# Genetic Reinforcement Learning through Symbiotic Evolution for Fuzzy Controller Design

Chia-Feng Juang, Jiann-Yow Lin, and Chin-Teng Lin, *Senior Member, IEEE*

***Abstract*—An efficient genetic reinforcement learning algorithm for designing fuzzy controllers is proposed in this paper. The genetic algorithm (GA) adopted in this paper is based upon *symbiotic evolution* which, when applied to fuzzy controller design, complements the local mapping property of a fuzzy rule. Using this Symbiotic-Evolution-based Fuzzy Controller (SEFC) design method, the number of control trials, as well as consumed CPU time, are considerably reduced when compared to traditional GA-based fuzzy controller design methods and other types of genetic reinforcement learning schemes. Moreover, unlike traditional fuzzy controllers, which partition the input space into a grid, SEFC partitions the input space in a flexible way, thus creating fewer fuzzy rules. In SEFC, different types of fuzzy rules whose consequent parts are singletons, fuzzy sets, or linear equations (TSK-type fuzzy rules) are allowed. Further, the free parameters (e.g., centers and widths of membership functions) and fuzzy rules are all tuned automatically. For the TSK-type fuzzy rule especially, which put the proposed learning algorithm in use, only the significant input variables are selected to participate in the consequent of a rule. The proposed SEFC design method has been applied to different simulated control problems, including the cart-pole balancing system, a magnetic levitation system, and a water bath temperature control system. The proposed SEFC has been verified to be efficient and superior from these control problems, and from comparisons with some traditional GA-based fuzzy systems.**

***Index Terms*—Genetic reinforcement, fitness value, fuzzy partition, symbiotic evolution, TSK-type fuzzy rules.**

## I. Introduction

FUZZY SYSTEMS have supplanted conventional technologies in some scientific applications and engineering systems in the past decade, especially in control systems [1]. Fuzzy logic has the ability to express the ambiguity of human thinking and translate expert knowledge into computable numerical data. Also, for real-time applications, its relatively low computational complexity makes it a good candidate.

A fuzzy system consists of a set of fuzzy if-then rules. Conventionally, the selection of fuzzy if-then rules often relies on a substantial amount of heuristic observation to express the knowledge of proper strategies. Obviously, it is difficult for human experts to examine all the input-output data from a complex system to find proper rules for the fuzzy system. To cope with this difficulty, several approaches to generating fuzzy if-then rules from numerical data have been proposed [2]–[5]. These methods were developed for supervised learning; i.e., the correct "target" output values are given for each input pattern to guide the network's learning. If supervised learning can be used in control, then it has been shown to be more efficient than reinforcement learning (e.g., [6], [7]). However, for some real-world control 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 fuzzy [8], [9] or neural controller [10]–[12] design. Recently, the development of genetic algorithms (GA's) has provided another approach for controller design when only sparse feedback information is available. GA's are probabilistic search procedures based on the mechanics of natural selection [13]. They explore a solution space and have been employed in two major areas: optimization and machine learning. Since GA's do not require derivative information, their most appropriate applications are problems where gradient information is unavailable or costly to obtain. Reinforcement learning is one example of such a domain, and the corresponding learning scheme is called "genetic reinforcement learning."

Some researchers have developed methods to design and implement fuzzy controllers by GA's. Karr used a GA to generate membership functions for a fuzzy controller [14]. However, in Karr's work, the user needs to declare an exhaustive rule set and then use the GA to design the membership functions only. Since the membership functions and rule set are co-dependent, using a hand-designed rule set with GA-designed membership functions, or hand-designed membership functions with a GA-designed rule set does not use the GA to its full advantage. In [15], a fuzzy controller design method that used the GA's to find the membership functions and the rule sets simultaneously was proposed. In [14] and [15], the input space is partitioned into a grid, with each cell representing a fuzzy if-then rule. The major disadvantage of such partitions is that the number of fuzzy rules, and hence the length of each chromosome in the GA, increases exponentially as the dimension of the input space increases. To reduce the number of rules required, in [16], an entry in the grid partition can be assigned blank, indicating no fuzzy set output for the corresponding rule. With this method, only the nonblank cells, instead of all the cells as in [14], are used to form a fuzzy system. Similarly, in [17], a coding method was proposed so that the user can make sure that only $n$ rules were selected from the possible rule combinations in a grid partition. The fuzzy rules used in the aforementioned techniques are rules with fuzzy sets

Manuscript received May 12, 1998; revised November 26, 1999. This work was supported by the R.O.C. National Science Council under Grant NSC88-2213-E-009-116. This paper was recommended by Associate Editor A. Kandel.

C.-F. Juang is with the Department of Control Engineering, Private Chung-Chou Junior College of Technology and Commerce, Changhua, Taiwan, R.O.C.

J.-Y. Lin and C.-T. Lin are with the Department of Electrical and Control Engineering, National Chiao-Tung University, Hsinchu, Taiwan, R.O.C.

Publisher Item Identifier S 1083-4419(00)02967-8.

in their consequents. In [18], the GA is used to tune the consequent parameters of TSK-type fuzzy rules, as well as the membership functions in the precondition parts. Since the consequent of a TSK-type fuzzy rule is a linear combination of all the input variables, the number of parameters increases as the dimension of the input space increases.

The fuzzy system design approaches mentioned above are all based on the traditional GA's, with each chromosome representing a fuzzy system. In this paper, the Symbiotic-Evolution-based Fuzzy Controller (SEFC) design method, a new genetic reinforcement learning algorithm, is proposed. Each chromosome represents only one fuzzy rule, and an $n$-rule fuzzy system is constructed by selecting and combining $n$ chromosomes from the population. Compared to normal evolution, symbiotic evolution promotes both cooperation and specialization, which ensures diversity and prevents the population from converging to suboptimal solutions [19]. In SEFC, which achieves fuzzy system design through symbiotic evolution, because no single individual can perform one problem well alone, the GA can search many different areas of the solution space concurrently. With the local property of a fuzzy rule, the fitness assignment performed by the symbiotic evolution is quite representative. In this way, symbiotic evolution and fuzzy system design can complement each other, which result in a fast, efficient genetic search. We have shown that the proposed method requires fewer trials and less CPU time than the traditional GA-based design methods for fuzzy systems. Moreover, with the proposed SEFC design method, a flexible partition of the input space is achieved, and hence the number of rules is relatively small compared to the grid partition. No prepartition of the input variables is needed, so less design effort is required. The SEFC design method can be used to learn different types of fuzzy rules, including fuzzy rules with singleton consequents, fuzzy rules with fuzzy set consequents, and TSK-type fuzzy rules. This can further reduce the number of parameters to be learned.

This paper is organized as follows. Section II describes the basic concept of symbiotic evolution and its link to fuzzy system design. A detailed implementation of the genetic reinforcement learning algorithm for the SEFC method is presented in Section III. In Section IV, the SEFC design method is applied to three control problems: the cart-pole balancing system, a magnetic levitation control system, and a water bath temperature control system. Finally, conclusions and future works on the proposed SEFC design method are summarized in the last section.

## II. Design of Fuzzy Systems Through Symbiotic Evolution

In this section, some basic concepts of GA's and symbiotic evolution are briefly reviewed, and then the idea of incorporating symbiotic evolution into the design of a fuzzy system is described. We shall show that the specialization property of symbiotic evolution matches the local property of a fuzzy rule well. This makes worthwhile the fuzzy system design achieved via symbiotic evolution.

### A. Basic Concept of GAs

Genetic algorithms are search algorithms inspired by the mechanics of natural selection, genetics, and evolution. It is widely accepted that the evolution of living beings is a process that operates on chromosome-organic devices for encoding the structure of living beings. Natural selection is the link between chromosomes and the performance of their decoded structures. Processes of natural selection cause chromosomes that encode successful structures to reproduce more often than those that do not. In addition to reproduction, mutations may cause the chromosomes of children to be different from those of their biological parents, and recombination processes may create different chromosomes in children by combining material from the chromosomes of their two parents. In a GA, a set of variables for a given problem is encoded into a string, analogous to a chromosome. Each unit in a string is called an allele. Like nature, GA's solve the problem of finding good chromosomes by manipulating the materials in the chromosomes without any knowledge about the type of problem they are solving. The only information they are given is an evaluation of each chromosome they produce. This evaluation is used to bias the selection of chromosomes, so that those with the best evaluations tend to reproduce more often than those with bad evaluations. GA's, using simple manipulations of chromosomes such as encoding and reproduction mechanisms, can display complicated behavior and solve some extremely difficult problems without any knowledge of the decoded world.

### B. Symbiotic Evolution and Fuzzy Systems

The idea of symbiotic evolution was first inspired by the proposed implicit fitness sharing algorithm used in an immune system model [20]. The authors developed artificial antibodies to match or identify artificial antigens. Because each antibody can only match one antigen, a difference population of antibodies is required to effectively defend against a variety of antigens. The similar model called co-adaptive genetic algorithm is based on competition but not cooperation [21]. In this system, each antibody must compete for survival with other antibodies in the subpopulation to identify the given antigen. And the fitness of each individual reflects how well it matches its opposing antigen; not how well it cooperates with other individuals. The antibodies are thus not dependent on other antibodies for identification of an antigen and only interact implicitly through competition. In [22], the author has proposed a method that a learning classifier system (LCS) can be mapped to a neural network. And each hidden node represents a classifier rule that must compete with other hidden nodes in a competition. In [23] and [24], the authors have developed a symbiotic evolutionary strategy called cooperative coevolutionary genetic algorithm (CCGA) and have applied it to both rule-based and neural network systems. Recently, the word "symbiotic evolution" was indeed appeared in [19]. The authors proposed a reinforcement learning method called symbiotic, adaptive neuro-evolution (SANE) which evolves a population of neurons through genetic algorithms to form a neural network.

Unlike normal genetic evolution algorithms, which operate on a population of full solutions to the problem, symbiotic evolution assumes that each individual in the population represents only a partial solution to the problem; complete solutions are formed by combining several individuals. In a normal evolution algorithm, a single individual is responsible for the overall performance, with the fitness value assigned to that individual according to its performance. In symbiotic evolution, the fitness of an individual (a partial solution) is calculated by summing the fitness values of all possible combinations of that individual with other current individuals (partial solutions) and dividing by the total number of combinations. As shown in [19], partial solutions can be characterized as *specializations*. The specialization property ensures diversity, which prevents the population from converging to suboptimal solutions [19]. A single partial solution cannot "take over" a population, since there must be other specializations present. Unlike the standard evolutionary approach, which always converges the population, hopefully at the global optimum, but often at a local one, the symbiotic evolution finds solutions in diverse, unconverged populations [19]. In addition to diversity, the symbiotic evolution appears to be a faster and more efficient search scheme than the normal evolutionary approaches. These properties of symbiotic evolution inspired our idea of incorporating it into fuzzy controller design.

The basic idea of incorporating symbiotic evolution into the design of a fuzzy system is to represent a single fuzzy rule by an individual (i.e., a chromosome). The final fuzzy system is then formed by combining $n$ randomly selected rules from the population. With the fitness assignment performed by symbiotic evolution, and the local property of a fuzzy rule, symbiotic evolution and the fuzzy system design can complement each other. If a normal GA evolution scheme is adopted for fuzzy system design, only the overall performance of a fuzzy system is known, not the performance of each fuzzy rule. The best method to replace the unsuitable fuzzy rules that degrade the overall performance of a fuzzy system is through random crossover operations, followed by observing the performance of the offspring. Only when the overall performance of the fuzzy system is good do we know that the unsuitable rules have been replaced. As for using symbiotic evolution for neural network design [19], the performance of a neuron (coded as an individual) depends on the other neurons with which it combines. A neuron that cooperates well with one set of neurons may cooperate badly with other sets of neurons. This problem also occurs in designing a neural network by a traditional GA method [25], [26]. Owing to this highly dependent (i.e., global instead of local) relationship in neural networks, the performance of a symbiotic-evolution-based neural network is not as good as the proposed SEFC. Moreover, if there is a *a priori* knowledge about some rule assignments (i.e., expert knowledge in the form of fuzzy if-then rules), we can incorporate it into the fuzzy controller directly to make our SEFC design more efficient.

Codings of a fuzzy system which contains $n$ fuzzy rules by the normal and symbiotic evolutions are shown in Fig. 1(a) and (b), respectively. For clarity, in Fig. 1 and the following figures, a rule with $j$ input variables and one output variable is shown. In Fig. 1(a), a string (an individual) represents a fuzzy system. The notation "MemFun" represents the free parameters of a membership function on each input variable which are to be optimized. Similarly, the term "ConPar" represents the free parameters on the consequent of a rule, which are to be optimized. Each string (individual) in Fig. 1(b) represents one fuzzy rule and is coded the same way as in Fig. 1(a).

## III. Genetic Reinforcement Learning Algorithm Through Symbiotic Evolution

A detailed implementation of the genetic reinforcement learning algorithm for the SEFC is introduced in this section. Like traditional GA's, this algorithm consists of three major operators: reproduction, crossover, and mutation. Before going into the details of these three genetic operators, the issues of coding, initialization, and fitness assignment are discussed. Coding concerns the way membership functions and fuzzy rules of a fuzzy system are represented as chromosomes suitable for symbiotic evolution. Initialization is the proper assignment of learning constants before entering the evolution process. Fitness assignment gives a proper fitness value to each fuzzy rule during the evolution process. The whole genetic reinforcement learning algorithm is described step by step below.

- Coding. The coding of a fuzzy rule in a chromosome is shown in Fig. 2. In this paper, a Gaussian membership function is used with variables $m$ and $\sigma$ representing the center and width of the membership function, respectively. Fig. 2(a) denotes a fuzzy rule whose consequent part is a singleton; i.e., the rule has the form of

$$\begin{aligned}&\text{IF } x_1 \text{ is } \mu(m_1,\sigma_1) \quad \text{and} \quad x_2 \text{ is } \mu(m_2,\sigma_2) \quad \text{and}\\ &\quad \cdots \quad \text{and} \quad x_j \text{ is } \mu(m_j,\sigma_j) \quad \text{THEN } y \text{ is } w_0\end{aligned}$$

  where $\mu(m_i,\sigma_i)$ represents a Gaussian membership function with center $m_i$ and width $\sigma_i$. Fig. 2(b) denotes a TSK-type fuzzy rule whose consequent part is a linear combination of the input variables plus a constant value; i.e., the rule has the form of

$$\begin{aligned}&\text{IF } x_1 \text{ is } \mu(m_1,\sigma_1) \quad \text{and} \quad x_2 \text{ is } \mu(m_2,\sigma_2) \quad \text{and}\\ &\quad \cdots \quad \text{and} \quad x_j \text{ is } \mu(m_j,\sigma_j)\\ &\text{THEN } y = w_0 + w_1 x_1 + \cdots + w_j x_j.\end{aligned}$$

  Unlike the traditional TSK-type fuzzy rule where all the input variables are linearly combined in the consequent part, only some significant input variables are used in the consequent parts of our SEFC, as shown in Fig. 2(b). The terms not used are denoted by "0". The problem of which terms should be present in the consequent part of each rule is left for the genetic search in our algorithm.

  For each chromosome, an gene is represented by a floating point number or an integer. The integer coding is adopted when only a limited choice of parameter values is allowed. For example, if the width of the membership function is an element of $\{0.3, 0.4, 0.5, 0.6\}$, then four integers "1," "2," "3," and "4" are used to represent the four candidate values. A flexible partition of the domain of a fuzzy rule is allowed, and no prepartition of the variables is required in the proposed learning algorithm.

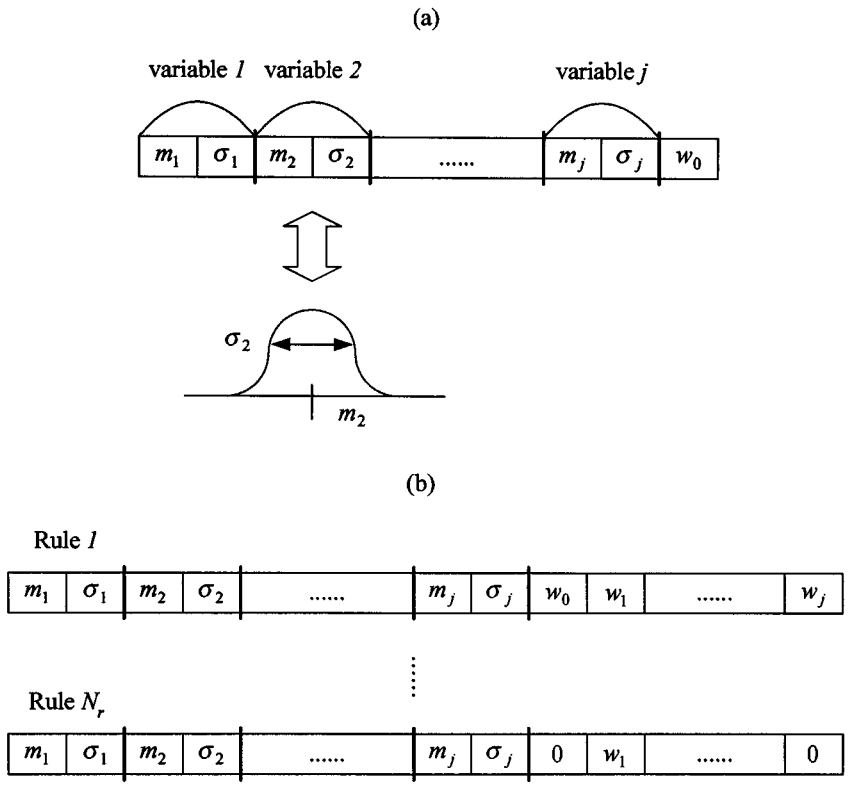



Fig. 2. Coding of a fuzzy rule into a chromosome in symbiotic evolution. (a) A rule with a singleton in the consequence. (b) A TSK-type fuzzy rule.

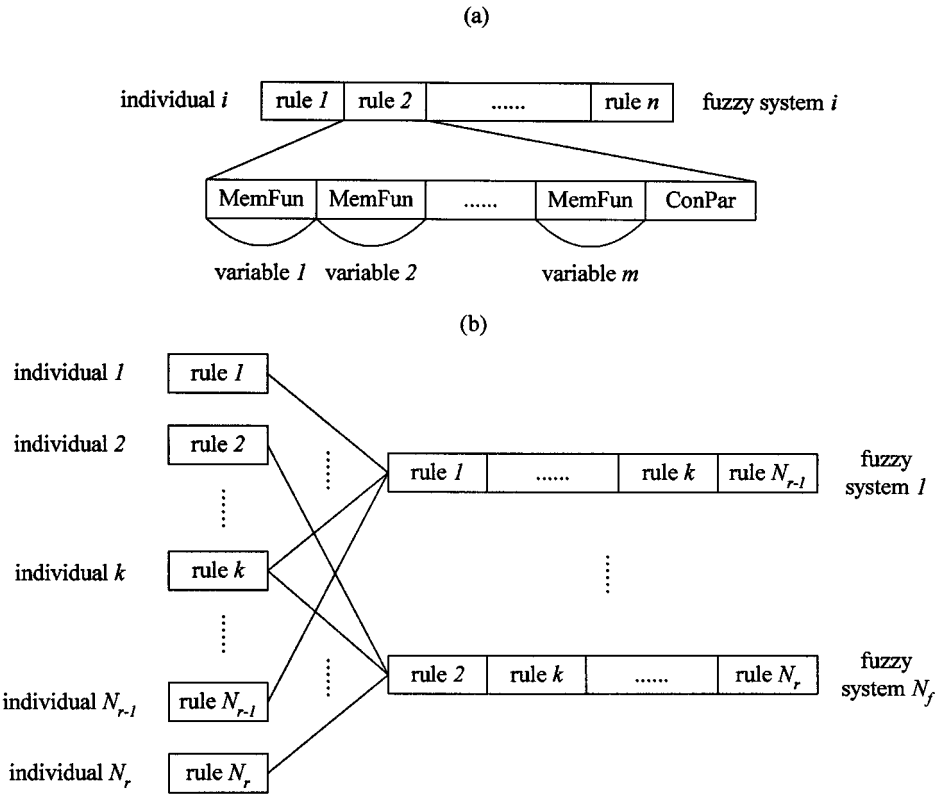



Fig. 1. Representation of a fuzzy system by (a) normal evolution, where an individual (a chromosome) represents a fuzzy system, and (b) symbiotic evolution, where an individual represents only a fuzzy rule.

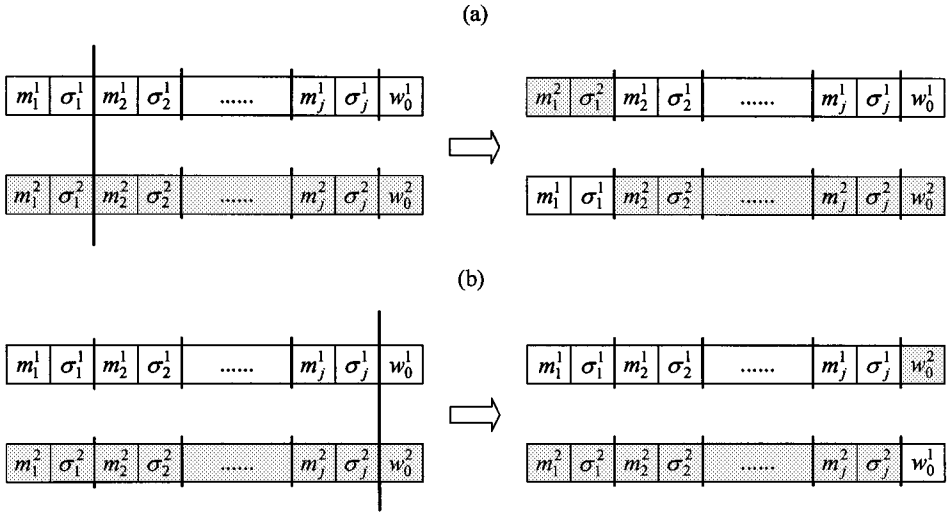



Fig. 3. Crossover operations on the rules (individuals). (a) Crossover at the precondition part of a rule. (b) Crossover at the consequent part of a rule.

- Initialization. Before proceeding with the SEFC design, $P_s$ individuals forming the initial population should be first generated randomly. The size of the population depends on the complexity of a problem. Besides the population size, some other parameters need to be specified. These are the number of fuzzy systems to be formed and evaluated per generation, $N_f$, the number of fuzzy rules that constitute a fuzzy system, $N_r$, and the probability of mutation, $p_m$.
- Fitness assignment. As previously stated, for symbiotic evolution, the fitness value of a rule (an individual) is calculated by summing the fitness values of all the possible combinations of that rule with all other $N_r - 1$ randomly selected rules, and then dividing the summed value by the total number of combinations. The detailed steps for assigning the fitness value are itemized as follows.

  Step 1) Randomly choose $N_r$ fuzzy rules from a population of $P_s$ rules, and form a fuzzy system from these $N_r$ rules.
  Step 2) Evaluate the performance of the combined fuzzy system to obtain a fitness value.
  Step 3) Divide the fitness value by $N_r$ and accumulate the divided fitness value to the fitness record of the $N_r$ selected rules with their fitness records set to zero initially.
  Step 4) Repeat $N_f$ times of the above process until each rule has been selected for a sufficient number of times and record the number of fuzzy systems in which each individual has participated.
  Step 5) Divide the accumulated fitness value of each individual by the number of times it has been selected.

The average fitness value represents the performance of a rule. For a TSK-type fuzzy rule, after computing the average fitness value of each rule, the average fitness value of each rule is further divided by "$K+$number of parameters in a rule," where $K$ is a constant, so a rule with less parameters is assigned a higher fitness value. In this way, only the significant terms are selected and used in the consequent part of a TSK-type fuzzy rule.

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

$$\phi = \prod_{i=1}^{j} \exp\left\{-\frac{(x_i - m_i)^2}{\sigma_i^2}\right\} = \exp\left\{-\sum_{i=1}^{j} \frac{(x_i - m_i)^2}{\sigma_i^2}\right\}. \quad (1)$$

The output of a fuzzy system is computed by

$$y^* = \frac{\sum_{i=1}^{N_r} \phi_i y_i}{\sum_{i=1}^{N_r} \phi_i} \quad (2)$$

where $y_i$ is the control action recommended by rule $i$.

- Reproduction. Reproduction is a process in which individual strings are copied according to their fitness values. To perform the process, the population is first sorted according to the fitness value of each individual. The top-half best-performing individuals in the population will advance to the next generation. The remaining half will be generated by performing crossover operations on individuals in the top half of the parent generation.
- Crossover. Reproduction directs the search toward the best existing individuals, but does not create any new individuals. New individuals are created mainly by crossover operations. In order to select the individuals for crossover, tournament selection [15] is performed on the top-half best-performing individuals. In the tournament selection, two or more individuals in the top-half population are selected at random, and their fitness values are compared. The individual with the highest fitness value is selected as one parent. The other parent is selected in the same way. The offspring is created by performing crossover on the selected parents. Some of the commonly used crossover techniques are one-point crossover, two-point crossover, multiple-point crossover, shuffle-exchange crossover [27], and uniform crossover [27]. In our problem, one-point crossover is used. Choosing a crossover site (a bit position) randomly on the two selected parents above. The individuals are crossed and separated at the site as shown in Fig. 3(a) and (b). In Fig. 3(a), new

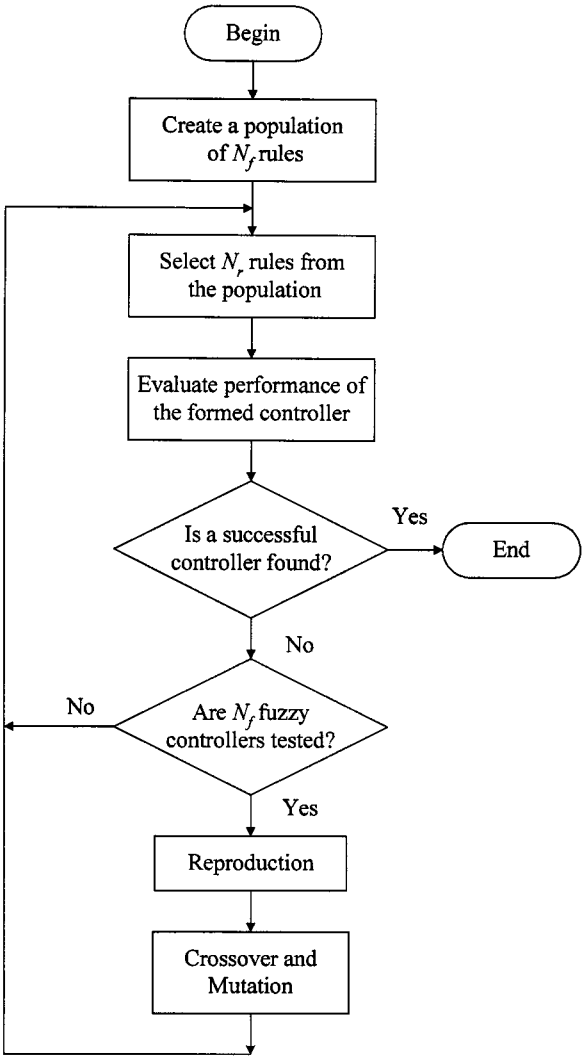



Fig. 4. Block diagram of the proposed SEFC design method.

individuals are created by exchanging some of the precondition parts of parent rules. In Fig. 3(b), new individuals are created by exchanging the consequent parts of parent rules. After operation, the individuals with poor performance will be replaced by the newly produced offsprings.

- Mutation. Mutation is an operator whereby the allele of a gene is altered randomly. With mutation, new genetic materials can be introduced into the population. Mutation should be used sparingly because it is a random search operator; otherwise, with high mutation rates, the algorithm will become little more than a random search. In the following simulations, a mutation probability $p_m = 0.01$ is used.

The process mentioned above is done repeatedly, and it is stopped when the predetermined condition is achieved. A block diagram of the SEFC design method is shown in Fig. 4. The SEFC design method can find a successful controller quickly. For complex design problems, the migration idea [28] is incorporated in the above algorithm for genetic search to increase the probability of global search and to reduce the effect of bad initial individual assignment which may cause a longer or even failure evolution. When the performance of the controller doesn't improve over a number of generations, half or more of the worst-performing individuals will migrate (implemented by random assignment of the individuals) to increase the searching ability in solving the global optimal problem.

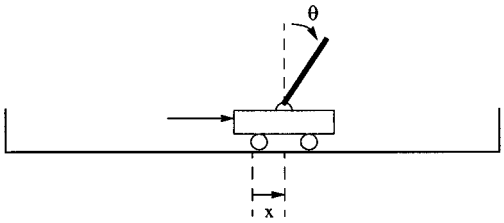



Fig. 5. Cart-pole balancing system.

## IV. Simulations

To verify the performance of the proposed SEFC, three control examples—the cart-pole balancing system, a magnetic levitation control system, and a water bath temperature control system—are presented in this section. Since the first example concerns a benchmark control problem attacked by several existing reinforcement learning algorithms, we shall compare the performance of SEFC to that of other approaches based on this problem. The other two examples illustrate the application of SEFC on two more practical problems.

*Example 1. Cart-Pole Balancing System:* In this example, we shall apply the SEFC design 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 [29], [30], or reinforcement learning schemes [11], [12], and is now used as a control benchmark. As shown in Fig. 5, the cart-pole balancing problem is the problem of learning how to balance an upright pole. The bottom of the pole is hinged to a cart that travels along a finite-length track to its right or left. Both the cart and the pole can move only in the vertical plane; that is, each has only one degree of freedom. There are four state variables in the system: $\theta$, the angle of the pole from an upright position (in degrees); $\dot{\theta}$, the angular velocity of the pole (in degrees/seconds); $x$, the horizontal position of the cart's center (in meters); and $\dot{x}$, the velocity of the cart (in meters/seconds). The only control action is $f$, which is the amount of force (Newtons) applied to cart to move it toward left or right. The system fails when the pole falls past a certain angle ($\pm 12^\circ$ is used here) or the cart runs into the bounds of its track (the distance is 2.4 m from the center to each bound of the track). The goal of this control problem is to determine a sequence of forces applying to the cart to balance the pole upright. The model and corresponding parameters of the cart-pole balancing system for our computer simulation are

$$\theta(t+1) = \theta(t) + \Delta\dot{\theta}(t) \tag{3}$$

$$\dot{\theta}(t+1) = \dot{\theta}(t) + \Delta\frac{mg\sin\theta_t - \cos\theta_t\left[f_t + m_p\ell\dot{\theta}_t^2\sin\theta_t\right]}{(4/3)m\ell - m_p\ell\cos^2\theta_t} \tag{4}$$

TABLE I
CPU TIME AND NUMBER OF TRIALS REQUIRED TO FIND A SUCCESSFUL CONTROLLER FROM A CENTERED POLE AND CART IN EXAMPLE 1. THE RESULTS ARE COMPUTED OVER 50 SIMULATIONS FOR EACH METHOD

| | CPU Seconds | | | | Pole Balance Trials | | | |
|---|---|---|---|---|---|---|---|---|
| Method | Mean | Best | Worst | SD | Mean | Best | Worst | SD |
| GENITOR | 30.9 | 13.0 | 146.3 | 24.1 | 1846 | 272 | 7052 | 1396 |
| SANE | 19.2 | 13.0 | 28.0 | 2.0 | 535 | 70 | 1910 | 329 |
| TGFC | 47.8 | 11.7 | 589.8 | 86.1 | 880 | 4 | 8223 | 1889 |
| SEFC | 13.2 | 11.7 | 19.3 | 2.4 | 111 | 4 | 323 | 93 |

TABLE II
CPU TIME AND NUMBER OF TRIALS REQUIRED TO FIND A SUCCESSFUL CONTROLLER FROM RANDOM POLE AND CART POSITIONS WITH RANDOM INITIAL VELOCITIES IN EXAMPLE 1. THE RESULTS ARE COMPUTED OVER 50 SIMULATIONS FOR EACH METHOD

| | CPU Seconds | | | | Pole Balance Trials | | | |
|---|---|---|---|---|---|---|---|---|
| Method | Mean | Best | Worst | SD | Mean | Best | Worst | SD |
| GENITOR | 31.9 | 13.0 | 175.5 | 25.7 | 2578 | 415 | 12964 | 2092 |
| SANE | 16.5 | 13.0 | 29.3 | 3.6 | 1691 | 46 | 4461 | 984 |
| TGFC | 75.9 | 12.8 | 563.1 | 126.5 | 2067 | 26 | 12691 | 2968 |
| SEFC | 15.2 | 12.8 | 31.4 | 2.5 | 661 | 10 | 5295 | 749 |

$$x(t+1) = x(t) + \Delta \dot{x}(t) \tag{5}$$

$$\dot{x}(t+1) = \dot{x}(t) + \Delta \frac{f_t + m_p \ell \left[\dot{\theta}_t^2 \sin\theta_t - \ddot{\theta}_t \cos\theta_t\right]}{m} \tag{6}$$

where

$\ell$ 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$^2$, acceleration due to the gravity;
$\Delta$ 0.02 (s), sampling interval.

The constraints on the variables are $-12^\circ \leq \theta \leq 12^\circ$, $-2.4\text{ m} \leq x \leq 2.4$ m, and $f = \pm 10$ N. A control strategy is deemed successful if it can balance a pole for 120 000 time steps.

In training, the SEFC for the cart-pole balancing problem, a fuzzy controller consisting of five rules with a singleton value in the consequent part of each rule is designed (i.e., $N_r = 5$). The number of rules is chosen so that the total number of parameters is nearly the same as that of other previously proposed genetic reinforcement learning algorithms [19], [26] applied to the same control problem. As in [19], the number of individuals in the population $P_s$ is set to 200. Two-hundred fuzzy controllers are formed and evaluated per generation (i.e., $N_f = 200$), and the evaluation of a fuzzy controller consists of a single trial on the cart-pole system. On average, a fuzzy rule participates in five fuzzy controllers per generation. The four state variables $(\theta, \dot{\theta}, x, \dot{x})$ and $f$ are normalized between $-1$ and 1 over the following ranges, $x : [-2.4, 2.4]$, $\dot{x} : [-1.5, 1.5]$, $\theta : [-12^\circ, 12^\circ]$, $\dot{\theta} : [-60^\circ, 60^\circ]$, $f : [-10, 10]$. The four normalized state variables are used as inputs to the fuzzy controller. The coding of a rule in a chromosome is of the form in Fig. 2(a). The values $m$ and $w_0$ are floating-point numbers randomly assigned between $-1$ and 1 initially. For simplicity, we allow $\sigma_i$ to be values from $\{0.3, 0.4, 0.5, 0.6\}$. Hence, the value of $\sigma_i$ is integer coded, with its value being "1," "2," "3," or "4" representing the actual normalized value of 0.3, 0.4, 0.5, or 0.6. The fitness value is the number of time steps in which the pole remains balanced.

Two experiments are performed on the SEFC. In the first experiment, the initial values of $(\theta, \dot{\theta}, x, \dot{x})$ are set to (0, 0, 0, 0). In the second experiment, the initial values of the four states are randomly assigned, with the values taken from the allowable range of each variable. A control is considered successful if it can balance the pole from any single starting state. For each experiment, 50 simulations are run. The number of pole-balance trials (which reflects the number of training episodes required) and the CPU time (the time from the first trial to the end of a successful control) are measured. The simulation results of both experiments on the SEFC are shown in Tables I and II.

Another fuzzy controller designed by the traditional GA's for the same control problem is used here for comparison. In the traditional genetic-fuzzy controller (TGFC) design, an individual represents a fuzzy controller, and the coding of each variable into a chromosome as well as the genetic operators are the same as those of the SEFC design. The number of rules in a population is set as in the SEFC design, so there are 40 individuals in a population with five rules in each individual. For further comparisons, previous genetic reinforcement learning algorithms, the GENITOR [26] and SANE (Symbiotic, Adaptive Neuro-Evolution) [19], are applied to the same control problem and the simulation results are listed in Tables I and II. In GENITOR, the normal evolution algorithm is 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 hidden units and one output unit. In SANE, the symbiotic evolution algorithm is used to evolve a two-layer neural network with five input units, eight hidden units, and two output units. An individual in the SANE represents a hidden unit with five specified connections to input and output units. The simulation data are adopted from [19] and the CPU time is converted to the CPU time of our working environment, a personal computer with Intel pentium-133 CPU inside. Although the 50 random initial states used in the SEFC and TGFC are different from those used in [19], on average, SEFC is superior to GENTOR and not second to SANE in the number of control trials and consumed CPU time. In fact, the proposed SEFC still obtains better (i.e., lower) values in most performance comparison indexes as shown in Tables I and II. It is shown that the fuzzy system design through symbiotic evolution is feasible and effective. Also in [19], the performance of SANE is shown to be much better than other reinforcement learning schemes including the single-layer Adaptive Heuristic Critic (AHC) [11], the two-layer AHC [12], and Q-learning [31]. Hence, the performance of these approaches are not compared to the proposed SEFC method. From the comparisons in Tables I and II, we can see that the SEFC design method achieves not only the minimum number of trials but also the least CPU time on average.

The CPU time conversion process in Tables I and II is detailed as follows. To compare the performance of our system with that of other existing reinforcement learning systems, we repeat the single-layer AHC design of the Barto's original system

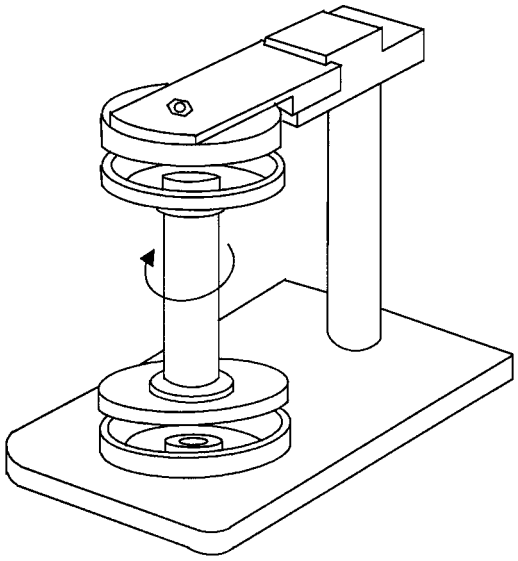


Fig. 6. Magnetic levitation control system.

[11]. Two neuro-like adaptive elements are integrated in this system. 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 the system. A bang-bang control scheme is used in the ASE, where the control output can have only two values: $+10$ N and $-10$ N. We divide the four-dimensional state space of the cart-pole system into disjoint regions. Here, 162 regions corresponding to all of the combinations of intervals are used. All of these settings are the same as those in [11] and [19]. The simulation is written in the C program and runs on a Pentium-133 personal computer. We have the average results computed over 50 simulations; it takes 424.8 s to find a successful controller from a centered pole and cart position, and 160.8 s to find a successful controller from random pole and cart positions with random initial velocities. Hence the AHC in our computation environment takes about 3.25 times of the CPU time taken by the same AHC in the computation environment in [19] to solve the same cart-pole balancing problem. Since the exactly same AHC simulation was done in [19], we can treat our simulation result and the counterpart in [19] as a basis of CPU time conversion from an IBM workstation to a personal computer. In this way, we have transferred the other simulation data in [19] into the counterparts in our working environment.

*Example 2. Magnetic Levitation Control System:* The goal of this example is to control a magnetic levitation system with a fuzzy controller. Most of the zero-power-consumption magnetic levitation (Maglev) systems are achieved by using hybrid magnets. The basic form of the Maglev system controlled in this example consists of a levitated object (rotor) and a pair of opposing E-shaped controlled-PM electromagnets with coil winding, as shown in Fig. 6. The attractive force each electromagnet exerts on the levitated object is proportional to the square of the current in each coil, and is inversely dependent on the square of the gap. The coil is highly inductive and the rate of change of the current is limited. The model of the magnetic levitation system

TABLE III
CPU TIME AND NUMBER OF TRIALS REQUIRED TO FIND A SUCCESSFUL FUZZY CONTROLLER BY SEFC AND TGFC FOR THE MAGNETIC LEVITATION CONTROL PROBLEM IN EXAMPLE 2

| | CPU Seconds | | | | Control Trials | | | |
|---|---|---|---|---|---|---|---|---|
| Method | Mean | Best | Worst | SD | Mean | Best | Worst | SD |
| TGFC | 82.3 | 0.9 | 719.8 | 141.9 | 1854 | 14 | 11076 | 2414 |
| SEFC | 5.2 | 0.9 | 22.8 | 4.7 | 332 | 13 | 1054 | 238.8 |

for our computer simulation is

$$x(k+1) = x(t) + \Delta\dot{x}(t) \tag{7}$$

$$\dot{x}(t+1) = \dot{x} + \Delta\left[\frac{F_1 - F_2 + f}{m} - g\right] \tag{8}$$

$$I(t+1) = I(t) + \Delta\frac{v - B\dot{x}(t) - RI(t)}{L} \tag{9}$$

where

| | |
|---|---|
| $g$ | 9.8 m/s$^2$, acceleration due to the gravity; |
| $m$ | 1.65 kg, mass of levitated object; |
| $\Delta$ | 0.001 s, sampling interval; |
| $f$ | 0.001, external disturbance; |
| $L$ | effective inductance of the magnetic circuit; |
| $B$ | effective flux density of the magnetic circuit; |
| $F_1$ | the upward force; |
| $F_2$ | the downward force. |

The control objective is to keep the levitated object balanced at the center. A fuzzy controller containing fuzzy rules with a singleton value in the consequent part of each rule is used. Inputs of the fuzzy controller are position $x$ and velocity $\dot{x}$ of the levitated object, and the output is the voltage $v$. The ranges of these variables are $x : [0.192, 0.2]$ m and $v : [-20, 20]$ V, which are normalized to $[-1, 1]$. The value of $\dot{x}$ is normalized by 0.2. The coding of a rule is the same as in Fig. 2(a). The initial values of $m$ and $w_0$ are randomly assigned floating-point numbers between $-1$ and 1. The $\sigma$ value is integer coded by $\{1\} \sim \{4\}$, with the integer value representing a value in $\{0.3, 0.4, 0.5, 0.6\}$. Five rules constitute a fuzzy controller (i.e., $N_r = 5$). The number of individuals in one population is $P_s = 100$, and one hundred fuzzy systems are formed and evaluated per generation ($N_f = 100$), which allows each rule to participate in five fuzzy controllers per generation on average. A control process is considered to be successful if the levitated object is controlled within the range $x = [0.195, 0.197]$ m for 2200 time steps, from both the top position $x = 0.2$ m and the bottom position $x = 0.192$ m.

The performance of a controller is measured by the number of time steps in which the controlled levitated object satisfies our demands. The requests we set are that the position of the levitated object should not touch the upper or lower bound during the control process, and after 150 control time steps, the position of the levitated object should be within $[0.195, 0.197]$ m; otherwise a failure occurs. The initial positions of the levitated object are randomly placed at the top ($x = 0.2$ m) or bottom ($x = 0.192$) positions with zero velocity ($\dot{x} = 0$). If the control

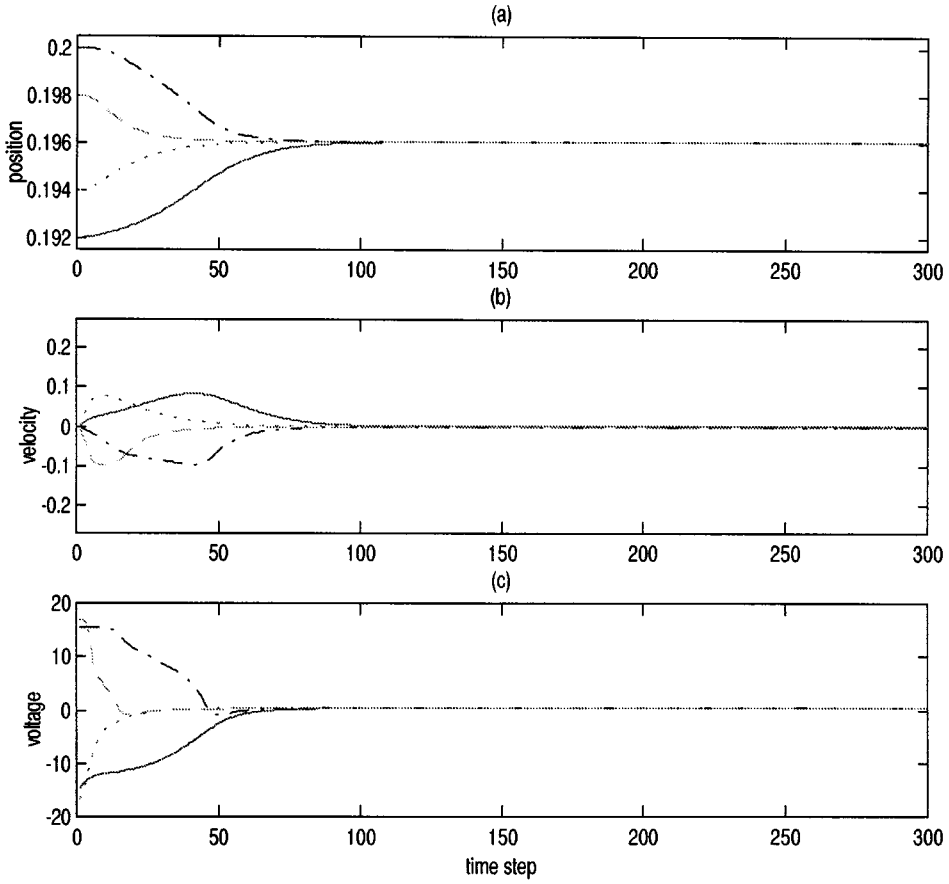



Fig. 7. Control results of the magnetic levitation system by the SEFC in Example 2. (a) Position of the levitated object. (b) Velocity of the levitated object. (c) The control voltage.

from one initial position is satisfied then the control from the other initial position is tried, with the total number of time steps accumulated. Fifty runs of the SEFC are simulated, and a run ends when a successful controller is found. The simulation results are listed in Table III. The control results of one of the fifty runs is shown in Fig. 7, where in addition to the trials starting from the two boundary positions used in the training process, the trials starting from other initial positions, $x = 0.198$ and $x = 0.194$ with $\dot{x} = 0$, are also performed for testing, and good results are obtained. The fuzzy rules learned in the SEFC are

Rule 1: IF $x(k)$ is $\mu(0.54, 0.4)$ and $\dot{x}(k)$ is $\mu(0.95, 0.4)$ THEN $v$ is $-0.40$.

Rule 2: IF $x(k)$ is $\mu(0.82, 0.6)$ and $\dot{x}(k)$ is $\mu(0.21, 0.3)$ THEN $v$ is 0.80.

Rule 3: IF $x(k)$ is $\mu(-0.26, 0.3)$ and $\dot{x}(k)$ is $\mu(-0.13, 0.3)$ THEN $v$ is $-0.86$.

Rule 4: IF $x(k)$ is $\mu(-0.76, 0.3)$ and $\dot{x}(k)$ is $\mu(-0.77, 0.4)$ THEN $v$ is $-0.96$.

Rule 5: IF $x(k)$ is $\mu(0.31, 0.4)$ and $\dot{x}(k)$ is $\mu(0.21, 0.3)$ THEN $v$ is 0.80.

where $\mu(m_i, \sigma_i)$ represents a Gaussian membership function with center $m_i$ and width $\sigma_i$.

A learning method called temporal difference and GA-based reinforcement (TDGAR) learning scheme has been also applied in the magnetic levitation system with the same simulation conditions [32]. The TDGAR learning method is a kind of hybrid GA algorithm and is constructed by integrating two feedforward multilayer networks. One neural network acts as a critic network for helping the learning of the other network, the action network, which determines the outputs of the TDGAR learning system. In the simulation, a neural fuzzy network is used as the action network. Given the current state of the plant, the action network selects an action by implementing an inference scheme based on fuzzy control rules. It can be represented as a network with five layers of nodes, each layer performing one stage of the fuzzy inference process. Comparing the simulation result of the TDGAR learning system with that of our SEFC design, the latter has similar control response time but less overshoot to maintain the levitated object around the equilibrium point.

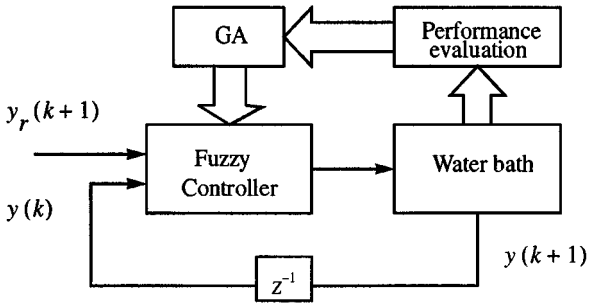



Fig. 8. Temperature control configuration of the water bath system in Example 3.

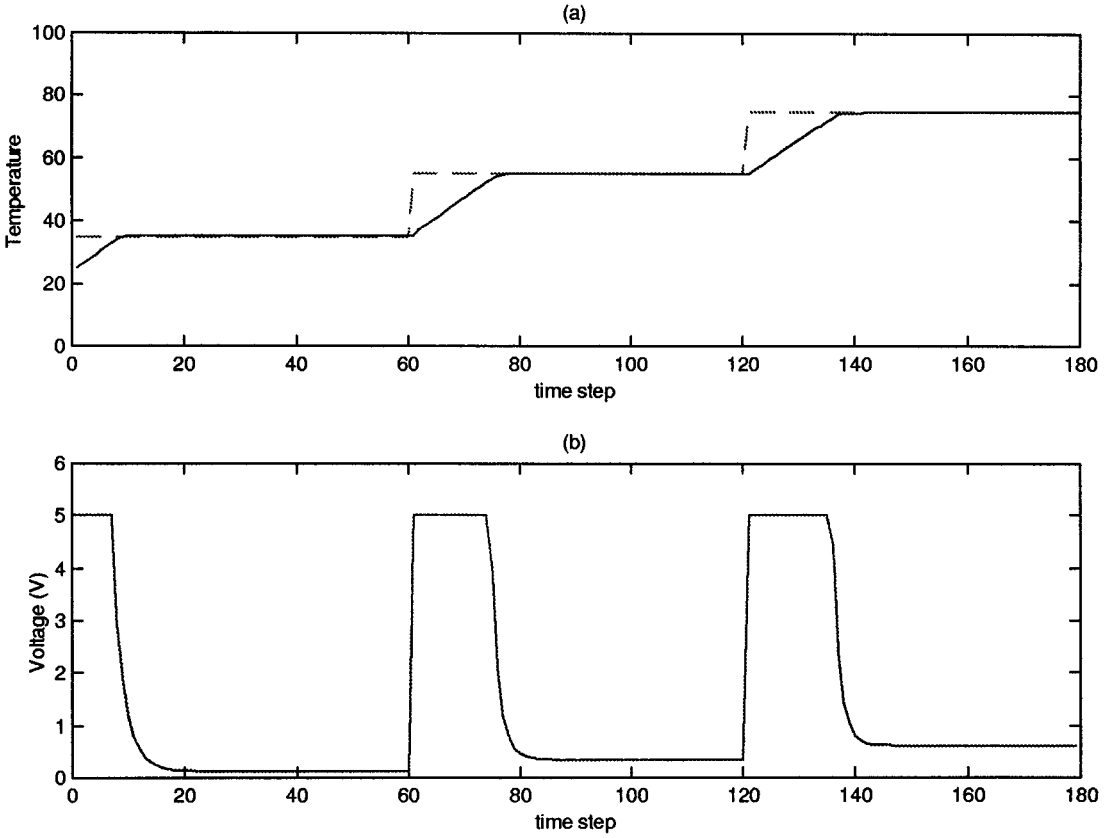



Fig. 9. Desired temperatures $(\cdots)$ and the controlled temperatures $(-)$ by SEFC (a) as well as the control outputs (b) in Example 3.

Also, a traditional GA-based fuzzy controller (TGFC) is designed for the same control problem to compare the learning capability of the SEFC design. The number of rules in one population is set as the same as that in the SEFC design, so there are 20 individuals in one population with each individual consisting of five rules. The coding and initial assignment of each variable is the same as those in the SEFC design. Fifty runs using the TGFC are simulated and the results are listed in Table III. From Table III, we find that the SEFC design method requires not only fewer trials but also less CPU time during learning.

*Example 3. Water Bath Temperature Control System:* The goal of this example is to control the temperature of a water bath system given by

$$\frac{dy(t)}{dt} = \frac{u(t)}{C} + \frac{Y_0 - y(t)}{RC} \tag{10}$$

where

- $y(t)$ system output temperature in °C;
- $u(t)$ heating flowing inward the system;
- $Y_o$ room temperature;
- $C$ equivalent system thermal capacity;
- $R$ equivalent thermal resistance between the system borders and surroundings.

Assuming that $R$ and $C$ are essentially constant, we rewrite the system in (10) into discrete-time form with some reasonable approximation. The system

$$y(k+1) = e^{-\alpha T_s} y(k) + \frac{\frac{\beta}{\alpha}(1 - e^{-\alpha T_s})}{1 + e^{0.5y(k)-40}} u(k) + [1 - e^{-\alpha T_s}] Y_0 \tag{11}$$

is obtained, where $\alpha$ and $\beta$ are some constant values describing $R$ and $C$. The system parameters used in this example are $\alpha = 1.001\,51e^{-4}$, $\beta = 8.679\,73e^{-3}$, and $Y_o = 25.0$(°C), which were obtained from a real water bath plant in [33]. The plant input $u(k)$ is limited between 0 and 5, and the sampling period is $T_s = 30$. The system configuration is shown in Fig. 8, where $y_r$ is the desired temperature of the controlled plant.

Unlike the previous two examples where only one desired target is set, the control target in this example may be any desired temperature in the controlled temperature range at different times, and the output of the controller may change sharply, so a fuzzy controller containing fuzzy rules with a singleton in the consequent part of each rule cannot meet the control requirement. This has been shown in our simulations. Hence, the TSK-type fuzzy rule is used in the SEFC for this control problem. The inputs to the fuzzy controller are the current temperature $y(k)$ of the water bath and the desired temperature $y_r(k)$, and the consequent part of each rule is of the form, $\hat{y} = w_0 + w_1 y(k) + w_2 y_r(k)$. The two inputs $y(k)$ and $y_r(k)$ are positively linearly transformed from $[0, 100]$ to $[0, 1]$. The coding of a rule is the same as in Fig. 2(b). The

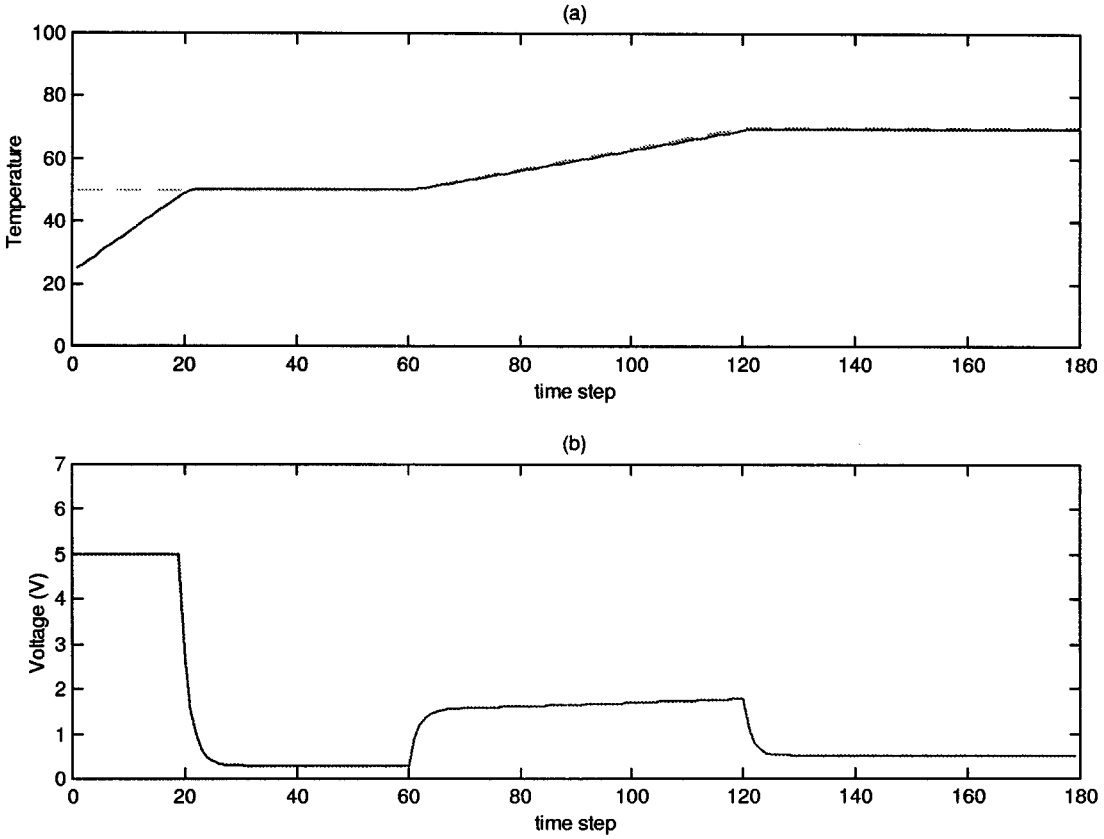



Fig. 10. Testing reference temperatures $(\cdots)$ and the controlled temperatures $(-)$ (a) with the corresponding control outputs (b) in Example 3.

initial values of $m$ and $w_0$ are randomly assigned floating-point numbers between 0 and 1. As before, for simplicity, $\sigma$ is integer coded by $\{1\} \sim \{4\}$, with the integer value representing a value in $\{0.3, 0.4, 0.5, 0.6\}$. The ranges of $w_1$ and $w_2$ are unknown, and a large range $[-45, 45]$ is assigned for them. Initial values of $w_1$ and $w_2$ are randomly assigned floating-point numbers in this range. A zero value of $w_i$ in the chromosome means that the corresponding term is not used. Initially, the number of terms used in the consequent part is assigned randomly for each rule. Five rules constitute a fuzzy controller (i.e., $N_r = 5$). The population size is set as $P_s = 100$. One hundred fuzzy controllers are formed and evaluated per generation (i.e., $N_f = 100$). The desired temperature profile used for training is shown in Fig. 9. The fitness of the controller is measured by the number of time steps in which the controlled temperature meets the constraint. The constraint is set as that for each reassigned regulation temperature $T = 35, 55$, and 75 in Fig. 9, starting from the current temperature and after 10 time steps, the controlled temperature error should be within $\pm 1.5$ °C; otherwise a failure occurs. Since a TSK-type fuzzy rule is used, as mentioned in Section III, the average fitness value of each rule is further divided by "30+number of parameters in a rule." In this way, only the significant terms are selected and used in the consequent part of a TSK-type fuzzy rule.

After 30 generations of training, a successful SEFC is obtained, where the fuzzy rules learned are

Rule 1: IF $y(k)$ is $\mu(0.11, 0.5)$ and $y_r(k)$ is $\mu(0.73, 0.4)$ THEN $y(k) = 37.7y(k) + 34.6y_r(k)$.

Rule 2: IF $y(k)$ is $\mu(0.12, 0.4)$ and $y_r(k)$ is $\mu(0.52, 0.6)$ THEN $y(k) = -26.4y(k)$.

Rule 3: IF $y(k)$ is $\mu(0.82, 0.4)$ and $y_r(k)$ is $\mu(0.39, 0.5)$ THEN $y(k) = -2.33 - 43.3y(k) + 18.2y_r(k)$.

Rule 4: IF $y(k)$ is $\mu(0.54, 0.6)$ and $y_r(k)$ is $\mu(0.60, 0.3)$ THEN $y(k) = -0.4 - 43.3y(k) + 30.8y_r(k)$.

Rule 5: IF $y(k)$ is $\mu(0.57, 0.6)$ and $y_r(k)$ is $\mu(0.81, 0.5)$ THEN $y(k) = -0.6 - 28.9y(k) + 43.8y_r(k)$.

where $\mu(m_i, \sigma_i)$ represents a Gaussian membership function with center $m_i$ and width $\sigma_i$. The controlled temperature curve obtained by the SEFC and the corresponding control outputs are shown in Fig. 9. To test the generalization ability of the designed controller, another desired temperature profile is fed into the controller. The desired temperature profile and the controlled one are shown in Fig. 10. A good control result is achieved.

The simulation results of the SEFC design are equivalent to those of two other control algorithms, an on-line neural controller training method [33] and the neural fuzzy inference network (NFIN) [34], for the same control problem. The on-line neural controller training method is based on performing multiple updating operations during each sampling period. It uses the direct inverse control error approach for small adjustments of the neural controller when it is already reasonably trained, and the predicted output error approach for minimizing the control error and improving convergence of the controller. And the NFIN is inherently a modified TSK-type fuzzy rule-based model possessing a neural network's learning ability. The rules in NFIN are created and adapted as on-line learning proceeds via

simultaneous structure and parameter identifications. Though the proposed SEFC method does not have superior performance to these two control algorithms, however, the SEFC is basically a reinforcement learning algorithm that takes the external reinforcement signal only after a long sequence of actions have been passed. And no a priori knowledge of the plant model or supervised input-output training data are required. Whereas these two compared algorithms are both supervised learning algorithms that require exact supervised input-output training data.

## V. Conclusion

A new genetic reinforcement learning algorithm, called the SEFC design method, is proposed for designing fuzzy controller in this paper. As compared to the traditional GA-based fuzzy controller design methods, the proposed SEFC design method requires fewer trials and less CPU time. The input space of the SEFC is partitioned in a flexible way. The rules used in the SEFC can be several types, e.g., a rule with a singleton in the consequent, or a TSK-type fuzzy rule. For the TSK-type fuzzy rule, only the significant terms are selected and combined in the linear equation of the consequent part. Since no *a priori* knowledge of the mathematical model of the controlled plant or supervised input-output training data are required, application domains of the proposed SEFC design method are believed to be numerous. Further works on the proposed SEFC design include 1) auto-determination of the number of fuzzy rules instead of preassignment, 2) on-line adaptation ability to meet the variations of the controlled plant or changing environments, and 3) coping with occasions when constraints on controller design are set.

## References


[1] C. C. Lee, "Fuzzy logic in control systems: Fuzzy logic controller—Part II," *IEEE Trans. Syst., Man, Cybern.*, vol. 20, pp. 419–435, 1990.

[2] C. T. Lin and C. S. G. Lee, *Neural Fuzzy Systems: A Neural-Fuzzy Synergism to Intelligent Systems*. Englewood Cliffs, NJ: Prentice-Hall, May 1996. (with disk).

[3] J. S. Jang, C. T. Sun, and E. Mizutani, *Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligent*. Englewood Cliffs, NJ: Prentice-Hall, 1997.

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

[5] L. X. Wang, *Adaptive Fuzzy Systems and Control*. Englewood Cliffs, NJ: Prentice-Hall, 1994.

[6] C. W. Anderson, "Learning and Problem Solving With Multilayer Connectionist Systems," Ph.D. dissertation, Univ. Massachusetts, Amherst, 1986.

[7] A. G. Barto and M. I. Jordan, "Gradient following without backpropagation in layered networks," in *Proc. IEEE 1st Annual Conf. Neural Networks*, vol. 2, San Diego, CA, 1987, pp. 629–636.

[8] H. R. Berenji and P. Khedkar, "Learning and tuning fuzzy logic controllers through reinforcements," *IEEE Trans. Neural Networks*, vol. 3, pp. 724–740, May 1992.

[9] C. T. Lin and C. S. G. Lee, "Reinforcement structure/parameter learning for neural-network-based fuzzy logic control systems," *IEEE Trans. Fuzzy Syst.*, vol. 2, pp. 46–63, Feb. 1994.

[10] G. E. Hinton, "Connectionist learning procedures," *Artif. Intell.*, vol. 40, no. 1, pp. 143–150, 1989.

[11] A. G. Barto, R. S. Sutton, and C. W. Anderson, "Neuronlike adaptive elements that can solve difficult learning control problems," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-13, pp. 834–846, 1983.

[12] C. W. Anderson, "Learning to control an inverted pendulum using neural networks," *IEEE Contr. Syst. Mag.*, vol. 9, pp. 31–37, 1989.

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

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

[15] A. Homaifar and E. McCormick, "Simultaneous design of membership functions and rule sets for fuzzy controllers using genetic algorithms," *IEEE Trans. Fuzzy Syst.*, vol. 3, pp. 129–139, April 1995.

[16] P. Thrift, "Fuzzy logic synthesis with genetic algorithms," in *Proc. 4th Int. Conf. Genetic Algorithms*, 1991, pp. 509–513.

[17] M. H. Lim, S. Rahardja, and B. H. Gwee, "A GA paradigm for learning fuzzy rules," *Fuzzy Sets and Systems*, vol. 82, pp. 177–186, 1996.

[18] M. A. Lee and H. Takagi, "Integrating design stages of fuzzy systems using genetic algorithms," in *Proc. IEEE Int. Conf. Fuzzy Systems*, New York, 1993, pp. 612–617.

[19] D. E. Moriarty and R. Miikkulainen, "Efficient reinforcement learning through symbiotic evolution," *Mach. Learn.*, vol. 22, pp. 11–32, 1996.

[20] R. E. Smith, S. Forrest, and A. S. Perelson, "Searching for diverse, cooperative populations with genetic algorithms," *Evol. Comput.*, vol. 1, no. 2, pp. 127–149, 1993.

[21] R. E. Smith and B. Gray, "Co-Adaptive Genetic Algorithms: An Example in Othello Strategy," Department of Engineering Science and Mechanics, University of Alabama, Tech. Rep. TCGA 94 002, 1993.

[22] R. E. Smith, "Is a learning classifier system a type of neural network?," *Evolutionary Computation*, vol. 2, no. 1, 1994.

[23] M. Potter and K. De Jong, "Evolving neural networks with collaborative species," presented at the Summer Computer Simulation Conf., Ottawa, Ont., Canada, 1995.

[24] M. Potter, K. De Jong, and J. Grefenstette, "A coevolutionary approach to learning sequential decision rules," presented at the 6th Int. Conf. Genetic Algorithms, Pittsburgh, PA, 1995.

[25] D. Whitley, T. Starkweather, and C. Bogart, "Genetic algorithm and neural networks: Optimizing connections and connectivity," *Parallel Computing*, vol. 14, pp. 347–361, 1990.

[26] D. Whitley, S. Dominic, R. Das, and C. W. Anderson, "Genetic reinforcement learning for neuro control problems," *Mach. Learn.*, vol. 13, pp. 259–284, 1993.

[27] L. J. Eshelman, R. A. Caruana, and J. Schaffer, "Biases in the crossover landscape," presented at the Proc. 3rd Int. Conf. Genetic Algorithms, 1989.

[28] J. J. Grefenstette, "Genetic algorithms for changing environment," in *Parallel Problem Solving from Nature, 2*. Amsterdam, The Netherlands: North Holland, 1992, pp. 145–154.

[29] R. H. Cannon, Jr., *Dynamics of Physical Systems*. New York: McGraw-Hill, 1967.

[30] K. C. Cheok and N. K. Loh, "A ball-balancing demonstration of optimal and disturbance-accommodating control," *IEEE Contr. Syst. Mag.*, pp. 54–57, 1987.

[31] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Mach. Learn.*, vol. 8, no. 3, pp. 279–292, 1992.

[32] C. P. Jou, "Temporal Difference and GA-Based Reinforcement Learning System," Ph.D. dissertation, Nat. Chiao-Tung Univ., Hsinchu, Taiwan, R.O.C., 1999.

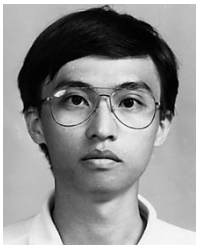
[33] J. Tanomaru and S. Omatu, "Process control by on-line trained neural controllers," *IEEE Trans. Ind. Electron.*, vol. 39, pp. 511–521, Dec. 1992.

[34] C. T. Lin, C. F. Juang, and C. P. Li, "Temperature control with a neural fuzzy inference network," *IEEE Trans. Syst., Man, Cybern.*, vol. 29, pp. 440–459, 1999.


**Chia-Feng Juang** received the B.S. and Ph.D. degrees in control engineering from the National Chiao-Tung University, Hsinchu, Taiwan, R.O.C., in 1993 and 1997, respectively.

He is currently an Assistant Professor in the Department of Control Engineering, Private Chung-Chou Junior College of Technology and Commerce, Changhua, Taiwan. His current research interests are neural networks, learning systems, fuzzy control, noisy speech recognition, and signal processing.

**Jiann-Yow Lin** received the B.S. degree in control engineering from the National Chiao-Tung University (NCTU) , Hsinchu, Taiwan, R.O.C., in 1997. He is currently pursuing the Ph.D. degree in the Department of Electrical and Control Engineering at NCTU. His current research interests are digital signal pcocessing, neural networks, fuzzy control, and learning systems.



**Chin-Teng Lin** (S'88–M'91–SM'99) received the B.S. degree in control engineering from the National Chiao-Tung University (NCTU), Hsinchu, Taiwan, R.O.C., in 1986, and the M.S.E.E. and Ph.D. degrees in electrical engineering from Purdue University, West Lafayette, IN, in 1989 and 1992, respectively.

Since August 1992, he has been with the College of Electrical Engineering and Computer Science, NCTU, Hsinchu, where he is currently a Professor of Electrical and Control Engineering. He has also been serving as the Deputy Dean of the Research and Development Office of NCTU since 1998. His current research interests are fuzzy systems, neural networks, intelligent control, human-machine interface, and video and audio processing. He is the coauthor of *Neural Fuzzy Systems—A Neuro-Fuzzy Synergism to Intelligent Systems* (Englewood Cliffs, NJ: Prentice-Hall, 1996), and the author of *Neural Fuzzy Control Systems with Structure and Parameter Learning* (New York: World Scientific, 1994). He has published over 50 journal papers in the areas of neural networks and fuzzy systems.

Dr. Lin is a member of Tau Beta Pi and Eta Kappa Nu. He has been the Executive Council Member of Chinese Fuzzy System Association (CFSA) since 1995, and the Supervisor of Chinese Automation Association since 1998. He was the Vice Chairman of IEEE Robotics and Automation Taipei Chapter in 1996 and 1997. He won the Outstanding Research Award granted by National Science Council (NSC), Taiwan, in 1997 and 1999, and the Outstanding Electrical Engineering Professor Award granted by the Chinese Institute of Electrical Engineering (CIEE) in 1997.