Using an Efficient Immune Symbiotic Evolution Learning for Compensatory Neuro-Fuzzy Controller

Cheng-Hung Chen, *Student Member, IEEE*, Cheng-Jian Lin, *Member, IEEE*, and Chin-Teng Lin, *Fellow, IEEE*

***Abstract*—This paper presents an efficient immune symbiotic evolution learning (ISEL) algorithm for the compensatory neuro-fuzzy controller (CNFC). The proposed ISEL method includes three major components--initial population, subgroup symbiotic evolution, and immune system algorithm. First, the self-clustering algorithm that determines proper input space partitioning and finds the mean and variance of the Gaussian membership functions and number of rules is applied to the initial population. Second, the subgroup symbiotic evolution method that uses each subantibody represents a single fuzzy rule and the evolution of the rule itself. Third, the immune system algorithm uses the clonal selection principle, such that antibodies between others of high similar degree are canceled, and these antibodies, after processing, will have higher quality, accelerating the search, and increasing the global search capacity. Finally, the proposed CNFC with ISEL (CNFC–ISEL) method is adopted to solve several nonlinear control problems. The simulation results have shown that the proposed CNFC–ISEL can outperform other methods.**

***Index Terms*—Compensatory fuzzy operator, immune system algorithm, neuro-fuzzy network, self-clustering algorithm (SCA), symbiotic evolution.**

I. Introduction

THE CONCEPT of fuzzy systems and artificial neural networks for resolving control problems has been extensively studied in recent years [1], [2], because classical control theory usually requires a mathematical model to design the controller. However, the inaccuracy of the mathematical modeling of the plants usually degrades the performance of the controller, especially for nonlinear and complex control problems [3]. On the contrary, the fuzzy system controller and the artificial neural network controller offer a key advantage over traditional adaptive control systems. Restated, they do not require mathematical models of the plants. Although traditional neural networks can learn from data and feedback, the meaning associated with each neuron and each weight in the network is not easily understood. Alternatively, fuzzy logical models are easily appreciated, because they use linguistic terms and the structure of if–then rules. However, fuzzy systems lack the learning capacity to fine-tune fuzzy rules and membership functions. In contrast to pure neural networks or fuzzy systems, neuro-fuzzy network representations have emerged as a highly effective approach to solve many problems [4]–[6].

Recent developments in genetic algorithms (GAs) have provided a method for fuzzy system and neuro-fuzzy network design. GAs represent highly effective techniques for evaluating system parameters and finding global solutions while optimizing the overall structure. Thus, many researchers have developed GAs to implement fuzzy systems and neuro-fuzzy networks in order to automate the determination of parameters and structures [7]–[16]. Lee and Takagi [7] proposed an automatic fuzzy system design method that integrates membership functions, the number of fuzzy rules, and the rule consequent by the GA. Ng and Li [8] applied a string (chromosomes) in the GA to optimize sophisticated membership functions for a nonlinear water-level control system. They concentrated on encoding all of the fuzzy rules into the chromosome while keeping the membership function fixed [9]. A small number of significant fuzzy if–then rules were selected by GA method by Ishibuchi *et al.* [10]. Lim *et al.* [11] described a paradigm for learning fuzzy rules using GA. The n rules were selected from the possible rule combinations. Evolutionary fuzzy expert systems have been discussed in which the membership function shapes and types and the fuzzy rule set are evolved using a GA by Shi *et al.* [12]. Seng *et al.* [13] proposed a neuro-fuzzy network that is based on the radial basis function neural network, all of whose parameters are simultaneously tuned using GA. A flexible position coding strategy of the neuro-fuzzy network parameters is also implemented to yield near optimal solutions. Lam *et al.* [14] presented the fuzzy nonlinear control systems that use GA with arithmetic crossover and nonuniform mutation. Karr [15] used a GA to generate membership functions for a fuzzy system. In Karr's work, a user needs to declare an exhaustive rule set and then use a GA to design only the membership functions. In [16], a fuzzy controller design method that used GAs to find the membership functions and the rule sets simultaneously was proposed. In [15] and [16], the input space was partitioned into a grid. In this grid scheme, a partitioned grid is taken as the initial state at the beginning of learning and is simple to implement. Unfortunately, the number of fuzzy rules (i.e., the length of each chromosome in the GA) increased exponentially as the number of inputs increased. However, such GAs encounter some important issues: 1) The input space is partitioned into a grid; 2) the number of fuzzy rules is determined by trial-and-error or by a expert system; 3) a fuzzy system was encoded into a chromosome; and 4) the rate of convergence of the evolution process is low.

Manuscript received May 4, 2006; revised February 8, 2007 and April 23, 2007; accepted May 27, 2007. First published April 30, 2008; current version published June 11, 2009. This work was supported in part by the Ministry of Economic Affairs, Taiwan, R.O.C., under Grant 96-EC-17-A-02-S1-032 and in part by the National Science Council, Taiwan, R.O.C., under Grant NSC 95-2221-E-009-180.

C.-H. Chen is with the Department of Electrical and Control Engineering, National Chiao-Tung University, Hsinchu 300, Taiwan, R.O.C. (e-mail: chchen.ece93g@nctu.edu.tw).

C.-J. Lin is with the Department of Computer Science and Information Engineering, National Chin-Yi University of Technology, Taichung County 411, Taiwan, R.O.C. (e-mail: cjlin@ncut.edu.tw).

C.-T. Lin is with the Department of Computer Science, National Chiao-Tung University, Hsinchu 300, Taiwan, R.O.C., with the Department of Electrical and Control Engineering, National Chiao-Tung University, Hsinchu 300, Taiwan, R.O.C., and also with the Brain Research Center, University System of Taiwan, Hsinchu 300, Taiwan, R.O.C. (e-mail: ctlin@mail.nctu.edu.tw).

Digital Object Identifier 10.1109/TFUZZ.2008.924186

This study presents an efficient immune symbiotic evolution learning (ISEL) algorithm for compensatory neuro-fuzzy controller (CNFC). The proposed CNFC is based on our previous research [17], [18]. The ISEL method includes three major components--initial population, subgroup symbiotic evolution, and immune system algorithm. First, the self-clustering algorithm (SCA) that determines proper input space partitioning and finds the mean and variance of the membership function and the number of rules is applied to the initial population. Second, the subgroup symbiotic evolution method in which each subantibody represents a single fuzzy rule differs from traditional GAs. A fuzzy system with R-rules is constructed by selecting and combining R subantibodies from a given population, and allowing the evolution of the rule itself. Third, the immune system algorithm uses the clonal selection principle to cancel antibodies between others of high similar degree, and these antibodies, after the process, will be of higher quality, accelerating the search and increasing the global search capacity. The advantages of the proposed ISEL method are summarized as follows: 1) The proposed SCA method can online cluster the input partitions and considers the variation of each dimension for the input data; 2) the subgroup symbiotic evolution method uses the subgroup-based population to evaluate the fuzzy rules locally; and 3) the adopted immune system algorithm can accelerate the search and increase global search capacity.

The rest of this paper is organized as follows. Section II describes the basic concept of symbiotic evolution and immune system. The structure of the CNFC is presented in Section III. Next, Section IV describes the SCA. An ISEL method is proposed in Section V. Section VI presents the results of the simulation of several nonlinear control problems. The conclusion and future works are finally drawn in Section VII.

II. Symbiotic Evolution and Immune System

This section describes basic concepts concerning symbiotic evolution and the immune system. The specialization property of symbiotic evolution and the immune system can match the local property of the neuro-fuzzy network. Therefore, the development of a neuro-fuzzy network based on symbiotic evolution and the immune system is valuable.

A. Basic Concept of Symbiotic Evolution

The notion of symbiotic evolution [19] is similar to the implicit fitness sharing used in an immune system model. The authors evolve artificial antibodies to match or detect artificial antigens. Each antibody can only match a single antigen, and different antibodies are needed to protect effectively against various antigens. These antibodies consist in separating the population into subpopulations and changing the way fitness values are assigned by fitness sharing. In the proposed method, an antibody is selected for replacement by randomly choosing a subset of the population and then selecting the member of that subset that is most similar to the new antibody. Therefore, summing the fitness values of all possible combinations of that antibody with other current antibodies and dividing the sum by the total number of combinations yields the fitness of an antibody. Moriarty and Miikkulainen [20] proposed a reinforcement learning method called symbiotic, adaptive neuro-evolution that evolves a population of neurons through GAs to form a neural network. The GA adopted by Juang *et al.* [21] is based upon symbiotic evolution, which, when applied in fuzzy controller design, complements the local mapping property of a fuzzy rule. Jamei *et al.* [22] exploited the capacity of symbiotic evolution, as a generic methodology, to elicit a fuzzy rule of the Mamdani type.

As shown in [20]–[22], partial solutions can be characterized as specializations. The specialization property ensures diversity, which prevents a population from converging to suboptimal solutions. A single partial solution cannot "take over" a population since there must be other specializations present. Unlike the standard evolutionary approach, which always causes a given population to converge, hopefully at the global optimum, but often, at a local one, the symbiotic evolution finds solutions in different, unconverted populations [20]–[22].

The basic idea of symbiotic evolution is that an individual is used to represent a single fuzzy rule. A fuzzy system is formed when several individuals, which are randomly selected from a population, are combined. With the fitness assignment performed by symbiotic evolution and with 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 the fuzzy system is known and 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 to use crossover operations, followed by observing the performance of the offspring.

B. Basic Concept of the Immune System

The natural immune system is the central paradigm of the proposed approach. In the function of an immune system, the diverse antibodies eliminate the antigens, while the lymphocytes produce the antibodies through the clonal proliferation [23]. The diverse antibodies can recognize all cells within the body as either antibodies or antigens. Therefore, the immune system of an organism includes a huge diversity of antibodies, waiting for many possible specific antigens. When an antigen is encountered, the number of antibodies that can kill it increases by many orders of magnitude. Therefore, clonal selection and affinity maturation principles are used to explain how the immune system responds to antigens and how it revises its ability to recognize eliminate antigens [24]. Recently, Chun *et al.* [25] proposed a method that employs the immune algorithm, which is different from GA in memory-educating system and the production system of various antibodies, as a search method to optimize the shape of an electromagnetic device. Jiao *et al.* [26] proposed the immune GA based on the theory of immunity, which mainly constructs an immune operator accomplished by a vaccination and an immune selection, into the canonical GA.

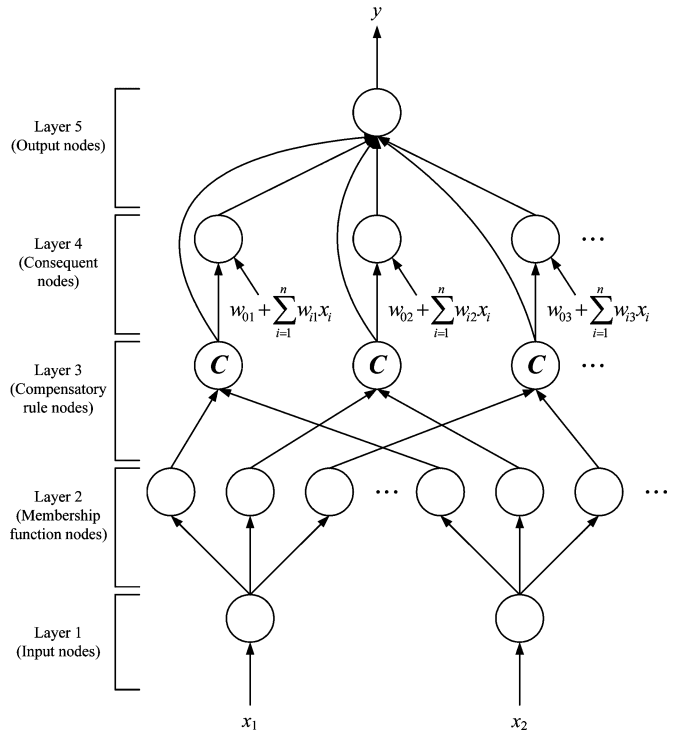

Fig. 1. Structure of the proposed CNFC.

III. Structure of Compensatory Neuro-Fuzzy Controller

The section describes CNFC [17], [18]. Compensatory operators are used to optimize fuzzy logic reasoning and to select optimal fuzzy operators. Therefore, an effective neuro-fuzzy controller should be able not only to adaptively adjust fuzzy membership functions but also to dynamically optimize adaptive fuzzy operators. Fig. 1 shows the structure of the CNFC, which is systematized into N input variables, R-term nodes for each input variable, M output nodes, and $N \times R$ membership function nodes. The CNFC consists of five layers, $R \times (N \times 2 + 2 + (N+1) \times M)$ parameters and $N + (N \times R) + 2 \times R + M$ nodes, where R denotes the number of existing rules. Nodes in layer 1 are input nodes, which represent input variables. Nodes in layer 2 are called membership functions nodes to express the input fuzzy linguistic variables. Nodes in this layer are used to calculate Gaussian membership values. Each node in layer 3 is called a compensatory rule node, and nodes in layer 4 are called consequent nodes. The number of nodes in layer 3 equals the number of compensatory fuzzy sets that correspond to each external linguistic input variable. Each compensatory rule node has a corresponding consequent node, which calculates a weighted linear combination of input variables. Nodes in layer 5 are called output nodes, each node of which is an individual output of the system.

The CNFC realizes a fuzzy model in the following form.

$$\begin{aligned}&\text{Rule} - j:\\ &\text{IF}\,[x_1 \text{ is } A_{1j} \ldots \text{ and } x_i \text{ is } A_{ij} \ldots \text{ and } x_N \text{ is } A_{Nj}]^{1-\gamma_j+\gamma_j/N}\\ &\text{THEN } y' \text{ is } w_{0j} + w_{1j}x_1 + \cdots + w_{ij}x_i + \cdots + w_{Nj}x_N \end{aligned} \quad (1)$$

where x_i is the input variable, y' is the output variable, A_{ij} is the linguistic term of the precondition part, $\gamma_j \in [0,1]$ is the compensatory degree, and w_{0j} and w_{ij} are the corresponding parameters of consequent part.

Next, the operation functions of the nodes in each layer of the CNFC are described. In the following, $u^{(l)}$ denotes the output of a node in the lth layer.

Layer 1 (Input node): No computation is performed in this layer. Each node in this layer is an input node, which corresponds to a single input variable, and only transmits directly input values to the next layer.

$$u_i^{(1)} = x_i. \quad (2)$$

Layer 2 (Membership function node): Nodes in this layer correspond to a single linguistic label of the input variables in layer 1. Therefore, the calculated membership value that specifies the degree to which an input value belongs to a fuzzy set in layer 2. The performed Gaussian membership function in layer 2 is

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

where m_{ij} and σ_{ij} are the mean and variance of the Gaussian membership function, respectively, of the jth term of the ith input variable x_i.

Layer 3 (Compensatory rule node): Nodes in this layer represent the precondition part of a fuzzy logic rule. They receive the 1-D membership degrees of the associated rule from the nodes of a set in layer 2. The compensatory fuzzy operator described elsewhere [17] is adopted to perform the IF-condition matching of fuzzy rules. As a result, the output function of each inference node is

$$u_j^{(3)} = \left(\prod_i u_{ij}^{(2)}\right)^{1-\gamma_j+\gamma_j/N} \quad (4)$$

where $\gamma_j = c_j^2/\left(c_j^2 + d_j^2\right) \in [0,1]$ is called the compensatory degree and $c_j, d_j \in [-1,1]$. Tuning c_j and d_j increases the adaptability of the fuzzy operator becomes more adaptive.

Layer 4 (Consequent node): Nodes in this layer are called consequent nodes. The input to a node of layer 4 is the output from layer 3, and the other inputs are the input variables from layer 1, as shown in Fig. 1. For such a node

$$u_j^{(4)} = u_j^{(3)}\left(w_{0j} + \sum_{i=1}^{N} w_{ij}x_i\right) \quad (5)$$

where the summation is over all the inputs, and w_{ij} are the corresponding parameters of consequent part.

Layer 5 (Output node): Each node in this layer corresponds to one output variable. The node integrates all of the actions recommended by layers 3 and 4 and acts as a defuzzifier with

$$y = u^{(5)} = \frac{\sum_{j=1}^{R} u_j^{(4)}}{\sum_{j=1}^{R} u_j^{(3)}} = \frac{\sum_{j=1}^{R} u_j^{(3)}\left(w_{0j} + \sum_{i=1}^{N} w_{ij}x_i\right)}{\sum_{j=1}^{R} u_j^{(3)}} \quad (6)$$

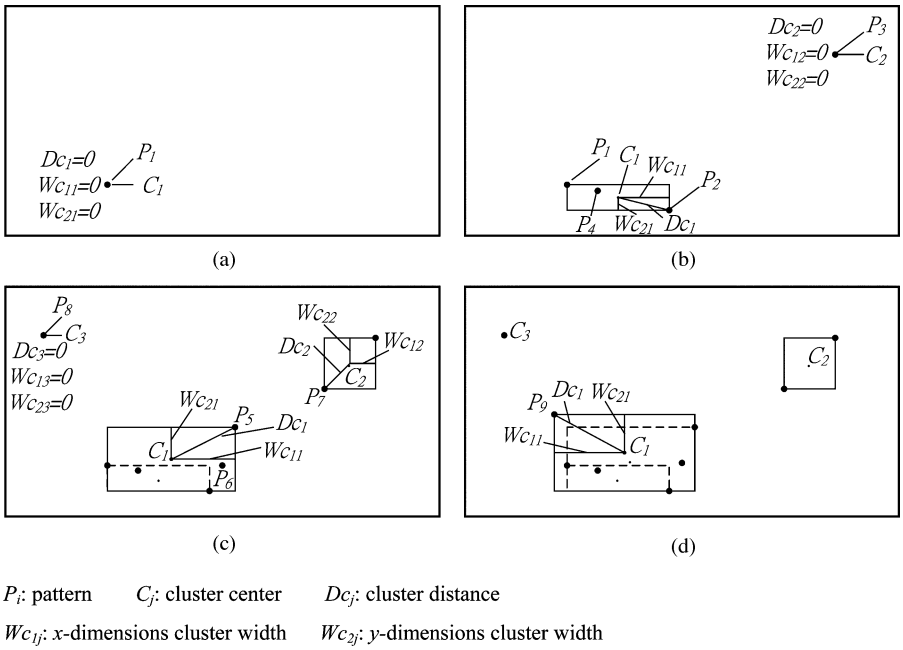

Fig. 2. Brief clustering process using the SCA with samples $P_1 - P_9$ in 2-D space. (a) Sample P_1 causes the SCA to create a new cluster center C_1. (b) P_2: update cluster center C_1; P_3: create a new cluster center C_2; P_4: do nothing. (c) P_5: update cluster C_1; P_6: do nothing; P_7: update cluster center C_2; P_8: create a new cluster C_3. (d) P_9: update cluster C_1.

where R is the number of fuzzy rules, and N is the number of input variables.

IV. Self-Clustering Algorithm

The number of fuzzy rules and approximate estimates of the corresponding parameters, such as means and variances, describing the fuzzy term sets in the precondition parts need to be extracted from given input–output pairs. Thus, the choice of clustering technique in neuro-fuzzy networks is an important consideration. This is due to the use of partition-based clustering techniques, such as fuzzy *C*-means (FCM) [27], possibilistic *C*-means (PCM) [28], linear vector quantization (LVQ) [29], fuzzy Kohonen partitioning (FKP), and pseudo FKP [30], to perform cluster analysis. However, such clustering techniques require prior knowledge such as the number of clusters present in a data set. To solve the aforementioned problem, online-based cluster techniques were proposed [31], [32], but there still is a problem with these methods, namely, the clustering methods [31], [32] only consider the total variations of the mean and variance in all dimensions of the input data. This is because the cluster numbers increase quickly.

In this paper, we use an SCA to partition the input space to create fuzzy rules. The proposed SCA is an online and distance-based clustering method, which is unlike the traditional clustering techniques [27]–[32]. The tendency of the traditional clustering techniques is to consider the total variations in all dimensions of the input data that will cause clusters to extend too fast. According to the aforementioned problems, the proposed SCA method considers the variation of each dimension for the input data.

The SCA is proposed to perform the scatter partitioning of the input space. With no optimization, the online SCA is a fast, one-pass algorithm that dynamically estimates the number of clusters in a set of data and finds the current centers of clusters in the input data space. It is a distance-based connectionist-clustering algorithm. In any cluster, the maximum distance between a sample point and the center is less than a threshold value, which has been set as a clustering parameter and which influences the number of clusters to be estimated.

In the clustering process, the data samples come from a data stream. The process begins with an empty set of clusters. When a new cluster is created, the cluster center is defined, and its cluster distance and cluster width *Dc* and *Wc* are initially set to zero. When more samples are presented one after another, some created clusters are updated by changing the positions of their centers and increasing the cluster distances and cluster width. Which cluster will be updated and the extent to which it is changed depends on the position of the current sample in the input space. A cluster will not be updated any more when its cluster distance *Dc* reaches the value equals the threshold value D_{thr}.

Fig. 2 shows the SCA clustering process in two-input space. The SCA is described as follows.

Step 0) The input data set should be preprocessed. Finding the optimal solution is difficult because the range of training data is wide. Therefore, the data must be normalized. Let x_i be transformed to the interval of [0,1]:

$$x_i' = \frac{x_i - x_{i\_\min}}{x_{i\_\max} - x_{i\_\min}} \tag{7}$$

where x_i' is the value after normalization, x_i is the vector of the ith dimension to be normalized, $x_{i\_\min}$

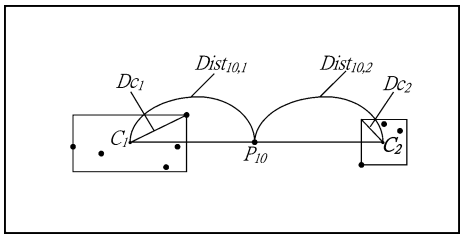

Fig. 3. Case of the equal distances between the sample and two the cluster.

is the minimum value of vector x_i and $x_{i\_\max}$ is the maximum value of vector x_i.

Step 1) Create the first cluster by simply setting the position of the first sample from the input stream as the first cluster center C_1 and setting its cluster distance Dc_1 and cluster widths Wc_{11} and Wc_{21} to zero, as shown in Fig. 2(a).

Step 2) If all samples of the data stream have been processed, then the algorithm is complete. Otherwise, the current input sample P_i is used, and the distances between this sample and all p already created cluster centers C_j, $Dist_{ij} = \|P_i - C_j\|$, $j = 1, 2, \ldots, p$ are calculated.

Step 3) If any distances $Dist_{ij}$ equals, or is less than, at least one of the distances Dc_j, $j = 1, 2, \ldots, p$, then the current sample P_i is in the cluster C_m with the minimum distance

$$Dist_{im} = \|P_i - C_m\| = \min\left(\|P_i - C_j\|\right),\ j = 1, 2, \ldots, p. \quad (8)$$

In such a case, neither is a new cluster created nor is any existing cluster updated, as in the cases P_4 and P_6 shown in Fig. 2, for example. The algorithm then returns to step 2. Otherwise, it proceeds to the next step.

Step 4) Find a cluster with center C_m and cluster distance Dc_m from all p existing cluster by calculating $S_{ij} = Wc_{ij} + Dc_j$, $j = 1, 2, \ldots, p$ and then choosing the cluster center C_m with the minimum value S_{im}:

$$S_{im} = Wc_{im} + Dc_m = \min(S_{ij}),\ j = 1, 2, \ldots, p. \quad (9)$$

In (8), the maximum distance from the center of any cluster to the samples in this cluster does not exceed the threshold D_{thr}, although the algorithm does not keep any information on the passed samples. However, the formulation involves only the distance between the input data and the cluster center in (9). If the two distances $Dist_{im}$ are the same, then the distances between the sample and any two cluster centers are equal. For example, the distances between a given point P_{10} and both cluster centers C_1 and C_2, $Dist_{10,1}$ and $Dist_{10,2}$ are as shown in Fig. 3. In the aforementioned scheme, the cluster C_2, which has a small distance Dc_2, is expanded according to (9). However, doing so causes the problem of rapidly increasing the number of clusters. The following condition is applied to solve this problem.

If (the distance between P_{10} and $Dist_{10,1}$ equals the distance between P_{10} and $Dist_{10,2}$) and ($Dc_1 > Dc_2$), then $Dc_m = Dc_1$.

This rule dictates that when the distances between the input data and both clusters are the same, the formula chooses the cluster with large distances Dc_1.

Step 5) If S_{im} exceeds D_{thr}, then the sample P_i is not in any existing cluster. A new cluster is created as described in step 1, as in cases P_3 and P_8 shown in Fig. 2, and the algorithm returns to step 2.

Step 6) If S_{im} does not exceed D_{thr}, then the cluster C_m is updated by moving its center, C_m, and increasing the cluster distance Dc_m and the cluster widths Wc_{1m} and Wc_{2m}. The parameters are updated according to the following equation:

$$Wc_{1m}^{\text{new}} = \frac{(\|C_{m\_x} - P_{i\_x}\| + Wc_{1m})}{2} \quad (10)$$

$$Wc_{2m}^{\text{new}} = \frac{(\|C_{m\_y} - P_{i\_y}\| + Wc_{2m})}{2} \quad (11)$$

$$C_{m\_x}^{\text{new}} = \|P_{i\_x} - D_{1m}^{\text{new}}\| \quad (12)$$

$$C_{m\_y}^{\text{new}} = \|P_{i\_y} - D_{2m}^{\text{new}}\| \quad (13)$$

$$Dc_m^{\text{new}} = \frac{S_{im}}{2} \quad (14)$$

where C_{m_x} is the distance in the *x* direction associated with C_m, C_{m_y} is the distance in the *y* direction associated with C_m, P_{i_x} is the distance in the *x* dimension associated with P_i, and P_{i_y} is the associated with in the *y* dimension for P_i, as in the cases P_2, P_5, P_7, and P_9, as shown in Fig. 2. The algorithm returns to step 2.

Accordingly, the maximum distance from the center of any cluster to the samples in the cluster does not exceed the threshold value D_{thr}, although the algorithm keeps no information on the passed samples. Thereafter, the number of rules, the mean, and the variance of the Gaussian membership function in CNFC are given by the following equation:

$$m_{ij} = C_{ij} \quad (15)$$

$$\sigma_{ij} = 50\% \times Wc_{ij} \quad (16)$$

$$R = \text{ the number of clusters} \quad (17)$$

where C_{ij}, Wc_{ij}, and the number of clusters are created by SCA.

In the clustering process, the threshold parameter D_{thr} is an important parameter. A low threshold value leads to the learning of coarse clusters (i.e., fewer rules are generated), whereas a high threshold value leads to the learning of fine clusters (i.e., more rules are generated). Therefore, the selection of the threshold value D_{thr} will critically affect the simulation results, and the value will be based on practical experimentation or on trial-and-error tests. We defined generally that D_{thr} is equal to 0.5–1 times the summation of the inputs variance.

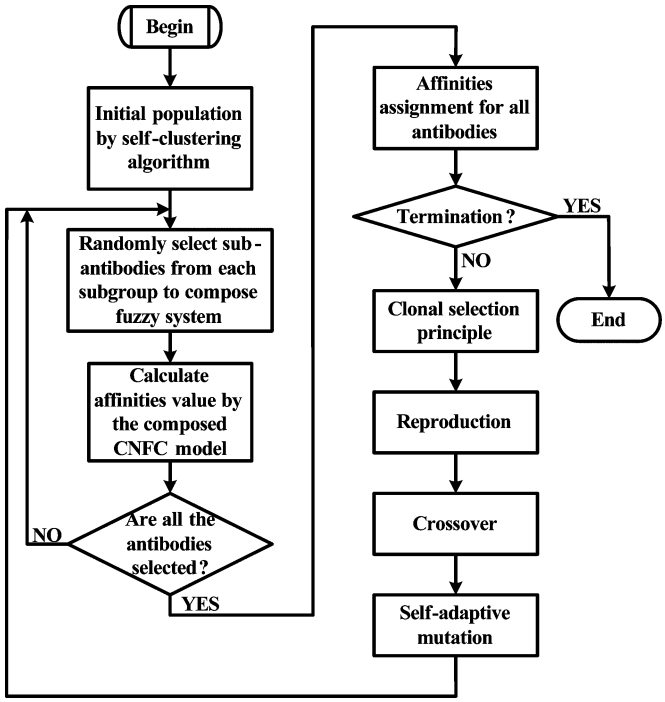

Fig. 4. Flowchart of the proposed ISEL method.

V. Immune Symbiotic Evolution Learning for the CNFC

This section describes an ISEL method for constructing the CNFC. The ISEL method comprises of three major components—initial population, subgroup symbiotic evolution, and immune system algorithm. First, the initial population uses the SCA that determines proper input space partitioning and finds the mean and variance of the Gaussian membership function and the number of rules. Second, in traditional symbiotic evolution, all rules generally evolve together; however, in the subgroup symbiotic evolution method, each rule evolves separately. Third, the immune system algorithm uses the clonal selection principle to accelerate the search and increase global search capacity. Fig. 4 presents the ISEL of the CNFC.

A. Initial Population

This subsection introduces the production of initial population, covering the coding and initialization steps. The coding step involves the membership functions and the fuzzy rules of a fuzzy system that represent subantibodies and are suitable for subgroup symbiotic evolution. The initialization step assigns the population values before the evolution process begins.

1) Coding Step: The first step in ISEL method is the coding of a fuzzy rule into a subantibody. Fig. 5 shows an example of a fuzzy rule coded into a subantibody where i and j are the ith dimension and jth rule. Fig. 5 describes a fuzzy rule given by (1), where m_{ij} and σ_{ij} are the mean and variance of a Gaussian membership function, respectively, c_j and d_j are compensatory parameters, and w_{ij} is the corresponding parameter of the consequent part associated with the jth rule node.

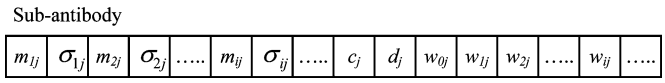

Fig. 5. Coding a fuzzy rule into a subantibody in subgroup symbiotic evolution.

2) Initialization Step: Before the ISEL method is designed, the individuals that will constitute an initial population must be created. In this study, the initial population was created according to the range of the mean and variance of the membership function, which were computed by the SCA method. The following formulations show the generation of the initial population:

$$\text{Mean: } Subantibody\,[index] = m_{ij} + random\,[0,1] \times \sigma_{ij}$$
$$\text{where } index = 1, 3, \ldots, 2 \times N - 1$$
$$\text{Variance: } Subantibody\,[index] = 4 \times random\,[0,1] \times \sigma_{ij}$$
$$\text{where } index = 2, 4, \ldots, 2 \times N$$
$$\text{Other parameters: } Subantibody\,[index] = random\,[-1,1]$$
$$\text{where } index > 2 \times N$$

where index is the site of a subantibody, and m_{ij} and σ_{ij} are the mean and variance, respectively, of the Gaussian membership function of the jth rule of the ith input variable. The size of the population depends on the complexity of the problem. Many experiments showed that the population size of 20 is most effective. Besides the population size, other parameters must be set. This is the number of fuzzy systems to be formed and evaluated in each generation. The crossover rate and the mutation rate and these parameters also depend on the complexity of the problem.

B. Subgroup Symbiotic Evolution

This section presents a novel method of subgroup symbiotic evolution. As described before, in the subgroup symbiotic evolution, the affinity value of a rule (a subantibody) is computed as the sum of the affinity values of all the feasible combinations of that rule with all other randomly selected rules and then dividing this sum by the total number of combinations. Fig. 6 shows the structure of the subantibody in the subgroup symbiotic evolution. The stepwise assignment of the affinity value is as follows.

Step 0) Divide the rules into subgroups of size *ps*.

Step 1) Randomly select R fuzzy rules (subantibody) from each of the aforementioned subgroups, and compose the fuzzy system using these R rules.

Step 2) Calculate affinities of the antibodies using the CNFC thus composed. In this study, the affinity value is given by the follow formula:

$$\text{Affinity value} = \frac{1}{1 + \sqrt{1/N_t \sum_{k=1}^{N_t} \left(y_k - y_k^d\right)^2}} \tag{18}$$

where y_k represents the *k*th model output, y_k^d represents the desired output, and N_t represents the number of input data.

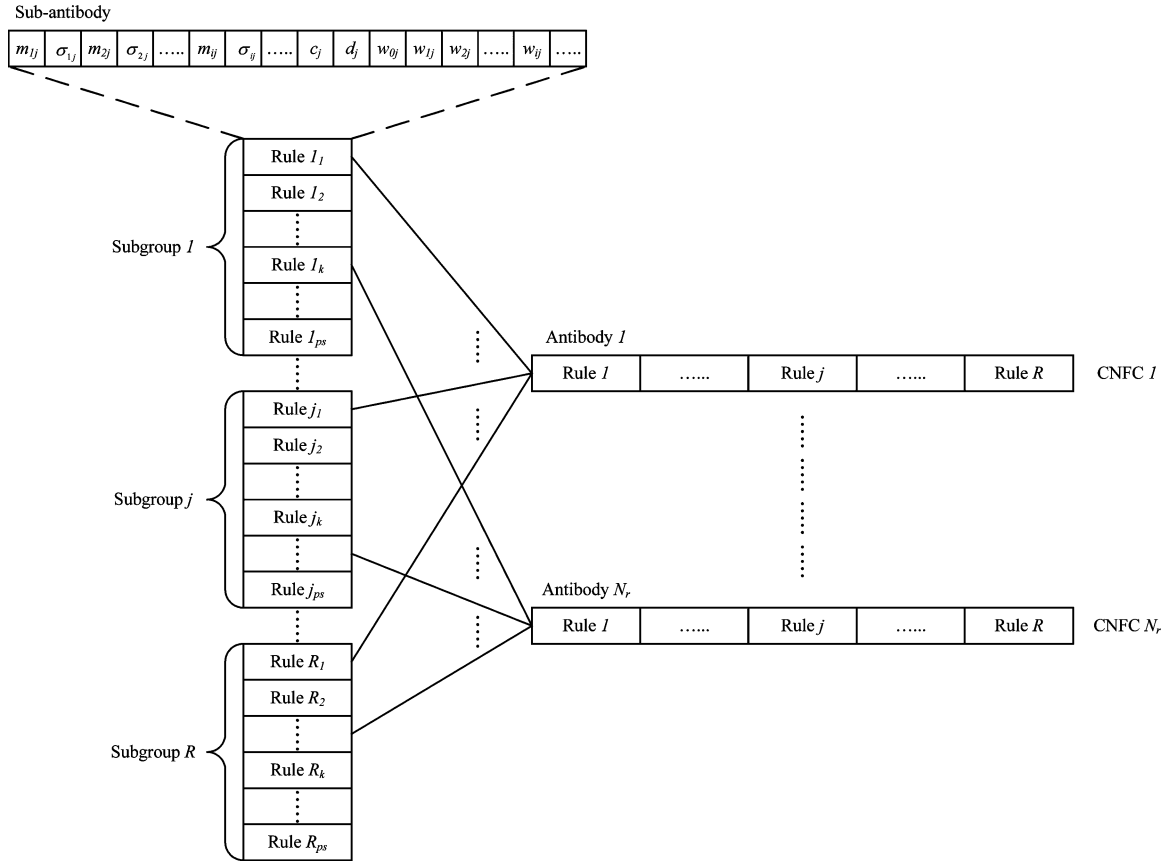

Fig. 6. Structure of subantibody in subgroup symbiotic evolution.

Step 3) Divide the affinity value by R, and accumulate the divided affinity value to the affinity record of the R selected rules with their recorded affinity values initially set to zero.

Step 4) Repeat the earlier steps until each rule (subantibody) in each subgroup has been selected a sufficient number of times, and record the number of fuzzy systems to which each subantibody has contributed.

Step 5) Divide the accumulated affinity of each subantibody by the number of times it has been selected.

Step 6) Sort these subantibodies in each subgroup in order of decreasing affinity.

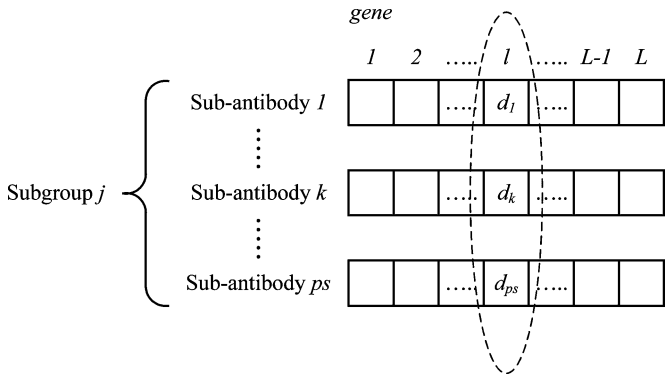

Fig. 7. Composition of subgroup with these subantibodies.

C. Immune System Algorithm

This section introduces the implementation of the immune system algorithm that is based on the clonal selection principle, the reproduction step, the crossover step, and the self-adaptive mutation step.

1) Clonal Selection Principle: The antibodies of this immune system that eliminate the antigen should exhibit at least a minimal threshold of diversity. Accordingly, the diversity operator is applied to these antibodies such that those between others of high similar degree are canceled, guaranteeing an improvement in quality. This process is called the clonal selection principle. Therefore, if the affinity between two antibodies exceeds the suppression threshold Th_{aff}, then these two antibodies are called similar. Now, the subgroup is assumed to be composed of *ps* subantibodies with *L* genes, as shown in Fig. 7.

Information theory yields the entropy value $EV_l(ps)$ of the lth gene in this jth subgroup:

$$EV_l(ps) = \sum_{k=1}^{ps} -P_{kl}\log P_{kl} \tag{19}$$

where P_{kl} is the probability of the kth allele from the lth gene and $P_{kl} = 1/(1 + \|d_k - d_{k'}\|)$ is defined, where d_k and $d_{k'}$ are the genes of the kth and the k'th subantibody, respectively. The diversity of the genes is calculated using (19). If all alleles of the lth gene are of the same form, then the entropy value $EV_l(ps)$

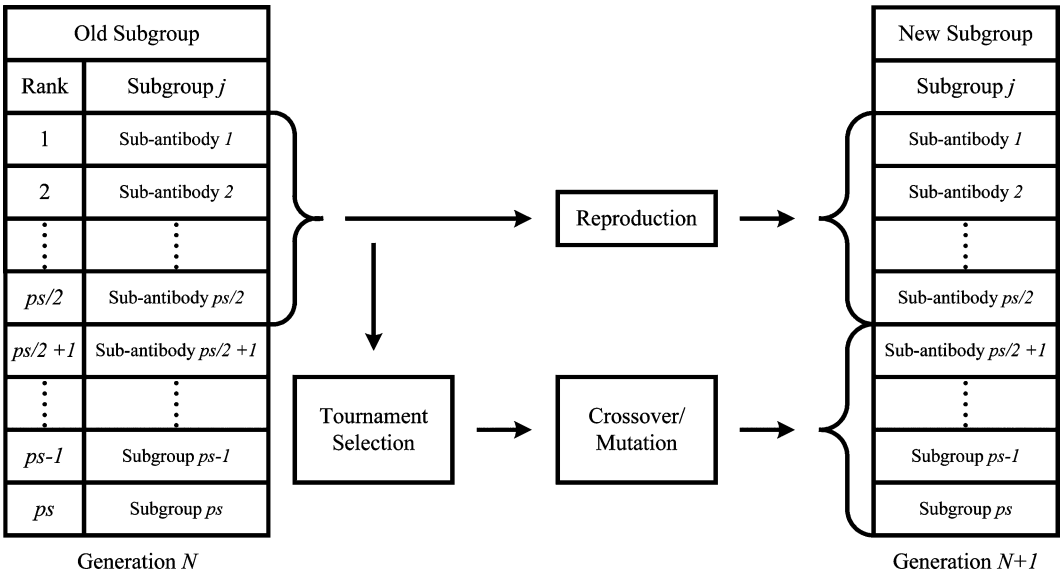

Fig. 8. Flowchart of reproduction, crossover, and mutation steps.

will be zero. The average entropy value *EV(ps)* of diversity is also given as follows:

$$EV(ps) = \frac{\sum_{l=1}^{L} EV_l(ps)}{L} \tag{20}$$

where L is the size of the gene in a subantibody. Equation (20) yields the diversity of the subantibody pool in terms of the entrop. Two affinity expressions are considered in the proposed approach. The first explains the relationships between the antibody and the antigen and is given by (18). The second describes the degree of association between subantibodies. The following equation computes the affinity between subantibodies:

$$\text{Affinity\_}Ab_{kk'} = \frac{1}{1 + EV(2)} \tag{21}$$

where Affinity$_Ab_{kk'}$ is the affinity between two subantibodies k and k', and *EV(2)* is the entropy of only the subantibodies k and k'. This affinity is constrained from zero to one. When *EV(2)* is zero, the genes of the kth subantibody and those of the k'th subantibody are the same. Therefore, the clonal selection principle is that if the Affinity$_Ab_{kk'}$ exceeds Th_{aff}, then the subantibody should be canceled. Reproduction, crossover, and mutation steps follow, according to the flowchart presented in Fig. 8.

2) Reproduction Step: Reproduction is a process in which subantibody strings are copied according to their affinity values. This process is an artificial version of natural selection. This study uses the roulette-wheel selection method [33]—a simulated roulette is spun—in this reproduction step. The best performing subantibodies in the top half of the subgroup evolves to the next generation. The other half is generated to perform crossover on subantibodies in the top half of the parent generation.

3) Crossover Step: Reproduction directs the search toward the best existing subantibodies but does not create any new subantibodies. In nature, an offspring has two parents and inherits genes from both. The main operator working on the parents is the crossover operator, the operation of which occurred for a selected pair with a crossover rate that was set to 0.5 in this study. Therefore, the new subantibodies are principally generated by crossover. In this process, the first step is to select two subantibodies for crossover. Two subantibodies for crossover are selected by tournament selection [33] on the top half of the best-performing subantibodies. Tournament selection randomly selects three subantibodies in each subgroup, and the subantibody with the best affinity is adopted as the designated subantibody. The other designated subantibody is selected in the same way. Two new subantibodies are generated by performing crossover on the selected designated subantibodies. In the second step, the two designated subantibodies are crossed and separated using a two-point crossover in which new subantibodies are created by exchanging the gene between the selected sites of the two designated subantibodies. In the crossover step, we also keep the same number of subantibodies in the other half of the population for each subgroup. After this operation, the new subantibodies replace the designated subantibodies with poor performances.

4) Self-Adaptive Mutation Step: After reproduction and the crossover step yielded new strings, no new information is introduced to the subgroup at the site of a subantibody. As a source of new sites, mutation should be used sparingly because it is a random search operator. In the following simulations, a mutation rate was set to 0.3. Mutation is an operator that randomly alters the allele of a gene. The self-adaptive mutation adopted in the immune system algorithm yields high diversity. The subantibody suffers from a mutation to avoid falling in a local optimal solution and to ensure the searching capacity of approximate global optimal solution. The self-adaptive mutation with real-valued representation can be used to produce mutants by adding a number randomly selected from some interval to a given antibody. The mutation value is generated according to

TABLE I
INITIAL PARAMETERS BEFORE TRAINING

Parameter	Value
Size of Subgroup (*ps*)	20
Crossover Rate	0.5
Mutation Rate	0.3
Threshold of Affinity (Th_{aff})	0.95
Coding Type	Real Number

(22), shown below. Following self-adaptive mutation, a new subantibody can be introduced into the population.

$$SAb[\text{index}] = \begin{cases} SAb[\text{index}] + (\max(SAb[:]) - SAb[\text{index}]) \times \lambda & \text{if } \alpha > 0.5 \\ SAb[\text{index}] + (\min(SAb[:]) - SAb[\text{index}]) \times \lambda & \text{if } \alpha \le 0.5 \end{cases} \tag{22}$$

where $\lambda = \beta \times (t/T)^{\beta}$, and $\alpha, \beta \in [0, 1]$ are random values, *index* represents the gene in a subantibody (*SAb*), *t* is the *t*th generation, and *T* is the maximum number of generations.

VI. ILLUSTRATIVE EXAMPLES

This study evaluated the performance of the proposed CNFC using ISEL (CNFC–ISEL) algorithm for nonlinear control systems. This section presents several examples and compares the performance with that of other neural fuzzy networks using symbiotic evolution (NFN–SE) [21], [22]. In the nonlinear control system problems, CNFC–ISEL is adopted to design controllers in four simulations—multi-input single-output (MISO) plant control, multi-input multi-output (MIMO) plant control [2], backing up the truck [34], and the ball-and-beam system [35]. Table I presents the initial parameters before training used in the four computer simulations.

A. Example 1: Multi-Input Single-Output Control

The controlled plant is the same as that used elsewhere [2] and is described by

$$y_p(k+1) = f[y_p, y(k-1)] + u(k) \tag{23}$$

where $f[y_p, y(k-1)] = [y_p(k)y_p(k-1)(y_p(k)+2.5)]/[1 + y_p^2(k) + y_p^2(k-1)]$. In designing the CNFC, the desired output y_r is specified by the following 250 pieces of data:

$$y_r(k+1) = 0.6y_r(k) + 0.2y_r(k-1) + r(k) \tag{24}$$

where $r(k) = 0.5\sin(2\pi k/45) + 0.2\sin(2\pi k/15) + 0.2\sin(2\pi k/90)$.

The inputs to the CNFC are $y_p(k), y_p(k+1), y_r(k)$, and $y_r(k+1)$, and the output is *u(k)*. For the SCA, we chose the parameter $D_{\text{thr}} = 0.85$, and then, four rules are generated by SCA in CNFC; hence, $R = 4$, and the number of total parameters is 60. Accordingly, the number of genes is also 15 in each subgroup. Fig. 9 plots the learning curves of the best performance of the CNFC–ISEL for the affinity/fitness value, the symbiotic evolution of the fuzzy controller (SEFC) [21], and the Mamdani-type fuzzy system using the symbiotic evolution algorithm (MFS–SE) [22], after the learning process of 500 generations. Fig. 10(a) plots the control results of the desired output and the model output after a learning process of 500 generations. Fig. 10(b) shows the errors of the proposed method. Table II presents the best and averaged affinity/fitness values after 500 generations of training. The comparison indicates that the best and averaged affinity/fitness values of CNFC–ISEL are better than those obtained using other methods.

B. Example 2: Multi-Input Multi-Output Control

In this example, the MIMO plants [2] to be controlled are described by

$$\begin{bmatrix} y_{p1}(k+1) \\ y_{p2}(k+1) \end{bmatrix} = \begin{bmatrix} 0.5[y_{p1}(k)/(1+y_{p2}^2(k))] \\ 0.5[y_{p1}(k)y_{p2}(k)/(1+y_{p2}^2(k))] \end{bmatrix} + \begin{bmatrix} u_1(k) \\ u_2(k) \end{bmatrix}. \tag{25}$$

The controlled outputs should follow the desired output y_{r_1} and y_{r_2}, as specified by the following 250 pieces of data:

$$\begin{bmatrix} y_{r1}(k) \\ y_{r2}(k) \end{bmatrix} = \begin{bmatrix} \sin(k\pi/45) \\ \cos(k\pi/45) \end{bmatrix}. \tag{26}$$

The inputs of the CNFC are $y_{p_1}(k), y_{p_2}(k), y_{r_1}(k)$, and $y_{r_2}(k)$, and the outputs are $u_1(k)$ and $u_2(k)$. For the SCA, we chose the parameter $D_{\text{thr}} = 1$, and then, six rules generated by SCA in CNFC: $R = 6$. Therefore, the total number of parameters is 120, and the number of genes is 20 in each subgroup.

Fig. 11 plots the learning curves of the best performance of the CNFC–ISEL model for the affinity/fitness value, the SEFC [21], and the Mamdani-type fuzzy system using symbiotic evolution algorithm (MFS–SE) [22], after the learning process of 500 generations. To demonstrate the control result, Fig. 12(a) and (b) plots the control results of the desired output and the model output after the learning process of 500 generations. Fig. 12(c) and (d) shows the errors of the proposed method. Table III presents the best and averaged affinity/fitness values after 500 generations of training. The comparison indicates that the best and averaged affinity/fitness values of CNFC–ISEL are better than those of other methods.

C. Example 3: Control of Backing Up the Truck

Backing a truck into a loading dock is difficult. It is a nonlinear control problem for which no traditional control method exists [34]. Fig. 13 shows the simulated truck and loading zone. The truck's position is exactly determined by three state variables ϕ, *x*, and *y*, where ϕ is the angle between the truck and the horizontal, and the coordinate pair (*x*, *y*) specifies the position of the center of the rear of the truck in the plane. The steering angle θ of the truck is the controlled variable. Positive values of θ represent clockwise rotations of the steering wheel, and negative values represent counterclockwise rotations. The truck is placed at some initial position and is backed up while being

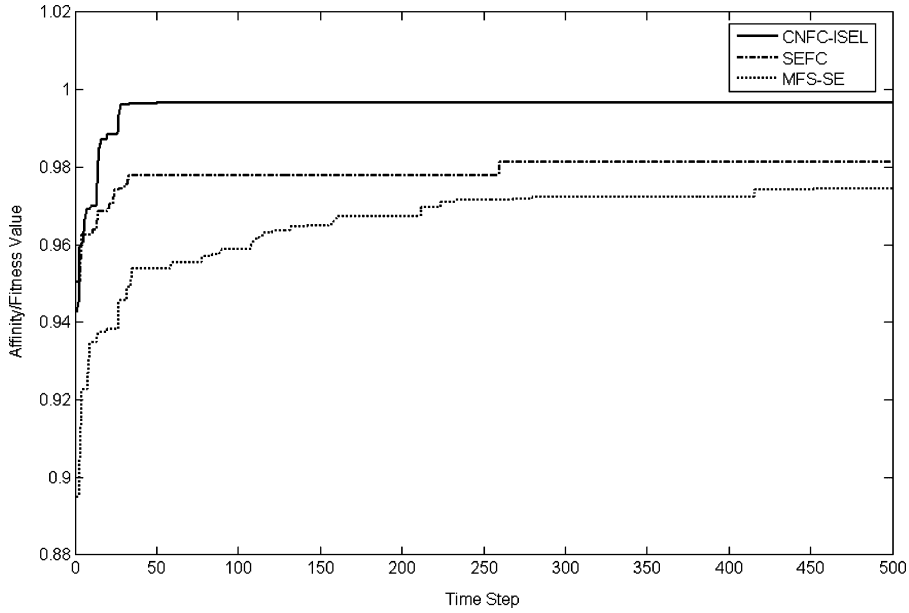

Fig. 9. Learning curves of best performance of the CNFC–ISEL, SEFC, and MFS–SE in Example 1.

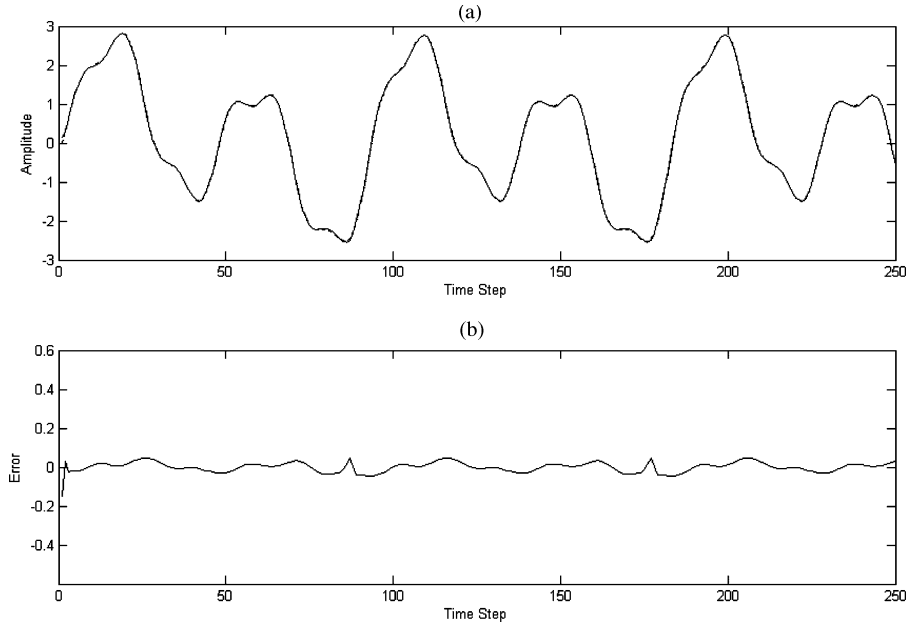

Fig. 10. (a) Desired (solid line) and model (dotted line) output generated by CNFC designed with ISEL in Example 1. (b) Errors of proposed CNFC–ISEL.

TABLE II
PERFORMANCE COMPARISONS WITH CNFC–ISEL, SEFC, AND MFS–SE IN EXAMPLE 1

Method	CNFC-ISEL	SEFC [21]	MFS-SE [22]
Affinity/Fitness Value (Ave)	0.9916	0.9793	0.9655
Affinity/Fitness Value (Best)	0.9961	0.9802	0.9745

steered by the controller. The objective of this control problem is to use backward only motion of the truck to make it arrive at the desired loading dock ($x_{\text{desired}}, y_{\text{desired}}$) at a right angle ($\phi_{\text{desired}} = 90^\circ$). The truck moves backward as the steering wheel moves through a fixed distance (d_f) in each step. The loading region is limited to the plane [0 1 0 0] $\times$ [0 1 0 0].

The input and output variables of the CNFC must be specified. The controller has two inputs: truck angle ϕ and cross position x. When the clearance between the truck and the loading dock is assumed to be sufficient, the y coordinate is not considered to be an input variable. The output of the controller is the steering angle θ. The ranges of the variables x, ϕ, and θ are as follows:

$$0 \le x \le 100 \tag{27}$$

$$-90^\circ \le \phi \le 270^\circ \tag{28}$$

$$-30^\circ \le \theta \le 30^\circ. \tag{29}$$

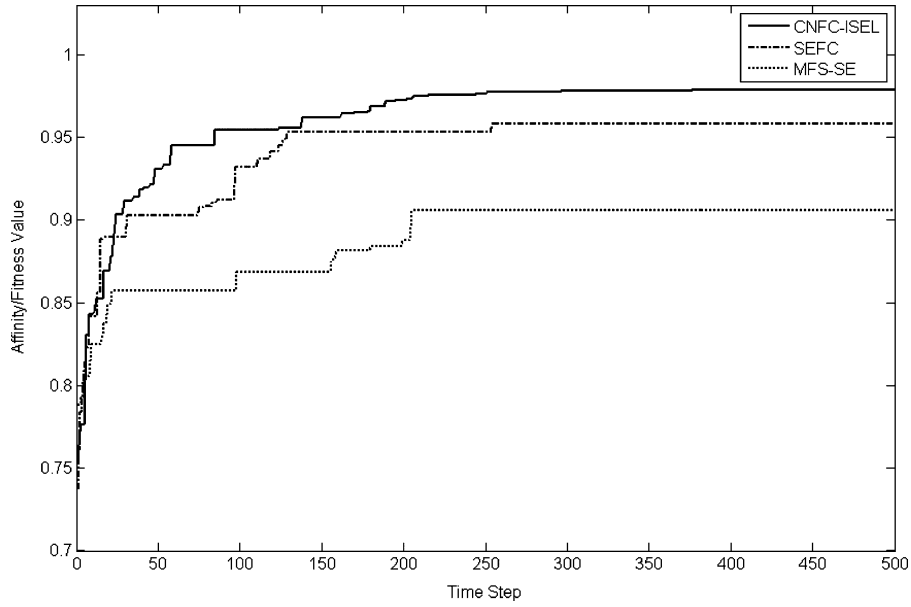

Fig. 11. Learning curves of best performance of the CNFC–ISEL, SEFC, and MFS–SE in Example 2.

The equations of backward motion of the truck are

$$\begin{aligned} x(k+1) &= x(k) + d_f\ \cos\theta(k) + \cos\phi(k) \\ y(k+1) &= y(k) + d_f\ \cos\theta(k) + \sin\phi(k) \\ \phi(k+1) &= \tan^{-1}\left[\frac{l\ \sin\phi(k) + d_f\ \cos\phi(k)\sin\theta(k)}{l\ \cos\phi(k) - d_f\ \sin\phi(k)\sin\theta(k)}\right] \end{aligned} \quad (30)$$

where l is the length of the truck. Equation (30) yields the next state from the present state.

Learning involves several attempts, each starting from an initial state and terminating when the desired state is reached; the CNFC is thus trained. For the SCA, we chose the parameter $D_{\text{thr}} = 0.45$. After SCA learning, 13 fuzzy logic rules are generated, and the parameters of the membership functions are determined. Hence, the total number of parameters is 117, and the number of genes is nine in each subgroup. The training process continues for 500 generations. The affinity of the CNFC is approximately 0.95587, and the learning curve of CNFC is compared with those obtained using various existing models [21], [22], as shown in Fig. 14. Fig. 15(a)–(d) plots the trajectories of the moving truck controlled by the CNFC, starting at initial positions (x, y, ϕ) = (40, 20, $-30°$), (10, 20, 150°), (70, 20, $-30°$), and (80, 20, 150°), after the training process has been terminated. The considered performance indices include the best affinity/fitness and the average affinity/fitness value. Table IV compares the results. According to these results, the proposed CNFC–ISEL outperforms various existing models.

D. Example 4: Control of the Ball-and-Beam System

Fig. 16 shows the ball-and-beam system [35]. The beam is made to rotate in the vertical plane by applying a torque at the center of rotation and the ball is free to roll along the beam. The ball must remain in contact with the beam.

TABLE III
COMPARISON OF PERFORMANCE OF CNFC–ISEL, SEFC, AND MFS–SE IN EXAMPLE 2

Method	CNFC-ISEL	SEFC [21]	MFS-SE [22]
Affinity/Fitness Value (Ave)	0.9721	0.9553	0.8503
Affinity/Fitness Value (Best)	0.9786	0.9581	0.8560

TABLE IV
COMPARISONS OF PERFORMANCE OF CNFC–ISEL, SEFC, AND MFS–SE IN EXAMPLE 3

Method	CNFC-ISEL	SEFC [21]	MFS-SE [22]
Affinity/Fitness Value (Ave)	0.9511	0.9451	0.9332
Affinity/Fitness Value (Best)	0.9558	0.9516	0.9398

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

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} = \begin{bmatrix} x_2 \\ B\left(x_1 x_4^2 - G\sin x_3\right) \\ x_4 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} u$$
$$y = x_1 \quad (31)$$

where $x = (x_1, x_2, x_3, x_4)^T \equiv (r, \dot{r}, \theta, \dot{\theta})^T$ is the state of the system, and $y = x_1 \equiv r$ is the output of the system. The control u is the angular acceleration ($\ddot{\theta}$), and the parameters $B = 0.7143$ and $G = 9.81$ are set in this system. The purpose of control is to determine $u(x)$ such that the closed-loop system output y will converge to zero from different initial conditions.

According to the input-/output-linearization algorithm [35], the control law $u(x)$ is determined as follows: For state x, compute $v(x) = -\alpha_3\phi_4(x) - \alpha_2\phi_3(x) - \alpha_1\phi_2(x) - \alpha_0\phi_1(x)$, where $\phi_1(x) = x_1$, $\phi_2(x) = x_2$, $\phi_3(x) = -BG\sin x_3$, $\phi_4(x) = -BGx_4\cos x_3$, and α_i are chosen so that $s^4 + \alpha_3 s^3 + \alpha_2 s^2 + \alpha_1 s + \alpha_0$ is a Hurwitz polynomial. Compute $a(x) =$

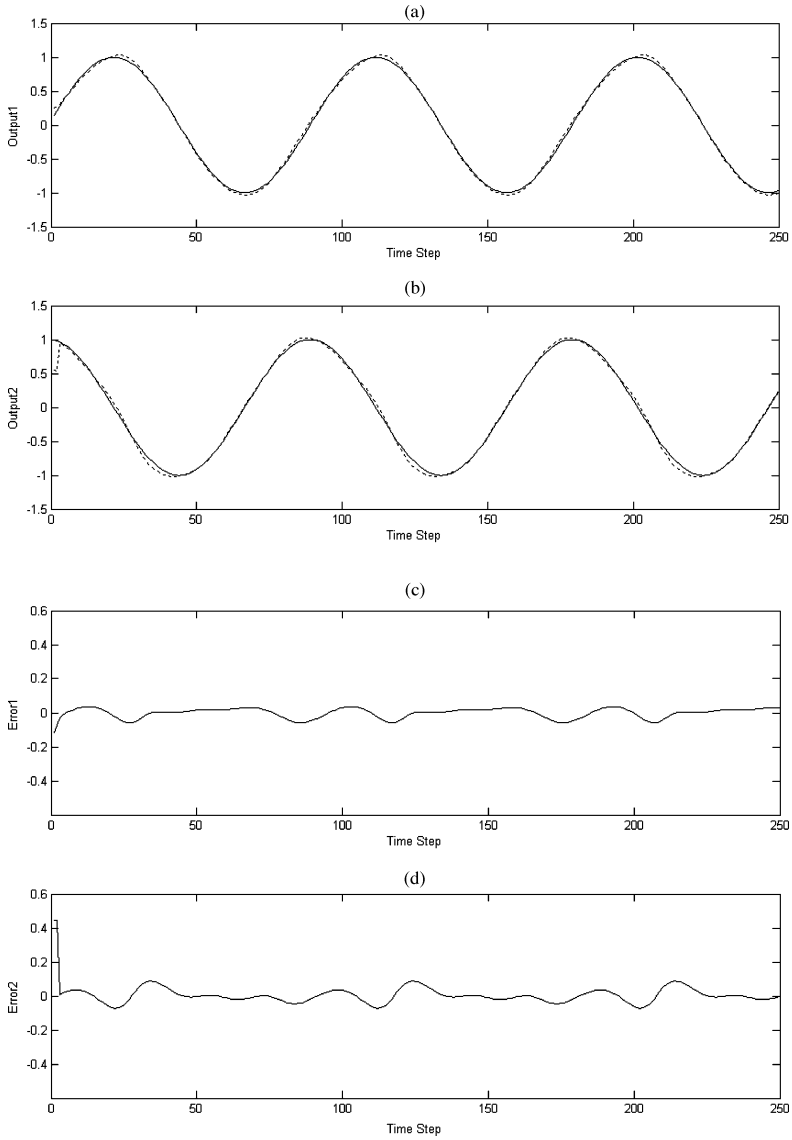

Fig. 12. (Solid line) desired and (dotted line) model output generated by CNFC designed with ISEL in Example 1. (a) Output 1. (b) Output 2. Errors of proposed CNFC–ISEL for (c) output 1 and (d) output 2.

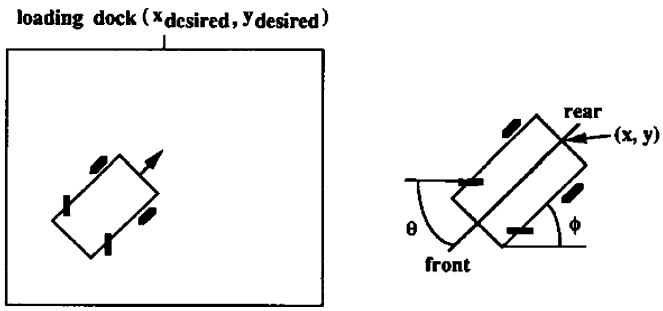

Fig. 13. Diagram of simulated truck and loading zone.

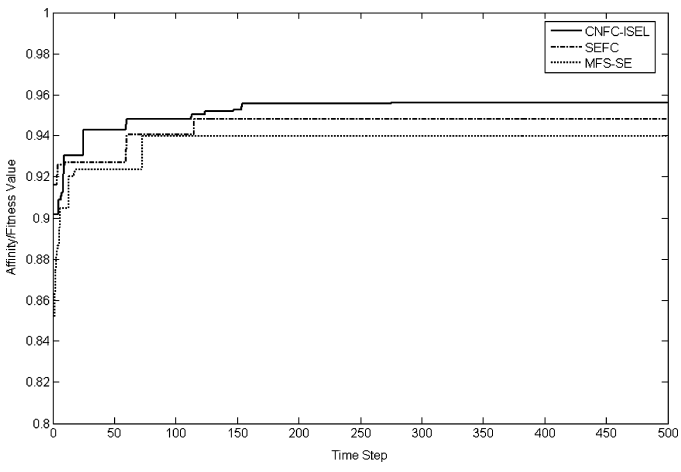

Fig. 14. Learning curves of best performance of the CNFC–ISEL, SEFC, and MFS–SE in Example 3.

$-BG\cos x_3$ and $b(x) = BGx_4^2 \sin x_3$; then, $u(x) = [v(x) - b(x)]/a(x)$.

In the simulation herein, the differential equations are solved using the second/third-order Runge–Kutta method. The CNFC

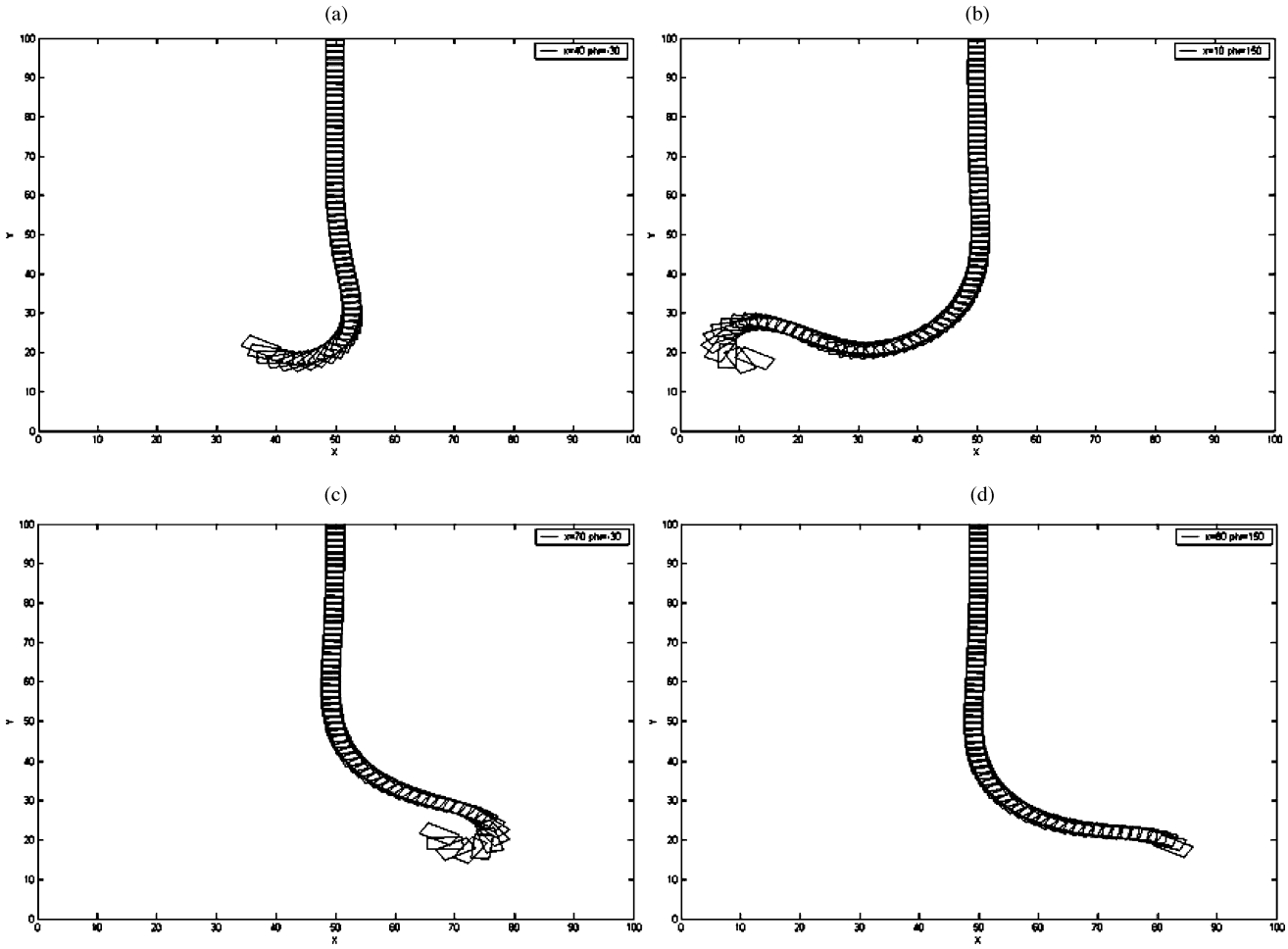

Fig. 15. Trajectories of truck, starting at four initial positions under the control of the CNFC–ISEL after learning using training trajectories.

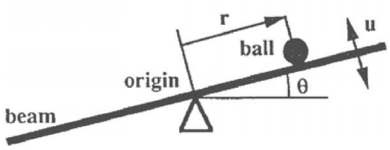

Fig. 16. Ball and beam system.

is trained to approximate the aforementioned conventional controller of a ball-and-beam system. $u(x) = [v(x) - b(x)]/a(x)$ is used to generate the input/output train pair with x obtained by randomly sampling 200 points in the region $U = [-5,5] \times [-3,3] \times [-1,1] \times [-2,2]$. For the SCA, we chose the parameter $D_{\rm thr} = 1.5$. After SCA learning, nine fuzzy logic rules are generated, and the corresponding parameters of membership functions are determined in our simulation. Therefore, the number of total parameters is 135. The number of genes is 15 in each subgroup. Fig. 17 plots the learning curves of the best performance of the CNFC–ISEL and various other models [21], [22] after a learning process of 1000 generations. The controller after learning was tested under the following four initial conditions: $x(0) = [2.4, -0.1, 0.6, 0.1]^T$, $[1.6, 0.05, -0.5, -0.05]^T$, $[-1.6, -0.05, 0.5, 0.05]^T$, and $[-2.4, 0.1, -0.6, -0.1]^T$. Fig. 18 plots the output responses of the closed-loop ball-and-beam system con-

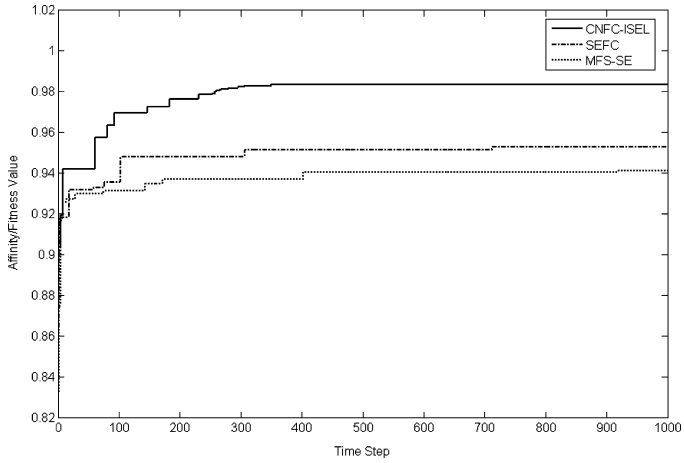

Fig. 17. Learning curves of best performance of the CNFC–ISEL, SEFC, and MFS–SE in Example 4.

trolled by the CNFC. These responses approximate those of the original controller under the four initial conditions. Fig. 19 also shows the behavior of the four states of the ball-and-beam system, starting from the initial condition $[-2.4, 0.1, -0.6, -0.1]^T$. In this figure, the four states of the system decay gradually

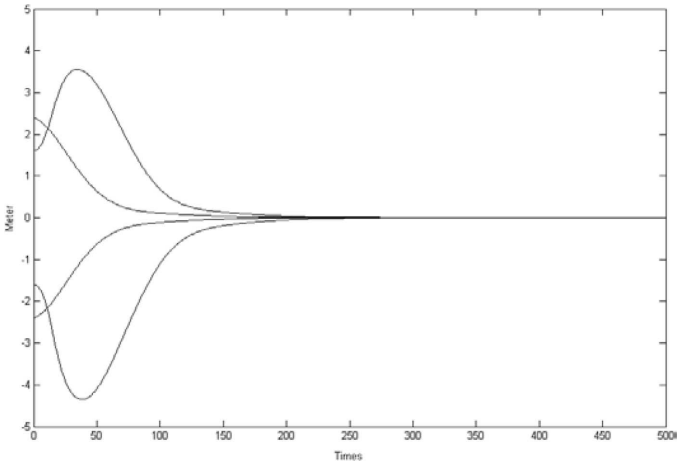

Fig. 18. Responses of the ball-and-beam system controlled by the CNFC under four initial conditions.

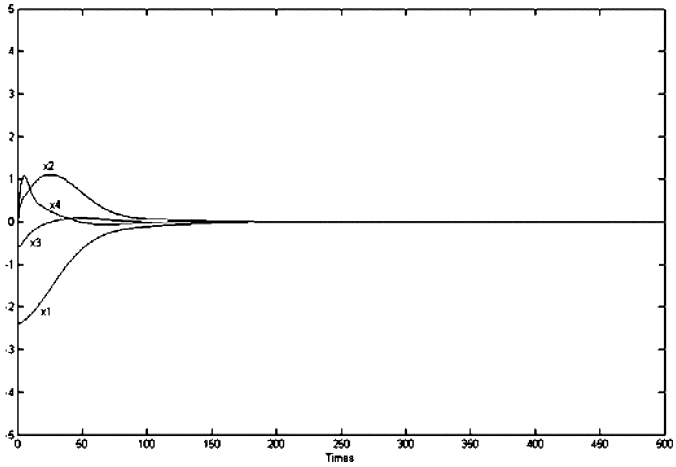

Fig. 19. Responses of four states of the ball-and-beam system under the control of the learned CNFC.

TABLE V
COMPARISON OF PERFORMANCE OF CNFC–ISEL, SEFC, AND MFS–SE IN EXAMPLE 4

Method	CNFC-ISEL	SEFC [21]	MFS-SE [22]
Affinity/Fitness Value (Ave)	0.9782	0.9478	0.9328
Affinity/Fitness Value (Best)	0.9836	0.9530	0.9414

to zero. The results show the perfect control capability of the trained CNFC–ISEL method. The performance of CNFC is compared with the SEFC [21] and the Mamdani-type fuzzy system using the symbiotic evolution algorithm (MFS–SE) [22]. Table V compares the results. These results indicate that the proposed CNFC–ISEL is better than various existing models.

VII. CONCLUSION AND FUTURE WORKS

This study proposes an efficient ISEL algorithm for CNFC. The ISEL method includes three major components—initial population, subgroup symbiotic evolution, and immune system algorithm. However, the SCA, which is an evolution of the rule itself, the clonal selection principle of antibodies, and self-adaptive mutation are achieved by the ISEL method. Moreover, the proposed CNFC–ISEL obtains better simulation results than alternative models in some circumstances, for example, achieving faster learning and higher design accuracy in many nonlinear control problems.

Two advanced topics on the proposed CNFC–ISEL should be addressed in future research. First, it would be better if the CNFC–ISEL has the ability to delete unnecessary or redundant rules. The fuzzy similarity measure [36] determines the similarity between two fuzzy sets in order avoid having the existing membership functions be too similar. Second, in addition to the simulations done in this paper, the proposed CNFC–ISEL will be used to solve many practical problems, including the control of magnetic levitation systems and the control of the inverted pendulum system in our laboratory.

REFERENCES

[1] C. C. Lee, "Fuzzy logic in control systems: Fuzzy logic controller—Parts I, II," *IEEE Trans. Syst., Man, Cybern.*, vol. 20, no. 2, pp. 404–435, Mar. 1990.

[2] K. S. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks," *IEEE Trans. Neural Netw.*, vol. 1, no. 1, pp. 4–27, Mar. 1990.

[3] K. J. Astrom and B. Wittenmark, *Adaptive Control*. Reading, MA: Addison-Wesley, 1989.

[4] J.-S. R. Jang, "ANFIS: Adaptive-network-based fuzzy inference system," *IEEE Trans. Syst., Man, and Cybern.*, vol. 23, no. 3, pp. 665–685, May 1993.

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

[6] C. J. Lin and C. T. Lin, "An ART-based fuzzy adaptive learning control network," *IEEE Trans. Fuzzy Syst.*, vol. 5, no. 4, pp. 477–496, Nov. 1996.

[7] M. A. Lee and H. Takagi, "Integrating design stage of fuzzy systems using genetic algorithms," in *Proc. IEEE Int. Conf. Fuzzy Syst.*, 1993, pp. 612–617.

[8] K. C. Ng and T. Li, "Design of sophisticated fuzzy logic controllers using genetic algorithms," in *Proc. 3rd IEEE Int. Conf. Fuzzy Syst.*, 1994, pp. 1708–1711.

[9] W. R. Hwang and W. E. Thompson, "Design of intelligent fuzzy logic controllers using genetic algorithms," in *Proc. IEEE Int. Conf. Fuzzy Syst.*, 1994, pp. 1383–1388.

[10] H. Ishibuchi, K. Nozaki, N. Yamamoto, and H. Tanaka, "Selecting fuzzy if–then rules for classification problems using genetic algorithms," *IEEE Trans. Fuzzy Syst.*, vol. 3, no. 3, pp. 260–270, Aug. 1995.

[11] M. H. Lim, S. Rahardja, and B. H. Gwee, "A GA paradigm for learning fuzzy rules," *Fuzzy Sets Syst.*, vol. 22, pp. 11–32, 1996.

[12] Y. Shi, R. Eberhart, and Y. Chen, "Implementation of evolutionary fuzzy systems," *IEEE Trans. Fuzzy Syst.*, vol. 7, no. 2, pp. 109–119, Apr. 1999.

[13] T. L. Seng, M. B. Khalid, and R. Yusof, "Tuning of a neuro-fuzzy controller by genetic algorithm," *IEEE Trans. Syst., Man, Cybern. B*, vol. 29, no. 2, pp. 226–236, Apr. 1999.

[14] H. K. Lam, F. H. F. Leung, and P. K. S. Tam, "Design and stability analysis of fuzzy model based nonlinear controller for nonlinear systems using genetic algorithm," *IEEE Trans. Syst., Man, Cybern. B*, vol. 33, no. 2, pp. 250–257, Apr. 2003.

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

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

[17] C. J. Lin and C. H. Chen, "Nonlinear system control using compensatory neuro-fuzzy networks," *IEICE Trans. Fundam. Electron. Comm. Comput. Sci.*, vol. E86-A, no. 9, pp. 2309–2316, 2003.

[18] C. J. Lin and C. H. Chen, "Identification and prediction using recurrent compensatory neuro-fuzzy systems," *Fuzzy Sets Syst.*, vol. 150, no. 2, pp. 307–330, Mar. 2005.

[19] 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.
[20] D. E. Moriarty and R. Miikkulainen, "Efficient reinforcement learning through symbiotic evolution," *Mach. Learn.*, vol. 22, pp. 11–32, 1996.
[21] C. F. Juang, J. Y. Lin, and C. T. Lin, "Genetic reinforcement learning through symbiotic evolution for fuzzy controller design," *IEEE Trans. Syst., Man, Cybern. B*, vol. 30, no. 2, pp. 290–302, Apr. 2000.
[22] M. Jamei, M. Mahfouf, and D. A. Linkens, "Elicitation and fine-tuning of fuzzy control rules using symbiotic evolution," *Fuzzy Sets Syst.*, vol. 147, no. 1, pp. 57–74, Oct. 2004.
[23] N. K. Jerne, "The immune system," *Sci. Amer.*, vol. 229, no. 1, pp. 52–60, 1973.
[24] G. L. Ada and G. J. V. Nossal, "The clonal selection theory," *Sci. Amer.*, vol. 257, pp. 50–57, 1987.
[25] J. S. Chun, M. K. Kim, and H. K. Jung, "Shape optimization of electromagnetic devices using immune algorithm," *IEEE Trans. Magn.*, vol. 33, no. 2, pp. 1876–1879, Mar. 1997.
[26] L. Jiao and L. Wang, "A novel genetic algorithm based on immunity," *IEEE Trans. Syst., Man, Cybern., Part A*, vol. 30, no. 5, pp. 552–561, Sep. 2000.
[27] M. C. Hung and D. L. Yang, "The efficient fuzzy c-means clustering technique," in *Proc. IEEE Int. Conf. Data Mining*, Dec. 2001, pp. 225–232.
[28] R. Krishnapuram and J. M. Keller, "The possibilistic c-means algorithm: Insights and recommendations," *IEEE Trans. Fuzzy Syst.*, vol. 4, no. 3, pp. 385–393, Aug. 1996.
[29] J. He, L. Liu, and G. Palm, "Speaker identification using hybrid LVQ-SLP networks," in *Proc. IEEE Int. Conf. Neural Netw.*, vol. 4, Dec. 1995, pp. 2052–2055.
[30] K. K. Ang, C. Quek, and M. Pasquier, "POPFNN-CRI(S): Pseudo outer product based fuzzy neural network using the compositional rule of inference and singleton fuzzifier," *IEEE Trans. Syst., Man, Cybern., Part B*, vol. 33, no. 6, pp. 838–849, Dec. 2003.
[31] N. K. Kasabov and Q. Song, "DENFIS: Dynamic evolving neural-fuzzy inference system and its application for time-series prediction," *IEEE Trans. Fuzzy Syst.*, vol. 10, no. 2, pp. 144–154, Apr. 2002.
[32] W. L. Tung and C. Quek, "GenSoFNN: A generic self-organizing fuzzy neural network," *IEEE Trans. Neural Netw.*, vol. 13, no. 5, pp. 1075–1086, Sep. 2002.
[33] O. Cordon, F. Herrera, F. Hoffmann, and L. Magdalena, "Advances in fuzzy systems-applications and theory," in *Genetic Fuzzy Systems Evolutionary Tuning and Learning of Fuzzy Knowledge Bases,* vol. 19, Hackensack, NJ: World Scientific, 2001.
[34] D. Nguyen and B. Widrow, "The truck backer-upper: An example of self-learning in neural network," in *Proc. IEEE Conf. Syst. Mag.*, vol. 10, no. 3, pp. 18–23, 1990.
[35] J. Hauser, S. Sastry, and P. Kokotovic, "Nonlinear control via approximate input–output lineariztion: The ball and beam example," *IEEE Trans. Autom. Control*, vol. 37, no. 3, pp. 392–398, Mar. 1992.
[36] C. J. Lin and W. H. Ho, "An asymmetry-similarity-measure-based neural fuzzy inference system," *Fuzzy Sets Syst.*, vol. 152, no. 3, pp. 535–551, Jun. 2005.

Cheng-Hung Chen (S'07) was born in Kaohsiung, Taiwan, in 1979. He received the B.S. and M.S. degrees in computer science and information engineering from the Chaoyang University of Technology, Taichung County, Taiwan, in 2002 and 2004, respectively, and the Ph.D. degree in electrical and control engineering from the National Chiao-Tung University, Hsinchu, Taiwan, in 2008.

His current research interests are fuzzy systems, neural networks, evolutionary algorithms, image processing, intelligent control, and pattern recognition.

Cheng-Jian Lin (S'93–M'95) received the B.S. degree in electrical engineering from Ta-Tung University, Taipei, Taiwan, in 1986 and the M.S. and Ph.D. degrees in electrical and control engineering from the National Chiao-Tung University, Hsinchu, Taiwan, in 1991 and 1996, respectively.

Currently, he is a full Professor with the Computer Science and Information Engineering Department, National Chin-Yi University of Technology, Taichung County, Taiwan. He served as the chairman of the Computer Science and Information Engineering Department, Chaoyang University of Technology, Taichiung County, from 2001 to 2005. He served as the library director of the Poding Memorial Library, Chaoyang University of Technology, from 2005 to 2007. He served as the Associate Editor of the *International Journal of Applied Science and Engineering* from 2002 to 2005. His current research interests are soft computing, pattern recognition, intelligent control, image processing, bioinformatics, and field programmable gate array design.

Dr. Lin is a member of Phi Tau Phi. He is also a member of the Chinese Fuzzy Systems Association (CFSA), the Chinese Automation Association, the Taiwanese Association for Artificial Intelligence (TAAI), the IEEE Systems, Man, and Cybernetics Society, and the IEEE Computational Intelligence Society. He was an executive committee member of the Taiwanese Association for Artificial Intelligence (TAAI) from 2003 to 2008. He was an executive committee member of the Chinese Fuzzy Systems Association (CFSA) from 2007 to 2008. He has received several honors and awards, including the 2006 Outstanding Paper Award from the 11th Conference on Artificial Intelligence and Applications, the 2007 Outstanding Paper Award from the 12th Conference on Artificial Intelligence and Applications, and the 2006 Best Paper Award of the *International Transactions on Computer Science and Engineering* (vol. 32, no. 1).

Chin-Teng Lin (S'88–M'91–SM'99–F'05) received the B.S. degree in control engineering from the National Chiao-Tung University (NCTU), Hsinchu, Taiwan, in 1996 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 was previously the Founding Dean of Computer Science College and is currently the Provost of Academic Affairs and the Chair Professor of Electrical and Control Engineering. He is the author or coauthor of more than 110 journal papers and the texbooks *Neural Fuzzy Systems* (Prentice–Hall, 1996) and *Neural Fuzzy Control Systems With Structure and Parameter Learning* (World Scientific, 1994). His research interests include intelligent technology, soft computing, brain–computer interfaces, intelligent transportation systems, robotics and intelligent sensing, nano-bio-information technologies, and cognitive science.

Dr. Lin was a member of the Board of Governors (BoG) of the IEEE Systems, Man, and Cybernetics Society from 2003 to 2005 and is a current BoG member of IEEE Circuits and Systems Society. He was an IEEE Distinguished Lecturer from 2003 to 2005. He is also the Deputy Editor-in-Chief of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS, PART II. He was the Program Chair of the 2006 IEEE International Conference on Systems, Man, and Cybernetics held in Taipei, Taiwan. He was the President of the BoG of the Asia Pacific Neural Networks Assembly from 2004 to 2005. He was the recipient of the Outstanding Research Award granted by the National Science Council, Taiwan, the Outstanding Professor Award granted by the Chinese Institute of Engineering in 2000, and the 2002 Taiwan Outstanding Information-Technology Expert Award. He was elected to be one of the 38th Annual Outstanding Rising Stars in Taiwan in 2000. He is a member of the Tau Beta Pi, Eta Kappa Nu, and Phi Kappa Phi honorary societies.