GA-Based Fuzzy Reinforcement Learning for Control of a Magnetic Bearing System

Chin-Teng Lin, *Senior Member, IEEE* and Chong-Ping Jou

***Abstract*—This paper proposes a TD (temporal difference) and GA (genetic algorithm)-based reinforcement (TDGAR) learning method and applies it to the control of a real magnetic bearing system. The TDGAR learning scheme is a new hybrid GA, which integrates the TD prediction method and the GA to perform the reinforcement learning task. The TDGAR learning system is composed of two integrated feedforward networks. One neural network acts as a critic network to guide the learning of the other network (the action network) which determines the outputs (actions) of the TDGAR learning system. The action network can be a normal neural network or a neural fuzzy network. Using the TD prediction method, the critic network can predict the external reinforcement signal and provide a more informative internal reinforcement signal to the action network. The action network uses the GA to adapt itself according to the internal reinforcement signal. The key concept of the TDGAR learning scheme is to formulate the internal reinforcement signal as the fitness function for the GA such that the GA can evaluate the candidate solutions (chromosomes) regularly, even during periods without external feedback from the environment. This enables the GA to proceed to new generations regularly without waiting for the arrival of the external reinforcement signal. This can usually accelerate the GA learning since a reinforcement signal may only be available at a time long after a sequence of actions has occurred in the reinforcement learning problem. The proposed TDGAR learning system has been used to control an active magnetic bearing (AMB) system in practice. A systematic design procedure is developed to achieve successful integration of all the subsystems including magnetic suspension, mechanical structure, and controller training. The results show that the TDGAR learning scheme can successfully find a neural controller or a neural fuzzy controller for a self-designed magnetic bearing system.**

***Index Terms*—Action network, active magnetic bearing, adaptive heuristic critic, critic network.**

NOMENCLATURE

B	Effective flux density of the magnetic circuit.
f	External disturbance.
F_1	Upward force.
F_2	Downward force.
$\mathrm{FIT}(t)$	Fitness function.
g	Acceleration of gravity.
G	Generation size.
L	Effective inductance of the magnetic circuit.
m	Mass of the levitated object.
N_c	Number of crossover point.
P_i	Permeance of the air gap and the leakage path.
POP	Population size.
r	External reinforcement signal.
$\hat{r}$	Internal reinforcement signal.
R	Effective resistance of the magnetic circuit.
R_c	Reluctance of the iron core of stator.
R_{gc}	Reluctance of the main air gap.
R_{gm}	Reluctance of leakage.
R_p	Reluctance of the permanent magnet.
R_r	Reluctance of the rotor.
V	Supplied voltage to the system.
v	Prediction of the external reinforcement value.
X_r	Leviated object position.
$\dot{X}_r$	Speed of X_r.
μ_0	Permeability of air.
μ_c	Relative permeability of the iron core of stator.
μ_p	Relative permeability of the permanent magnet.
μ_r	Relative permeability of the rotor.
Δ	Sampling interval.
Φ	Magnetic flux of the magnetic circuit.
γ	Discount rate.
η	Learning rate.

Manuscript received May 12, 1998; revised January 5, 2000. 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.

The authors are with the Department of Electrical and Control Engineering, National Chiao-Tung University, Hsinchu 300, Taiwan, R.O.C.

Publisher Item Identifier S 1083-4419(00)02966-6.

I. INTRODUCTION

MAGNETIC levitation is a subject of considerable scientific interest. It offers many advantages and opportunities for a wide variety of industrial, medical, and scientific applications [1]–[3], and high-speed ground transportation systems [4], [5]. An active magnetic bearing (AMB) is a collection of electromagnets used to suspend an object via feedback control. Traditionally, the term *magnetic bearing* has referred to devices for the suspension of a rotor. Its obvious feature is noncontact motion. Commercial applications include pumps, compressors, flywheels, milling and grinding spindles, turbine engines, and centrifuges. Magnetic suspension offers a number of practical advantages over conventional bearings such as lower rotating losses, higher speeds, elimination of the lubrication system and lubricant contanimation of the process, operation at temperature extremes and in vacuum, and longer life. However, magnetic suspensions have also been applied to nonrotating objects (sometimes referred to as *flotors*) for applications as varied as precision motion, wind tunnel model levitation, vibration isolation systems, and the treatment of brain tumors. AMB applications often require the solution of very interesting and formidable control problems because of the inherent instability and the nonlinear relationship between the lift force and the air gap distance. Thus, many control methods have been proposed to

solve these problems, such as gain schedule H_∞ robust control [6], nonlinear control [7], [8], adaptive autocentering control [9], and sliding control [10]. However, all these approaches need the precise mathematical model of the AMB system. In this paper, we propose a GA-based reinforcement learning scheme to solve the control problem of a self-designed and self-built AMB system with low power consumption.

In neural learning methods, supervised learning is more efficient than the reinforcement learning when the input-output training data are available [11]. However, many control problems require selecting control actions whose consequences emerge over uncertain periods for which input-output training data are not readily available. In such a case, reinforcement learning can be used to learn the unknown desired outputs by providing the system with a suitable evaluation of its performance [12]. Hence, the reinforcement learning techniques are more appropriate than the supervised learning for practical systems such as the real AMB system studied in this paper. Two general approaches can achieve the goal of reinforcement learning; the *actor-critic* architecture and the genetic algorithm (GA).

Barto and his colleagues built the original actor-critic architecture using the neuron-like adaptive elements in [13]. This architecture and the adaptive heuristic critic (AHC) algorithm were fully developed in [14]. The AHC algorithm belongs to a class of adaptive critic reinforcement learning algorithms that rely upon both a learned critic function and a learned action function. Adaptive critic algorithms are designed for reinforcement learning with delayed rewards. The AHC algorithm uses the temporal difference method to train a critic network that learns to predict failure. The prediction is then used to heuristically generate plausible target outputs at each time step, thereby allowing the use of backpropagation in a separate neural network that maps state variables to output actions. They also proposed the associative reward-penalty (A_{R-P}) algorithm for adaptive elements called A_{R-P} elements [15]. Several generalizations of A_{R-P} algorithm have been proposed [11]. Williams formulated the reinforcement learning problem as a gradient-following procedure [16], and he identified a class of algorithms, called REINFORCE algorithms, that possess the gradient ascent property; i.e., for any such algorithm the average update direction in weight space lies in a direction for which the performance measure (i.e., average reinforcement) is increasing. In other words, if the reinforcement indicates improved performance, then the weights are changed so as to make the element more likely to do whatever it did; otherwise, if the reinforcement indicates decreased performance, then the weights are changed to make the element more likely to do something else. Hence, the REINFORCE algorithms are gradient-descent optimization algorithms. Anderson [17] developed a numerical connectionist learning system by replacing the neuron-like adaptive elements in the actor-critic architecture with multilayer networks. With multilayer networks, the learning system can learn more new features that are required by or that facilitate the search for the task's solution. In [18] and [19], neural fuzzy networks were used to replace the neuron-like adaptive elements in the actor-critic architecture, so that the neural fuzzy network could house available expert knowledge to speed up its learning. The main drawback of these actor-critic architectures is that they usually suffer from the local minima problem in network learning due to the use of gradient descent learning method.

Another approach to reinforcement learning is by the use of GA's. GA's are general purpose optimization algorithms with a probabilistic component that provides a means to search poorly understood, irregular spaces [20]–[24]. From the network learning point of view, GA's only need a suitable evaluation of the network performance to yield the fitness values for evolution. Because GA's do not require or use derivative information, the most appropriate applications are problems where gradient information is unavailable or costly to obtain, such as reinforcement learning problems. GA's have been widely used for the design of fuzzy or neural controllers [25]–[30]. Karr [31] used GA's to alter the shape of the fuzzy sets used in a given rule base. Lee and Takagi [32] used a GA to optimize the rule base (including the number of rules and fuzzy sets per domain). Kropp and Baitinger [33] proposed the use of a GA to optimize rule bases such that the problem of generating meaningless genes using crossover and mutation was avoided. The application of GAs to the evolution of neural network topologies has also produced interesting results [34]–[39]. Whitley and his colleagues [40] demonstrated how GA's can be used to train neural networks for reinforcement learning and neurocontrol applications. They used the external reinforcement signal from the environment as the fitness function for the GA's. Moriarty and Miikkulainen [41] proposed a new reinforcement learning method called SANE (Symbiotic Adaptive Neuro-Evolution). SANE achieves efficient learning through *symbiotic evolution*, where each individual in the population represents only a partial solution to the problem, and complete solutions are formed by combining several individuals. As compared to the aforementioned actor-critic architecture, all the above GA-based reinforcement learning schemes use only the action networks. Without the predictions (internal critics) of the critic network, the GA cannot proceed to the next generation until the arrival of the external reinforcement signals. This constitutes the main drawback of these pure GA approaches, since a reinforcement signal may only be available at a time long after a sequence of actions has occurred in the reinforcement learning problems.

In this paper, we integrate the actor-critic architecture and the GA into a new reinforcement learning method. This scheme can solve the local minima problem in the actor-critic architecture by making use of the global optimization capability of GA's. Also, this method uses a critic network to provide the action network with a more informative internal reinforcement signal, so that the GA can perform a more effective search on the weight space of the action network. The proposed method is called the TD (temporal difference) and GA (genetic algorithm)-based reinforcement (TDGAR) learning method, and integrates the TD prediction method and the GA into the actor-critic architecture to perform the reinforcement learning task. The TDGAR learning system is constructed by integrating two feedforward multilayer networks. One neural network acts as a critic network to guide the learning of the action network, and the other is a neural network or a neural fuzzy network acting as an action network for determining the outputs (actions) of the TDGAR learning system. Using the TD prediction method, the critic net-

work can predict the external reinforcement signal and provide a more informative internal reinforcement signal to the action network. The action network uses the GA to adapt itself according to the internal reinforcement signal. The key concept of the proposed TDGAR learning scheme is to formulate the internal reinforcement signal as the fitness function for the GA. By using the internal reinforcement signal as the fitness function, the GA can evaluate the candidate solutions (chromosomes) regularly even during periods without external feedback from the environment. Hence, the GA can proceed to new generations regularly without waiting for the arrival of the external reinforcement signal. The proposed TDGAR learning scheme has been used to train, respectively, a neural controller and a neural fuzzy controller for a self-designed active magnetic bearing system. Both trained controllers can suspend the rotor successfully. Also, the trained neural fuzzy controller does achieve the goal of low-power-consumption control.

This paper is organized as follows. Section II describes the mechanical structure of the magnetic bearing system, and its design issues. The structure of the proposed TDGAR learning system and the corresponding learning algorithm is presented in Section III. In Section IV, the TDGAR learning method is applied to control the magnetic bearing system. Finally, conclusions are made in Section V.

II. Design of the Active Magnetic Bearing System

The basic form of our magnetic bearing system consists of a levitated object (rotor) and a pair of opposing E-shaped controlled-PM (permanent magnet) electromagnets with coil winding, as shown in Fig. 1. 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 main advantages of such a system lie in its higher lift-to-weight ratio and lower power consumption as compared to the conventional magnetic suspension system. In this section, the preliminary analysis of the magnetic circuit is carried out and the design process is introduced.

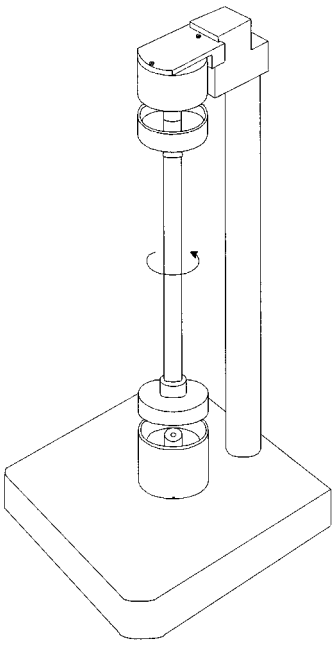
Because of the symmetric structure of this system, only one of the two opposing electromagnetics is considered, as shown in Fig. 2. The permanent magnet produces not only the main air gap flux, but also a significant amount of leakage flux. Let P_i represent the permeance of the air gap and the leakage path. Expressions of these parameters are given as follows:

$$P_1 = \mu_0 \frac{\pi l_p^2}{4z} \tag{1}$$

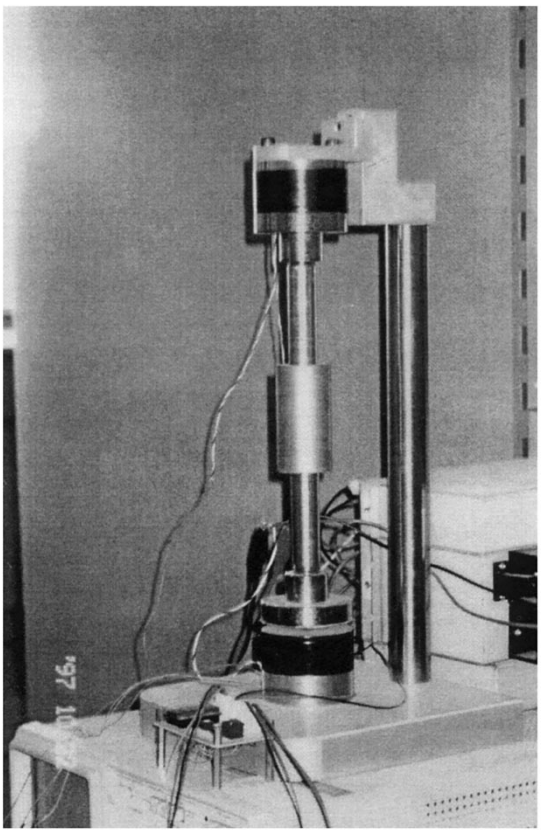
$$P_2 = 0.26\mu_0 \cdot 2\pi \left(\frac{1}{2}l_p + \frac{1}{4}z\right) \tag{2}$$

$$P_3 = \mu_0 \frac{l_g \cdot 2\pi \left(\frac{1}{2}l_p + W + \frac{1}{2}l_g\right)}{z} \tag{3}$$

$$P_4 = 0.26\mu_0 \cdot 2\pi \left(\frac{1}{2}l_p + W - \frac{1}{4}z\right) \tag{4}$$



(a)



(b)

Fig. 1. (a) Schematic of the self-designed magnetic bearing system. (b) Picture of the real magnetic bearing system.

$$P_5 = 0.26\mu_0 \cdot 2\pi \left(\frac{1}{2}l_p + W + l_g + \frac{1}{4}z\right) \tag{5}$$

where μ_0 is the permeability of air.

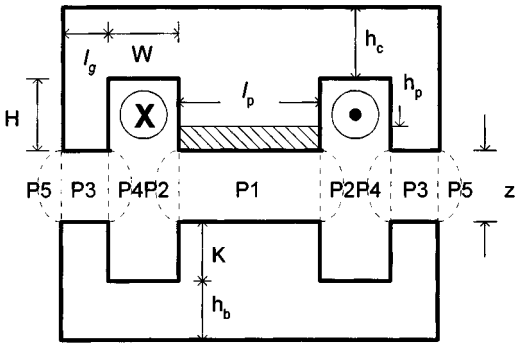


Fig. 2. Air gap and flux leakage for one of the two opposing E-shaped controlled-PM (permanent magnet) electromagnets with coil winding in the magnetic bearing system in Fig. 1.

By paralleling P_1P_2 and $P_3P_4P_5$, the main air gap and leakage reluctance can be obtained as follows:

$$R_{gm} = \frac{1}{P_{gm}} = \frac{1}{P_1 + P_2} \tag{6}$$

$$R_{gc} = \frac{1}{P_{gc}} = \frac{1}{P_3 + P_4 + P_5}. \tag{7}$$

The reluctance of the permanent magnet and the iron core of stator and rotor are

$$R_p = \frac{4h_p}{\mu_p \pi l_p^2} \tag{8}$$

$$R_c = \frac{1}{\mu_c}\left[\frac{H - h_p}{\frac{\pi}{4}l_p^2} + \frac{l_p/4 + W + l_g/2}{\frac{1}{2}(2\pi h_c(W + l_p/2 + l_g))} + \frac{H}{l_g \cdot 2\pi(l_p/2 + W + l_g/2)}\right] \tag{9}$$

$$R_r = \frac{1}{\mu_r}\left[\frac{K}{\frac{\pi}{4}l_p^2} + \frac{l_p/4 + W + l_g/2}{\frac{1}{2}(2\pi h_c(W + l_p/2 + l_g))} + \frac{K}{l_g \cdot 2\pi(l_p/2 + W + l_g/2)}\right] \tag{10}$$

where μ_p, μ_c, μ_r are relative permeabilities with respect to PM, core, and rotor.

The overall magnetic circuit with the permanent magnet as the mmf source is shown in Fig. 3. From Fig. 3, we apply the equivalence of Kirchhoff's voltage law to the magnetic circuit and obtain the magnetic flux in the circuit as

$$\Phi_k = \frac{NI - bH_c \cdot h_p}{R_c + R_{gc} + R_r + R_{gm} + R_p}, \quad k = 1, 2 \tag{11}$$

and

$$\phi_1 = \frac{P_1}{P_1 + P_2}\Phi_k \tag{12}$$

$$\phi_2 = \frac{P_2}{P_1 + P_2}\Phi_k \tag{13}$$

$$\phi_3 = \frac{P_3}{P_3 + P_4 + P_5}\Phi_k \tag{14}$$

$$\phi_4 = \frac{P_4}{P_3 + P_4 + P_5}\Phi_k \tag{15}$$

$$\phi_5 = \frac{P_5}{P_3 + P_4 + P_5}\Phi_k \tag{16}$$

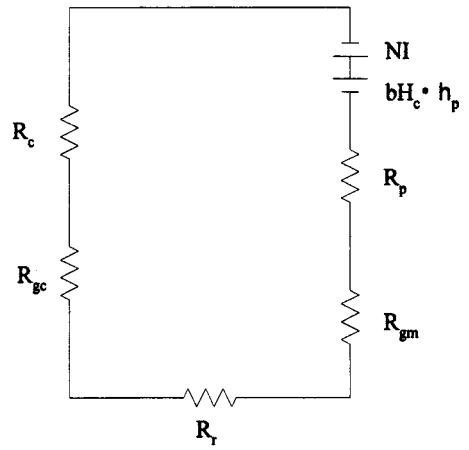


Fig. 3. Magnetic circuit of the magnetic bearing system in Fig. 1.

where ϕ_i is the magnetic flux with respect to the ith path in Fig. 3, for $i = 1, 2, 3, 4, 5$.

The average cross-sectional area of each path S_i is expressed as follows:

$$S_1 = \frac{\pi l_p^2}{4} \tag{17}$$

$$S_2 = 0.322 \cdot z \cdot \pi\left[l_p + \frac{z}{2}\right] \tag{18}$$

$$S_3 = l_g \cdot 2\pi\left[\frac{l_p}{2} + W + \frac{l_g}{2}\right] \tag{19}$$

$$S_4 = 0.322 \cdot z \cdot \pi\left[l_p + 2W - \frac{1}{2}z\right] \tag{20}$$

$$S_5 = 0.322 \cdot z \cdot \pi\left[l_p + 2W + 2l_g + \frac{1}{2}z\right]. \tag{21}$$

According to (12)–(21), the lift force is obtained from the equation

$$f_i = \frac{\phi_i^2}{2\mu_0 S_i}, \quad i = 1, 2, \ldots, 5. \tag{22}$$

Therefore, we get the total force as

$$F_k = \sum_{i=1}^{5} f_i, \quad k = 1, 2 \tag{23}$$

where F_1 is the upward force and F_2 is the downward force.

By Newton's second law, the force equilibrium equation is

$$m\ddot{X}_r = F_1 - F_2 - mg \tag{24}$$

where m and X_r are the mass and position of the rotor, respectively.

In addition, the characteristic of electricity can be obtained by applying the equivalence of Kirchhoff's voltage law to the system. The characteristic of electricity describes the dynamic relationship among the applied voltage to the system and the resulting current, the effective resistance, and the flux of the two electromagnets of our system. The diagram of characteristic

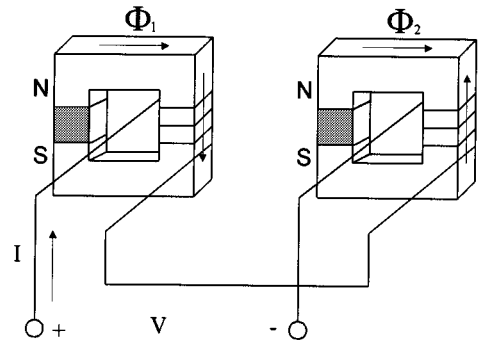


Fig. 4. Diagram for obtaining the characteristic of electricity of the proposed magnetic bearing system.

of electricity is shown in Fig. 4. Applying the equivalence of Kirchhoff's voltage law to the circuit in this figure, we can obtain

$$\begin{aligned} V &= IR + \frac{d}{dt}(\Phi_1 + \Phi_2) \\ &= IR + \frac{d(\Phi_1 + \Phi_2)}{dX_r}\frac{dX_r}{dt} + \frac{d(\Phi_1 + \Phi_2)}{dI}\frac{dI}{dt} \\ &= RI + B\dot{X}_r + L\dot{I} \end{aligned} \tag{25}$$

V supplied voltage to the system;
R effective resistance of the magnetic circuit;
L effective inductance of the magnetic circuit;
B effective flux density of the magnetic circuit.

Hence, we have a simplified dynamic model for the magnetic bearing system.

The above formulas are used to analyze and design our AMB system. From Fig. 2, we see that there are eight parameters to be decided in this system; they are $H, W, K, l_g, l_p, h_p, h_c$, and h_b. In this paper, we propose the use of GA to perform this difficult combinatorial design task. We code the eight parameters into a chromosome (string) using the real-value coding scheme (to be discussed in Section III-B2), eight bits for each parameter. The population size is set as 40. A fitness function is defined to evaluate if a given AMB system has the desired magnetic force, Cu-wire resistance, and the flux density in the weak corner. Also, the eight parameters selected have to avoid crush problems; that is, to avoid the leviated object (rotor) bumping into the upper or lower electromagnets. In GA evolution, for each chromosome we compute the total magnetic force according to (23) for the corresponding AMB system. With the knowledge of total magnetic force, we can evaluate the fitness value for the AMB system and produce the next generation by applying genetic operators (reproduction, crossover, and mutation). This process continues until a satisfactory design is obtained. We have verified that the final design successfully passed the FEM (finite element method) checking [42].

III. TD AND GA-BASED REINFORCEMENT LEARNING SYSTEM

The proposed TDGAR learning method is an hybrid GA algorithm [43]. Traditional simple GA's, though robust, are generally not the most successful optimization algorithm on any particular domain. Hybridizing a GA with domain-specific algorithms can produce an algorithm better than the GA and the current algorithms [44]–[50]. GA's may be crossed with various problem-specific search techniques to form a hybrid that exploits the global perspective of the GA (*global search*) and the convergence of the problem-specific technique (*local search*). In some situations, hybridization entails using the representation as well as optimization techniques already in use in the domain, while tailoring the GA operators to the new representation. According to this concept, the proposed TDGAR learning method is a hybrid of GA and the actor-critic architecture, which is a mature technique in the reinforcement learning domain. We shall introduce the structure and learning algorithm of the TDGAR learning system in the following subsections.

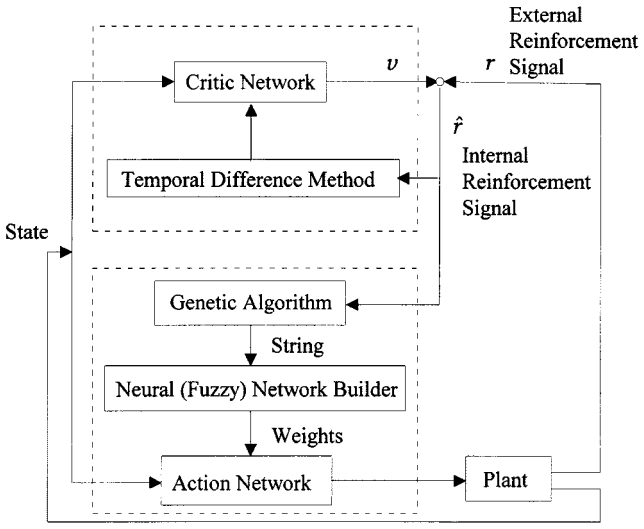


Fig. 5. Proposed TDGAR learning system.

A. Structure of the TDGAR Learning System

The proposed TDGAR learning system is constructed by integrating two feedforward multilayer networks. One neural network acts as a critic network for guiding the learning of the action network, which is a neural controller or a neural fuzzy controller for determining the outputs (actions) as shown in Fig. 5. The TDGAR learning system is basically in the form of the actor-critic architecture [14]. Since we want to solve reinforcement learning problems in which the external reinforcement signal is available only after a long sequence of actions have been acted on the environment, we need a multistep critic network to predict the external reinforcement signal. In the TDGAR learning system, the critic network models the environment such that it can perform a multistep prediction of the external reinforcement signal caused by the current action given by the action network. With the multistep prediction, the critic network can provide a more informative internal reinforcement signal to the action network. The action network can then determine a better action for the next time step, according to the current environment state and the internal reinforcement signal. The internal reinforcement signal from the critic network enables both the action network and the critic network to learn at each time step without waiting for the arrival of an external reinforcement signal, greatly accelerating the learning of both networks. The structures and functions of

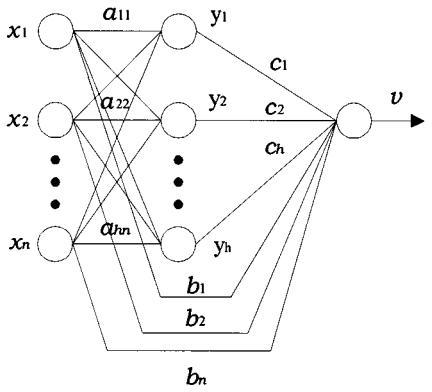


Fig. 6. Structure of the critic network in the TDGAR learning system.

the critic network and the action network are described in the following subsections.

1) The Critic Network: The critic network constantly predicts the reinforcement associated with different input states, and thus evaluates the "goodness" of the control actions determined by the action network. The only information received by the critic network is the state of the environment in terms of state variables and whether or not a failure has occurred. The critic network is a standard three-layer feedforward network with sigmoids in the hidden layer and output layer. The input to the critic network is the state of the plant, and the output is an evaluation of the state, denoted by v. This value is suitably discounted and combined with the external failure signal to produce the internal reinforcement signal, $\hat{r}(t)$.

Fig. 6 shows the structure of the critic network. It includes h hidden nodes and n input nodes including a bias node (i.e., $x_1, x_2, \ldots, x_n$). In this network, each hidden node receives n inputs and has n weights, while each output node receives $n+h$ inputs and has $n+h$ weights. The output of the node in the hidden layer is given by

$$y_i[t, t+1] = g\left(\sum_{j=1}^{n} a_{ij}[t] x_j[t+1]\right) \tag{26}$$

where

$$g(s) = \frac{1}{1+e^{-s}} \tag{27}$$

t and $t+1$ are successive time steps, and a_{ij} is the weight from the jth input node to the ith hidden node. The output node of the critic network receives inputs from the nodes in the hidden layer (i.e., y_i) and directly from the nodes in the input layer (i.e., x_j):

$$v[t, t+1] = g\left(\sum_{j=1}^{n} b_j[t] x_j[t+1] + \sum_{i=1}^{h} c_i[t] y_i[t, t+1]\right) \tag{28}$$

where v is the prediction of the external reinforcement value, b_j is the weight from the jth input node to output node, and c_i is the weight from the ith hidden node to output node.

The critic network evaluates the action recommended by the action network and represents the evaluated result as the internal reinforcement signal. The internal reinforcement signal is a function of the external failure signal and the change in state

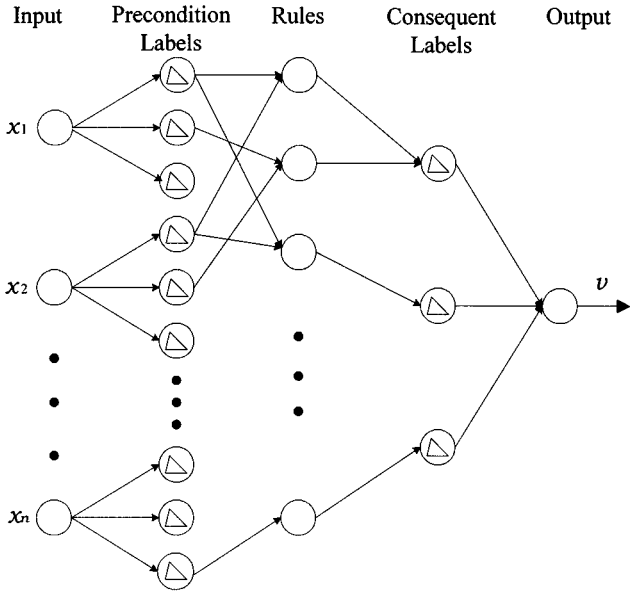


Fig. 7. Structure of a neural fuzzy controller as the action network.

evaluation based on the state of the system at time $t+1$

$$\hat{r}(t+1) = \begin{cases} 0, & \text{start state} \\ r[t+1] - v[t,t], & \text{failure state} \\ r[t+1] + \gamma v[t, t+1] - v[t,t], & \text{otherwise} \end{cases} \tag{29}$$

where $0 \le \gamma \le 1$ is the discount rate. In other words, the change in the value of v plus the value of the external reinforcement signal constitutes the heuristic or internal reinforcement signal, $\hat{r}(t)$, where the future values of v are discounted more the further they are from the current state of the system.

2) The Action Network: The action network determines a proper action acting on the environment (plant) according to the current environment state. If a neural controller is used, the structure of the action network is exactly the same as that of the critic network shown in Fig. 6. If a neural fuzzy controller is used, the structure is shown in Fig. 7. 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. The connections are feedforward, with each node performing a local computation. Layer 1 is the input layer, consisting of the real-value input variables. These can also be thought of as the linguistic variables of interest. No computation is done at these nodes. Nodes at layer 2 act as membership functions to represent the terms of the respective linguistic variable at layer 1. Each node at layer 3 is a rule node which represents one fuzzy logic rule. Thus, all layer-three nodes form a fuzzy rule base. Each layer 4 node corresponds to a consequent label. Its inputs come from all rules which use this particular consequent label. The node in layer 5 is an output node reprensenting one output linguistic variable.

The only information received by the action network is the state of the environment in terms of state variables and the internal reinforcement signal from the critic network. Since the GA is used to train the action network for control applications, the connection weights of the action network are encoded as a real-value string,

where for using neural fuzzy controller as the action network, the shapes and positions of the membership functions of the input/output variables are viewed as parameters to be learned. Initially, the GA generates a population of real-value strings randomly. An action network corresponding to each string then runs in a feedforward fashion to produce control actions acting on the environment according to (26) and (28). At the same time, the critic network constantly predicts the reinforcement associated with changing environment states under the control of the current action network. After a fixed time period, the internal reinforcement signal from the critic network will indicate the "fitness" of the current action network. This evaluation process continues for each string (action network) in the population. When each string in the population has been evaluated and given a fitness value, the GA can look for a better set of strings and apply genetic operators on them to form a new population as the next generation. Better actions can thus be chosen by the action network in the next generation. After a fixed number of generations, or when the desired control performance is achieved, the evolution process stops, and the string with the largest fitness value in the last generation is selected and decoded into the final action network. The detailed learning scheme for the action network will be discussed in the next subsection.

B. Learning Algorithm of the TDGAR Learning System

The flowchart of the TDGAR learning algorithm is shown in Fig. 8. In the following subsections, we first consider the reinforcement learning scheme for the critic network of the TDGAR system, and then introduce the GA-based reinforcement learning scheme for the action network of the TDGAR system.

1) Learning Algorithm for the Critic Network: When both the reinforcement signal and input patterns from the environment depend arbitrarily on the past history of the action network outputs and the action network only receives a reinforcement signal after a long sequence of outputs, the credit assignment problem becomes severe. This *temporal credit assignment* problem results because we need to assign credit or blame to each individual step in a long sequence leading up to eventual success or failure. Thus, to handle this class of reinforcement learning problems, we need to solve the temporal credit assignment problem along with solving the original structural credit assignment problem concerning attribution of network errors to different connections or weights. The solution to the temporal credit assignment problem in the TDGAR system is to use a multistep critic network that predicts the reinforcement signal at each time step in the period without any external reinforcement signal from the environment. This can ensure that both the critic network and the action network can update their parameters during the period without any evaluative feedback from the environment. To train the multistep critic network, we use a technique based on the temporal difference method, which is often closely related with the dynamic programming techniques [13], [51], [52]. Unlike the single-step prediction and the supervised learning methods which assign credit according to the difference between the predicted and actual outputs, the temporal difference methods assign credit according to the difference between temporally successive predictions. Note that the term "multistep prediction" used here means the critic network

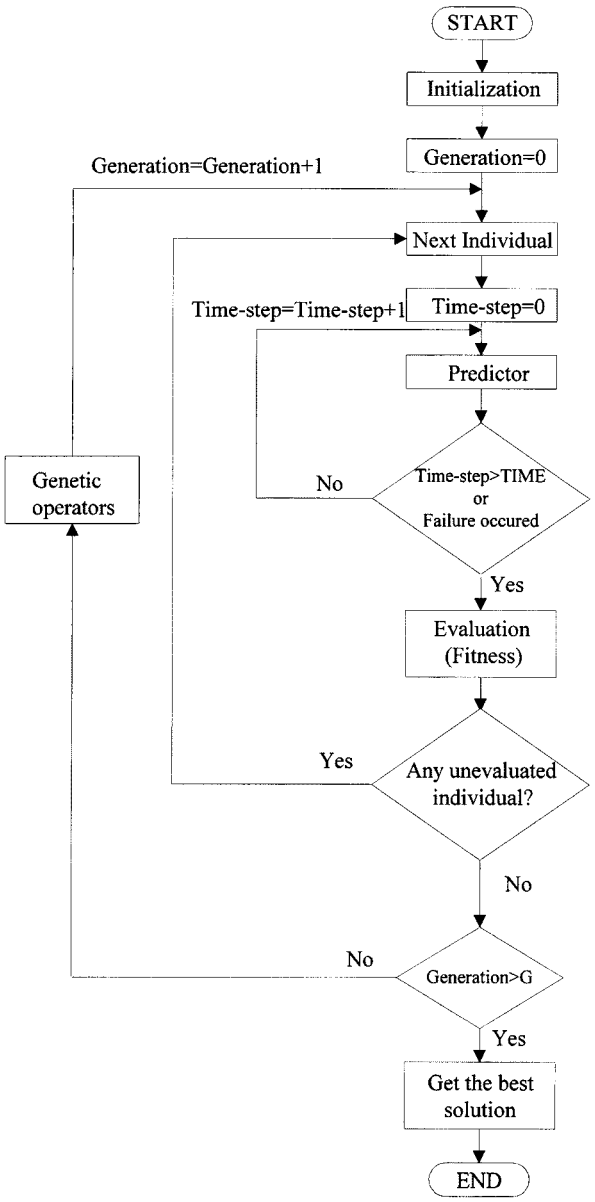


Fig. 8. Flowchart of the proposed TDGAR learning method.

can predict a value that will be available several time steps later, although it does such prediction at each time step to improve its prediction accuracy.

The goal of training the multistep critic network is to minimize the prediction error. It is similar to a reward/punishment scheme for updating the weights in the critic network. If positive (negative) internal reinforcement is observed, the values of the weights are rewarded (punished) by being changed in the direction which increases (decreases) its contribution to the total sum. The weights on the links connecting the nodes in the input layer directly to the nodes in the output layer are updated according to the following rule:

$$b_i[t+1] = b_i[t] + \eta \hat{r}[t+1] x_i[t] \tag{30}$$

where $\eta > 0$ is the learning rate and $\hat{r}[t+1]$ is the internal reinforcement signal at time $t+1$.

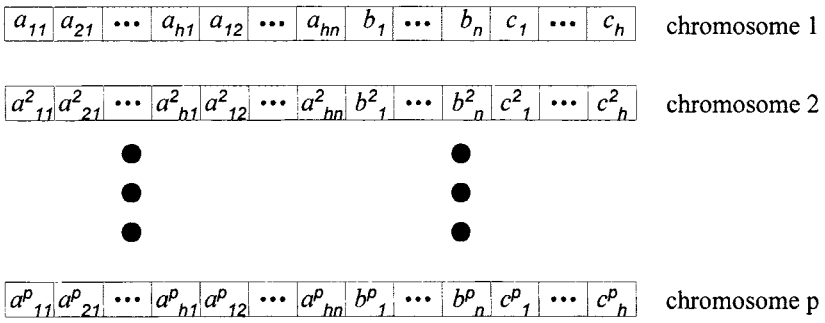


Fig. 9. Encoding of the neural network in Fig. 6 on chromsomes.

Similarly, for the weights on the links between the hidden layer and the output layer, we have the following weight update rule

$$c_i[t+1] = c_i[t] + \eta \hat{r}[t+1] y_i[t,t]. \tag{31}$$

The weight update rule for the hidden layer is based on a modified version of the error backpropagation algorithm [53]. Since no direct error measurement is possible (i.e., knowledge of correct action is not available), $\hat{r}$ plays the role of an error measure in the update of the output node weights. In Section III-A1, we have stated that v is the prediction of the external reinforcement value. Changes in v due to problem-state transitions are combined with the failure signal r to form $\hat{r}$ [see (29)]. For all states but those corresponding to failure, r is zero and $\hat{r}$ is just the difference between successive values of v. After learning, a positive change in the prediction of failure means the present state entered a state from which failure occures less often or further in the future than from the previous state. An action is considered desirable if it leads to a positive change in failure prediction and undesirable if it leads to a negative change. Hence, a positive (or negative) value of $\hat{r}$ results in an increase (or decrease) of the preceding state's evaluation, effectively shifting evaluations to earlier states. Therefore, the equation for updating the hidden weights is

$$a_{ij}[t+1] = a_{ij}[t] + \eta \hat{r}[t+1] y_i[t,t](1 - y_i[t,t]) \times \operatorname{sgn}(c_i[t]) x_j[t]. \tag{32}$$

Note that the sign of a hidden node's output weight is used, rather than its value. The variation is based on Anderson's empirical study [17] that the algorithm is more robust if the sign of the weight is used rather than its value. Their results suggest that this variation speeds learning and decreases the sensitivity to the value of the learning rate parameter η.

2) Learning Algorithm for the Action Network: The GA is used to train the action network by using the internal reinforcement signal from the critic network as the fitness function. Initially, the GA randomly generates a population of real-value strings, each of which represents one set of parameters for the action network. Roughly speaking, GA's manipulate strings of binary digits, 1's and 0's, called chromosomes which represent multiple points in the search space through proper encoding mechanism. GA's carry out simulated evolution on populations of such chromosomes. Bit string encoding is the most common encoding technique used by GA researchers because of its ease of creating and manipulating [20]. However, with a binary string encoding the resulting search time is much longer than that of using real-value string encoding [55]. Hence, the real-value encoding scheme instead of the normal binary encoding scheme is used here, so recombination can only occur between weights. An example in Fig. 9 shows the encoding of the action network on chromosomes with population size p . Also, one example illustrating the crossover operation for real-value encoding is demonstrated in Fig. 10. Another example illustrating the mutation operation for real-value encoding is demonstrated in Fig. 11. The details of the real-value-coded GA can be found in [40] and reference therein. A small population is used in our learning scheme. The use of small population reduces the exploration of the multiple (representationally dissimilar) solutions for the same network.

After a new real-value string is created, an interpreter takes this real-value string and uses it to set the parameters in the action network. The action network then runs in a feedforward fashion to control the environment (plant) for a fixed time period (determined by the constant "TIME" in Fig. 8) or until a failure occurs. At the same time, the critic network predicts the external reinforcement signal from the controlled environment and provides an internal reinforcement signal to indicate the "fitness" of the action network. In this way, according to a defined fitness function, a fitness value is assigned to each string in the population, where high fitness values mean good fit. It is very free to choose the fitness function, FIT; functional characteristics such as nonlinear, nondifferentiable, discontinuous, and positive are all permissible, because the GA only needs a fitness value assigned to each string. In this paper, we use the internal reinforcement signal from the critic network to define the fitness function. The fitness function is defined by

$$\mathrm{FIT}(t) = \frac{1}{|\hat{r}(t)|} \tag{33}$$

which reflects the fact that smaller internal reinforcement values mean higher fitness of the action network, where t is the current time step, $1 \le t \le \mathrm{TIME}$, and the constant "TIME" is a fixed time period during which the performance of the action network is evaluated by the critic network. If a failure signal occurs from the environment before the time limit (i.e., $t \le \mathrm{TIME}$), then an

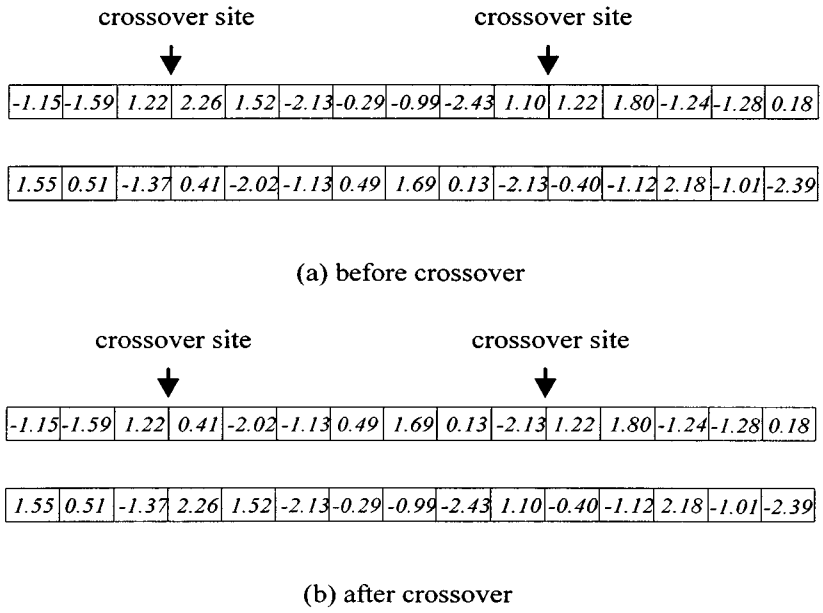


Fig. 10. Illustration of crossover operation on real-value strings.

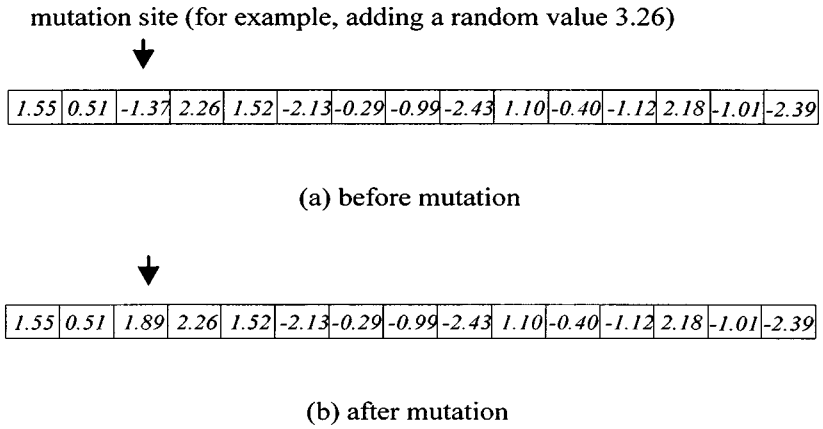


Fig. 11. Illustration of mutation operation on a real-value string.

action network having higher t will obtain higher fitness value. With the definition of fitness function in (33), the GA is not engaged at the start state of learning, so the divided-by-zero problem will not occure in the first beginning. However, there might be a divided-by-zero problem occurred in other situations. In such case, we can assign a defaulted big value to the evaluated action network as its fitness value. The above fitness function is different from that defined by Whitley and his colleagues [40]. Their relative measure of fitness takes the form of an accumulator that determines how long the experiment is still "success." Hence, a string (action network) cannot be assigned a fitness value until an external reinforcement signal arrives to indicate the final success or failure of the current action network.

When each string in the population has been evaluated and given a fitness value, the GA then looks for a better set of strings to form a new population as the next generation by using genetic operators (i.e., the reproduction, crossover, and mutation operators). In basic GA operators, the crossover operation can be generalized to multipoint crossover in which the number of crossover point (N_c) is defined. With N_c set to 1, generalized crossover reduces to simple crossover. The multipoint crossover can solve one major problem of the simple crossover; one-point crossover cannot combine certain combinations of features encoded on chromosomes. In the proposed GA-based reinforcement learning algorithm, we choose $N_c = 2$. Although we tried to use higher N_c values such as $N_c = 2$, $N_c = 3$, and $N_c = 4$ in our experiments, the results showed no remarkable change in the TDGAR learning system. For the mutation operator, since we use the real-value encoding scheme, we use a higher mutation probability (0.1) in our algorithm. This is different from the traditional GA's that use the binary encoding scheme. The latter are largely driven by recombination, not mutation. The above learning process continues to new generations until the number of generations meets a predetermined stop criterion. After the whole evolution process is stopped, the string with the largest fitness value in the last generation is selected and decoded into the final action network.

The major feature of the proposed hybrid GA learning scheme is that we formulate the internal reinforcement signal as the fitness function for the GA based on the actor-critic architecture.

In this way, the GA can evaluate the candidate solutions (the weights of the action network) regularly during periods without external feedback from the environment. The GA can thus proceed to new generations in fixed time steps [specified by the constant "TIME" in (33)] without waiting for the arrival of the external reinforcement signal. In other words, we can keep the time steps (TIME) for evaluating each string (action network) and the generation size (G) fixed in our learning algorithm (see the flowchart in Fig. 8), since the critic network can give predicted reward/penalty information to a string without waiting for the final success or failure. This can usually accelerate the GA learning since an external reinforcement signal may only be available at a time long after a sequence of actions has occurred in the reinforcement learning problems. This is similar to the fact that we usually evaluate a person according to his/her potential or performance during a period, not after he/she has done something really good or bad.

IV. Control of the Magnetic Bearing System

In this section, we shall apply the proposed TDGAR learning method to control the magnetic bearing system (Fig. 1) designed in Section II. Two controllers, a neural controller and a neural fuzzy controller, are trained and their performance is compared. The control problem is to keep around an operational point of the magnetic bearing system (i.e., to maintain the leviated object position of the magnetic bearing system around an equilibrium point, 0.196 mm). This system is nonlinear, autonomous and intrinsically unstable. There are three input state variables in this system: X_r, levitated object position; $\dot{X}_r$, speed of X_r; and I, the coil current. The only control action is the voltage V, supplied to the two E-shaped electromagnets. The physical constraints of the magnetic levitation system are: $0.192\text{ mm} \leq X_r \leq 0.2\text{ mm}$ and $-20.0v \leq V \leq 20.0v$. Since no exact teaching information is available, this is a typical reinforcement learning problem and the feedback failure signal serves as the external reinforcement signal.

In our experiments, we first apply the TDGAR learning system on the simplified dynamical model of our magnetic bearing system given in Section II to train the controller (action network) off-line. This will result in a TDGAR learning system with a good initial controller (action network) for the real magnetic bearing system. Then the TDGAR learning is performed continuously on the real magnetic bearing system on-line to obtain the final controller. Such a two-step learning procedure can shorten the number of trials required by the TDGAR learning on the real system and thus make the experiments more feasible, since the off-line trained action network has gained most of the control skills according to the simplified dynamic model of the real system. The equations of motion that we used are from (24) and (25):

$$X_r(t+1) = X_r(t) + \Delta \cdot \dot{X}_r(t) \tag{34}$$

$$\dot{X}_r(t+1) = \dot{X}_r(t) + \Delta \cdot \left[\frac{(F_1 - F_2 + f)}{m} - g\right] \tag{35}$$

$$I(t+1) = I(t) + \Delta \cdot \frac{(V - B\dot{X}_r(t) - RI(t))}{L} \tag{36}$$

where

$g = 9.8\text{ m/s}^2$	acceleration of gravity;
$m = 1.65\text{ kg}$	mass of the levitated object;
$\Delta = 0.001$	sampling interval;
$R = 3.3\text{ ohm}$	effective resistance of the magnetic circuit;
$f = 0.001$	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.

(37)

Notice that, since the real AMB system was built based on the mathematical model derived in Section II, where some key mechanical parameters were chosen through GA, and the final design successfully passed the FEM (finite element method) checking, the derived mathematical model should have captured the dynamical characteristics of the real system precisely. However, due to the measurement errors of some system parameters [e.g., the parameters in (37)] and the discretization error resulting from transforming a continuous system [(24) and (25)] to a discrete-time system [(34)–(36)] in digital simulations, (34)–(36) can only form a mathematical approximation of the real AMB system. Our testing in [42] showed that (34)–(36) and the real system have quite similar system responses (e.g., response shop, time constant) but differ in response details (e.g., overshoot value, ripples). Hence, the TDGAR learning system can learn the "skeleton" of the final controller in the off-line learning step via simplified dynamic model, and then is tuned finely in the on-line learning step via the real system.

The (external) reinforcement signal in this problem is defined as (38), shown at the bottom of the page. To allow the transient behavior of a controller, we set less strict conditions of failure in the beginning of learning. Hence, during the first 0.1 s of learning, the leviated object position at $X_r(t) < 0.192$ mm or $X_r(t) > 0.2$ mm will be treated as a failure. Later on, since we hope to control the leviated object position around the equilibrium point $X_r(t) = 0.196$ mm after 0.1 s, $X_r(t) < 0.1957$ mm or $X_r(t) > 0.1963$ mm for $t > 0.1$ s will also be treated

$$r(t) = \begin{cases} -1, & \text{if } X_r(t) < 0.192 \quad \text{or} \quad X_r(t) > 0.2 \quad \text{for } t \leq 0.1\text{ s}, \quad \text{and} \\ & \text{if } X_r(t) < 0.1957 \quad \text{or} \quad X_r(t) > 0.1963 \quad \text{for } t > 0.1\text{ s}, \\ 0, & \text{otherwise.} \end{cases} \tag{38}$$

as a failure. It is understood that the external reinforcement signal may only be available after a long sequence of time steps in this failure avoidance task. Hence we need the multistep prediction capability of the critic network in the TDGAR learning system.

In the first set of experiments, a neural network in Fig. 6 is used as the action network. We use the fitness function defined in (33), i.e., $\mathrm{FIT}(t) = (1/|\hat{r}(t)|)$, in the TDGAR learning system, where $\hat{r}(t)$ is the internal reinforcement signal from the critic network. The critic network and action network both have four input nodes, four hidden nodes, and one output node. Hence, there are 24 weights in each network. A bias node fixed at 0.5 is used as the fifth input to the network; a weight from the bias node to a hidden node (or to the output node) in effect changes the threshold behavior of that node. The learning parameters used in the TDGAR system are the learning rate $\eta = 0.1$, the population sizes $\mathrm{POP} = 100$, the time limit $\mathrm{TIME} = 1000$, and the generation size G is not limited here. Initially, we set all the weights in the critic network and action network to random values between -2.5 and 2.5. To assign credit to the individual action of the action sequence, the critic network evaluates each individual and sort the population according to the fitness. In reproduction, we use the roulette wheel selection technique [43] to pick two parent individuals, and then apply two-point crossover and mutation operators to these two parents to generate two new individuals according to the crossover probability and mutation probability. After 50 occurences of reproduction, 100 new individuals are produced. These 100 new individuals are evaluated, their fitness are calculated, and they replace the current 100 individuals to form the next generation. The crossover probability is set as 0.9 and the crossover sites are randomly seclected. For the mutation operator, the mutation probablity is set as 0.1, and the mutation site is randomly seclected and a random value with range ± 2.0 is added to the chosen site. A control strategy is deemed successful if the individual with maximum fitness could maintain the leviated object position around the equilibrium point [i.e., to satisfy (38)]. Then the GA stops; otherwise the evolution of the next generation starts.

Fig. 12 shows the control performance of the neural controller trained by the TDGAR learning method on the real magnetic bearing system for different initial positions. Note that although the trained neural controller can levitate the rotor and keep it balanced around the equilibrium point, it cannot stabilize the rotor well and causes obvious oscillations. Fig. 13 shows one set of control signals (the other sets of control signals are quite similar to this). It shows that the power needed to control the magnetic bearing system is large. This dose not satisfy the goal of low power consumption.

In the second set of experiments, the neural fuzzy network in Fig. 7 is used as the action network. There are two input variables, error (the difference of the levitated object position and the operational point) and error change (the speed of the levitated object), and one output variable, the voltage applied to the driver of the magnetic bearing system. Fig. 14 shows the types of membership functions used for the input and output variables. Each variable has three fuzzy partitions, with membership functions labeled as $\{\mathrm{NE}, \mathrm{ZE}, \mathrm{PO}\}$, where $\mathrm{NE} =$ Negative, $\mathrm{ZE} =$ Zero, and $\mathrm{PO} =$ Positive. Nine fuzzy control rules are designed for the magnetic bearing system as follows:

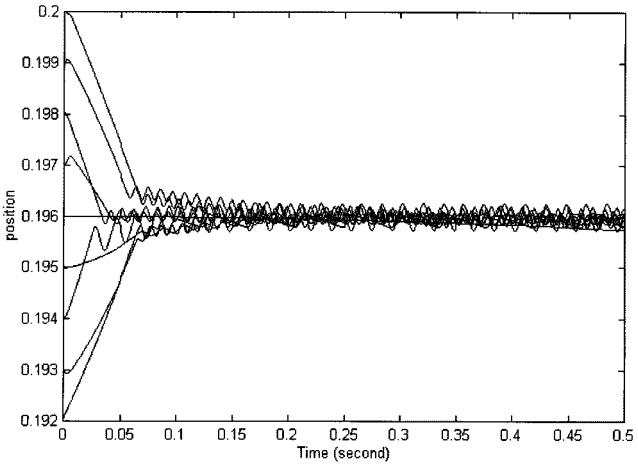


Fig. 12. Position variations of the rotor under the control of the neural controller trained by the TDGAR learning method for different initial positions.

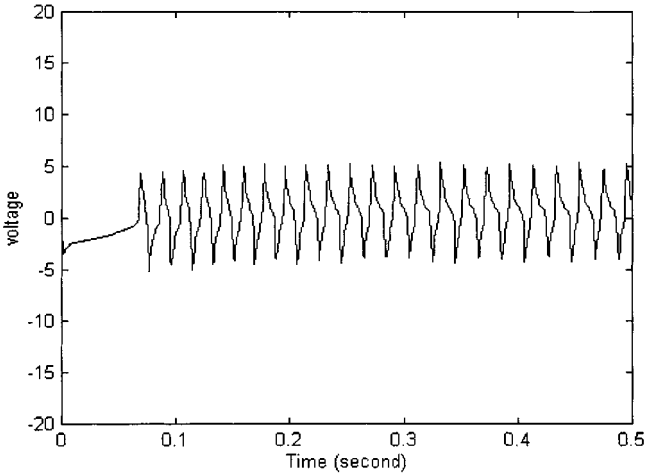


Fig. 13. One set of control signals corresponding to the control performance of Fig. 12.

R1: IF error is NE AND *error change* is NE THEN *output* is NE

R2: IF *error* is NE AND *error change* is ZE THEN *output* is NE

R3: IF *error* is NE AND *error change* is PO THEN *output* is ZE

R4: IF *error* is ZE AND *error change* is NE THEN *output* is NE

R5: IF *error* is ZE AND *error change* is ZE THEN *output* is ZE

R6: IF *error* is ZE AND *error change* is PO THEN *output* is PO

R7: IF *error* is PO AND *error change* is NE THEN *output* is ZE

R8: IF *error* is PO AND *error change* is ZE THEN *output* is PO

R9: IF *error* is PO AND *error change* is PO THEN *output* is PO

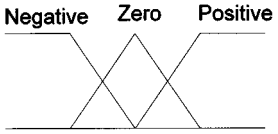


Fig. 14. Types of membership functions used for the input and output variables in the neural fuzzy controller in Fig. 7.

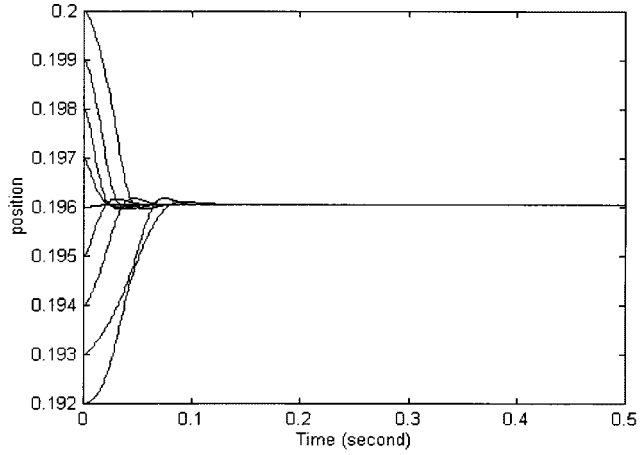


Fig. 15. Position variations of the rotor under the control of the neural fuzzy controller trained by the TDGAR learning method for different initial positions.

In the GA learning, an individual of the population represents one trial set of fuzzy membership functions. As mentioned above, there are three input variables (X_r, $\dot{X}_r$, and I) in the magnetic bearing system and one control action, the voltage V, supplied to the system. Each variable has three fuzzy partitions, with membership functions labeled as Negative, Zero, and Positive (as shown in Fig. 14), respectively. A membership function is specified by three parameters, so each variable has nine parameters. Because the relationship between V and I is $V = IR$, where R is the effective resistance of the magnetic circuit, chooosing either V or I as a variable for learning is enough. Therefore, only three variables are used (X_r, $\dot{X}_r$, and V) for GA learning here. These 27 parameters (9 membership functions $\times$ 3) are concatenated into a real-value string as an individual for GA learning. For the input variables, the universe of discourse is $X_r \in [-5, 5]$, $\dot{X}_r \in [-0.2, 0.2]$, and the universe of discourse of the output variable is $V \in [-20, 20]$. The initial population is randomly generated. The learning parameters are the same as before, the fitness function is defined as in (33), the (external) reinforcement signal is defined as in (38), and the critic network is kept the same as in the neural controller case.

Fig. 15 shows the control performance of the neural fuzzy controller trained by the TDGAR learning method on the real magnetic bearing system for different initial positions. They appear to be better than those of the previous neural controller in Fig. 12. Fig. 16 shows the control signals for different initial conditions. It is observed that the trained neural fuzzy controller does achieve the goal of low power consumption. Fig. 17 shows some of the membership functions trained by the TDGAR learning method for the input and output variables. Note that the incorporation of expert knowledge into the neural fuzzy network does speed up the learning, and generates better controllers.

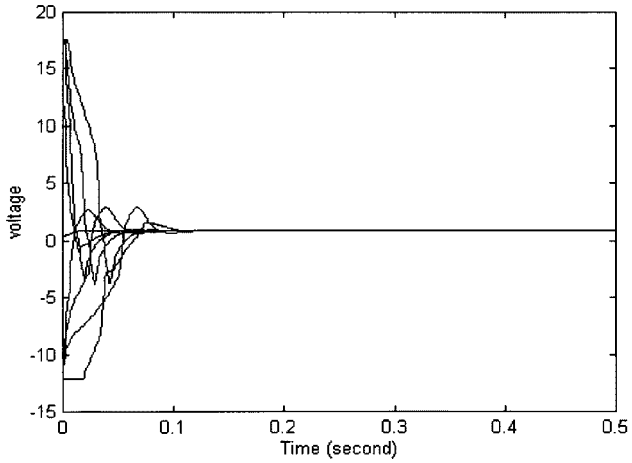


Fig. 16 Sets of control signals corresponding to the control performance of Fig. 15.

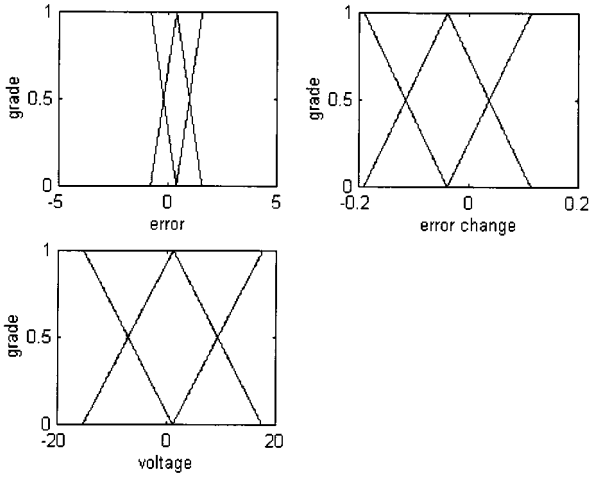


Fig. 17. Examples of the learned membership functions for the neural fuzzy controller after the TDGAR learning.

V. Conclusion

This paper integrates the temporal difference (TD) technique, gradient descent method, and genetic algorithm (GA) into the actor-critic architecture to form a new reinforcement learning system, called the TDGAR learning system. Using the TDGAR learning system, we can train a neural controller or a neural fuzzy controller for the plant according to a simple reinforcement signal. The proposed TDGAR learning method makes the design of neural fuzzy controllers or neural controllers more feasible and practical for real-world applications, since it greatly lessens the quality and quantity requirements of the teaching signals, and reduces the long training time of a pure GA approach. In this paper, we also design and build a magnetic bearing system by using the GA for parameter selection. The proposed TDGAR learning method has been applied to control the magnetic bearing system in practice, and achieves very satisfactory results.

REFERENCES

[1] R. L. Hollis, S. E. Salcudean, and A. P. Allan, "A six-degree-of-freedom magnetically levitated variable compliance fine-motion-wrist: Design, modeling and control," *IEEE Trans. Robot. Automat.*, vol. 7, pp. 320–332, Mar. 1991.

[2] T. Eiichi, K. Kiyoshi, and M. Yuji, "An ionization chamber using a magnetically suspended electrode and its application to environmental radiation monitoring," *IEEE Trans. Nucl. Sci.*, vol. 38, pp. 491–493, Feb. 1991.

[3] G. Schweitzer, H. Bleuler, and A. Traxler, *Active Magnetic Bearing*, Zürich, Switzerland, 1994.

[4] H. Tsuruga, "Superconductive maglev system on the Yamanashi maglev test line," *SAE Maglev*, pp. 7–17, 1992.

[5] W. W. Dickhart, III, "The trasrpid maglev system," *SAE Maglev Levitation Technology for Advanced Transit System*, pp. 13–21, 1989.

[6] F. Mastumura, T. Namerikawa, K. Hagiwara, and M. Fujita, "Application of gain schedule H_∞ robust controllers to a magnetic bearing," *IEEE Trans. Control Syst. Technol.*, vol. 4, no. 5, pp. 484–493, 1996.

[7] A. Charara, J. De Miras, and B. Caron, "Nonlinear control of a magnetic levitation system without premagnetization," *IEEE Trans. Control Syst. Technol.*, vol. 4, pp. 513–523, May 1996.

[8] M. S. de Queiroz and D. M. Dawson, "Nonlinear control of active magnetic bearings: A backstepping approach," *IEEE Trans. Control Syst. Technol.*, vol. 4, pp. 545–552, May 1996.

[9] K. Y. Lum, V. T. Coppola, and D. S. Bernstein, "Adaptive autocentering control for an active magnetic bearing supporting a rotor with unknown mass imbalance," *IEEE Trans. Control Syst. Technol.*, vol. 4, pp. 587–597, May 1996.

[10] A. E. Rundell, S. V. Drakunov, and R. A. DeCarlo, "A sliding mode observer and controller for stabilization of rotation motion of a vertical shaft magnetic bearing," *IEEE Trans. Control Syst. Technol.*, vol. 4, pp. 598–608, May 1996.

[11] A. G. Barto and M. I. Jordan, "Gradient following without backpropagation in layered network," in *Proc. Int. Joint Conf. Neural Networks*, vol. II, San Diego, CA, 1987, pp. 629–636.

[12] S. Mikami and Y. Kakazu, "Extended stochastic reinforcement learning for the acquisition of cooperative motion plants for dynamically constrained agents," in *Proc. IEEE Conf. Systems, Man and Cybernetics Control*, New York, 1993, pp. 257–262.

[13] A. G. Barto, R. S. Sutton, and C. W. Anderson, "Neuronlike adaptive elements that can solve difficult learning control problem," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-13, pp. 834–847, 1983.

[14] R. S. Sutton, "Temporal Credit Assignment in Reinforcement Learning," Ph.D. dissertation, Univ. Massachusetts, Amherst, MA, 1984.

[15] A. G. Barto and P. Anandan, "Pattern-recognizing stochastic learning automata," *IEEE Trans. Syst,. Man, Cybern.*, vol. SMC-15, pp. 360–375, 1985.

[16] R. J. Williams, "A class of gradient-estimating algorithms for reinforcement learning in neural networks," in *Proc. Int. Joint Conf. Neural Networks*, vol. II, San Diego, CA, 1987, pp. 601–608.

[17] C. W. Anderson, "Strategy learning with multilayer connectionist representations," in *Proc. 4th Int. Workshop on Mach. Learn.*, Irvine, CA, June 1987, pp. 103–114.

[18] 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.

[19] C. T. Lin and C. S. G. Lee, "Reinforcement structure/parameter learning for an integrated fuzzy neural network," *IEEE Trans. Fuzzy Syst.*, vol. 2, pp. 46–63, Feb. 1994.

[20] J. H. Holland, *Adaptation in Natural and Artificial System*. Ann Arbor, MI: Univ. of Michigan Press, 1975.

[21] T. Mariti, H. Ishibuchi, and K. H. Lee, "Application of two-objective genetic algorithm to flowshop scheduling problems with interval processing time," in *Proc. 4th European Congress Intelligent Techniques and Soft Computing*, Germany, 1996, pp. 443–447.

[22] S. M. Sait and H. Youssef, "Timing-influenced general-cell genetic floorplanner ," *Microelectron. J.*, vol. 28, no. 2, pp. 151–166, Feb. 1997.

[23] S. Aiguo, Z. Qingjun, and H. Weiyi, "Identification and control of bilateral telerobot with time delay," in *Proc. IEE Int. Conf. Intelligent Robots and Systems, IROS'96*, New York, USA, 1996, pp. 1353–1358.

[24] E. Cox, "A model-free trainable fuzzy system for the analysis of financial time-series data with fuzzy set morphology rule association optimization through a genetic optimizer," in *Proc. of AI Applications on Wall Street*. Gaithersburg, MD, 1993, pp. 280–285.

[25] T. H. Heung and T. K. Ho, "Hierarchical fuzzy logic traffic control at a road junction using genetic algorithms," in *Proc. IEEE Int. Conf. Fuzzy Systems*, New York, 1998, pp. 1170–1175.

[26] S. Tanterdtid, W. Steanputtanagul, and W. Benjapolakul, "Adaptive ATM network configuration based on the modified genetic algorithm," in *ACE: Proc. World Telecommunications Conf. Global Network Evolution: Convergence or Collision?*. Toronto, Ont., Canada, 1997, pp. 619–625.

[27] T. Kawabe, T. Tagami, and T. Katayama, "A genetic algorithm based minimax optimal design of robust I-PD controller," in *Proc. IEE Int. Conf. Control*, London, U.K., 1996, pp. 436–441.

[28] A. Delgado, L. Puigjaner, K. Sanjeevan, and I. Sole, "Hybrid system: Neural networks and genetic algorithms applied in nonlinear regression and time series forecasting," in *Proc. 12th Symp. Computational Statistics*, Heidelberg, Germany, 1996, pp. 217–222.

[29] H. Kobayashi, C. C. Hsu, S. I. Yamada, and H. Fujikawa, "A design of a model reference fuzzy adaptive controller for linear systems with time delay using MSGA," in *Proc. IEEE Int. Conf. Industrial Electronics, Control and Instrumentation, IECON'97*, New York, 1997, pp. 1040–1045.

[30] Z. Zhang, J. Rayner, A. Cheetham, and T. Lund, "A simplified fuzzy model to mimic a nonlinear system, applied to a plasma source," *IEEE Trans. Plasma Sci.*, vol. 25, pp. 27–36, Feb. 1997.

[31] C. Karr, "Genetic algorithms for fuzzy controllers," *AI Expert*, vol. 2, pp. 27–33, 1991.

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

[33] K. Kropp, "Optimization of fuzzy logic controller inference rules using a genetic algorithm," in *Proc. EUFIT'93*, Aachen, Germany, 1993, pp. 1090–1096.

[34] S. Harp, T. Samad, and A. Guha, "Designing application-specific neural networks using the genetic algorithm," in *Neural Information Processing Systems*. San Mateo, CA: Morgan Kaufman, 1990, vol. 2.

[35] J. D. Schaffer, R. A. Caruana, and L. J. Eshelman, "Using genetic search to exploit the emergent behavior of neural networks," *Physica D*, vol. 42, pp. 244–248, 1990.

[36] Z. J. Yang, T. Hachino, and T. Tsuji, "On-line identification of continuous time-delay systems combining least-squares techniques with a genetic algorithm," *Int. J. Control*, vol. 66, no. 1, pp. 23–42, Jan. 1997.

[37] K. S. Tang, K. F. Man, S. Kwong, and Q. He, "Genetic algorithms and their applications," *IEEE Signal Processing Mag.*, vol. 13, pp. 22–37, Nov. 1996.

[38] Z. Zibo and F. Naghdy, "SISO system identification using genetic algorithms," in *Proc. Electrical Engineering Congr., EEC'94*. Barton, ACT, Australia, 1994, pp. 337–343.

[39] T. Nakatsuji, S. Seki, S. Shibuya, and T. Kaku, "Artificial intelligence approach for optimizing traffic signal timing on an urban road network," *Trans. Inst. Syst., Contr. Inform. Eng.*, vol. 7, no. 11, pp. 470–478, 1996.

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

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

[42] C. T. Lin and C. H. Wu, "Design and Realization of a Magnetic Bearing System," Dept. Elect. Contr. Eng., National Chiao-Tung Univ., Hsinchu, Taiwan, R.O.C., Tech. Rep. CN97-10.

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

[44] *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, New York, 1991.

[45] D. Adler, "Genetic algorithms and simulated annealing: A marriage proposal," in *Proc. IEEE Int. Conf. Neural Networks*, vol. II, San Francisco, CA, 1993, pp. 1104–1109.

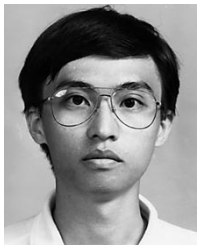
[46] L. Tsinas and B. Dachwald, "A combined neural and genetic learning algorithm," *Proc. IEEE Int. Conf. Neural Networks*, vol. I, pp. 770–774, 1994.

[47] V. Petridis, S. Kazarlis, A. Papaikonomou, and A. Filelis, "A hybrid genetic algorithm for training neural networks," in *Artificial Neural Networks 2*, I. Aleksander and J. Taylor, Eds. Amsterdam, The Netherlands: North Holland, 1992, pp. 953–956.

[48] J. S. Yim and C. M. Kyung, "Datapath layout optimization using genetic algorithm and simulated annealing," *Proc. Inst. Elect. Eng., Comput. Digital Techniques*, vol. 145, no. 2, pp. 1170–1175, Mar. 1998.

[49] M. Salami, "Genetic algorithm processor on reprogrammable architectures," in *Proc. 5th Annual Conf. Evolutionary Programming*. Cambridge, MA, 1996, pp. 355–361.

[50] P. M. Stanfield, R. E. King, and T. J. Hodgson, "Multi-objective stochastic scheduling of job ready times," *Ann. Oper. Res.*, vol. 70, pp. 221–239, 1997.
[51] R. S. Sutton, "Learning to predict by the methods of temporal difference," *Mach. Learn.*, vol. 3, pp. 9–44, 1988.
[52] P. J. Werbos, "A menu of design for reinforcement learning over time," in *Neural Networks for Control*, W. T. Miller III, R. S. Sutton, and P. J. Werbos, Eds. Cambridge, MA: M.I.T. Press, 1990, ch. 3.
[53] D. Rumelhart, G. Hinto, and R. J. Williams, "Learning internal representation by error propagation," in *Parallel Distributed Processing*, D. Rumelhart and J. McCelland, Eds. Cambridge, MA: MIT Press, 1986, pp. 318–362.
[54] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Mach. Learn.*, vol. 8, pp. 229–256, 1992.
[55] D. Whitley and T. Starkweather, "Optimizing small neural networks using a distributed genetic algorithm," *Int. Joint Conf. Neural Networks*, vol. 1, pp. 206–209, 1990.



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, where he is currently a Professor of Electrical and Control Engineering. He also serves 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 co-author 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 45 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. Dr. Lin 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.

Chong-Ping Jou was born in Taiwan, R.O.C., in 1960. He received the B.S. degree in electrical engineering from the Chung-Cheng Institute of Technology, Taiwan, in 1982, the M.S. degree in electrical engineering from the National Taiwan Institute of Technology in 1988, and the Ph.D. degree from the Department of Electrical and Control Engineering from Chiao-Tung University, Hsinchu, Taiwan, in 1999.

He is currently an Assistant Researcher at the Chung-Shan Institute of Science and Technology, Tao-Yuan, Taiwan. His research interests include neural network, chaos, and fuzzy control.