computation provides a more robust and efficient approach for solving complex real-world problems [10–12]. Many evolutionary algorithms, such as genetic algorithms (GA) [13], genetic programming [14], evolutionary programming [15], and evolution strategies [16], have been proposed. Since they are heuristic and stochastic, they are less likely to get stuck at the local minimum, and 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. For this reason, an evolutionary fuzzy model is discussed in this paper, and a new algorithm, the hybrid evolution learning algorithm (HELA), is proposed.

The evolutionary fuzzy model generates a fuzzy system automatically by incorporating evolutionary learning procedures [10–16], where the well-known procedure is GA. Several genetic fuzzy models, i.e. fuzzy models augmented by a learning process based on GAs, have been proposed [17–20]. In [17], Karr applied GAs to the design of the membership functions of a fuzzy controller, with the fuzzy rule set assigned in advance. Since the membership functions and rule sets are co-dependent, simultaneous design of these two approaches 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 [18–20]. Differences between the approaches depend mainly on the type of coding and the way in which the membership functions are optimized. To enhance the searching capability of GA, in [21] a relatively new evolutionary algorithm, the particle swarm optimization [22,23], was incorporated, and a hybrid of GA and particle swarm optimization was proposed for recurrent neural/fuzzy networks design. However, in the aforementioned approaches, the number of fuzzy rules had to be assigned in advance. To overcome these inconveniences, the capability to do automatic searches for the number of fuzzy rules is included in the proposed HELA. To obtain this capability, the idea of variable-length genotypes [24] is incorporated, and the idea of compact GA and its corresponding evolution is also proposed in the HELA.

Let the length of each individual denote the total number of genes in it. In the proposed HELA for fuzzy model design, the initial length of each individual may be different from each other, depending on the total number of rules encoded in it. Individuals with an equal number of rules constitute the same group, so initially there are several groups in a population. In this paper, 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 evolution of groups in HELA consists of three operations: the elite-based reproduction strategy using the compact genetic algorithm, variable two-part crossover, and variable two-part mutation. The effectiveness of the proposed HELA method will be verified by identification and control problems.

This paper is organized as follows. Section 2 describes the TSK-type fuzzy model (TFM). In Section 3, the proposed HELA that constructs the TFM automatically is introduced. Section 4 presents the simulation results. The conclusions are given in the last section.

2. Structure of TSK-type fuzzy model (TFM)

A TSK-type fuzzy model employs different implication and aggregation methods from a standard Mamdani fuzzy model. Instead of using fuzzy sets, the conclusion part of a rule is a linear combination of the crisp inputs.

$$\begin{aligned}&\mathit{IF}\ x_1\ \mathit{is}\ A_{1j}(m_{1j},\sigma_{1j})\ \mathit{and}\ x_2\ \mathit{is}\ A_{2j}(m_{2j},\sigma_{2j})\ldots\ \mathit{and}\ x_n\ \mathit{is}\ A_{nj}(m_{nj},\sigma_{nj})\\&\qquad \mathit{Then}\ y' = w_{0j} + w_{1j}x_1 + \cdots + w_{nj}x_n.\end{aligned} \tag{1}$$

In this paper, we adopt a TSK-type fuzzy model (TFM) with a hybrid evolutionary learning algorithm (HELA) to perform various identification and control problems. The structure of a TFM is shown in Fig. 1, where n and R are the number of input dimensions and the number of rules respectively. It is a five-layer network structure. In the proposed TFM model, the firing strength of a fuzzy rule is calculated by performing the following “AND” operation on the truth

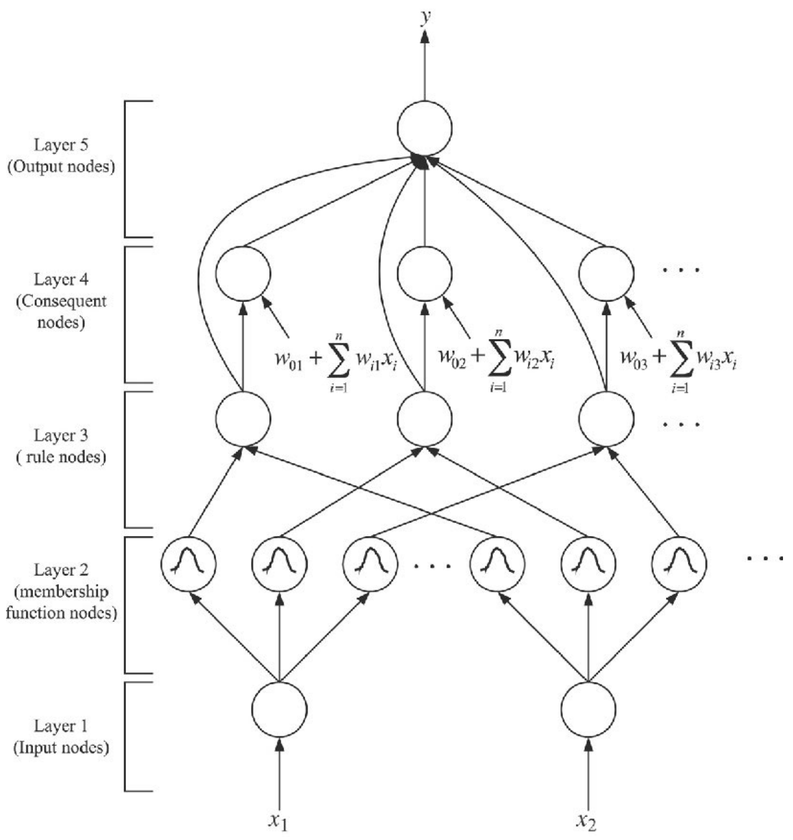
y
Layer 5
(Output nodes)
Layer 4
(Consequent nodes)
Layer 3
(rule nodes)
Layer 2
(membership function nodes)
Layer 1
(Input nodes)
$w_{01} + \sum_{i=1}^{n} w_{i1}x_i$
$w_{02} + \sum_{i=1}^{n} w_{i2}x_i$
$w_{03} + \sum_{i=1}^{n} w_{i3}x_i$
x_1
x_2

Fig. 1. Structure of the TSK-type fuzzy model.

values of each variable to its corresponding fuzzy sets by:

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

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

The output of a fuzzy system is computed by:

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

where $u^{(5)}$ is the output of the 5th layer and R is the number of fuzzy rules.

3. The hybrid evolutionary learning algorithm (HELA)

This section will introduce the proposed hybrid evolutionary learning algorithm (HELA). Recently, many efforts to enhance traditional GAs have been made [25]. Among them, one category focuses on modifying the structure of a population or the role an individual plays in it [26–29]. Examples in this category include the distributed GA [26], the cellular GA [27], and the symbiotic GA [29].

In a normal evolution algorithm, the number of fuzzy rules in a fuzzy model must be predefined. Our proposed HELA combines the compact genetic algorithm (CGA) and the modified variable-length genetic algorithm (MVGA). In 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 fuzzy rules.

Individuals with an equal number of rules constitute the same group. Thus, initially, there are several groups in a population. Unlike the traditional VGA, Bandyopadhyay [24] used "#" to mean "does not care". In this study, 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. In VTC and VTM, we do not use "#" to mean "does not care".

We divide a chromosome into two parts. The first part of the chromosome gives the antecedent parameters of the fuzzy rules (i.e., the parameters of the membership functions), and the second part of the chromosome gives the consequent parameters of the fuzzy rules (i.e., the coefficients of the linear combination). Each part of the chromosome can be performed using the VTC on the overlapping genes of two chromosomes. The structure of the chromosomes in MVGA is shown in Fig. 2.

In traditional VGA, Bandyopadhyay [24] only evaluated the performance of each chromosome in a population. The performance of the number of rules was not evaluated in [24]. In this study, we use the elite-based reproduction strategy to keep the best group with the same length chromosomes. 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 suitable to the environment as they acquire knowledge from society.

In the proposed HELA method, we adopt the compact genetic algorithm (CGA) [30] 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 behavior of the simple GA [31]. The pseudo code of the CGA is shown in Fig. 3. As shown in Fig. 3, we can see that the CGA uses the probability vectors to represent the gene is suitable to 1 or 0. The advantage of the CGA is that it processes each gene independently and requires less memory than the normal GA. The learning diagram of the proposed HELA method is shown in Fig. 4. The building blocks (BBs) in CGA represent the suitable lengths of the chromosomes and reproduce the chromosomes according to the BBs.

The learning process of the HELA involves seven major operators: coding, initializing, evaluating, sorting, elite-based reproduction strategy, variable two-part crossover, and variable two-part mutation. Fig. 5 shows the flowchart of the learning process. The whole learning process is described step-by-step as follows:

3.1. Coding

The coding step consists of the coding done by the MVGA and the CGA. The MVGA codes a TSK-type fuzzy model into a chromosome, as shown in Fig. 6. Fig. 6 shows a fuzzy rule that has the form in Eq. (2), where m_{ij} and σ_{ij} represent a Gaussian membership function with mean and deviation, respectively, of the ith dimension and the jth rule node. The CGA codes the probability vector into building blocks (BBs), as shown in Fig. 7, where each probability vector represents the suitability of the rule of a TFM. In Fig. 7 we can see that we must predefine the maximum number of rules ($R_{\max}$) and minimum number of rules ($R_{\min}$) to prevent less or more fuzzy rules from being generated in a TFM.

3.2. Initializing

The initializing step sets initial values in the MVGA and the CGA. In the MVGA, individuals should be randomly generated initially to construct a population. In order to keep the same number of chromosomes in each group, the number of chromosomes in each group needs to be generated α chromosomes (that we predefined). Therefore, the population size must be set to $\alpha^*(R_{\max} - R_{\min})$. In the CGA, the probability vectors of BBs are set to 0.5 initially.

3.3. Evaluating

The evaluating step is to evaluate each chromosome in a population. The fitness function is defined as follows:

$$\text{fit} = 1/(1 + E(y, \bar{y})) \tag{4}$$

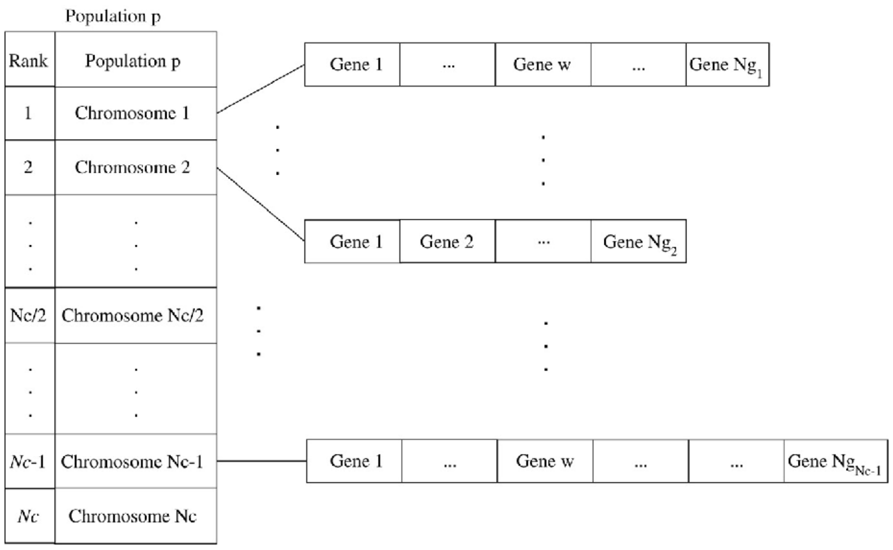

Fig. 2. The structure of the chromosome in the MVGA.

```
Procedure of compact genetic algorithm (CGA)
1. Initialize probability vector :
   for k=1 to BL do
        $V_k$=0.5;
2. Generate two individuals from the vector
   a: generate(p);
   b: generate(p);
3. Compare
   Winner=compete(a,b);
4. Update the probability vector towards the better one from step. 2
   for k=1 to BL do Figure 3: The pseudo code of the CGA
        if winner[k]<>loser[k] then
              if winner[k]=1 then $V_{k=}$ $V_k$+1/PN;
              else $V_{k=}$ $V_k$-1/PN;
5. To check the vector converged
   for k=1 to BL do
        if $V_k$>0 and $V_k$<1 then
              return to Step. 2;
6. V represents the final solution

The parameters of CGA:
        BL: Chromosome length.
        PN: Population Size.
```

Fig. 3. The pseudo code of the CGA.

$$\text{where } E(y, \bar{y}) = \sqrt{\frac{1}{N}\sum_{z=1}^{N}(y_z - \bar{y}_z)^2}, \qquad \text{for } z = 1, 2, \ldots, N \tag{5}$$

where N represents the number of input data; y and $\bar{y}$ represent the model output and desired output.

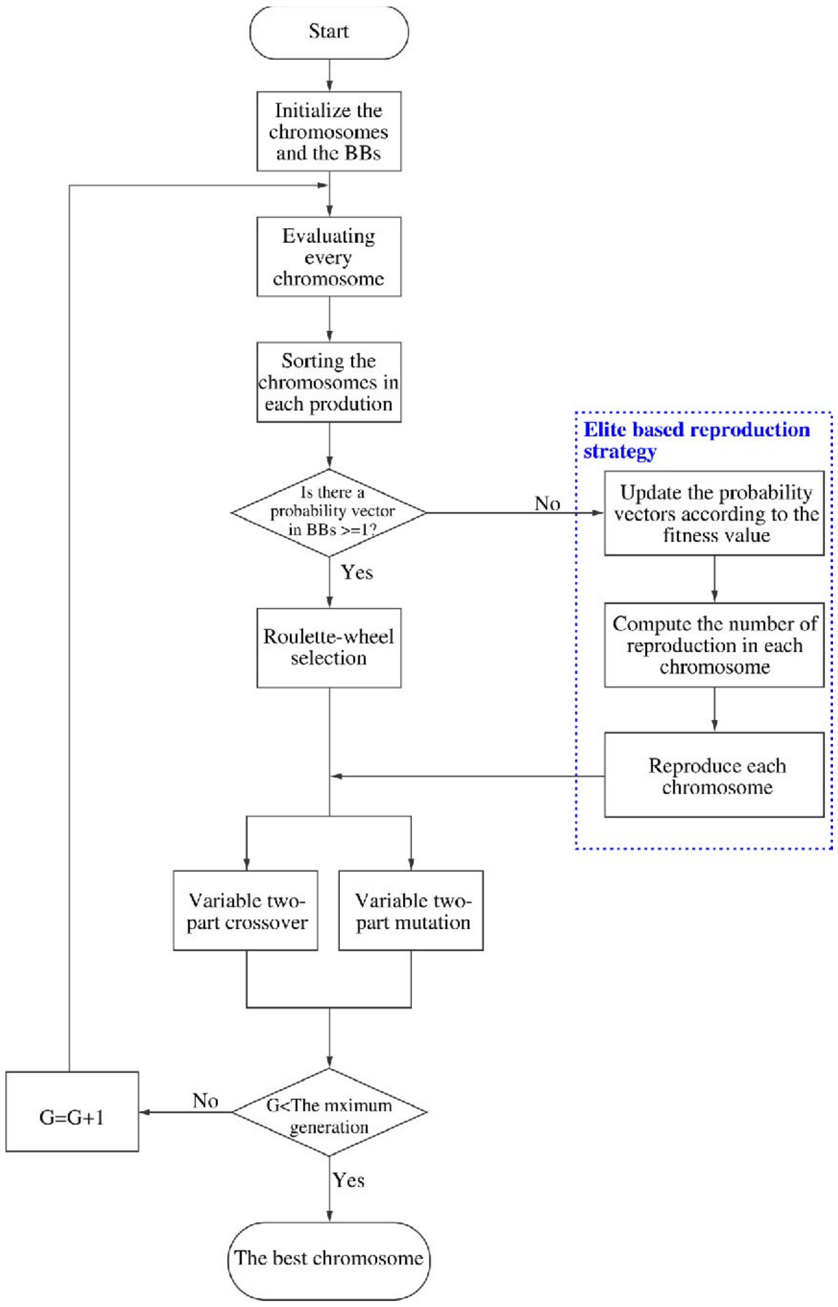
Start
Initialize the chromosomes and the BBs
Evaluating every chromosome
Sorting the chromosomes in each prodution
Is there a probability vector in BBs >=1?
No
Yes
Elite based reproduction strategy
Update the probability vectors according to the fitness value
Compute the number of reproduction in each chromosome
Reproduce each chromosome
Roulette-wheel selection
Variable two-part crossover
Variable two-part mutation
G<The mximum generation
No
G=G+1
Yes
The best chromosome

Fig. 4. The whole learning diagram of the proposed HELA.

3.4. Sorting

We sort the chromosomes in each population according to the decreasing fitness values. After sorting the chromosomes in each population, the algorithm goes to the next step.

3.5. 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 Eqs. (4) and (5). The higher a fitness value, the better the fitness. In this study, we use

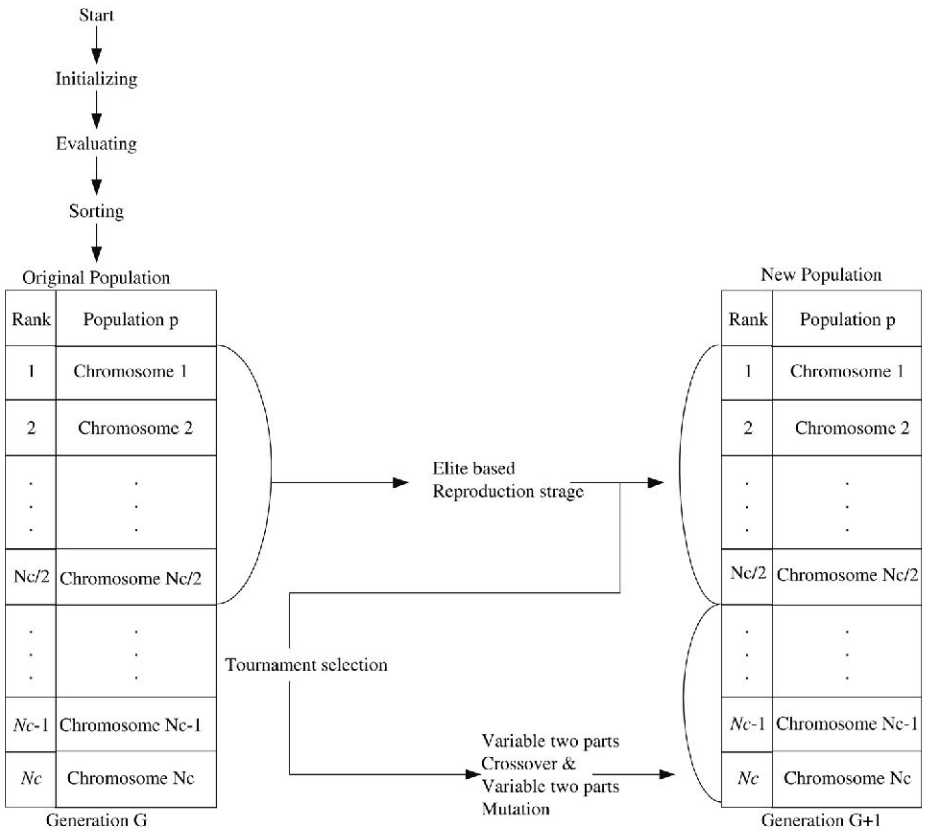

Fig. 5. The flowchart of the parameter learning in the HELA.

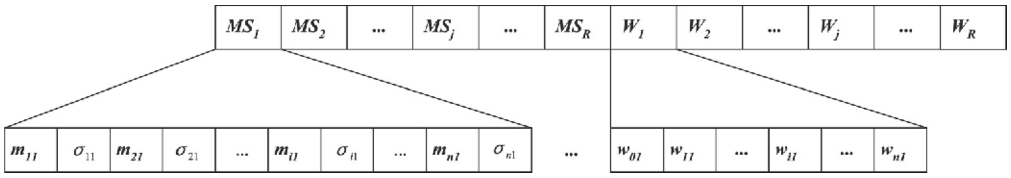
MS_1 MS_2 ... MS_j ... MS_R W_1 W_2 ... W_j ... W_R
m_{11} σ_{11} m_{21} σ_{21} ... m_{i1} σ_{i1} ... m_{n1} σ_{n1} ... w_{01} w_{11} ... w_{i1} ... w_{n1}

Fig. 6. Coding a fuzzy rule into a chromosome in the MVGA.

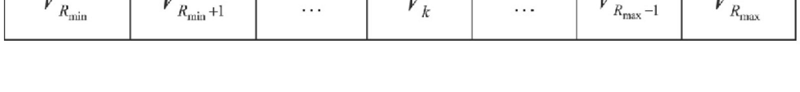
$V_{R_{\min}}$ $V_{R_{\min}+1}$ … V_k … $V_{R_{\max}-1}$ $V_{R_{\max}}$

Fig. 7. Coding of the CGA.

an elite-based reproduction strategy (ERS) to mimic the maturing phenomenon in society, where individuals become more suitable to the environment as they acquire more knowledge from society. The CGA is used here to perform 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 length of the chromosomes and reproduces the chromosomes according to the probability vector in the BBs. The best performing

```
Procedure of elite-based reproduction strategy
Begin
        Let $k$= $R_{min}$,$Temp=0$;
        Repeat
                Update $V_k$ by(6)to(9);
                If $V_k \geqq 1$ then
                        $Temp=k$; $Break$;
                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 (10) to (11)
                        $k=k+1$;
              Until $k$= $R_{max}$;
        End if
End
```

Fig. 8. The pseudo code for the ERS.

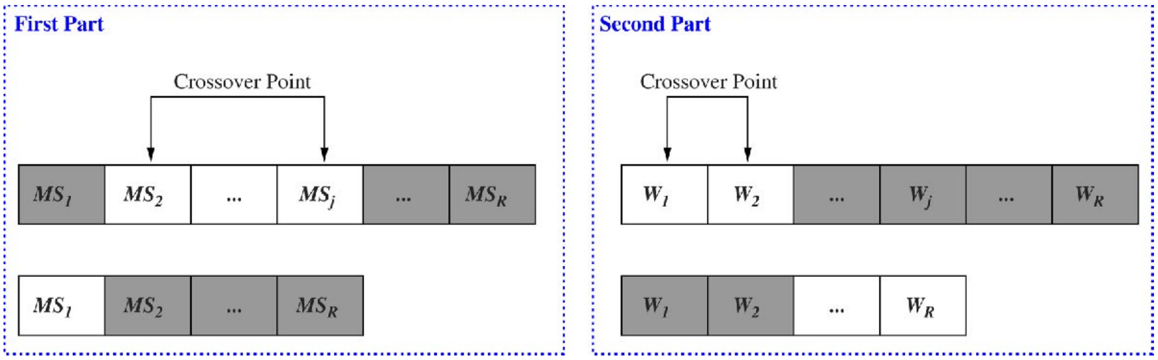

Fig. 9. Variable two-part crossover operation in the HELA.

individuals in the top half of each population are used to perform ERS. According to the results of the ERS, the other half is generated by using crossover and mutation operations. The details of the ERS are as follows:

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

$$\begin{cases} V_k = V_k + (\text{Upt\_value}_k^* \lambda), & \text{if Avg} <= \text{Max\_fit}_k \\ V_k = V_k - (\text{Upt\_value}_k^* \lambda), & \text{otherwise} \end{cases} \tag{6}$$

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

$$\text{Avg} = \sum_{p=1}^{Nc} \text{fit}_p / Nc \tag{7}$$

$$\text{Upt\_value}_k = \text{Total\_fit}_k \bigg/ \sum_{p=1}^{Nc} \text{fit}_p \tag{8}$$

$$\text{Total\_fit}_k = \sum_{p}^{N_k} \text{fit}_{kp} \tag{9}$$

where V_k is the probability vector in the BBs and presents the suitable chromosome in the group with k rules in a population; λ is a threshold value we predefined; Max_fit_k is the best fitness value in the kth group; Avg represents the average fitness value in the whole population; Nc is the population size; and N_k is the kth group size. As show in Eq. (10), if $\text{Max\_fit}_k >=$ Avg that means the suitable chromosomes in the kth group should be increased. On the other hand, if $\text{Max\_fit}_k <$ Avg that means the suitable chromosomes in the kth group should be decreased. Eq. (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:

$$\text{Rep}_k = (P_{\text{size}}/2) * (V_k/\text{Total\_Velocy}) \\ \text{where } k = [R_{\max}, R_{\min}] \tag{10}$$

$$\text{Total\_Velocy} = \sum_{k=R_{\min}}^{R_{\max}} V_k \tag{11}$$

where P_{size} represents the population size; Rep_k is the reproduction number that represents Rep_k chromosomes in the kth group need to be generated, and a chromosome has k rules for constructing a TFM.

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 [33].

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. In order to keep the same number of rules in every group, the redundant genes of chromosomes in different groups are removed. Moreover, the lack of genes in different groups is generated randomly. To replace the ERS step, we use the roulette-wheel selection method [33] — a simulated roulette is spun — for this reproduction process. The pseudo code for the ERS is shown in Fig. 8.

3.6. *Variable two-part crossover*

Although the reproduction 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 the variable two-part crossover (VTC) to perform this step. In VTC, parents are selected from the enhanced elites. Tournament selection [33] is used, in which two enhanced elites are selected, and their fitness values are compared. The elite with the higher fitness value is selected as one parent. The other parent is also selected in the same way. The two parents may be selected from the same or different groups. Performing crossover on the selected parents creates the offspring. Since parents may be of different lengths (i.e. from different groups), we must avoid misalignment of individuals in the crossover operation. Therefore, variable two-part crossover is proposed. The first part of the chromosome gives the antecedent parameters of the fuzzy rules (i.e., the parameters of the membership functions), and the second part of the chromosome gives the consequent parameters of the fuzzy rules (i.e., the coefficients of the linear combination). 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. Two individuals of different lengths using the variable two-part crossover operation are shown in Fig. 9. After the VTC operation, the individuals with poor performance are replaced by the new offspring.

3.7. *Variable two-part mutation*

Although reproduction and crossover operations 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 the variable two-part mutation (VTM) to perform the mutation operation. The proposed VTM is different from the traditional mutation used to mutate chromosomes and is applicable to chromosomes of different lengths. The first part of the chromosome gives the antecedent parameters of the fuzzy rules, and the second part of the chromosome gives the consequent parameters of the fuzzy rules. In each part of a chromosome, the uniform mutation

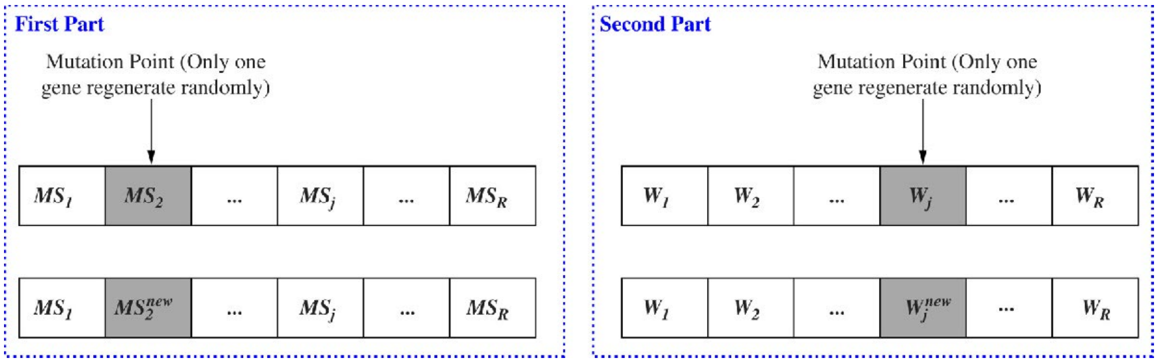
First Part
Mutation Point (Only one gene regenerate randomly)
MS1
MS2
...
MSj
...
MSR
MS1
MS2new
...
MSj
...
MSR
Second Part
Mutation Point (Only one gene regenerate randomly)
W1
W2
...
Wj
...
WR
W1
W2
...
Wjnew
...
WR

Fig. 10. Variable two-part mutation operation in the HELA.

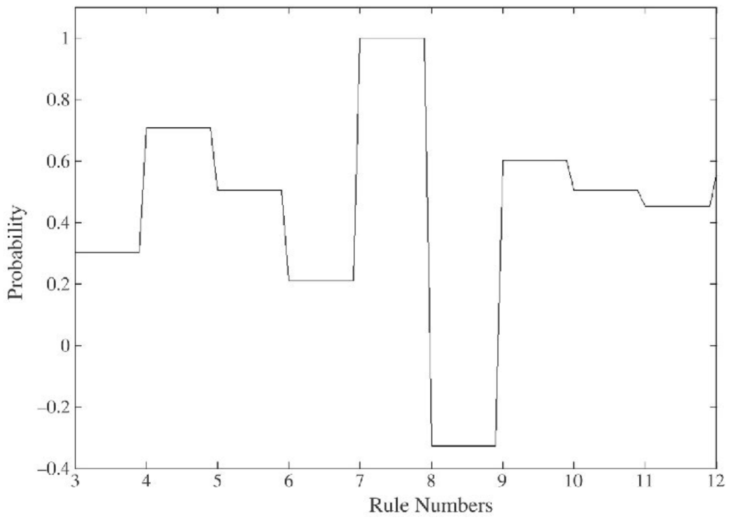

Fig. 11. Results of the probability vectors of the ERS step in the proposed HELA.

is adopted. Originally, mutation was used only for the binary code chromosome. To adopt the concept of introducing variations into the chromosome, a random mutation [33] has been used for the real code chromosome. The equation of a random mutation [33] is as follows:

$$g = g + \psi(\mu, \sigma) \tag{12}$$

where g is the real value gene; ψ is a random function and μ, σ are the mean and variance from the domain of the corresponding variable, respectively. Normally, a random function may be Gaussian or normally distributed. In this paper, we adopt a normally distributed random function. The VTM operation of each individual is shown in Fig. 10.

After the above-mentioned operations, the problem of groups 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 can eliminate unsuitable groups and fuzzy rules.

4. Simulations

To verify the performance of the proposed HELA method, we use two different simulations. The first simulation used the example given by Narendra and Parthasarathy [34]. Secondly, the simulation was used to control the water bath temperature control system that is described in [35]. For the two examples, the initial parameters are given in Table 1 before training.

Table 1
The initial parameters before training

Parameters	Value
Population size	14
Crossover rate	0.5
Mutation rate	0.5
$[\sigma_{\min}, \sigma_{\max}]$	[0,1]
$[m_{\min}, m_{\max}]$	[0,1]
$[w_{\min}, w_{\max}]$	[−20,20]
$R_{\max}$	12
$R_{\min}$	3
λ	0.01
α	2

Example 1: Identification of nonlinear dynamic system

The first example used for identification is described by the difference equation

$$y(k+1) = \frac{y(k)}{1+y^2(k)} + u^3(k). \tag{13}$$

The output of this equation depends nonlinearly on both its past value and the input, but the effects of the input and output values are additive. The 200 training input patterns were generated with $u(k) = \sin(2\pi k/25)$. The evolution progressed for 1000 generations and was repeated 50 times. After 1000 generations, the final average root mean square (rms) errors of the training data and testing data approximated 0.004 and 0.0051. Fig. 11 shows the results of the probability vector in the building blocks (BBs) of the ERS step at 1000 generations. The building blocks (BBs) in CGA represent the suitable lengths of the chromosomes and reproduce the chromosomes according to the BBs. As shown in Fig. 11, we can see that the final average optimal number of rules is 7. The fuzzy rules of the TFM using the HELA method are described as follows:

Rule 1: *IF* x_1 *is* $A_{11}(0.23, 0.66)$ *and* x_2 *is* $A_{21}(0.11, 0.47)$
THEN $y' = -1.09 - 0.37x_1 + 0.99x_2$
Rule 2: *IF* x_1 *is* $A_{12}(0.58, 0.61)$ *and* x_2 *is* $A_{22}(0.43, 0.24)$
THEN $y' = 8.82 - 9.88x_1 - 0.14x_2$
Rule 3: *IF* x_1 *is* $A_{13}(0.31, 0.68)$ *and* x_2 *is* $A_{23}(0.65, 0.32)$
THEN $y' = -0.43 - 0.65x_1 + 1.01x_2$
Rule 4: *IF* x_1 *is* $A_{14}(0.71, 0.41)$ *and* x_2 *is* $A_{24}(0.19, 0.96)$
THEN $y' = -0.93 + 0.26x_1 + 1.09x_2$
Rule 5: *IF* x_1 *is* $A_{15}(0.94, 0.57)$ *and* x_2 *is* $A_{25}(0.52, 0.50)$
THEN $y' = -0.25 + 0.20x_1 + 0.04x_2$
Rule 6: *IF* x_1 *is* $A_{16}(0.95, 0.37)$ *and* x_2 *is* $A_{26}(0.17, 0.80)$
THEN $y' = 0.33 + 0.62x_1 + 0.58x_2$
Rule 7: *IF* x_1 *is* $A_{17}(0.04, 0.62)$ *and* x_2 *is* $A_{27}(0.14, 0.98)$
THEN $y' = 0.97 + 1.09x_1 - 1.97x_2$.

To show the effectiveness and efficiency of the proposed HELA, a TFM using symbiotic evolution (SE) [32] and the genetic algorithm (GA) [17] was applied to the same problem. In GA and SE, the population size was set to 200 and the crossover and mutation probabilities were set to 0.5 and 0.3, respectively. We set seven rules to construct the TFM in the SE and the GA. The evolution process progressed for 1000 generations and was repeated 50 times. After 1000 generations, the final average rms errors of the training data for the SE and the GA approximated 0.009 and 0.10. Fig. 12(a)–(c) show the outputs of the three methods for the input $u(k) = \sin(2\pi k/25)$. As shown in Fig. 12(a)–(c),

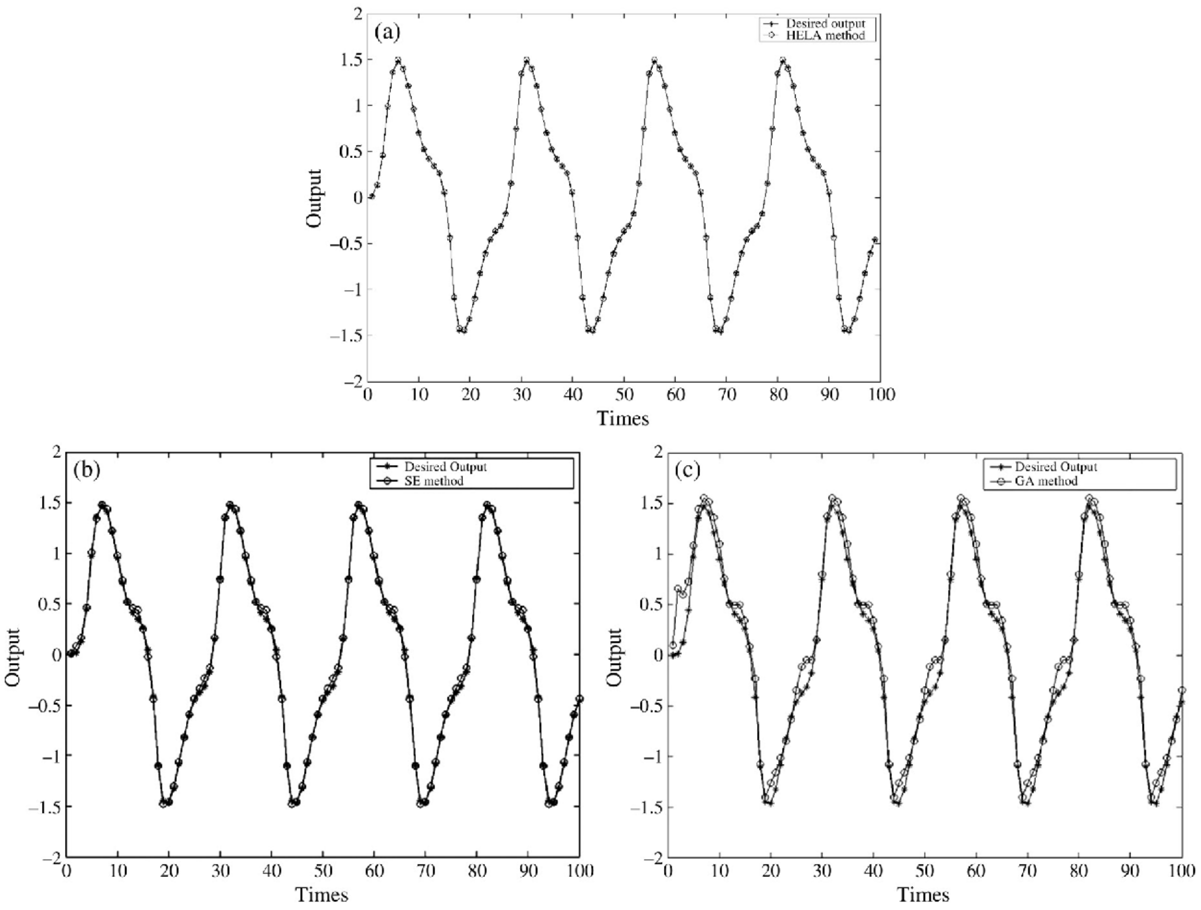

Fig. 12. Results of the desired output and (a) the proposed HELA, (b) the SE [32], and (c) the GA [17].

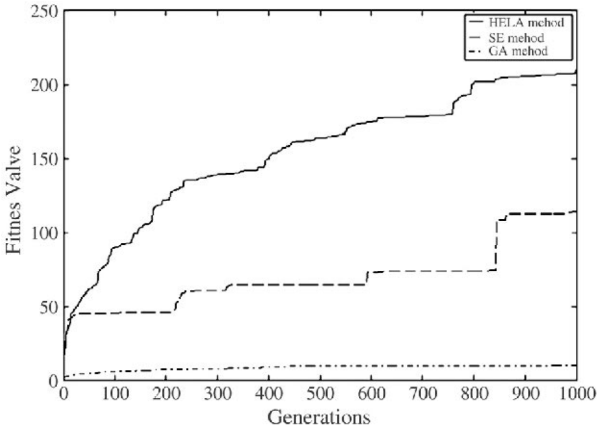

Fig. 13. The learning curves of the proposed HELA method, the SE [32] and the GA [17].

the identification ability of the HELA method was better than those of the SE and GA methods. Fig. 13 shows the learning curves of the three methods. In this figure, we find that the proposed HELA method obtains a lower rms error than the others.

We also compare the performance of our model with some existing models [6,7,34,36]. The performance indices considered include rms error, number of parameters, and training steps and CPU times. The comparison results are

Table 2
Performance comparison of various existing models

	Evolution algorithm			Backpropagation algorithm			
	HELA method	SE method [32]	GA method [17]	ANFIS [6]	SOFIN [7]	Hybrid system [36]	Neural networks [34]
RMS error (training)	**0.004**	0.009	0.10	0.0061	0.013	0.032	0.07
RMS error (testing)	**0.0051**	0.012	0.11	0.007	0.015	0.040	0.079
Number of parameters	**49**	49	49	49	29	112	270
Generations	**1000**	1000	1000	–	–	–	–
Training steps	–	–	–	2000	50 000	30 000	100 000
CPU times (seconds)	**71**	267	227	30	46	198	652

tabulated in Table 2. All the compared algorithms in Table 2 were repeated 50 times. Jang [6] proposed a model, called adaptive-network-based fuzzy inference system architecture, for learning and tuning a fuzzy identifier. In [6], there are seven fuzzy logic rules given in advance by experts. After 2000 training steps, the average rms error of the training data and testing data approximate 0.0061 and 0.007. The results are shown in the fourth column of Table 2. Juang and Lin [7] proposed the SONFIN model. The proposed SONFIN is inherently a modified TSK-type fuzzy rule-based model possessing neural networks. In the SONFIN model the training is performed for 5000 time steps. The average rms error of the training data and testing data approximate 0.013 and 0.015. The results are shown in the fifth column of Table 2. Lin [36] proposed a hybrid system and incorporates a priori knowledge into the selection of initial parameter values. In [36], the fuzzy system contains seven rules and the neural network contains seven hidden units. After the parameter learning, the average rms error of the training data and testing data approximate 0.032 and 0.040. The results are shown in the sixth column of Table 2. Narendra and Paethasarathy [34] using neural networks with two hidden layers. In [34], the identification process is 100 000 training steps. After the parameter learning, the average rms error of the training data and testing data approximate 0.07 and 0.079. The results are shown in the seventh column of Table 2. Besides to compare the performance of the training data and testing data, as shown in Table 2, we also compare the CPU times of the existing models [17,32,6,7,34,36]. In this experiment, we used a Pentium 4 1.5 GHz CPU, 512 MB of main memory, and the Visual C++ 6.0 simulation software. In Table 2, we can know that some backpropagation learning algorithms ([6] and [7]) obtain shorter CPU times than the proposed HELA method. This is because the backpropagation learning algorithms can converge more quickly than the evolution algorithms. However, the BP algorithm may have the local minima problem. As shown in Table 2, the proposed HELA obtains a smaller rms error than other existing models. Moreover, in the evolution algorithms (the HELA, [17], and [32]), our proposed HELA still obtains smaller CPU times than [17], and [32].

Example 2: Water bath temperature control system

The goal of this simulation is to control the temperature of a water bath system given by

$$\frac{\mathrm{d}y(t)}{\mathrm{d}t} = \frac{u(t)}{C} + \frac{Y_0 - y(t)}{R_1 C} \tag{14}$$

where $y(t)$ is the system output temperature in °C; $u(t)$ is heating flowing inward the system; Y_0 is room temperature; C is the equivalent system thermal capacity; and R_1 is the equivalent thermal resistance between the system borders and surroundings.

Assuming that R_1 and C are essentially constant, we rewrite the system in Eq. (14) into discrete-time form with some reasonable approximation. The system

$$y(t+1) = \mathrm{e}^{-\alpha Ts} y(k) + \frac{\frac{\beta}{\alpha}(1 - \mathrm{e}^{-\alpha Ts})}{1 + \mathrm{e}^{0.5y(k)-40}} u(k) + [1 - \mathrm{e}^{-\alpha Ts}] y_0 \tag{15}$$

is obtained, where α and β are some constant values describing R and C. The system parameters used in this example are $\alpha = 1.0015\mathrm{e}^{-4}$, $\beta = 8.67973\mathrm{e}^{-3}$ and $Y_0 = 25.0$ (°C), which were obtained from a real water bath plant in [35]. The input $u(k)$ is limited between $0v$ and $5v$ (i.e., $0v \le u(k) \le 5v$). The sampling period is $Ts = 30$. The system configuration is shown in Fig. 14, where y_{ref} is the desired temperature of the controlled plant.

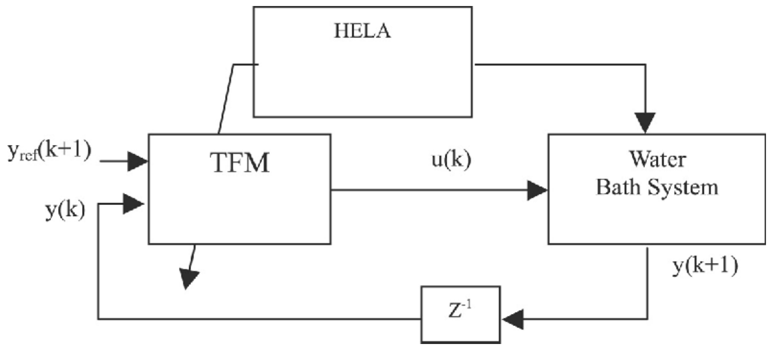

Fig. 14. Flow diagram of the TFM using the HELA for temperature control problem.

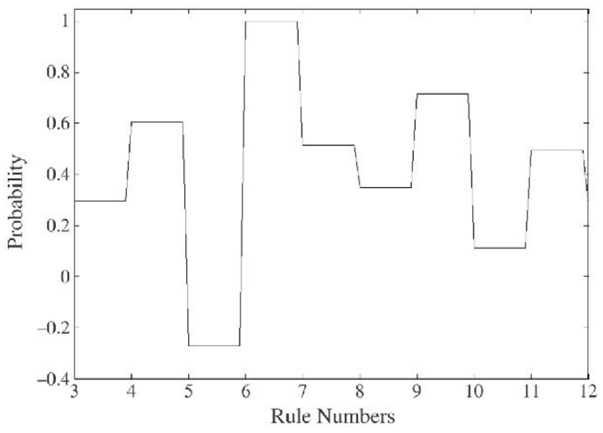

Fig. 15. Results of the probability vectors of the ERS step in the proposed HELA.

In the TFM, a sequence of random input signals $u_{rd}(k)$ limited to 0 and $5V$ is injected directly into the simulated system described in Eq. (15). The 120 training patterns are chosen from the input–outputs characteristic in order to cover the entire reference output. The initial temperature of the water is 25 °C, and the temperature rises progressively when random input signals are injected. The two input variables y_{ref} and $y(k)$ and the output $u(k)$ are normalized between 0 and 1 over the following ranges, y_{ref}: [25,85], $y(k)$: [25,85], $u(k)$: [0,5]. The evolution progressed for 1000 generations and was repeated 50 times. Fig. 15 shows the result of the probability vectors in CGA after 1000 generations. As shown in Fig. 15 we can see that the final average optimal number of rules is 6. The fuzzy rules of the TFM using the HELA method are described as follows:

$$\begin{aligned}
&\textit{Rule } 1\text{: } \textit{IF } x_1 \textit{ is } A_{11}(0.89, 0.43) \textit{ and } x_2 \textit{ is } A_{21}(0.49, 0.15)\\
&\quad \textit{THEN } y' = 13.42 - 10.04x_1 - 0.92x_2\\
&\textit{Rule } 2\text{: } \textit{IF } x_1 \textit{ is } A_{12}(0.55, 0.50) \textit{ and } x_2 \textit{ is } A_{22}(0.16, 0.98)\\
&\quad \textit{THEN } y' = -7.96 + 7.51x_1 + 5.29x_2\\
&\textit{Rule } 3\text{: } \textit{IF } x_1 \textit{ is } A_{13}(0.40, 0.78) \textit{ and } x_2 \textit{ is } A_{23}(0.12, 0.85)\\
&\quad \textit{THEN } y' = -5.37 - 6.75x_1 - 0.083x_2\\
&\textit{Rule } 4\text{: } \textit{IF } x_1 \textit{ is } A_{14}(0.91, 0.62) \textit{ and } x_2 \textit{ is } A_{24}(0.22, 0.12)\\
&\quad \textit{THEN } y' = -3.09 + 9.21x_1 - 8.85x_2\\
&\textit{Rule } 5\text{: } \textit{IF } x_1 \textit{ is } A_{15}(0.96, 0.69) \textit{ and } x_2 \textit{ is } A_{25}(0.58, 0.24)\\
&\quad \textit{THEN } y' = 19.68 - 19.71x_1 - 0.59x_2
\end{aligned}$$

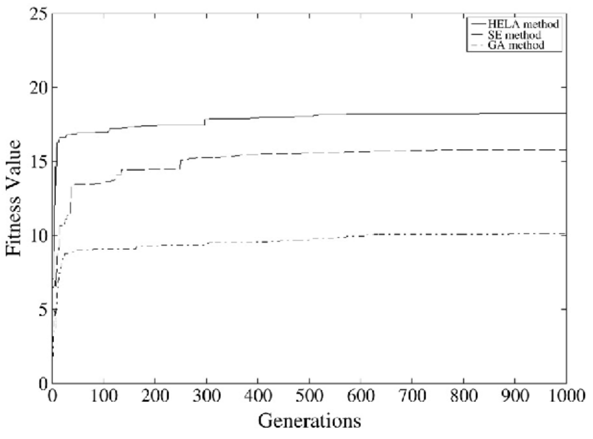

Fig. 16. The learning curves of the proposed HELA, the SE [32], and the GA [17].

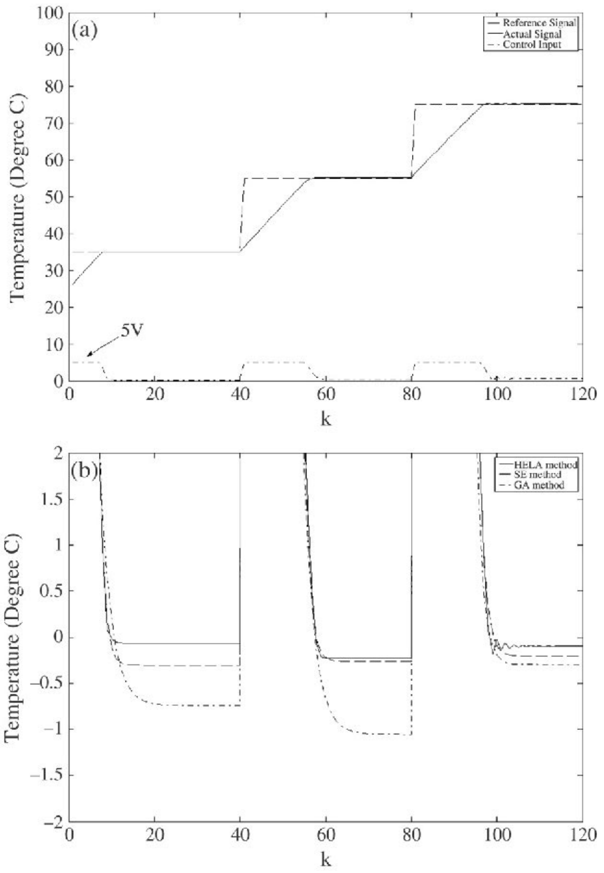

Fig. 17. (a) Final regulation performance of the TFM training by the proposed HELA for water bath system. (b) The error curves of the HELA, the SE [32], and the GA [17].

Rule 6: *IF* x_1 *is* $A_{16}(0.39, 0.30)$ *and* x_2 *is* $A_{26}(0.51, 0.70)$
THEN $y' = -9.27 - 2.41x_1 + 0.40x_2$.

In this paper, as with example 1, we also compared the performance of the HELA method with the symbiotic evolution (SE) [32] and the genetic algorithm (GA) [17]. The GA and SE used are the same as those used in example 1.

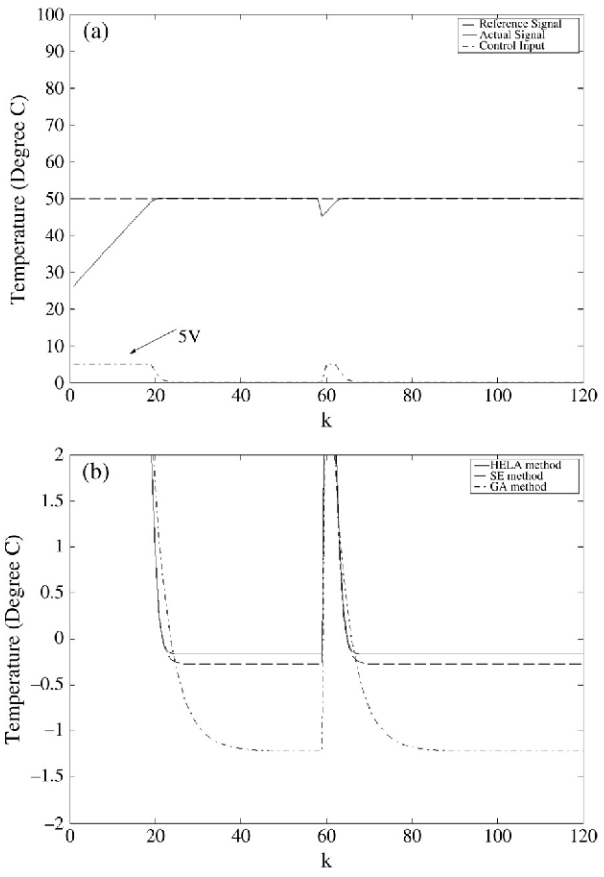

Fig. 18. Behavior of the TFM training by the proposed HELA under the impulse noise for the water bath system. (b) The error curves of the HELA, the SE [32], and the GA [17].

The evolution also progressed for 1000 generations and was repeated 50 times. Fig. 16 shows the learning curves of the four models. In Fig. 16 we can also see that the proposed HELA obtains a better fitness value than other models. To test the performance of the three models, the comparison performance measures include set-points regulation, the influence of impulse noise, and a large parameter variation in the system, and tracking capability of the controllers.

The first task is to control the simulated system to follow three set-points.

$$y_{\text{ref}}(k) = \begin{cases} 35\ ^\circ\text{C}, & \textit{for}\ k \leq 40 \\ 55\ ^\circ\text{C}, & \textit{for}\ 40 < k \leq 80 \\ 75\ ^\circ\text{C}, & \textit{for}\ 80 < k \leq 120. \end{cases} \tag{16}$$

The regulation performance of the HELA is shown in Fig. 17(a). We also test the regulation performance by using the SE [32] and the GA [17]. The error curves of the HELA, the SE, and the GA are shown in Fig. 17(b). In this figure, the HELA obtains smaller errors than the other two controllers.

The second set of simulations is carried out for the purpose of studying the noise-rejection ability of the four models when some unknown impulse noise is imposed on the process. One impulse noise value $-5\ ^\circ$C is added to the plant output at the 60th sampling instant. A set-point of 50 °C is performed in this set of simulations. The behaviors of the HELA under the influence of impulse noise and the corresponding errors are shown in Fig. 18(a)–(b).

One common characteristic of many industrial-control processes is that their parameters tend to change in an unpredictable way. To test the robustness of the four controllers, a value of $0.7 * u(k-2)$ is added to the plant input after the 60th sample in the fourth set of simulations. A set-point of 50 °C is used in this set of simulations. The behaviors of the HELA when there is a change in the plant dynamics are shown in Fig. 19(a). The corresponding errors for the HELA, the SE, and the GA are shown in Fig. 19(b).

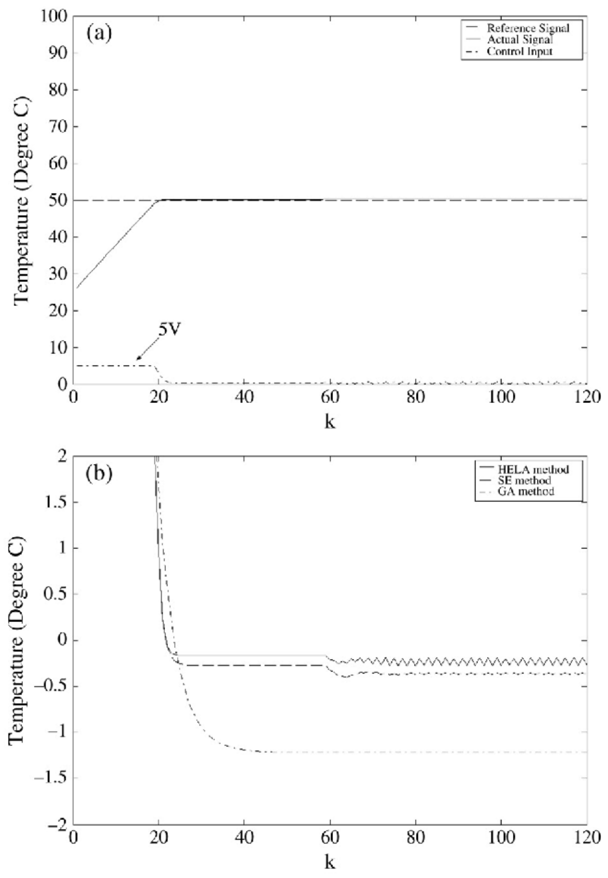

Fig. 19. (a) Behavior of the TFM training by the proposed HELA when a change occurs in the water bath system. (b) The error curves of the HELA, the SE [21], and the GA [17].

In the final set of simulations, the tracking capability of the HELA with respect to ramp-reference signals is studied. We define

$$y_{\text{ref}}(k) = \begin{cases} 34\ ^\circ\text{C} & \textit{for}\ k \le 30 \\ (34 + 0.5 * (k - 30))\ ^\circ\text{C} & \textit{for}\ 30 < k \le 50 \\ (44 + 0.8 * (k - 50))\ ^\circ\text{C} & \textit{for}\ 50 < k \le 70 \\ (60 + 0.5 * (k - 70))\ ^\circ\text{C} & \textit{for}\ 70 < k \le 90 \\ 70\ ^\circ\text{C} & \textit{for}\ 90 < k \le 120. \end{cases} \tag{17}$$

The tracking performance of the HELA is shown in Fig. 20(a). The corresponding errors for the HELA, the SE [32], and the GA [17] are shown in Fig. 20(b).

Table 3 shows the results given in the relevant literature plus the error found using TFM training by the proposed HELA. The column sum of absolute error of the table is used to test their regulation performance; a performance index, sum of absolute error, is defined by

$$\text{Sum of absolute error} = \sum_k |y_{\text{ref}}(k) - y(k)| \tag{18}$$

where $y_{\text{ref}}(k)$ and $y(k)$ are the reference output and the actual output of the simulated system, respectively. For the aforementioned simulation results, Table 3 has shown that the proposed HELA method has better performance than other methods. For the proposed HELA, however, no controller parameters have to be decided in advance.

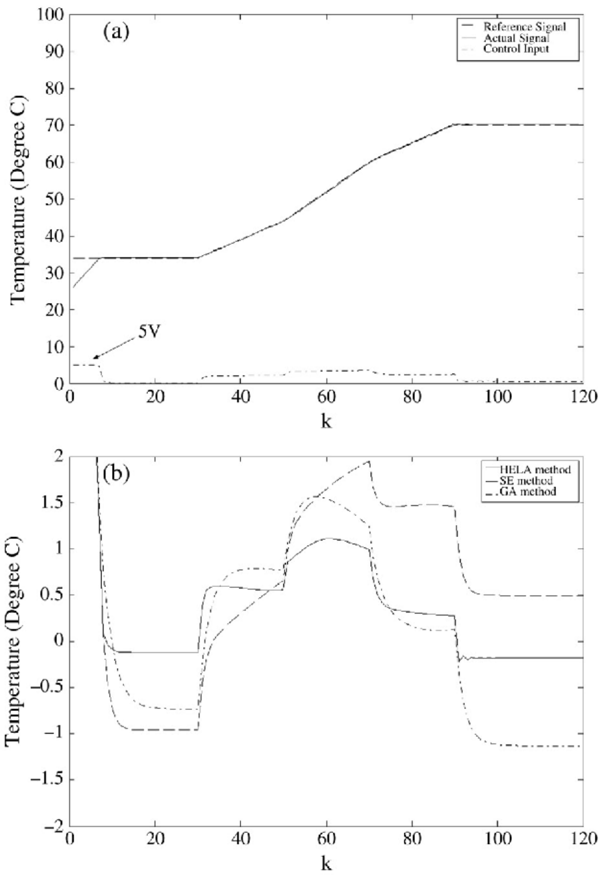

Fig. 20. The tracking of the TFM training by the proposed HELA when a change occurs in the water bath system. (b) The error curves of the HELA, the SE [32], and the GA [17].

Table 3
Performance comparison of various controllers

Sum of absolute error	**HELA**	PID	Manually designed fuzzy controller	GA [17]	SE [32]
Regulation performance	**358.07**	418.5	401.5	378.02	365.20
Influence of impulse noise	**253.93**	311.5	275.8	262.77	258.80
Effect of change in plant dynamics	**237.44**	322.2	273.5	280.38	244.14
Tracking performance	**50.71**	100.6	88.1	100.22	87.18

5. Conclusion

In this paper, a hybrid evolutionary learning algorithm (HELA), which combines the compact genetic algorithm (CGA) and the modified variable-length genetic algorithm (MVGA), was proposed for solving identification and control problems. In HELA, individuals of the same length are constituted into the same group and there are multiple groups in a population. The proposed HELA can create and train the TFM in a highly autonomous way. Thus, users need not give it any a priori knowledge or even any initial information on these. More notably, the HELA can partition input space, tune membership functions, and find proper fuzzy rules dynamically upon receiving training data. Computer simulations have shown that the proposed HELA method perform better than some existing methods.

Acknowledgement

This research is supported by the National Science Council of R.O.C. under grant NSC 94-2213-E-324- 004.

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. SMC-15 (1985) 116–132.

[6] J.-S.R. Jang, ANFIS: Adaptive-network-based fuzzy inference system, IEEE Trans. Syst. Man Cybern. 23 (1993) 665–685.

[7] C.F. Juang, C.T. Lin, An self-constructing neural fuzzy inference network and its applications, IEEE Trans. Fuzzy Syst. 6 (1) (1998) 12–31.

[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 Netw. 3 (5) (1992) 752–759.

[10] T. Bäck, H.P. Schwefel, An overview of evolutionary algorithms for parameter optimization, Evol. Comput. 1 (1) (1993) 1–23.

[11] D.B. Fogel, Evolutionary Computation: Toward a New Philosophy of Machine Intelligence, IEEE Press, Piscataway, NJ, 1995.

[12] X. Yao (Ed.), Evolutionary Computation: Theory and Applications, World Scientific, Singapore, 1999.

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

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

[15] 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.

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

[17] C.L. Karr, Design of an adaptive fuzzy logic controller using a genetic algorithm, in: Proc. Fourth Int. Conf. Genetic Algorithms, 1991, pp. 450–457.

[18] A. Homaifar, E. McCormick, Simultaneous design of membership functions and rule sets for fuzzy controllers using genetic algorithms, IEEE Trans. Fuzzy Syst. 3 (2) (1995) 129–139.

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

[20] 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.

[21] C.F. Juang, A hybrid of genetic algorithm and particle swarm optimization for recurrent network design, IEEE Trans. Syst. Man Cybern. B 34 (2) (2004) 997–1006.

[22] R. Eberchart, J. Kennedy, A new optimizer using particle swarm theory, in: Proc. of 6th Int. Sym. on Micro Machine and Human Science, Nagoya, Japan, 1995, pp. 39–43.

[23] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proc. of IEEE Int. Conf. on Neural Networks, Perth, Australia, 1995, pp. 1942–1948.

[24] S. Bandyopadhyay, C.A. Murthy, S.K. Pal, VGA-classifier: Design and applications, IEEE Trans. Syst. Man Cyber. B 30 (2000) 890–895.

[25] Z. Michalewicz, Genetic Algorithms+Data Structures=Evolution Programs, Springer-Verlag, New York, 1999.

[26] J. Arabas, Z. Michalewicz, J. Mulawka, GAVaPS—A genetic algorithm with varying population size, in: Proc. IEEE Int. Conf. on Evolutionary Computation, Orlando, 1994, pp. 73–78.

[27] R. Tanese, Distributed genetic algorithm, in: Proc. Int. Conf. Genetic Algorithms, 1989, pp. 434–439.

[28] R.J. Collins, D.R. Jefferson, Selection in massively parallel genetic algorithms, in: Proc. Int. Conf. Genetic Algorithms, 1991, pp. 249–256.

[29] D.E. Moriarty, R. Miikkulainen, Efficient reinforcement learning through symbiotic evolution, Mach. Learn. 22 (1996) 11–32.

[30] G.R. Harik, F.G. Lobo, D.E. Goldberg, The compact genetic algorithm, IEEE Trans. Evol. Comput. 3 (4) (1999) 287–297.

[31] K.Y. Lee, Xiaomin Bai, Y.M. Park, Optimization method for reactive power planning by using a modified simple genetic algorithm, IEEE Trans. Power Syst. 10 (4) (1995) 1843–1850.

[32] C.F. Juang, J.Y. Lin, C.T. Lin, Genetic reinforcement learning through symbiotic evolution for fuzzy controller design, IEEE Trans. Syst. Man Cybern. B 30 (2) (2000) 290–302.

[33] O. Cordon, F. Herrera, F. Hoffmann, L. Magdalena, Genetic fuzzy systems evolutionary tuning and learning of fuzzy knowledge bases, in: Advances in Fuzzy Systems—Applications and Theory, vol. 19, World Scientific Publishing, NJ, 2001.

[34] K.S. Narendra, K. Parthasarathy, Identification and control of dynamical systems using neural networks, IEEE Trans. Neural Netw. 1 (1) (1990) 4–27.

[35] J. Tanomaru, S. Omatu, Process control by self-constructing trained neural controllers, IEEE Trans. Ind. Electron. 39 (1992) 511–521.

[36] C.-J. Lin, C.-N. Tsai, Using context-sensitive neuro-fuzzy system for nonlinear system identification, J. Chin. Inst. Eng. 27 (1) (2004) 1–8.