

A novel genetic reinforcement learning for nonlinear fuzzy control problems

Cheng-Jian Lin*, Yong-Ji Xu

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

Received 21 September 2004; received in revised form 11 May 2005; accepted 23 September 2005

Available online 7 February 2006

Communicated by M.-J. Er

Abstract

Unlike a supervise learning, a reinforcement learning problem has only very simple “evaluative” or “critic” information available for learning, rather than “instructive” information. A novel genetic reinforcement learning, called reinforcement sequential-search-based genetic algorithm (R-SSGA), is proposed for solving the nonlinear fuzzy control problems in this paper. Unlike the traditional reinforcement genetic algorithm, the proposed R-SSGA method adopts the sequential-search-based genetic algorithms (SSGA) to tune the fuzzy controller. Therefore, the better chromosomes will be initially generated while the better mutation points will be determined for performing efficient mutation. The adjustable parameters of fuzzy controller are coded as real number components. We formulate a number of time steps before failure occurs as a fitness function. Simulation results have shown that the proposed R-SSGA method converges quickly and minimizes the population size.

© 2006 Elsevier B.V. All rights reserved.

Keywords: Reinforcement learning; Genetic algorithm; Fuzzy system; Nonlinear control; Efficient mutation; Sequential-search

1. Introduction

In recent years, the concept of the fuzzy logic or artificial neural networks for control problems has grown into a popular research area [17,24,27]. The reason is that classical control theory usually requires a mathematical model for designing controllers. The inaccuracy of mathematical modeling of plants usually degrades the performance of the controllers, especially for nonlinear and complex control problems [2,11]. Fuzzy logic has the ability to express the ambiguity of human thinking and translate expert knowledge into computable numerical data.

A fuzzy system consists of a set of fuzzy IF–THEN rules that describe the input–output mapping relationship of the networks. Obviously, it is difficult for human experts to examine all the input–output data from a complex system

to find proper rules for a fuzzy system. To cope with this difficulty, several approaches that are used to generate the fuzzy IF–THEN rules from numerical data have been proposed [1,3,15,21]. These methods were developed for supervised learning; i.e., the correct “target” output values are given for each input pattern to guide the learning of the network. However, most of the supervised learning algorithms for neural fuzzy networks require precise training data to tune the networks for various applications. For some real world applications, precise training data are usually difficult and expensive, if not impossible, to obtain. For this reason, there has been a growing interest in reinforcement learning algorithms for use in fuzzy [22,30] or neural controller [10,29] design.

In the design of a fuzzy controller, adjusting the required parameters is important. To do this, back-propagation (BP) training was widely used in [10,23,29]. It is a powerful training technique that can be applied to networks with a forward structure. Since the steepest descent technique is used in BP training to minimize the error function, the

*Corresponding author.

E-mail address: cjlin@mail.cyut.edu.tw (C.-J. Lin).

algorithms may reach the local minima very fast and never find the global solution.

The development of genetic algorithms (GAs) has provided another approach for adjusting parameters in the design of controllers. GA is a parallel and global technique [16,22]. Because it simultaneously evaluates many points in a search space, it is more likely to converge toward the global solution. Some researchers have developed methods to design and implement fuzzy controllers by using GAs. Karr [17] used a GA to generate membership functions for a fuzzy system. In Karr’s work, a user needs to declare an exhaustive rule set and then use a GA to design only the membership functions. In [12], a fuzzy controller design method that used a GA to find the membership functions and the rule sets simultaneously was proposed. Lin [19] proposed a hybrid learning method which combines the GA and the least-squares estimate (LSE) method to construct a neural fuzzy controller. In [12,19], the input space was partitioned into a grid. The number of fuzzy rules (i.e., the length of each chromosome in the GA) increased exponentially as the dimension of the input space increased. To overcome this problem, Juang [14] adopted a flexible partition approach in the precondition part. The method has the admirable property of small network size and high learning accuracy.

Recently, some researchers [9,16,18,22] applied GA methods to implement reinforcement learning in the design of fuzzy controllers. Lin and Jou [22] proposed GA-based fuzzy reinforcement learning to control magnetic bearing systems. In [16], Juang and his colleagues proposed genetic reinforcement learning in designing fuzzy controllers. The GA adopted in [16] was based upon traditional symbiotic evolution which, when applied to fuzzy controller design, complements the local mapping property of a fuzzy rule. In [9], Er and Deng proposed dynamic Q-Learning for on-line tuning the fuzzy inference systems. Kaya and Alhaji [18] proposed a novel multiagent reinforcement learning approach based on fuzzy OLAP association rules mining. However, these approaches encountered one or more of the following major problems: (1) the initial values of the populations were generated randomly; (2) the mutational value was generated by the constant range while the mutation point is also generated randomly; (3) the population sizes always depend on the problem which is to be solved.

In this paper, we propose a reinforcement sequential-search-based genetic algorithm (R-SSGA) method to solve above-mentioned problems. Unlike the traditional reinforcement learning, in this paper, we formulate a number of time steps before failure occurs as the fitness function. The new sequential-search-based genetic algorithm (SSGA) is also proposed to perform parameter learning. Moreover, the SSGA method is different from traditional GA, which the better chromosomes will be initially generated while the better mutation points will be determined for performing efficient mutation. Compared with traditional GA, the SSGA method generates initialize population efficiently

and decides efficient mutation points to perform mutation. The advantages of the proposed R-SSGA method are summarized as follows: (1) The R-SSGA method can reduce the population sizes to a minimum size (4); (2) The chromosome which has the best performance will be chosen to perform the mutation operator in each generation; (3) The R-SSGA method converges more quickly than existing traditional genetic methods.

This paper is organized as follows. Section 2 introduces the SSGA. A R-SSGA is presented in Section 3. In Section 4, the proposed R-SSGA method is evaluated using two different control problems, and its performances are benchmarked against other structures. Finally, conclusions on the proposed algorithm are summarized in the last section.

2. The sequential-search-based genetic algorithm

A new genetic learning algorithm, called SSGA, is proposed to adjust the parameters for the desired outputs. The proposed SSGA method is different from a traditional GA [16,22]. The SSGA method generates initial population efficiently and decides efficient mutation points to perform mutation. Like traditional GA [16,22], the proposed SSGA method consists of two major operators: reproduction, crossover. Before the details of these two operators are explained, coding, initialization and efficient mutation are discussed as follows:

(a) *Coding step*: The first step in the SSGA method is to code a fuzzy controller into a chromosome. In this paper, we adopt a Takagi–Sugeno–Kang (TSK) type fuzzy controller [26] to be the structure of the proposed SSGA method. A TSK-type fuzzy controller employs different implication and aggregation methods than the standard Mamdani controller [24]. Instead of using fuzzy sets the conclusion part of a rule, is a linear combination of the crisp inputs.

$$\begin{aligned} \text{IF } x_1 \text{ is } A_{1j} (m_{1j}, \sigma_{1j}) \quad \text{and } x_2 \text{ is } A_{2j}(m_{2j}, \sigma_{2j}) \dots \text{and } x_n \text{ is } \\ A_{nj}(m_{nj}, \sigma_{nj}) \\ \text{THEN } y' = w_0 + w_1x_1 + \dots + w_ix_i, \end{aligned} \tag{1}$$

where m_{ij} and σ_{ij} represent a Gaussian membership function with mean and deviation with i th dimension and j th rule node. A fuzzy rule in Fig. 1 is represented the form in Eq. (1).

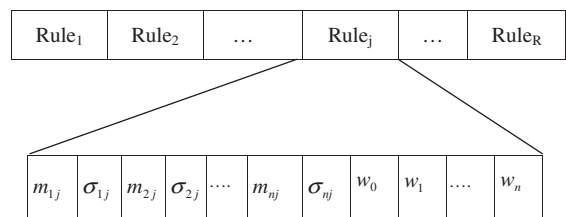


Fig. 1. Coding a fuzzy controller into a chromosome in the SSGA method.

(b) *Initialization step*: Before the SSGA method is designed, individuals forming an initial population should be generated. Unlike traditional GA, an initial population is generated randomly within a fixed range. In the SSGA method, the initial population is generated efficiently to ensure that chromosomes with good genes can be generated. The detailed steps of the initialization method are described as follows:

Step 0: The first chromosome that represents a TSK-type fuzzy controller will be generated initially. The following formulations show how to generate the chromosomes:

$$\text{Deviation: } Chr_j[p] = \text{random}[\sigma_{\min}, \sigma_{\max}],$$

$$\text{where } p = 2, 4, 6, \dots, 2n, \quad (2)$$

$$\text{Mean: } Chr_j[p] = \text{random}[m_{\min}, m_{\max}],$$

$$\text{where } p = 1, 3, 5, \dots, 2n - 1, \quad (3)$$

$$\text{Weight: } Chr_j[p] = \text{random}[w_{\min}, w_{\max}],$$

$$\text{where } p = 2n + 1, \dots, 2n + (1 + n), \quad (4)$$

where Chr_j means chromosome in i th rule and p represent the p th gene in a Chr_j ; $[\sigma_{\min}, \sigma_{\max}]$, $[m_{\min}, m_{\max}]$, and $[w_{\min}, w_{\max}]$ represent the predefined ranges of deviation, mean, and weight. The ranges are determined by practical experimentation or trial-and-error tests.

Step 1: To generate the other chromosomes, we use the SSGA method to generate the new chromosomes. The search algorithm of the SSGA method is similar to the local search procedure in [13]. In the SSGA method, every gene in the previous chromosomes is selected using a sequential search and the gene's value is updated to evaluate the performance based on the fitness value. The details of the SSGA method are as follows:

(a) Sequentially search for a gene in the previous chromosome.

(b) Update the chosen gene in (a) according to the following formula:

$$Chr_j[p] = \begin{cases} Chr_j[p] + \Delta(\text{fitness_value}, \\ \sigma_{\max} - Chr_j[p]), & \text{if } \alpha > 0.5, \\ Chr_j[p] - \Delta(\text{fitness_value}, \\ Chr_j[p] - \sigma_{\min}), & \text{if } \alpha < 0.5, \end{cases}$$

$$\text{where } p = 2, 4, 6, \dots, 2n, \quad (5)$$

$$Chr_j[p] = \begin{cases} Chr_j[p] + \Delta(\text{fitness_value}, \\ m_{\max} - Chr_j[p]), & \text{if } \alpha > 0.5, \\ Chr_j[p] - \Delta(\text{fitness_value}, \\ Chr_j[p] - m_{\min}), & \text{if } \alpha < 0.5, \end{cases}$$

$$\text{where } p = 1, 3, 5, \dots, 2n - 1, \quad (6)$$

$$Chr_j[p] = \begin{cases} Chr_j[p] + \Delta(\text{fitness_value}, \\ w_{\max} - Chr_j[p]), & \text{if } \alpha > 0.5, \\ Chr_j[p] - \Delta(\text{fitness_value}, \\ Chr_j[p] - w_{\min}), & \text{if } \alpha < 0.5, \end{cases}$$

$$\text{where } p = 2n + 1, \dots, 2n + (1 + n), \quad (7)$$

$$\text{where } \Delta(\text{fitness_value}, v) = v\lambda(1/\text{fitness_value})^2, \quad (8)$$

where $\alpha, \lambda \in [0, 1]$ are the random values; fitness_value is the fitness computed using Eq. (11); p represents the p th gene in a chromosome; j represents the j th rule, respectively. The function $\Delta(\text{fitness_value}, v)$ returns a value, such that $\Delta(\text{fitness_value}, v)$ comes close to 0 as fitness_value increases. This property causes the mutation operator to search the space uniformly during the initial stage (when fitness_value is small) and locally during the later stages, thus increasing the probability of generating children closer to its successor than a random choice and reducing the number of generations.

(c) If the new gene that is generated from (b) can improve the fitness value, then replace the old gene with the new gene in the chromosome. If not, recover the old gene in the chromosome. After this, go to (a) until every gene is selected. The pseudo-code for the SSGA method is listed in Fig. 2. The $Chr_{k,j}$ represents the k th chromosome and j th rule in a fuzzy controller. And N_f denote the size of the population, $\text{fitness}(Chr_{k,j_new})$ is a fitness function by Eq. (11) using the k th new chromosome.

Step 2: If no genes are selected to improve the fitness value in Step 1, then the new chromosome will be generated according to Step 0. After the new chromosome is generated, the initialization method returns to Step 1 until the total number of chromosomes is generated.

In this paper, the firing strength of a fuzzy rule is calculated by performing the “AND” operation on the truth values of each variable to its corresponding

Procedure Sequential-Search-Based Genetic Algorithm

```

Begin
  Let  $p=0, i=0$ ;
  Repeat
     $k=k+1$ ;
    Repeat
       $j=j+1$ ;
      Repeat
         $p=p+1$ ;
        Perform  $Chr_{k,j\_new} = \text{initialize}(Chr_{k,j\_old}[p])$ ; by (5) to (8);
        Evaluate  $\text{fitness}(Chr_{k,j\_new})$  and  $\text{fitness}(Chr_{k,j\_old})$  by (11);
        If  $\text{fitness}(Chr_{k,j\_new}) > \text{fitness}(Chr_{k,j\_old})$  Then
           $Chr_{k,j\_old} = Chr_{k,j\_new}$ ; else  $Chr_{k,j\_new} = Chr_{k,j\_old}$ ;
        Until  $p=2*n+(1+n)$ ;
      Until  $j=R$ ;
    Until  $k=N_f$ ;
End

```

Fig. 2. The pseudo-code for the SSGA method.

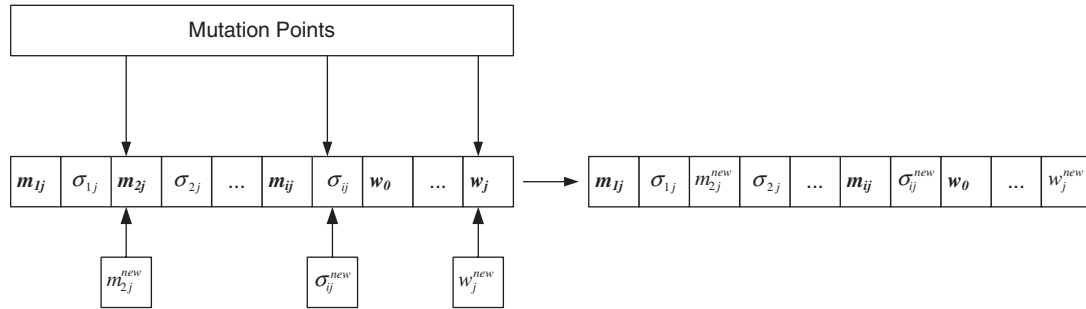


Fig. 3. Efficient mutation operation using 3 mutation points with j th rule.

fuzzy sets,

$$u_j = \prod_{i=1}^n \exp\left(-\frac{[u_i^{(1)} - m_{ij}]^2}{\sigma_{ij}^2}\right), \quad (9)$$

where m_{ij} and σ_{ij} are, respectively, the center and the width of the Gaussian membership function of the j th term of the i th input variable x_i . The output of a fuzzy system is computed by

$$u_{out} = \frac{\sum_j u_j \sum_{i=0}^n w_{ij} x_i}{\sum_j u_j}, \quad (10)$$

where the weight w_j is the output action strength associated with the j th rule and u_{out} is the output of the network.

(c) *Efficient mutation step*: Although reproduction and crossover will produce many new strings, they do not introduce any new information to the population at the site of an individual. Mutation is an operator that randomly alters the allele of a gene. In this paper, we propose using efficient mutation, which is unlike the traditional mutation, to mutate the chromosomes. In the SSGA method, we perform efficient mutation using the best fitness value chromosome of every generation. And we use the SSGA method to decide on the mutation points. When the mutation points are selected, we use Eqs. (5)–(7) to update the genes. The efficient mutation of an individual is shown in Fig. 3.

(d) *Reproduction step*: Reproduction is a process in which individual strings are copied according to their fitness value. In this study, we use the roulette-wheel selection method [8]—a simulated roulette is spun—for this reproduction process. The best performing individuals in the top half of the population [16] advances to the next generation. The other half is generated to perform crossover and mutation operations on individuals in the top half of the parent generation.

(e) *Crossover step*: Reproduction directs the search toward the best existing individuals but does not create any new individuals. In nature, an offspring has two parents and inherits genes from both. The main operator working on the parents is the crossover operator, the operation of which occurred for a selected pair with a crossover rate that was set to 0.5 in this study. The first step is to select the individuals from the population for the

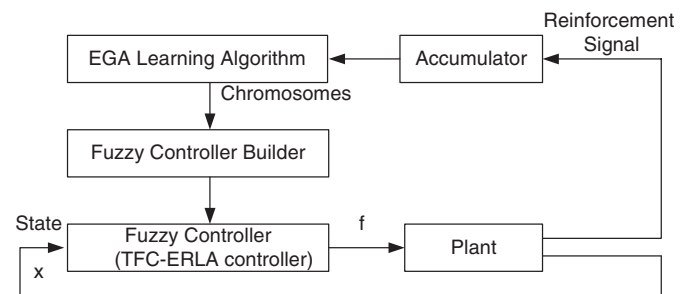


Fig. 4. The proposed R-SSGA method.

crossover. Tournament selection [8] is used to select the top-half of the best performing individuals [16]. The individuals are crossed and separated using a two-point crossover that is the new individuals are created by exchanging the site’s values between the selected sites of parents’ individual. After this operation, the individuals with poor performances are replaced by the newly produced offspring.

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

3. Reinforcement sequential-search-based genetic algorithm (R-SSGA)

Unlike the supervised learning problem, in which the correct “target” output values are given for each input pattern to perform fuzzy controller learning, 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. 4 shows how the R-SSGA method and its training environment interact in a reinforcement learning problem. The environment supplies a time-varying input vector to the R-SSGA method, receives its time-varying output/action vectors and then provides a reinforcement signal. In this paper, the reinforcement signal indicates whether a success or a failure occurs.

As show in Fig. 4, the R-SSGA method consists of a TSK-type fuzzy controller which acts as the control

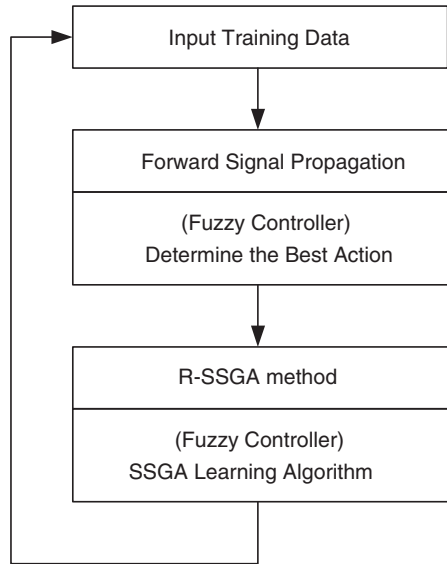


Fig. 5. Flowchart of the R-SSGA method.

Table 1
The initial parameters before training

Parameters	Value
Population size	4
Crossover rate	0.5
Coding type	Real number
$[\sigma_{\min}, \sigma_{\max}]$	[0,1]
$[m_{\min}, m_{\max}]$	[0,1]
$[w_{\min}, w_{\max}]$	[-20, 20]

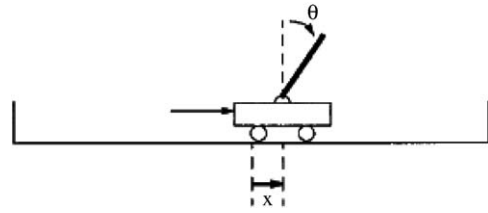


Fig. 6. The cart–pole balancing system.

network to determine a proper action according to the current input vector (environment state). The structure of the R-SSGA method is different from Barto and his colleagues' actor-critic architecture [4,5]. Two neuron-like adaptive elements are integrated in this system [4,5]. They are the associative search element (ASE) used as a controller, and the adaptive critic element (ACE) used as a predictor. Temporal difference techniques and single-parameter stochastic exploration are used in [4]. The input to the R-SSGA method is the state of the plant, and the output is a control action of the state, denoted by f . The only available feedback is a reinforcement signal that notifies the R-SSGA method only when a failure occurs. An accumulator plays a role which is a relative performance measure shown in Fig. 4. It accumulates the number of time steps before a failure occurs [20]. 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-SSGA method. That is, the accumulator will indicate the “fitness” of the current R-SSGA method. The key to this learning algorithm is formulating a number of time steps before failure occurs and using this formulation as the fitness function of the R-SSGA method. The advantage of the proposed method need not use the critical network as either a multi-step or single-step predictor.

Fig. 5 shows the flowchart of the R-SSGA method. The R-SSGA method runs in a feedforward fashion to control the environment (plant) until a failure occurs. Our relative measure of 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 high fitness values means good fit. In this paper, we use a number of time steps before failure occurs to define the fitness function. The fitness function is

defined by

$$fitness_value(i) = TIME-STEP(i), \quad (11)$$

where $TIME-STEP(i)$ represents how long the experiment is still a “success” about the i th population. Eq. (11) reflects the fact that long-time steps before failure occurs (to keep the desired control goal longer) mean higher fitness of the R-SSGA method.

4. Illustrative examples

To verify the performance of the proposed R-SSGA method, two control examples—the cart–pole balancing system and a water bath temperature control system—are presented in this section. For the two computer simulations, the initial parameters are given in Table 1 before training.

4.1. Example 1. Cart–pole balancing system

In this example, we shall apply the R-SSGA method to the classic control problem of the cart–pole balancing. This problem is often used as an example of inherently unstable and dynamic systems to demonstrate both modern and classic control techniques [6,7] or reinforcement learning schemes [16,23], and is now used as a control benchmark. As shown in Fig. 6, the cart–pole balancing problem is the problem of learning how to balance an upright pole. The bottom of the pole is hinged to the left or right of a cart that travels along a finite-length track. 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 in 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 m/s). The only control action is f , which is the amount of force (in N) applied to the cart to move it left or right. The system fails when the pole falls past a certain angle ($\pm 12^\circ$ is used here) or when the cart runs into the boundary of the track (the distance is 2.4 m from the center to each boundary of the track). The goal of this control problem is to determine a sequence of forces that, when applied to the cart, balance the pole so that it is upright. The motion equations that we used were:

$$\theta(t+1) = \theta(t) + \Delta \dot{\theta}(t), \quad (12)$$

$$\begin{aligned} \dot{\theta}(t+1) = \dot{\theta}(t) + \Delta & \frac{(m+m_p)g \sin \theta(t)}{(4/3)(m+m_p)l - m_p l \cos^2 \theta(t)} \\ & - \frac{\cos \theta(t)[f(t) + m_p l \dot{\theta}(t)^2 \sin \theta(t) - \mu_c \operatorname{sgn}(\dot{x}(t))]}{(4/3)(m+m_p)l - m_p l \cos^2 \theta(t)} \\ & - \frac{\mu_p(m+m_p)\dot{\theta}(t)}{m_p l} \\ & - \frac{m_p l}{(4/3)(m+m_p)l - m_p l \cos^2 \theta(t)}, \end{aligned} \quad (13)$$

$$x(t+1) = x(t) + \Delta \dot{x}(t), \quad (14)$$

$$\begin{aligned} \dot{x}(t+1) = \dot{x}(t) + \Delta & \frac{f(t) + m_p l [\dot{\theta}(t)^2 \sin \theta(t) - \ddot{\theta}(t) \cos \theta(t)]}{(m+m_p)} \\ & - \frac{\mu_c \operatorname{sgn}(\dot{x}(t))}{(m+m_p)}, \end{aligned} \quad (15)$$

where

$l = 0.5$ m, the length of the pole;

$m = 1.1$ kg, combined mass of the pole and the cart;

$m_p = 0.1$ kg, mass of the pole;

$g = 9.8$ m/s, acceleration due to the gravity;

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

$\mu_p = 0.000002$, coefficient of friction of the pole on the cart;

$\Delta = 0.02$ (s), sampling interval. (16)

The constraints on the variables were $-12^\circ \leq \theta \leq 12^\circ$, $-2.4 \text{ m} \leq x \leq 2.4 \text{ m}$, and $-10 \text{ N} \leq f \leq 10 \text{ N}$. A control strategy was deemed successful if it balanced 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 0 and 1 over the following ranges, $\theta \in [-12, 12]$, $\dot{\theta} \in [-60, 60]$, $x \in [-2.4, 2.4]$, $\dot{x} \in [-3, 3]$, $f_t \in [-10, 10]$. The fitness function in this example is defined in Eq. (11) to train the R-SSGA method where Eq. (11) is used to calculate how long it takes the cart–pole balancing system to fail and receives a penalty signal of -1 when the pole falls past a certain angle ($|\theta| > 12^\circ$) and when the cart runs into the boundaries of the tracks falls ($|x| > 2.4$ m). In this experiment, the initial values were set to (0, 0, 0, 0). And we set four rules constitute a TSK-Type fuzzy controller.

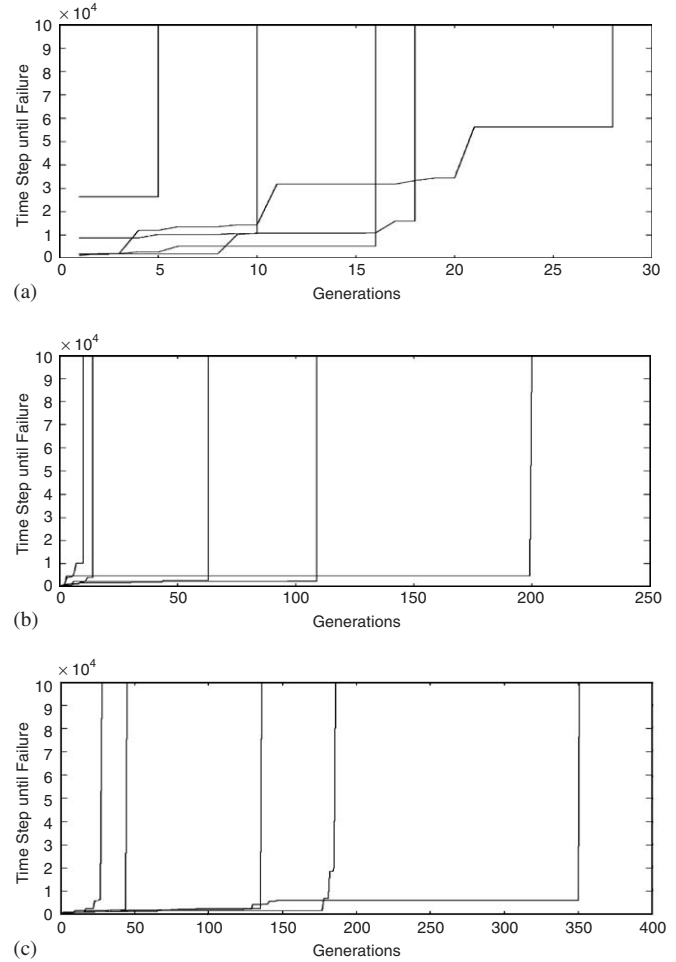


Fig. 7. The performance of (a) the R-SSGA method, (b) the SEFC method [16], and (c) the TGFC method [22] on the cart–pole balancing system.

A total of five runs were performed. Each run started at the same initial state. The simulation result in Fig. 7(a) shows that the R-SSGA method learned on average to balance the pole at the 16th generation. In this figure, each run indicates that the largest fitness value in the current generation was selected before the cart–pole balancing system failed. When the proposed R-SSGA learning method is stopped, we choose the best string in the population in the final generation and tested it on the cart–pole balancing system. The final fuzzy rules generated by the R-SSGA method are described as follows:

Rule 1: IF x_1 is $A_{11}(0.38, 0.35)$ and x_2 is $A_{21}(5.67, 0.32)$ and x_3 is $A_{31}(0.19, 1.91)$ and x_4 is $A_{41}(0.40, 0.825)$

THEN $y' = -2.94 + 0.42x_1 - 0.20x_2 - 0.70x_3 + 0.40x_4$.

Rule 2: IF x_1 is $A_{12}(0.52, 1.70)$ and x_2 is $A_{22}(7.43, 0.39)$ and x_3 is $A_{32}(0.37, 14.9)$ and x_4 is $A_{42}(1.28, 0.44)$

THEN $y' = 12.21 + 12.16x_1 - 0.25x_2 + 0.32x_3 + 4.66x_4$.

Rule 3: IF x_1 is $A_{13}(0.52, 6.66)$ and x_2 is $A_{23}(12.1, 0.39)$ and x_3 is $A_{33}(0.37, 9.64)$ and x_4 is $A_{43}(1.28, 0.44)$

THEN $y' = 11.93 + 9.63x_1 - 0.25x_2 + 0.32x_3 + 9.64x_4$.

Rule 4: IF x_1 is $A_{14}(0.52, 17)$ and x_2 is $A_{24}(9.29, 0.39)$ and x_3 is $A_{34}(0.37, 3.98)$ and x_4 is $A_{44}(1.28, 0.44)$

THEN $y' = 11.93 - 3.98x_1 - 0.25x_2 + 0.32x_3 + 10.29x_4$.

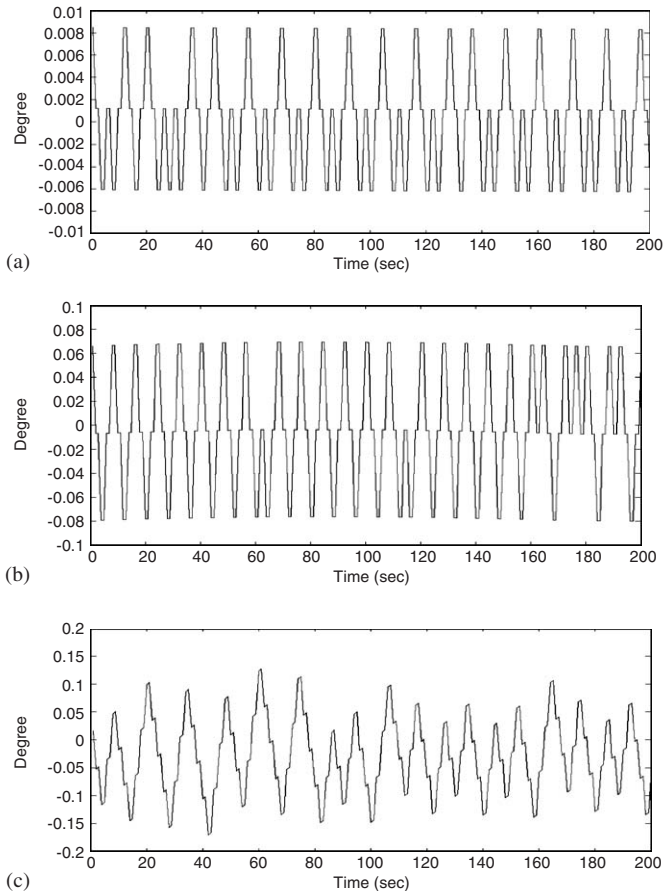


Fig. 8. Angular deviation of the pole by a trained (a) the R-SSGA method, (b) the SEFC method [16], and (c) the TGFC method [22].

Fig. 8(a) show the angular deviation of the pole when the cart–pole balancing system was controlled by the well-trained R-SSGA method starting at the initial state: $r(0) = 0, \dot{r}(0) = 0, \theta(0) = 0, \dot{\theta}(0) = 0$. The average angular deviation was 0.006° .

In this paper, we compare the performance of our system with the symbiotic evolution fuzzy controller (SEFC) [16] and the traditional genetic fuzzy controller (TGFC) [22]. In the SEFC and TGFC, the population sizes were also set to 50, and the crossover and mutation probabilities were set to 0.5 and 0.3, respectively. Figs. 7(b) and (c) show that the SEFC method and the TGFC method learned on average to balance the pole at the 80th and 149th generation. In this example, we compare the CPU times of the R-SSGA method with the SEFC and the TGFC methods. Table 2 shows the CPU times of the three methods. As shown in Table 2, our method obtains shorter CPU times than the SEFC and the TGFC methods. Figs. 8(b) and (c) show the angular deviation of the pole when the cart–pole balancing system was controlled by Juang et al. [16] and Lin and Jo [22] models. The average angular deviation of Juang et al. [16] and Lin and Jo [22] models were 0.06° and 0.1° . We also try to control the cart–pole balancing system at a different initial state: $r(0) = 0.6, \dot{r}(0) = 0.3, \theta(0) = 3, \dot{\theta}(0) = 1$.

Table 2

Performance comparison of the R-SSGA, the SEFC, and the TGFC methods

Method	Mean (s)	Best (s)	Worst (s)
R-SSGA	20	3	60
SEFC [16]	36	4	236
TGFC [22]	165	8	412

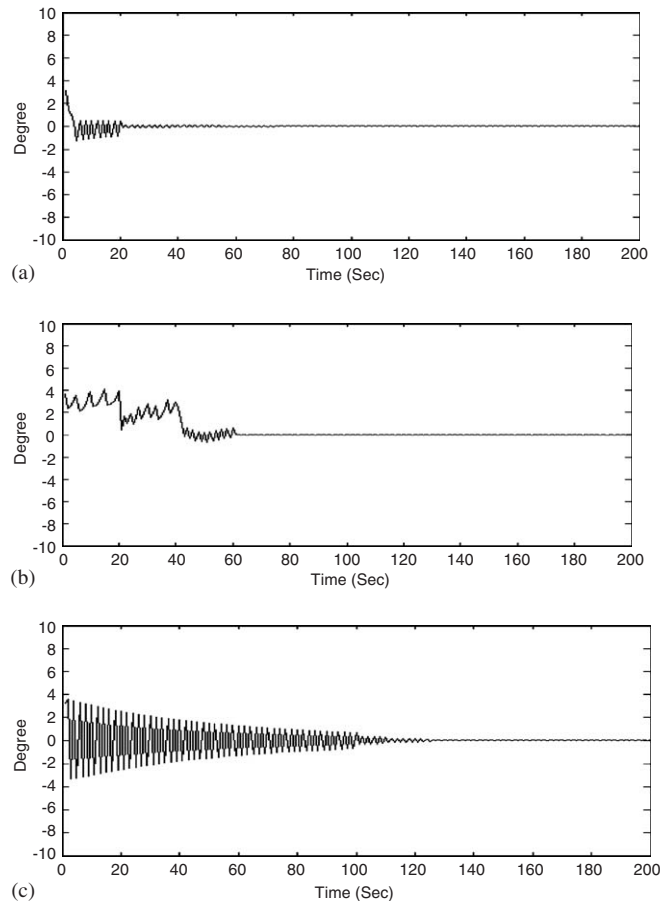


Fig. 9. Angular deviation of the pole by a trained (a) the R-SSGA method, (b) the SEFC method [16], and (c) the TGFC method [22] at the initial state: $r(0) = 0.6, \dot{r}(0) = 0.3, \theta(0) = 3, \dot{\theta}(0) = 1$.

Fig. 9(a)–(c) shows the angular deviation of the pole when the cart–pole balancing system was controlled by the R-SSGA, the SEFC [16], and the TGFC [22] models at the initial state: $r(0) = 0.6, \dot{r}(0) = 0.3, \theta(0) = 3, \dot{\theta}(0) = 1$.

Table 3 shows the number of pole-balance trials (which reflects the number of training episodes required) measured. The GENITOR [28] and symbiotic adaptive neuro-evolution (SANE) [25] were applied to the same control problem, and the simulation results are listed in Table 3. In GENITOR, 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 has five input units, five

Table 3
Performance comparison of various existing models in Example 1

Method	Mean	Best	Worst
GENITOR [28]	2578	415	12964
SANE [25]	1691	46	4461
TGFC [22]	80	26	200
SEFC [16]	149	10	350
R-SSGA	17	5	29

hidden units and one output unit. In SANE, the traditional symbiotic evolution algorithm was used to evolve a two-layer neural network with five input units, eight hidden units, and two output units. An individual in SANE represents a hidden unit with five specified connections to the input and output units. In Table 3 we can see that the proposed method is feasible and effective. And the proposed R-SSGA method only took four rules and the population size was 4.

In this example, to verify the performance of our proposed method, we use five different initial states for the R-SSGA, the SEFC, and the TGFC methods. The five different initial states are shown as follows:

$$S1: r(0) = 0.8, \dot{r}(0) = 0.2, \theta(0) = 8, \dot{\theta}(0) = 3,$$

$$S2: r(0) = 0.3, \dot{r}(0) = 0.1, \theta(0) = 2, \dot{\theta}(0) = 0,$$

$$S3: r(0) = 0.5, \dot{r}(0) = 0.1, \theta(0) = 4, \dot{\theta}(0) = 2,$$

$$S4: r(0) = 0.7, \dot{r}(0) = 0.4, \theta(0) = 6, \dot{\theta}(0) = 3,$$

$$S5: r(0) = 0.2, \dot{r}(0) = 0.1, \theta(0) = 2, \dot{\theta}(0) = 1.$$

Fig. 10(a)–(c) show that the R-SSGA, the SEFC, and the TGFC methods learned on average to balance the pole at the 78th, 105th, and 166th generation. Fig. 11(a)–(c) show the angular deviation of the pole when the cart–pole balancing system was controlled by the R-SSGA method, the SEFC method [16], and the TGFC method [22] that starting at the initial state: $r(0) = 0, \dot{r}(0) = 0, \theta(0) = 0, \dot{\theta}(0) = 0$. The average angular deviations were 0.01° , 0.04° , and 0.08° . Table 4 shows the number of pole-balance trials measured of the R-SSGA, the SEFC [16], and the TGFC [22] methods. In Table 4, we see that the proposed method obtains a better performance than some existing methods [16,22].

4.2. Example 2. Water bath temperature control system

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

$$\frac{dy(t)}{dt} = \frac{u(t)}{C} + \frac{Y_0 - y(t)}{RC}, \quad (17)$$

where $y(t)$ is the system output temperature in $^\circ\text{C}$; $u(t)$ is the heat flowing into the system; Y_0 is the room temperature; C is the equivalent system thermal capacity;

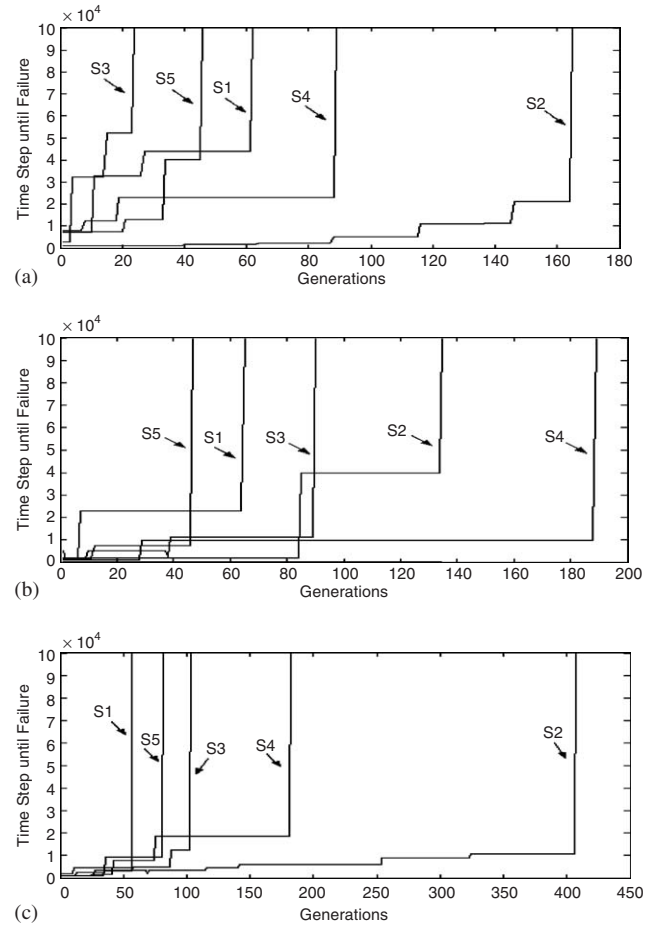


Fig. 10. The performance of (a) the R-SSGA method, (b) the SEFC method [16], and (c) the TGFC method [22] on the cart–pole balancing system starting at five different initial states.

and R is the equivalent thermal resistance between the system borders and the surroundings.

Assuming that R and C are essentially constant, we rewrite the system in Eq. (17) into discrete-time form with some reasonable approximation. The system

$$y(t+1) = e^{-\alpha T_s} y(k) + \frac{(\beta/\alpha)(1 - e^{-\alpha T_s})}{1 + e^{0.5y(k)-40}} u(k) + [1 - e^{-\alpha T_s}] y_0 \quad (18)$$

is obtained, where α and β are constant values describing R and C . The system parameters used in this example were $\alpha = 1.0015e^{-1}$, $\beta = 8.67973e^{-3}$, and $Y_0 = 25.0(^\circ\text{C})$, which were obtained from a real water bath plant in [27]. The input $u(k)$ was limited to 0, and the voltage was 5 V. The sampling period was $T_s = 30$. The system configuration is shown in Fig. 12, where y_{ref} was the desired temperature of the controlled plant.

In this example, y_{ref} and $y(k)$ and the output $u(k)$ were normalized between 0 and 1 over the following ranges: $y_{\text{ref}} : [25, 85]$, $y(k) : [25, 85]$, and $u(k) : [0, 5]$. The values of floating-point numbers were initially assigned using the R-SSGA method initially. The fitness function was set for each reassigned regulation temperature $T = 35, 55, \text{ and } 75$,

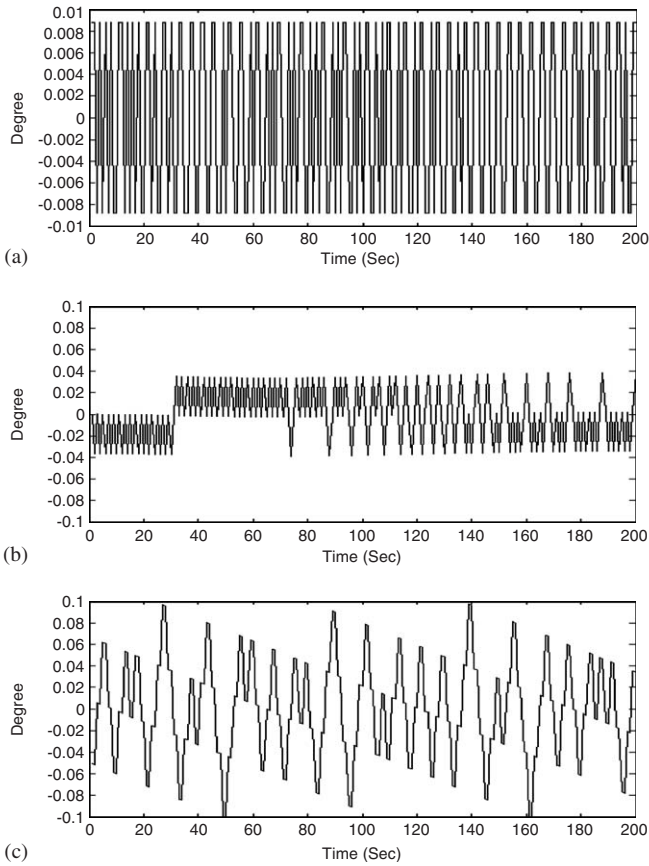


Fig. 11. Angular deviation of the pole by a trained (a) the R-SSGA method, (b) the SEFC method [16], and (c) the TGFC method [22].

Table 4
Performance comparison of existing models in Example 1

Method	Mean	Best	Worst
TGFC [22]	166	57	407
SEFC [16]	105	47	189
R-SSGA	78	24	165

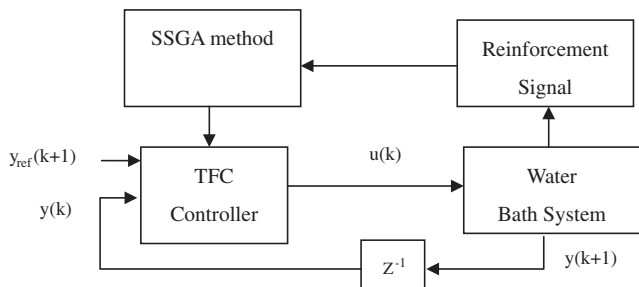


Fig. 12. Flow diagram of using the R-SSGA method for solving the temperature control problem.

starting from the current temperature and again after 10 time steps. The control temperature error should be within $\pm 1.5^\circ\text{C}$; otherwise failure occurs. In the R-SSGA method, we set five rules constitute a TSK-Type fuzzy controller

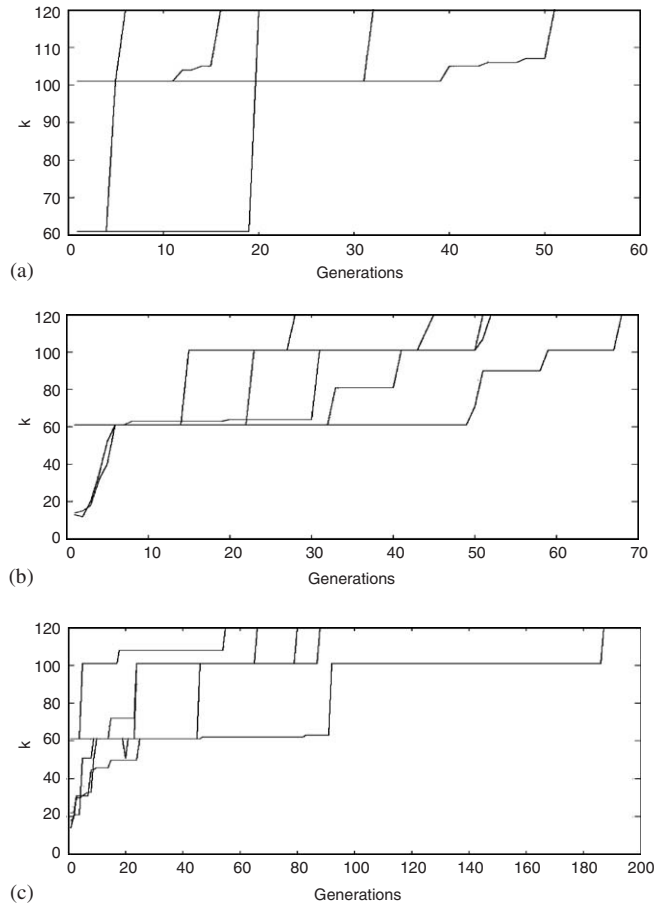


Fig. 13. The performance of the water bath system for (a) the R-SSGA method, (b) the SEFC method [16] and, (c) the TGFC method [22].

using the proposed R-SSGA method. A total of five runs were performed. Each run started at same initial state.

The simulation result in Fig. 13(a) shows that the R-SSGA method learned on average to succeed at the 25th generation. In this figure, each run indicates that the largest fitness value in the current generation was selected before the water bath temperature system failed. When the R-SSGA learning is stopped, we chose the best string in the population in the final generation and tested it with two different examples in the water bath temperature control system. The final fuzzy rules of a TSK-Type fuzzy controller by the R-SSGA method are described as follows:

- Rule 1: IF x_1 is $A_{11}(1.23, 0.75)$ and x_2 is $A_{21}(0.13, 0.81)$ THEN $y' = 7.09 + 8.50x_1 + 1.51x_2$.
- Rule 2: IF x_1 is $A_{12}(0.18, 0.352)$ and x_2 is $A_{22}(1.09, 0.45)$ THEN $y' = -19.41 - 14.051x_1 - 16.81x_2$.
- Rule 3: IF x_1 is $A_{13}(0.19, 0.36)$ and x_2 is $A_{23}(1.10, 0.46)$ THEN $y' = -19.42 - 14.05x_1 - 16.80x_2$.
- Rule 4: IF x_1 is $A_{14}(0.0001, 1.27)$ and x_2 is $A_{24}(1.09, 0.45)$ THEN $y' = 5.40 + 8.47x_1 - 16.81x_2$.
- Rule 5: IF x_1 is $A_{15}(5.0, 0.66)$ and x_2 is $A_{25}(0.14, 0.08)$ THEN $y' = -4.85 - 5.88x_1 + 9.45x_2$.

where $A_{ij}(m_{ij}, \sigma_{ij})$ represents a Gaussian membership function with mean m_{ij} and deviation σ_{ij} in i th input dimension and j th rule. In this example, as with Example 1,

we also compare the performance of our system with the SEFC method [16] and the TGFC method [22]. Figs. 13(b) and 10(c) show the performance of Juang et al. [16] and Lin and Jo [22] methods. In this figure we can see that the SEFC and TGFC methods learned on average to balance the pole at the 49th and 96th generation but in our model just take 25 generations.

For testing the controller system, we compare the three methods (the R-SSGA, SEFC, and TGFC methods). The three methods are applied to the water bath temperature control system. The comparison performance measures included a set points regulation and a change of parameters.

The first task was to control the simulated system to follow three set points

$$y_{\text{ref}}(k) = \begin{cases} 35^\circ\text{C} & \text{for } k \leq 40, \\ 55^\circ\text{C} & \text{for } 40 < k \leq 80, \\ 75^\circ\text{C} & \text{for } 80 < k \leq 120. \end{cases} \quad (19)$$

The regulation performance of the R-SSGA method is shown in Fig. 14(a). The error curves of the three methods are shown in Fig. 14(b). In this figure, the R-SSGA method obtains smaller errors than others.

In the second set of simulations, the tracking capability of the R-SSGA method with respect to ramp-reference signals is studied. We define

$$y_{\text{ref}}(k) = \begin{cases} 34^\circ\text{C} & \text{for } k \leq 30, \\ (34 + 0.5(k - 30))^\circ\text{C} & \text{for } 30 < k \leq 50, \\ (44 + 0.8(k - 50))^\circ\text{C} & \text{for } 50 < k \leq 70, \\ (60 + 0.5(k - 70))^\circ\text{C} & \text{for } 70 < k \leq 90, \\ 70^\circ\text{C} & \text{for } 90 < k \leq 120. \end{cases} \quad (20)$$

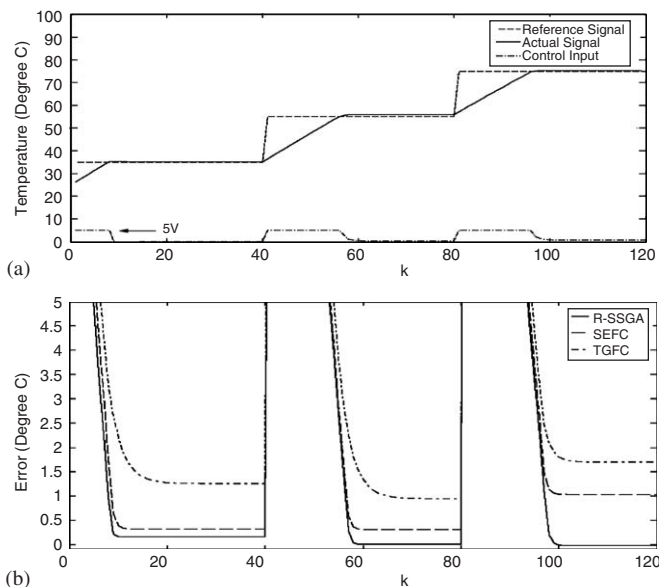


Fig. 14. (a) Final regulation performance of the R-SSGA method for water bath system. (b) The error curves of the R-SSGA method, the SEFC method and the TGFC method.

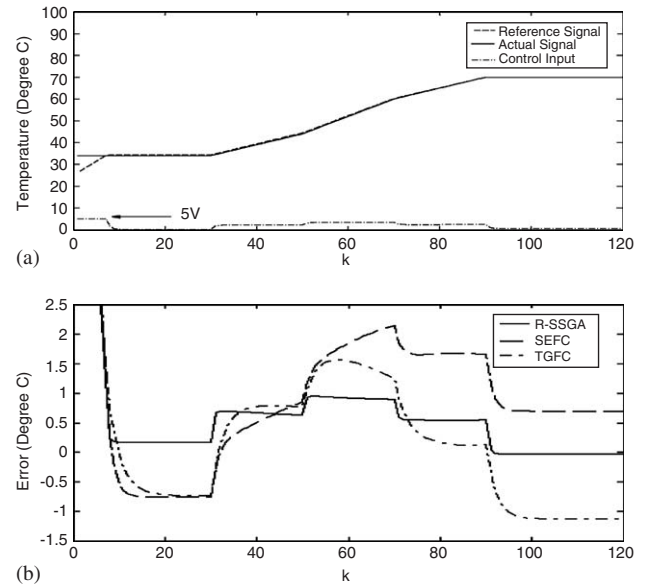


Fig. 15. (a) The tracking of the R-SSGA method when a change occurs in the water bath system. (b) The error curves of the R-SSGA method, the SEFC method [16], and the TGFC method [22].

Table 5
Performance comparison of various existing models in Example 2

$SAE = \sum_{k=1}^{120} y_{\text{ref}}(k) - y(k) $	R-SSGA	SEFC [16]	TGFC [22]
Regulation performance	360.04	370.12	400.12
Tracking performance	54.187	90.81	104.221

The tracking performance of the R-SSGA method is shown in Fig. 15(a). The corresponding errors of the three methods are shown in Fig. 15(b). The results show the good control and disturbance rejection capabilities of the trained R-SSGA method in the water bath system.

To test their regulation performance, a performance index, sum of absolute error (SAE), is defined by

$$SAE = \sum_k |y_{\text{ref}}(k) - y(k)|, \quad (21)$$

where $y_{\text{ref}}(k)$ and $y(k)$ are the reference output and the actual output of the simulated system, respectively. Table 5 shows the comparison the SAE among the R-SSGA method, the SEFC method, and the TGFC method. As show in Table 5, the proposed R-SSGA method has better performance than that of the others. And the proposed method only takes 5 rules and the populations' size is minimized to 4.

5. Conclusions

In this paper, a novel reinforcement sequential-search-based genetic algorithm (R-SSGA) is proposed. The better chromosomes will be initially generated while the better mutation points will be determined for performing efficient mutation. We formulate a number of time steps before

failure occurs as the fitness function. The proposed R-SSGA method makes the design of TSK-type fuzzy controllers more practical for real-world applications, since it greatly lessens the quality and quantity requirements of the teaching signals. Two typical examples were presented to show the fundamental applications of the proposed R-SSGA method. Simulation results have shown that (1) the R-SSGA method converges quickly; (2) the R-SSGA method requires a small number of population sizes (only 4); (3) the R-SSGA method obtains a smaller average angular deviation than other methods.

Acknowledgment

This research is supported by the National Science Council of ROC under Grant NSC 94-2213-E-324-004.

References

- [1] C.W. Anderson, Learning and problem solving with multilayer connectionist systems, Ph.D. dissertation, University of Massachusetts, Amherst, 1986.
- [2] K.J. Astrom, B. Wittenmark, Adaptive Control, Addison-Wesley, Reading, MA, 1989.
- [3] A.G. Barto, M.I. Jordan, Gradient following without backpropagation in layered networks, in: Proceedings of IEEE First Annual Conference on Neural Networks, vol. 2, San Diego, CA, 1987, pp. 629–636.
- [4] A.G. Barto, R.S. Sutton, Landmark learning: an illustration of associative search, *Biol. Cybern.* 42 (1981) 1–8.
- [5] A.G. Barto, R.S. Sutton, C.W. Anderson, Neuron like adaptive elements that can solve difficult learning control problem, *IEEE Trans. Syst. Man Cybern. SMC-13* (5) (1983) 834–847.
- [6] R.H. Cannon Jr., Dynamics of Physical Systems, Mc-Graw-Hill, New York, 1967.
- [7] K.C. Cheok, N.K. Loh, A ball-balancing demonstration of optimal and disturbance-accommodating control, *IEEE Control Syst. Mag.* (1987) 54–57.
- [8] O. Cordon, F. Herrera, F. Hoffmann, L. Magdalena, Genetic fuzzy systems evolutionary tuning and learning of fuzzy knowledge bases, *Advances in Fuzzy Systems-Applications and Theory*, vol. 19, World Scientific Publishing, NJ, 2001.
- [9] M.J. Er, C. Deng, Online tuning of fuzzy inference systems using dynamic Q-Learning, *IEEE Trans. Syst. Man Cybern. B* 34 (3) (2004) 1478–1489.
- [10] O. Grigore, Reinforcement learning neural network used in control of nonlinear systems, in: Proceedings of the IEEE International Conference on Industrial Technology, vol 1, January 2000, pp. 19–22.
- [11] J. Hauser, S. Sastry, P. Kokotovic, Nonlinear control via approximate input–output linearization: the ball and beam example, *IEEE Trans. Autom. Control* 37 (1992) 392–398.
- [12] A. Homaifar, E. McCormick, Simultaneous design of membership functions and rule sets for fuzzy controllers using genetic algorithms, *IEEE Trans. Fuzzy Syst.* 3 (9) (1995) 129–139.
- [13] J.-S.R. Jang, C.T. Sun, E. Mizutani, *Neuro-Fuzzy and Soft Computing*, Prentice-Hall, Englewood Cliffs, NJ, 1997 (Chapter 17).
- [14] C.F. Juang, A TSK-type recurrent fuzzy network for dynamic systems processing by neural network and genetic algorithms, *IEEE Trans. Fuzzy Syst.* 10 (2) (2002) 155–170.
- [15] C.F. Juang, C.T. Lin, An online self-constructing neural fuzzy inference network and its applications, *IEEE Trans. Fuzzy Syst.* 6 (1) (1998) 12–32.
- [16] 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.
- [17] C.L. Karr, E.J. Gentry, Fuzzy control of ph using genetic algorithms, *IEEE Trans. Fuzzy Syst.* 1 (1993) 46–53.
- [18] M. Kaya, R. Alhajj, Fuzzy OLAP association rules mining-based modular reinforcement learning approach for multiagent systems, *IEEE Trans. Syst. Man Cybern. B* 35 (2) (2005) 326–338.
- [19] C.J. Lin, A GA-based neural fuzzy system for temperature control, *Fuzzy Sets and Systems* 143 (2004) 311–333.
- [20] C.J. Lin, A GA-based neural network with supervised and reinforcement learning, *J. Chin. Inst. Electr. Eng.* 9(1) (2002) 11–24.
- [21] C.J. Lin, C.H. Chen, Nonlinear system control using compensatory neuro-fuzzy networks, *IEICE Trans. Fund. E86-A* (9) (2003) 2309–2316.
- [22] C.T. Lin, C.P. Jo, GA-based fuzzy reinforcement learning for control of a magnetic bearing system, *IEEE Trans. Syst. Man Cybern. B* 30 (2) (2000) 276–289.
- [23] C.T. Lin, C.S.G. Lee, Reinforcement structure/parameter learning for neural-network-based fuzzy logic control systems, *IEEE Trans. Fuzzy Syst.* 2 (1994) 46–63.
- [24] C.T. Lin, C.S.G. Lee, *Neural Fuzzy Systems: A Neural-Fuzzy Synergism to Intelligent Systems*, Prentice-Hall, Englewood Cliffs, NJ, 1996 (with disk).
- [25] D.E. Moriarty, R. Miikkulainen, Efficient reinforcement learning through symbiotic evolution, *Mach. Learn.* 22 (1996) 11–32.
- [26] T. Takagi, M. Sugeno, Fuzzy identification of systems and its applications to modeling and control, *IEEE Trans. Syst. Man Cybern.* 1 (1) (1985) 116–132.
- [27] J. Tanomaru, S. Omatu, Process control by on-line trained neural controllers, *IEEE Trans. Ind. Electron.* 39 (1992) 511–521.
- [28] D. Whitley, S. Dominic, R. Das, C.W. Anderson, Genetic reinforcement learning for neuro control problems, *Mach. Learn.* 13 (1993) 259–284.
- [29] X. Xu, H.G. He, Residual-gradient-based neural reinforcement learning for the optimal control of an acrobat, in: Proceedings of the IEEE International Conference on Intelligent Control, October 2002, pp. 27–30.
- [30] X.W. Yan, Z.D. Deng, Z.Q. Sun, Competitive Takagi–Sugeno fuzzy reinforcement learning, in: Proceedings of the IEEE International Conference on Control Applications, September 2001, pp. 878–883.



Cheng-Jian Lin received the B.S. degree in electrical engineering from Ta-Tung University, Taiwan, ROC, in 1986 and the M.S. and Ph.D. degrees in electrical and control engineering from the National Chiao-Tung University, Taiwan, ROC, in 1991 and 1996. From April 1996 to July 1999, he was an Associate Professor in the Department of Electronic Engineering, Nan-Kai College, Nantou, Taiwan, ROC. Since August 1999, he has been with the Department of Computer Science and Information Engineering, Chaoyang University of Technology. Currently, he is a Professor of Computer Science and Information Engineering Department, Chaoyang University of Technology, Taichung, Taiwan, ROC. He served as the chairman of Computer Science and Information Engineering Department from 2001 to 2005. His current research interests are neural networks, fuzzy systems, pattern recognition, intelligence control, bioinformatics, and FPGA design. He has published more than 60 papers in the referred journals and conference proceedings. Dr. Lin is a member of the Phi Tau Phi. He is also a member of the Chinese Fuzzy Systems Association (CFSA), the Chinese Automation Association, the Taiwanese Association for Artificial Intelligence (TAAI), the IEICE (The Institute of Electronics, Information and Communication Engineers), and the IEEE Computational Intelligence Society. He is an executive committee member of the Taiwanese Association for Artificial Intelligence (TAAI). Dr. Lin currently serves as the Associate Editor of International Journal of Applied Science and Engineering.



Yong-Ji Xu received the B.S. degree in information management from Ming-Hsin University of Science and Technology, Taiwan, ROC, in 2002 and the M.S. degree at the Department of Computer Science and Information Engineering,

Chaoyang University of Technology, Taiwan, ROC, in 2005. He is currently pursuing Ph.D. degrees in electrical and control engineering from the National Chiao-Tung University, Taiwan, ROC. His research interests include neural networks, fuzzy systems, and genetic algorithms.