Efficient Self-Evolving Evolutionary Learning for Neurofuzzy Inference Systems

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

***Abstract*—This study proposes an efficient self-evolving evolutionary learning algorithm (SEELA) for neurofuzzy inference systems (NFISs). The major feature of the proposed SEELA is that it is based on evolutionary algorithms that can determine the number of fuzzy rules and adjust the NFIS parameters. The SEELA consists of structure learning and parameter learning. The structure learning attempts to determine the number of fuzzy rules. A subgroup symbiotic evolution is adopted to yield several variable fuzzy systems, and an elite-based structure strategy is adopted to find a suitable number of fuzzy rules for solving a problem. The parameter learning is to adjust parameters of the NFIS. It is a hybrid evolutionary algorithm of cooperative particle swarm optimization (CPSO) and cultural algorithm, called cultural CPSO (CCPSO). The CCPSO, which uses cooperative behavior among multiple swarms, can increase the global search capacity using the belief space. Experimental results demonstrate that the proposed method performs well in predicting time series and solving nonlinear control problems.**

***Index Terms*—Cooperative particle swarm optimization (CPSO), cultural algorithm (CA), elite-based structure strategy (ESS), neurofuzzy inference system (NFIS), symbiotic evolution.**

I. Introduction

NEUROFUZZY inference systems (NFISs) [1]–[7] have become a popular research topic. Such systems bring not only the low-level learning and computational power of neural networks into fuzzy systems, but also the high-level human-like thinking and reasoning of fuzzy systems to neural networks. Recently, genetic fuzzy systems [8]–[10] have received increasing attention mainly because they combine the approximate reasoning method of fuzzy systems with the learning capabilities of evolutionary algorithms.

An NFIS requires technologies to train the system parameters and find the global solution while optimizing the overall structure. Many recent developments in evolutionary algorithms have provided several strategies for NFIS design. Three main strategies, including Pittsburgh-type, Michigan-type, and the iterative rule learning genetic fuzzy systems, focus on generating and learning fuzzy rules in genetic fuzzy systems. First, the Pittsburgh-type genetic fuzzy system [11] was characterized by using a fuzzy system as an individual in genetic operators. Second, the Michigan-type genetic fuzzy system was used for generating fuzzy rules in [12] and [13], where each fuzzy rule was treated as an individual. Thus, the rule generation methods in [12] and [13] were referred to as fuzzy classifier systems. Third, the iterative rule learning genetic fuzzy system [14]–[16] was adopted to search one adequate rule set for each iteration of the learning process. Moreover, Ishibuchi *et al.* [17]–[20] proposed genetic algorithms (GAs) for constructing a fuzzy system consisting of a small number of linguistic rules. Mitra and coworkers [21]–[25] presented some approaches that exploit the benefits of soft computation tools for rule generation. Moriarty and Miikkulainen [26] proposed a reinforcement learning method, called symbiotic, adaptive neuro-evolution, which evolves a population of neurons through GAs to form a neural network. The GA adopted by Juang *et al.* [27] is based on symbiotic evolution that, when applied in fuzzy controller design, complements the local mapping property of a fuzzy rule. Lin and Xu [28] presented a group-based symbiotic evolution learning method that uses a group-based population to evaluate the fuzzy rule locally. Each group that represents a set of the individuals performs the evolution process in each generation. Additionally, Harik *et al.* [29] applied a compact GA that adopts a probability distribution to represent a population over the set of solutions, and uses probability vectors to estimate gene values as 1 or 0. Lin and Xu [30] proposed a hybrid evolutionary learning algorithm that uses probability vectors to ensure that the best group can be reproduced many times in each generation. Therefore, this study proposes a structure learning strategy. The strategy adopts a subgroup symbiotic evolution (SSE) to yield several variable fuzzy systems, and uses an elite-based structure strategy (ESS) to find a suitable number of fuzzy rules using probability values.

Moreover, a novel optimization algorithm called particle swarm optimization (PSO) performs better than the backpropagation algorithm. PSO is an evolutionary computation technique that was developed by Kennedy and Eberhart [31], [32]. The underlying concept of the PSO algorithm is the social behavior in animals, such as bird flocking, fish schooling, and swarming. PSO has been successfully applied to many optimization problems, such as control problems [33]–[35] and feedforward neural network design [36], [37]. However, PSO suffers from the burden of many dimensions, such that its performance falls as the dimensionality of the search space increases.

Manuscript received August 23, 2007; revised December 12, 2007; accepted February 16, 2008. Current version published December 19, 2008. This work was supported in part by the National Science Council, Taiwan, under Grant NSC 95-2221-E-324-028-MY2 and Grant NSC-96-2221-E-009-058 and in part by the Taiwan Information Security Center (TWISC) under the National Science Council Grant NSC96-2219-E-009-013.

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

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

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

Digital Object Identifier 10.1109/TFUZZ.2008.2005935

Therefore, Bergh and Engelbrecht [38] proposed a cooperative approach that employs cooperative behavior, called cooperative particle swarm optimization (CPSO), which uses multiple swarms to improve upon traditional PSO. However, the CPSO still uses the formula (the local best (*Lbest*) position of each particle and global best (*Gbest*) position in the swarm) of the traditional PSO to evolve, and therefore, may find a suboptimal solution. This study presents a novel parameter learning strategy, called cultural CPSO (CCPSO), which combines the CPSO and the cultural algorithm (CA), to increase global search capacity, thus avoiding trapping in a suboptimal solution, and ensuring that a nearby global optimal solution can be found.

This study proposes an efficient self-evolving evolutionary learning algorithm (SEELA) for NFISs. The NFIS is based on our previous research [39], and combines a fuzzy system with a functional link neural network (FLNN) [40]. The consequent part of the fuzzy rules that corresponds to an FLNN comprises the functional expansion of input variables.

The proposed SEELA consists of structure learning to determine the number of fuzzy rules, and parameter learning to adjust the NFIS parameters. The structure learning adopts an SSE to yield several variable fuzzy systems, and uses an ESS to find a suitable number of fuzzy rules. The SSE in which each subparticle represents a single fuzzy rule differs from original symbiotic evolution. A fuzzy system with R-rules is constructed by selecting and combining R subparticles from each subgroup, and allowing the rule itself to evolve. The ESS adopts probability values to find a suitable number of fuzzy rules of a fuzzy system using the fitness values of variable fuzzy systems. The parameter learning is performed using an efficient hybrid method called CCPSO that combines CPSO and CAs. The proposed CCPSO method with cooperative behavior among multiple swarms increases the global search capacity using the belief space. Cooperative behavior among multiple swarms involves interaction by exchanging information with each other to solve a problem. The belief space is the information repository in which the individuals can store their experiences for other individuals to learn from them indirectly.

The advantages of the proposed method are summarized as follows: 1) the proposed SEELA uses SSE and ESS to determine the number of fuzzy rules; 2) the proposed SEELA adopts CCPSO to adjust the NFIS parameters. The proposed CCPSO method with cooperative behavior among multiple swarms increases the global search capacity using the belief space; 3) as demonstrated in Section 5, the proposed SEELA can obtain a smaller rms error than the other methods.

The rest of this paper is organized as follows. Section II describes the basic concept of symbiotic evolution, PSO, and CA. Section III presents the structure of the NFIS. Next, Section IV presents the SEELA. The results of the simulation of predictive applications and nonlinear control problems are described in Section V. Section VI draws conclusions.

II. Symbiotic Evolution, Particle Swarm Optimization, and Cultural Algorithm

This section describes basic concepts concerning symbiotic evolution, PSO, and the CA. The specialization property of symbiotic evolution, PSO, and CA is consistent with the learning algorithm property of the NFIS. Therefore, the development of an NFIS based on symbiotic evolution, PSO, and the CA is valuable.

A. Symbiotic Evolution

The notion of symbiotic evolution [26], [27] 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 effectively protect against various antigens. These antibodies are used to separate the population into subpopulations and change 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.

Partial solutions can be characterized as *specializations* in [26] and [27]. 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 [26], [27].

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, who 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, most of the overall performance of the fuzzy system is known, and less of 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. Particle Swarm Optimization

In 1995, Kennedy and Eberhart introduced the PSO algorithm [31], [32]. The PSO is a population-based optimization approach, in which the population is called a swarm. Furthermore, each swarm consists of many particles. Each particle has a velocity vector v_i and a position vector x_i, which represents a possible solution. Consider an optimization problem that requires the simultaneous optimization of variables. A collection or swarm of particles is defined, in which each particle is assigned a random position in the P-dimensional problem space so that the position of each particle corresponds to a candidate solution to the optimization problem. Then, the particles move rapidly around and search the solution space using the moving

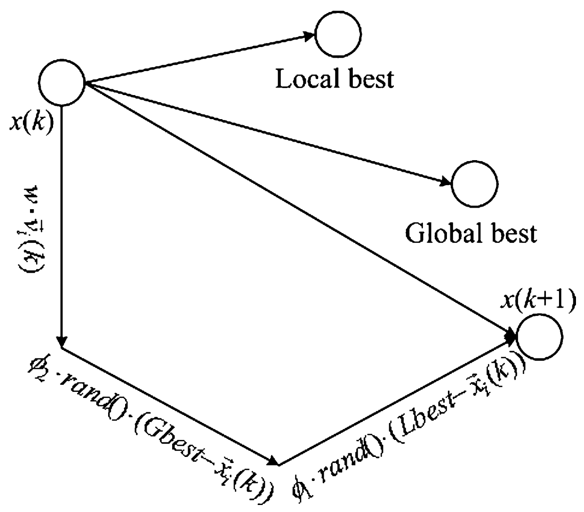


Fig. 1. Diagram of the updated velocity in the PSO.

velocity of each particle. PSO applies a simple rule. Each of these particle positions is scored to obtain a fitness value, based on how to define the solution of the problem. The *Lbest* position of each particle and the *Gbest* position in the swarm are used to yield a new velocity for each particle

$$v_i(k+1) = \omega * v_i(k) + \phi_1 * \text{Rand}() * (\text{Lbest} - x_i(k)) + \phi_2 * \text{Rand}() * (\text{Gbest} - x_i(k)) \quad (1)$$

where ω, ϕ_1, and ϕ_2 are called the coefficient of the inertia term, the cognitive term, and the society term, respectively. In the original PSO algorithm, ω is set in range [0.9, 1.2], and ϕ_1 and ϕ_2 are both set to 2. Rand() yields uniformly distributed random numbers in [0, 1]. The term v_i is limited to the range $\pm v_{\max}$. If the velocity violates this limit, then it is set to the actual limit. Fig. 1 presents the concept of the updated velocity.

Changing the velocity enables each particle to search around its individual best position and *Gbest* position. Based on the updated velocities, each particle changes its position according to

$$x_i(k+1) = x_i(k) + v_i(k+1). \quad (2)$$

C. Cultural Algorithm

CAs [41], [42] involve acquiring the belief space from the evolving population space, and then, exploiting that information to guide the search. Fig. 2 presents the CA components. CAs can be described in terms of two basic components—the belief space and the population space. The belief space is the information repository in which the individuals can store their experiences for other individuals to learn from them indirectly. In CAs, the information acquired by an individual can be shared with the entire population, unlike in most evolutionary techniques, in which the information can be shared only with the offspring of the individual. The population space comprises a set of possible solutions to the problem, and can be modeled using any population-based approach. The belief space and the population space are linked using a scheme that states rules that

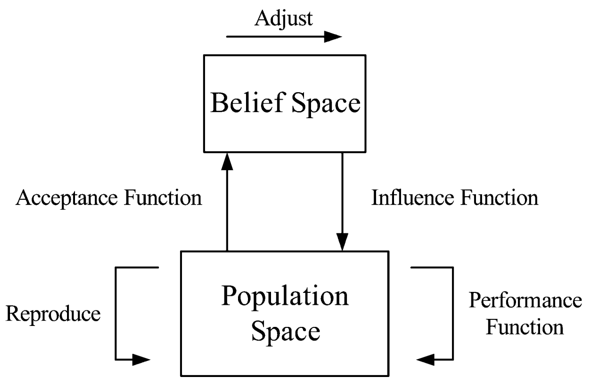


Fig. 2. Framework of the CA.

govern the individuals of the population space that can contribute to the belief space based on its experiences (according to the acceptance function), and the belief space can influence the new individuals of the population space (according to the influence function).

The acceptance function selects the top 20% of individuals that can directly affect the formation of the current belief space. The influence function adopts the normative knowledge containing the intervals for the variables where good solutions have been found, in order to move novel solutions toward those intervals. The following expression shows the influence of the normative knowledge on the variation operators:

$$x_i = \begin{cases} x_i + |\text{Rand}() \cdot (u-l)|, & \text{if } x_i < l \\ x_i - |\text{Rand}() \cdot (u-l)|, & \text{if } x_i > u \\ x_i + \frac{\text{Rand}() \cdot (u-l)}{m}, & \text{otherwise} \end{cases} \quad (3)$$

where x_i is the ith variable of the individual; u and l are the upper and lower bounds of all individuals of the belief space, respectively, and m is the number of all individuals of the belief space.

III. Structure of an NFIS

This section describes the NFIS model, which uses a nonlinear combination of input variables (FLNN). The NFIS is based on our previous research [39]. Each fuzzy rule corresponds to a sub-FLNN [40], comprising a functional link. Fig. 3 presents the structure of the proposed NFIS model.

The NFIS model realizes a fuzzy IF–THEN rule in the following form:

Rule$_j$:

IF $\hat{x}_1$ is A_{1j} and $\hat{x}_2$ is A_{2j} $\cdots$ and $\hat{x}_i$ is A_{ij} $\cdots$ and $\hat{x}_N$ is A_{Nj}

$$\text{THEN } \hat{y}_j = \sum_{k=1}^{M} w_{kj}\phi_k = w_{1j}\phi_1 + w_{2j}\phi_2 + \cdots + w_{Mj}\phi_M \quad (4)$$

where $\hat{x}_i$ and $\hat{y}_j$ are the input and local output variables, respectively, A_{ij} is the linguistic term of the precondition part with a Gaussian membership function, N is the number of input variables, w_{kj} is the link weight of the local output, ϕ_k is the

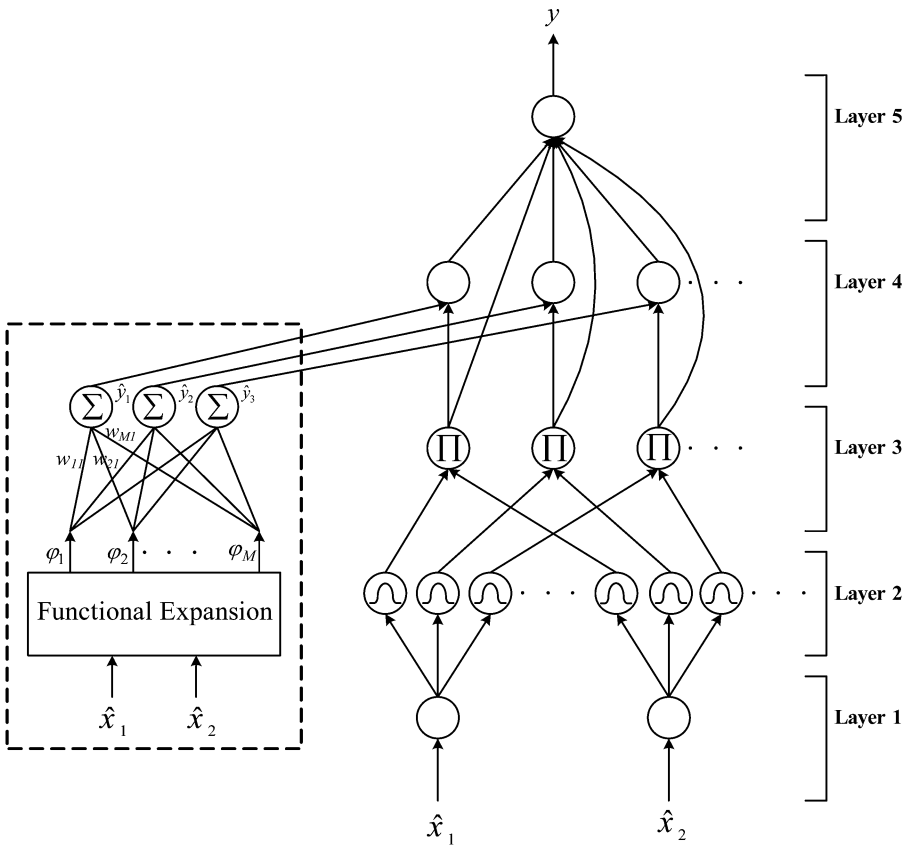


Fig. 3. Structure of the proposed NFIS model.

basis trigonometric function of input variables, M is the number of basis functions, and Rule_j is the jth fuzzy rule.

The operation functions of the nodes in each layer of the NFIS model are now described. In the following description, $u^{(l)}$ denotes the output of a node in the lth layer.

A. Layer 1 (Input Node)

No computation is performed in this layer. Each node in this layer is an input node, which corresponds to one input variable, and only transmits input values to the next layer directly

$$u_i^{(1)} = \hat{x}_i. \tag{5}$$

B. Layer 2 (Membership Function Node)

Nodes in this layer correspond to a single linguistic label of input variables in layer 1. Therefore, the calculated membership value specifies the degree to which an input value belongs to a fuzzy set in layer 2. The implemented Gaussian membership function in layer 2 is

$$u_{ij}^{(2)} = \exp\left(-\frac{[u_i^{(1)} - m_{ij}]^2}{\sigma_{ij}^2}\right) \tag{6}$$

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 $\hat{x}_i$.

C. Layer 3 (Rule Node)

Nodes in this layer represent the preconditioned part of a fuzzy logic rule. They receive 1-D membership degrees of the associated rule from the nodes of a set in layer 2. Here, the product operator described earlier is adopted to perform the IF-condition matching of the fuzzy rules. As a result, the output function of each inference node is

$$u_j^{(3)} = \prod_i u_{ij}^{(2)} \tag{7}$$

where $\prod_i u_{ij}^{(2)}$ of a rule node represents the firing strength of its corresponding rule.

D. Layer 4 (Consequent Node)

Nodes in this layer are called consequent nodes. The input to a node in layer 4 is the output from layer 3, and the other inputs are nonlinear combinations of input variables from an FLNN. For such a node,

$$u_j^{(4)} = u_j^{(3)} \cdot \sum_{k=1}^{M} w_{kj}\phi_k \tag{8}$$

where w_{kj} is the corresponding link weight of the FLNN and ϕ_k is the functional expansion of input variables. The functional expansion uses a trigonometric polynomial basis function, given by $[\hat{x}_1,\ \sin(\pi\hat{x}_1),\ \cos(\pi\hat{x}_1), \hat{x}_2,\ \sin(\pi\hat{x}_2),\ \cos(\pi\hat{x}_2)]$ for 2-D input variables. Therefore, M is the number of basis functions,

$M = 3 \times N$, where N is the number of input variables. Moreover, the output nodes of an FLNN depend on the number of fuzzy rules of the NFIS model.

E. Layer 5 (Output Node)

Each node in this layer corresponds to a single output variable. The node integrates all of the actions recommended by layers 3 and 4 and acts as a *center of area* defuzzifier with

$$y = u^{(5)} = \frac{\sum_{j=1}^{R} u_j^{(4)}}{\sum_{j=1}^{R} u_j^{(3)}} = \frac{\sum_{j=1}^{R} u_j^{(3)} \left(\sum_{k=1}^{M} w_{kj}\phi_k\right)}{\sum_{j=1}^{R} u_j^{(3)}} = \frac{\sum_{j=1}^{R} u_j^{(3)} \hat{y}_j}{\sum_{j=1}^{R} u_j^{(3)}} \tag{9}$$

where R is the number of fuzzy rules and y is the output of the NFIS model. As described earlier, the number of tuning parameters for the NFIS model is known to be $5 \times N \times R$, where N and R denote the number of inputs and existing rules, respectively.

IV. Self-Evolving Evolutionary Learning Algorithm for the NFIS Model

This section describes the proposed SEELA. The SEELA comprises structure learning and parameter learning. The structure learning consists of an SSE and an ESS. In the SSE, the fitness value of a rule (a subparticle) is computed as the sum of the fitness 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. The concept of ESS is based on the maturing phenomenon in society, where individuals adapt to the environment as they acquire more knowledge from the society. The ESS can find a suitable number of rules and the combinations of rules according to probability values.

The parameter learning adopts a CCPSO to adjust the NFIS parameters. The traditional PSO uses one swarm of particles defined by the P-dimension vectors to evolve. The CPSO method [38] can change a traditional PSO into P swarms of 1-D vectors, such that each swarm represents a dimension of the original problem. Fig. 4(a) and (b) shows the framework of the traditional PSO and CPSO methods. The key point is that, instead of using one swarm (of I particles) to find the optimal P-dimension vector, the vector is split into its components so that P swarms (of I particles each) optimize a 1-D vector where each 1-D vector represents each swarm, as shown in Fig. 4(b). Notably, the function that is being optimized still requires a P-dimension vector to be evaluated. Additionally, each swarm aims to optimize a single component of the solution vector essentially solving a 1-D optimization problem. Unfortunately, the CPSO still employs just the *Lbest* position and the *Gbest* position of the traditional PSO to the evolution process. The trajectory of each particle in the search space is adjusted according to the *Lbest* position of the particle and the *Gbest* position in the same search space, but it is unable to yield high diversity of particles to increase the search space. That is, it is lacking enough capability to satisfy the requirements of exploration [43], [44].

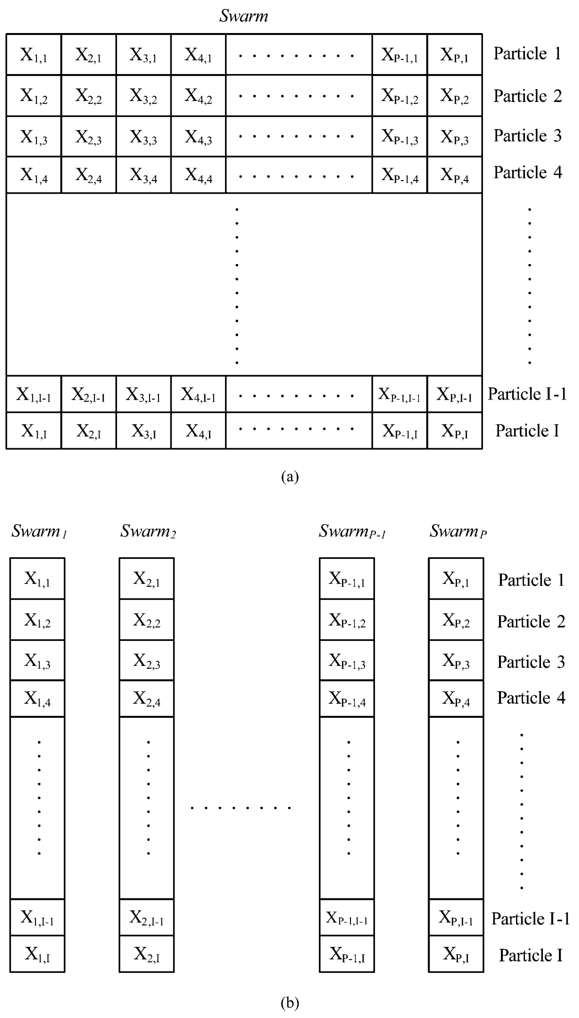

Swarm
$X_{1,1}$ $X_{2,1}$ $X_{3,1}$ $X_{4,1}$ $X_{P-1,1}$ $X_{P,1}$ Particle 1
$X_{1,2}$ $X_{2,2}$ $X_{3,2}$ $X_{4,2}$ $X_{P-1,2}$ $X_{P,2}$ Particle 2
$X_{1,3}$ $X_{2,3}$ $X_{3,3}$ $X_{4,3}$ $X_{P-1,3}$ $X_{P,3}$ Particle 3
$X_{1,4}$ $X_{2,4}$ $X_{3,4}$ $X_{4,4}$ $X_{P-1,4}$ $X_{P,4}$ Particle 4
$X_{1,I-1}$ $X_{2,I-1}$ $X_{3,I-1}$ $X_{4,I-1}$ $X_{P-1,I-1}$ $X_{P,I-1}$ Particle I-1
$X_{1,I}$ $X_{2,I}$ $X_{3,I}$ $X_{4,I}$ $X_{P-1,I}$ $X_{P,I}$ Particle I
(a)
$Swarm_1$ $Swarm_2$ $Swarm_{P-1}$ $Swarm_P$
(b)

Fig. 4. Framework. (a) PSO. (b) CPSO.

Therefore, the CPSO may fall into a suboptimal solution. Recently, CAs [41], [42] exploited the information of the specific belief space to guide the feasible search space and can change the direction of each individual in the solution space. Hence, the proposed CCPSO method, which combines the CPSO and the CA to increase the global search capacity, is proposed to avoid trapping in a suboptimal solution and ensure the ability to search for a near global optimal solution. Fig. 5 shows the framework of the proposed CCPSO method, which is based on a CPSO, all of whose parameters are simultaneously tuned using the belief space of the CA. In the aforementioned scheme, the proposed CCPSO method can avoid falling into a suboptimal solution and ensures that the approximate global optimal solution can be found.

The SEELA has three major phases: initialization, structure learning, and parameter learning. First, the initialization phase can create initial swarms and the belief space. Second, the structure learning phase includes SSE and ESS to determine the number of fuzzy rules. Third, the parameter learning phase uses CCPSO to adjust the NFIS parameters. The detailed flowchart of the proposed SEELA method is presented in Fig. 6.

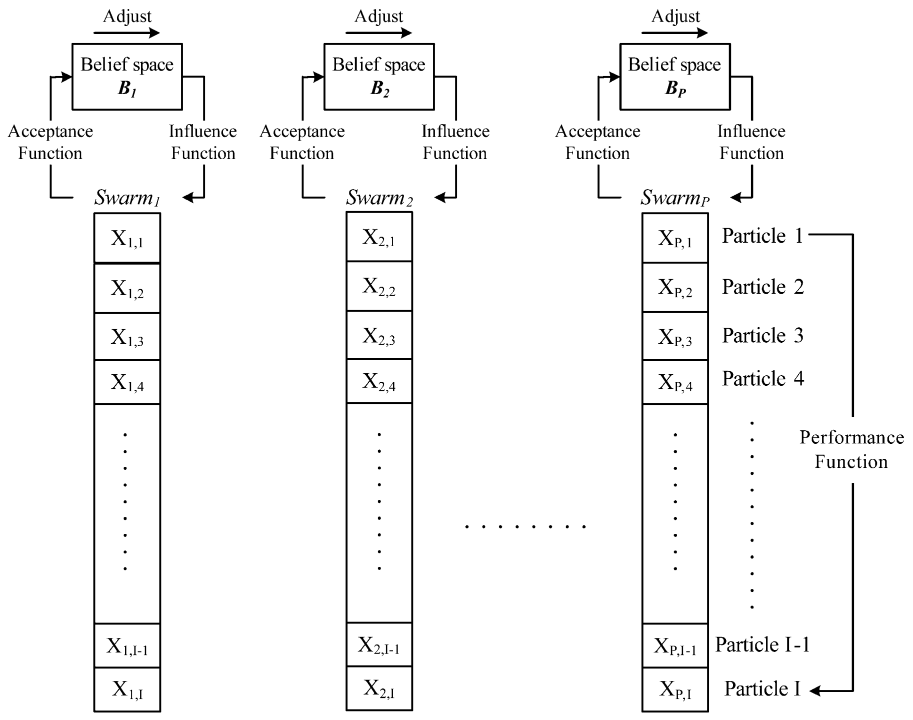


Fig. 5. Framework of the proposed CCPSO method.

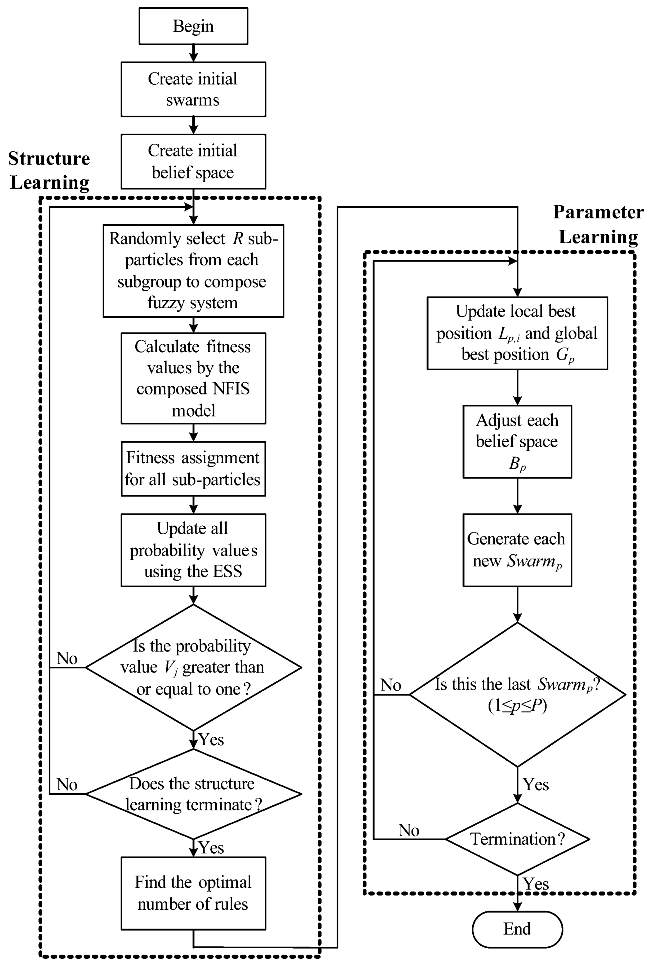

Begin
Create initial swarms
Create initial belief space
Structure Learning
Randomly select R sub-particles from each subgroup to compose fuzzy system
Calculate fitness values by the composed NFIS model
Fitness assignment for all sub-particles
Update all probability values using the ESS
Is the probability value V_j greater than or equal to one?
No
Yes
Does the structure learning terminate?
No
Yes
Find the optimal number of rules
Parameter Learning
Update local best position $L_{p,i}$ and global best position G_p
Adjust each belief space B_p
Generate each new $Swarm_p$
Is this the last $Swarm_p$? ($1 \leq p \leq P$)
No
Yes
Termination?
No
Yes
End

Fig. 6. Flowchart of the proposed SEELA method.

Sub-particle

m_{1j}	σ_{1j}	m_{2j}	σ_{2j}	…..	m_{ij}	σ_{ij}	…..	w_{1j}	w_{2j}	…..	w_{Mj}

Fig. 7. Coding a fuzzy rule into a subparticle in the proposed SEELA.

A. Initialization Phase

1) Coding Step: The foremost step in SEELA is the coding of a fuzzy rule into a subparticle. Fig. 7 shows an example of the coding of parameters of a fuzzy rule into a subparticle where i and j represent the ith input variable and the jth rule, respectively. In this study, a Gaussian membership function is adopted with variables that represent the mean and deviation of the membership function. Fig. 7 represents a fuzzy rule given by (4), where m_{ij} and σ_{ij} are the mean and deviation of a Gaussian membership function, respectively, and w_{kj} represents the corresponding link weight of the consequent part that is connected to the jth rule node. In this study, a real number represents the position of each subparticle.

2) Create Initial Swarms: Before the SEELA method is applied, every position $x_{p,i}(t)$ must be created randomly in the range [0, 1] in each subgroup, where $p = 1, 2, \ldots, P$ represents the pth swarm, $i = 1, 2, \ldots, I$ represents the ith particle, and t denotes the t-th generation.

3) Create Initial Belief Spaces: The belief space is the information repository in which the particles can store their experiences for other particles to learn from them indirectly. Create P belief spaces, $B_p (p = 1, 2, \ldots, P)$. Each initial B_p is defined as an empty set.

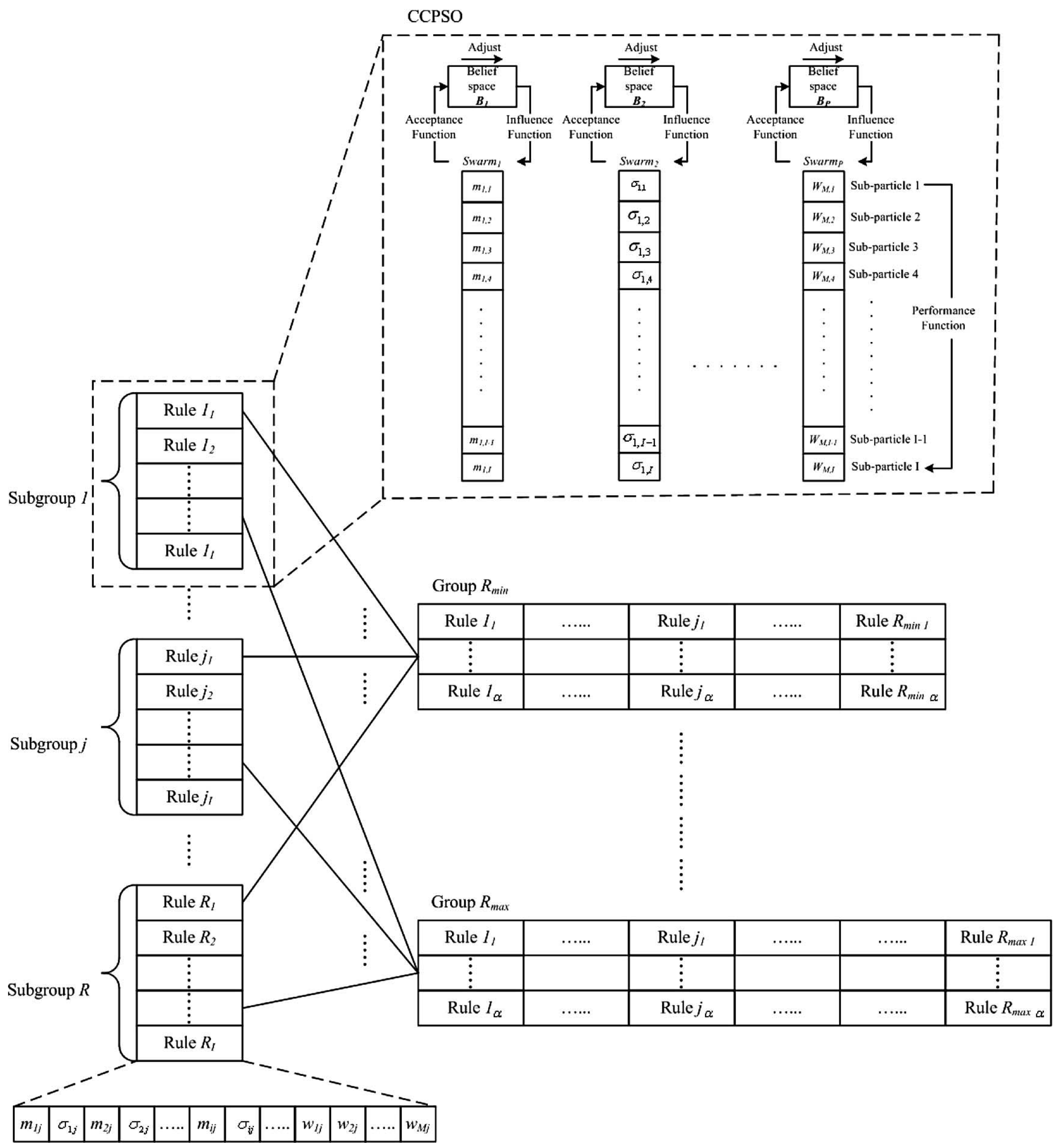


Fig. 8. Structure of the subparticle in SSE.

B. Structure Learning Phase

1) Subgroup Symbiotic Evolution: In order to keep the same rule numbers of each fuzzy system in each group, the size α needs to be defined in each group, i.e., the size of each group is α. Therefore, the size of all groups should be set to $\alpha \cdot (R_{\max} - R_{\min} + 1)$, where $R_{\max}$ and $R_{\min}$ represent the maximum number of rules and the minimum number of rules, respectively. In this step, the fitness value of a rule (a subparticle) is computed as the sum of the fitness 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. 8 shows the structure of the subparticle in the SSE. The stepwise assignment of the fitness value is shown as follows.

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

Step 2: Calculate the fitness value of the particles using the NFIS thus composed. In this study, the fitness value is given by the following formula:

$$F = \sqrt{\frac{1}{D}\sum_{d=1}^{D}(y_d - \overline{y}_d)^2} \tag{10}$$

where y_d represents the dth model output, $\overline{y}_d$ represents the dth desired output, and D represents the number of training data.

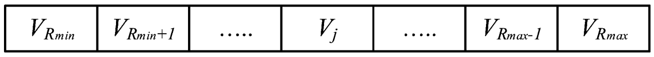


Fig. 9. Coding probability values into BBs.

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

Step 4: Repeat the aforementioned steps until the space of each group has been filled a sufficient number of times, and record the number of fuzzy systems to which each subparticle has contributed.

Step 5: Divide the accumulated fitness of each subparticle by the number of times it has been selected.

Step 6: Sort these subparticles in each subgroup in the order of increasing fitness.

2) Elite-Based Structure Strategy: The foremost step in ESS is the coding of the probability value V_j into building blocks (BBs), as shown in Fig. 9, where the probability value represents the appropriate degree of rule numbers of a fuzzy system. In Fig. 9, $R_{\max}$ and $R_{\min}$ are predefined to prevent less or more fuzzy rules from being generated in a fuzzy system. According to the results of the ESS, the appropriate rule numbers and rule combinations can be found. The details of ESS are shown as follows.

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

$$\begin{cases} V_j(t_s+1) = V_j(t_s) & \text{if } \mathrm{Avg}_j \text{ is the best} \\ \quad + (\mathrm{Upt\_value}_j \cdot \lambda), & \quad \text{performance} \\ V_j(t_s+1) = V_j(t_s), & \text{otherwise} \end{cases} \tag{11}$$

where $j = [R_{\min}, R_{\max}]$, $\mathrm{Avg}_j = \sum_{\alpha'=1}^{\alpha} F_{\alpha'}/\alpha$, $\mathrm{Upt\_value}_j = \sum_{\alpha'=1}^{\alpha} F_{\alpha'} / \sum_{R'=R_{\min}}^{R_{\max}} \sum_{\alpha'=1}^{\alpha} F_{\alpha'}$, where V_j is a probability value in the BBs and presents the appropriate rule numbers of a fuzzy system, λ is a constant, Avg_j is a average fitness value in the jth group, $F_{\alpha'}$ is the fitness value of each composed fuzzy system in each group, and α is the size of each group.

Step 2: Find the appropriate rule numbers. The probability values of BBs are initially set to zero. Repeat step 1 until V_j is greater than or equal to one. The finished prior step including subgroup symbiotic evolution and elite-based structure strategy is called one structure learning. Therefore, the maximum number of generations for structure learning should be set to N_s. Accumulate probability values of each structure learning in BBs and divide the accumulated probability values by N_s to find the appropriate rule numbers and rule combinations.

C. Parameter Learning Phase

The parameter learning phase adopts a CCPSO that combines the CPSO and the CA. The parameter learning process is described step-by-step as follows.

Step 1: Update *Lbest* position $L_{p,i}$ and *Gbest* position G_p. The *Lbest* position $L_{p,i}$ is the best previous position that yielded the best fitness value of the pth swarm of the ith particle, and the *Gbest* position G_p is generated by the whole *Lbest* position. In step 1, the first step updates the *Lbest* position. Compare the fitness value of each current particle with that of its *Lbest* position. If the fitness value of the current particle exceeds those of its *Lbest* position, then the *Lbest* position is replaced with the position of the current particle. The second step updates the *Gbest* position. Compare the fitness value of all particles in their *Lbest* positions with that of the particle in the *Gbest* position. If the fitness value of the particle in the *Lbest* position is better than those of the particles in the *Gbest* position, then the *Gbest* position is replaced with the current *Lbest* position

$$L_{p,i}(t+1) = \begin{cases} x_{p,i}(t), & \text{if } F(x_{p,i}(t)) < F(L_{p,i}(t)) \\ L_{p,i}(t), & \text{if } F(x_{p,i}(t)) \geq F(L_{p,i}(t)) \end{cases}$$
$$G_p(t+1) = \arg\min_{L_{p,i}} F(L_{p,i}(t+1)), \quad 1 \leq i \leq I. \tag{12}$$

Step 2: Adjust each belief space B_p using an acceptance function. The first part of step 2 sorts these particles in each *Swarm*$_p$ in the order of increasing fitness. Then, the paragon of each *Swarm*$_p$ is put into the belief space B_p using an acceptance function. This function yields the number of particles that are used to adjust each belief space and is shown as follows. The number of accepted particles decreases as the number of generations increases

$$N_{\text{accepted}} = n\% \cdot I + \frac{n\%}{t} \cdot I \tag{13}$$

where $n\%$ is a parameter that is set by the user, and must specify the top performing 20% [45], I is the number of particles, and t represents the t-th generation. The second step adjusts B_p. The interval of the belief space BI_p is defined $BI_p = [l_p, u_p] = \{x | l_p \leq x \leq u_p, x \in \Re\}$, where l_p is the lower bound of the belief space B_p and u_p is the upper bound of the belief space B_p. Then, the position of each particle in B_p is compared with the lower bound l_p. If the position of the particle is smaller than the lower bound l_p, then the lower bound l_p is replaced with the current position. Furthermore, the position of each particle in the B_p is compared with the upper bound u_p. If the position of the particle is greater than the upper bound u_p, then the upper bound u_p is replaced with the current position. These rules are given as follows:

$$l_p = \begin{cases} x_{p,i}, & \text{if } x_{p,i} \le l_p \\ l_p, & \text{otherwise} \end{cases}$$

$$u_p = \begin{cases} x_{p,i}, & \text{if } x_{p,i} \ge u_p \\ u_p, & \text{otherwise.} \end{cases} \tag{14}$$

Step 3: Generate each new $Swarm_p$ using $l_p, u_p, L_{p,i}$, and G_p. In step 3, the first step adjusts every position of each $Swarm_p$ using an influence function (15). This step can change the direction of each particle in the solution space, not easily being trapped at a local optimum. Then, the second step updates velocity and position of each particle to generate each new $Swarm_p$ using (16) and (17)

$$x_{p,i}(t) = \begin{cases} x_{p,i}(t) + |\text{Rand}() \cdot (u_p - l_p)|, & \text{if } x_{p,i} < l_p \\ x_{p,i}(t) - |\text{Rand}() \cdot (u_p - l_p)|, & \text{if } x_{p,i} > u_p \end{cases} \tag{15}$$

$$\begin{aligned} v_{p,i}(t+1) = {} & w \cdot v_{p,i}(t) \\ & + c_1 \cdot \text{Rand}() \cdot [L_{p,i}(t+1) - x_{p,i}(t)] \\ & + c_2 \cdot \text{Rand}() \cdot [G_p(t+1) - x_{p,i}(t)] \end{aligned} \tag{16}$$

$$x_{p,i}(t+1) = x_{p,i}(t) + v_{p,i}(t+1) \tag{17}$$

where c_1 and c_2 denote acceleration coefficients; Rand() is generated from a uniform distribution in the range [0, 1], and w controls the magnitude of $v_{p,i}(t)$.

D. Parameters Analysis of SEELA

This section analyzes all predefined parameters of SEELA. These parameters are explained as follows:

Notation	Description
$R_{\min}$	the minimum number of fuzzy rules in SSE and ESS
$R_{\max}$	the maximum number of fuzzy rules in SSE and ESS
α	the size of each group in SSE and ESS
λ	a constant value can influence the updated rate of V_j in (11)
N_s	the maximum number of generations for structure learning
$n\%$	a parameter of acceptance function determines the number of individuals that can enter the belief space in (13)
w	the coefficient of the inertia term can control the magnitude of velocity in (16)
c_1	the coefficient of the cognitive term for local best in (16)
c_2	the coefficient of the society term for global best in (16)

$R_{\min}$, $R_{\max}$, α, λ, and N_s influence structure learning, and $n\%, w, c_1$, and c_2 influence parameter learning of SEELA. $R_{\min}$, $R_{\max}$, α, and N_s depend on the complexity of the problem. The selection of parameter λ critically affects the analysis of V_j. The parameter λ, which uses the range (0, 1], was carefully examined in extensive experiments, and defined as [0.01, 0.5]. The variable $n\%$ is a parameter of the acceptance function given by the user in (0, 50%]; Saleem [45] suggests using 20%. The variables w, c_1, and c_2 are three coefficients of updated velocity. The determination of these three coefficients are based on practical experimentation and then defined by the user in [0.4, 0.9], [1, 2], and [1, 2].

TABLE I
INITIAL PARAMETERS BEFORE TRAINING

Parameter	Value
w	0.4
c_1	1.6
c_2	2
$R_{\min}$	2
$R_{\max}$	12
λ	0.05
α	10
N_s	10
$n\%$	20%
Coding Type	Real Number

V. Experimental Results

This section discusses three examples that were considered to evaluate the NFIS model with the SEELA method. The first example involves predicting a chaotic time series [46], the second example involves forecasting the number of sunspots [47], and the third example involves controlling backing up the truck [48]. Table I presents the initial parameters before training used in the three simulations.

A. Example 1: Prediction of Chaotic Time Series

The Mackey–Glass chaotic time series $x(t)$ was generated using the following delay differential equation:

$$\frac{dx(t)}{dt} = \frac{0.2x(t-\tau)}{1 + x^{10}(t-\tau)} - 0.1x(t). \tag{18}$$

Cowder [46] extracted 1000 input–output data pairs $\{x, y^d\}$ using four past values of $x(t)$

$$[x(t-18), x(t-12), x(t-6), x(t); x(t+6)] \tag{19}$$

where $\tau = 17$ and $x(0) = 1.2$. Four inputs to the NFIS model corresponded to these values of $x(t)$, and one output was $x(t + \Delta t)$, where Δt is the time interval into the future. The first 500 pairs [from $x(1)$ to $x(500)$] were the training dataset, while the remaining 500 pairs [from $x(501)$ to $x(1000)$] were the testing data used to validate the proposed method. For the simulation,

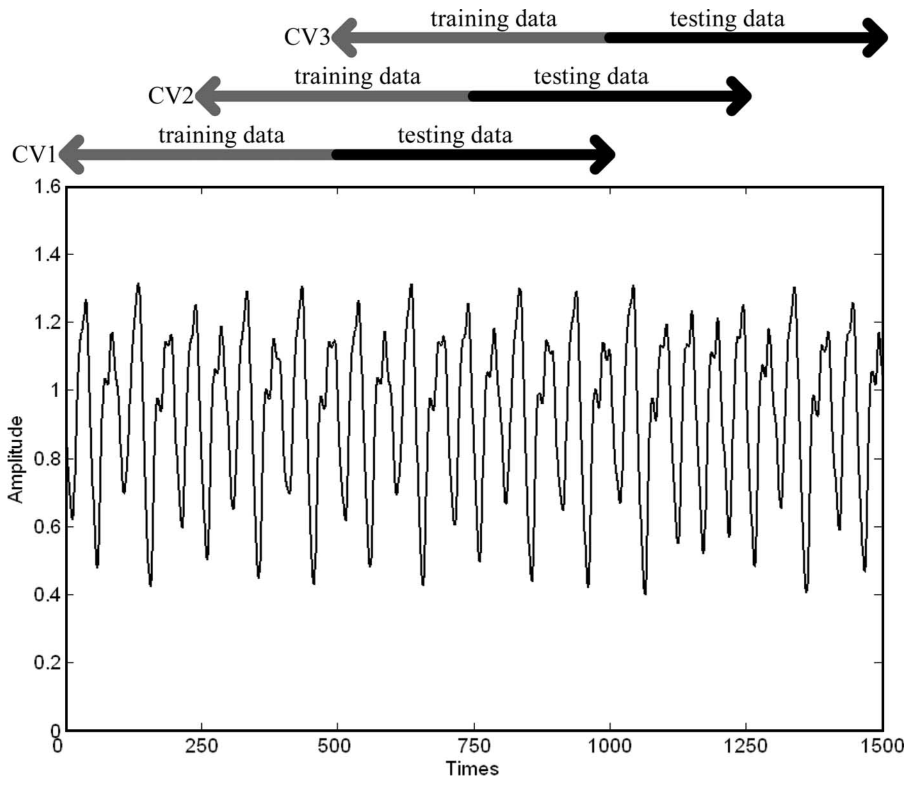


Fig. 10. Three cross-validation groups of training and testing data.

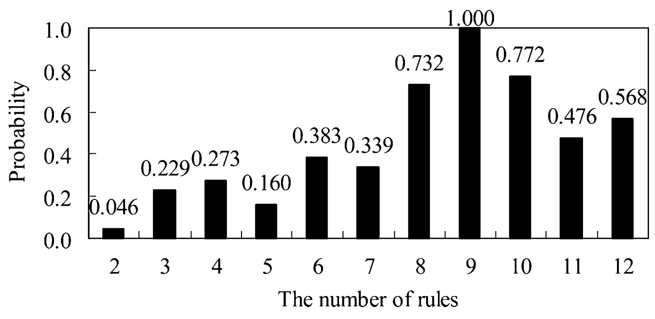


Fig. 11. Results of probability values of the ESS step in the proposed SEELA in CV1.

three cross-validation groups of training and testing data are used. They are CV1, CV2, and CV3. The training and testing windows are labeled in Fig. 10.

Fig. 11 shows the result of probability values after structure learning in CV1. As shown in Fig. 11, nine rules are adopted using ESS. The learning stage entered parameter learning through the CCPSO method. The learning proceeded for 1000 generations, and was performed ten times. The final rms error of the prediction output is about 0.00747 in CV1.

In this example, PSO [31], CPSO [38], and GA [49] were applied to the same problem to show the effectiveness and efficiency of the NFIS model with the SEELA method. In the PSO and CPSO, the coefficient w was set to 0.4, the cognitive coefficient c_1 was set to 1.6, and the society coefficient c_2 was set to 2. Nine rules are set to construct the fuzzy model. The learning proceeded for 1000 generations, and was performed ten times. Fig. 12 plots the learning curves of the NFIS model with the SEELA method, PSO, CPSO, and GA in CV1. The proposed method yields better prediction results than the other methods. Table II shows that the performance of the SEELA was compared with those of PSO [31], CPSO [38], and GA [49] in CV1, CV2, and CV3. Table III lists the generalization capabilities of other methods [1], [6], [27], [46]. The generalization capabilities were measured by using each model to predict 500 points immediately following the training dataset. The results show that the proposed NFIS model with the SEELA method offers a smaller rms error than other methods.

B. Example 2: Forecast of the Number of Sunspots

The number of sunspots varied nonlinearly from 1700 to 2004, in nonstationary, and non-Gaussian cycles that are difficult to predict [47]. In this example, the NFIS model with the SEELA method was used to forecast the number of sunspots. The inputs x_i of the NFIS model are defined as $x_1(t) = y_1^d(t-1), x_2(t) = y_1^d(t-2)$, and $x_3(t) = y_1^d(t-3)$, where t represents the year and $y_1^d(t)$ is the number of sunspots in the year t. In this example, the number of sunspots of the first 151 years (from 1703 to 1853) was used to train the NFIS model with the SEELA method while the number of sunspots of all 302 years (from 1703 to 2004) was used to test the trained NFIS model.

The learning stage involved parameter learning by the CCPSO method. Fig. 13 shows the result of probability values after structure learning. As shown in Fig. 13, nine rules are adopted according to ESS. The learning proceeded for 1000 generations,

TABLE II
COMPARISON OF BEST PERFORMANCE OF SEELA, PSO, CPSO, AND GA IN EXAMPLE 1

	SEELA		PSO [31]		CPSO [38]		GA [49]	
	RMS error (training)	RMS error (predicting)	RMS error (training)	RMS error (predicting)	RMS error (training)	RMS error (predicting)	RMS error (training)	RMS error (predicting)
CV1	**0.00745**	**0.00747**	0.00978	0.00989	0.00851	0.00866	0.016176	0.016341
CV2	**0.00572**	**0.00576**	0.00861	0.00883	0.00747	0.00762	0.014396	0.014707
CV3	**0.00692**	**0.00702**	0.00909	0.00928	0.00798	0.00816	0.014882	0.015361
Avg.	**0.00670**	**0.00675**	0.00919	0.00933	0.00799	0.00815	0.015151	0.015470

TABLE III
COMPARISON OF PERFORMANCE OF VARIOUS EXISTING MODELS

Method	RMSE $_{\text{Prediction}}$
NFIS-SEELA	**0.00675**
Back-propagation NN	0.02
Six-order polynomial	0.04
Cascaded-correlation	0.06
Auto regressive model	0.19
Linear predictive	0.55
ANFIS [1]	0.07
DENFIS [6]	0.033
SEFC [27]	0.032

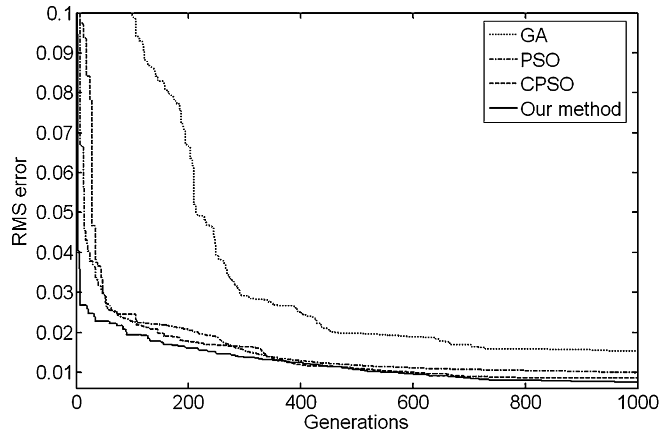


Fig. 12. Learning curves of best performance of the proposed SEELA method, PSO [31], CPSO [38], and GA [49] in CV1.

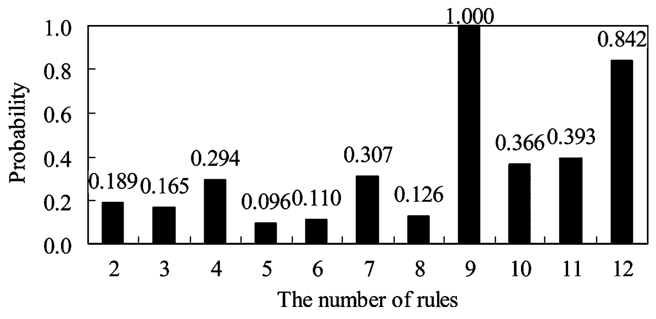


Fig. 13. Results of probability values of the ESS step in the proposed SEELA.

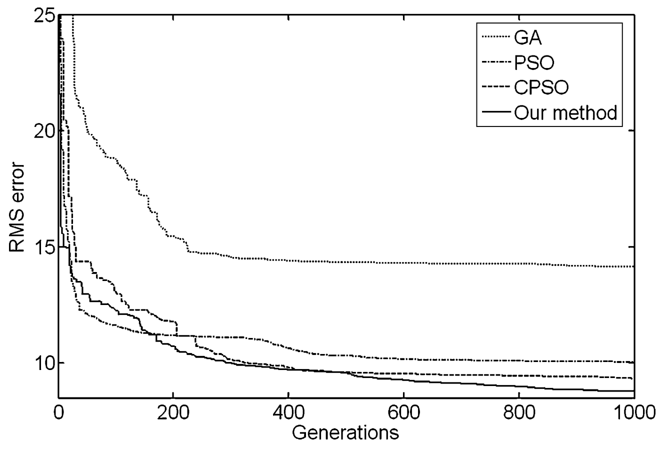


Fig. 14. Learning curves of best performance of the proposed SEELA method, PSO [31], CPSO [38], and GA [49].

TABLE IV
COMPARISON OF BEST PERFORMANCE OF SEELA, PSO, CPSO, AND GA IN EXAMPLE 2

	SEELA	PSO[31]	CPSO[38]	GA [49]
Training Times	**1000**	1000	1000	1000
RMS error (training)	**8.78006**	10.0453	9.33866	14.143368
RMS error (forecasting)	**14.7885**	20.7664	17.9984	19.726994
Average error	**10.214250**	11.963119	11.152235	13.269692

and was performed ten times. The final rms error of the forecast output is about 8.78006.

In this example, as in Example 1, the performance of the NFIS model with the SEELA method was compared with those of other methods. In PSO [31], CPSO [38], and GA [49], the parameters are the same as in Example 1. Nine rules are used to construct the fuzzy model. The learning proceeded for 1000 generations, and was performed ten times. Fig. 14 plots the learning curves of the NFIS model with SEELA, PSO [31], CPSO [38], and GA [49] methods. The proposed method yields better forecast results than the other methods. Table IV presents

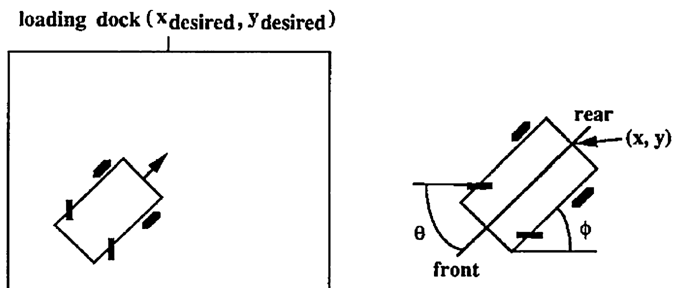


Fig. 15. Diagram of simulated truck and loading zone.

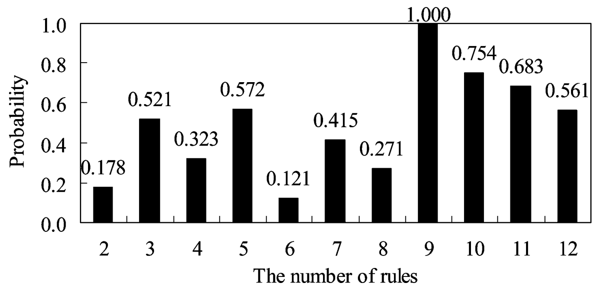


Fig. 16. Results of probability values of the ESS step in the proposed SEELA.

the rms error for training and forecasting and the average error. As presented in Table IV, the proposed NFIS model with SEELA method outperforms the other methods.

C. Example 3: Control of Backing Up the Truck

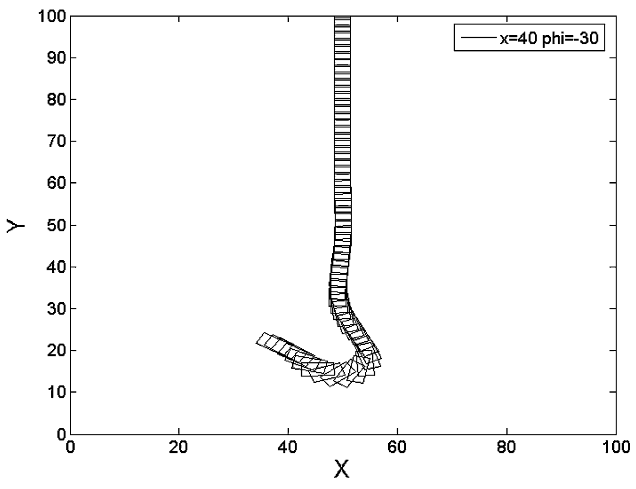
Backing a truck into a loading dock is difficult. It is a nonlinear control problem for which no traditional control method exists [48]. Fig. 15 shows the simulated truck and loading zone. The truck position is exactly determined by three state variables ϕ, x, and y, where ϕ is the angle between the 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 steered by the controller. The objective of this control problem is to use backward-only motions of the truck to make the truck arrive in 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 [0,100] × [0,100].

The input and output variables of the NFIS 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 as 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:

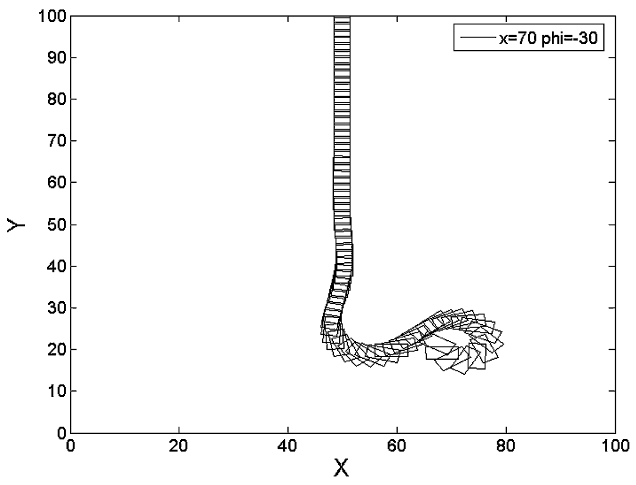
$$0 \le x \le 100 \tag{20}$$

$$-90^\circ \le \phi \le 270^\circ \tag{21}$$

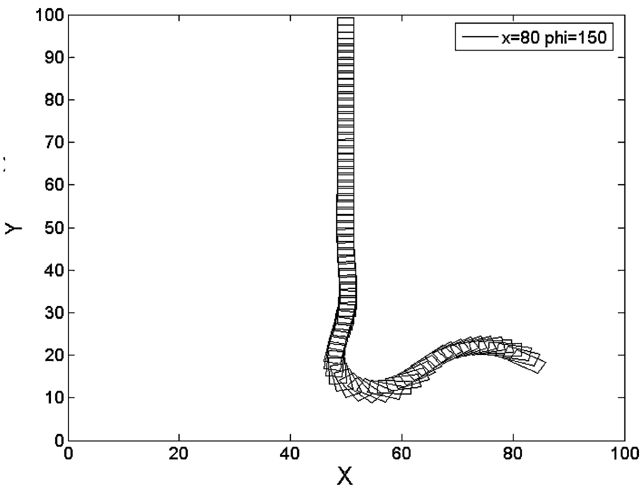
$$-30^\circ \le \theta \le 30^\circ. \tag{22}$$



(a)



(b)



(c)

Fig. 17. Trajectories of truck, starting at three initial positions under the control of the NFIS-SEELA after learning using training trajectories.

TABLE V
COMPARISONS OF BEST PERFORMANCE OF VARIOUS EXISTING MODELS IN EXAMPLE 3

	NFIS-SEELA	SEFC [27]	FALCON[4]	Neurofuzzy [50]	Neural Networks
Training Epochs	**500**	500	600	500	2000
Number of Rules /Hidden Nodes	**6**	9	19	35	30
RMS error	**0.0391**	0.0581	2.3	0.0924	0.0978

The equations of backward motion of the truck are

$$\begin{aligned} x(k+1) &= x(k) + d_f \cos\theta(k) + \cos\phi(k) \\ y(k+1) &= y(k) + d_f \cos\theta(k) + \sin\phi(k) \\ \phi(k+1) &= \tan^{-1}\left[\frac{l \sin\phi(k) + d_f \cos\phi(k) \sin\theta(k)}{l \cos\phi(k) - d_f \sin\phi(k) \sin\theta(k)}\right] \end{aligned} \quad (23)$$

where l is the length of the truck. Equation (23) 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 NFIS is thus trained. Fig. 16 shows the result of probability values after structure learning. As shown in Fig. 16, nine rules are adopted according to ESS. The learning proceeded for 500 generations, and was performed ten times. The final rms error of the forecast output is about 0.0391. Fig. 17(a)–(c) plots the trajectories of the moving truck controlled by the NFIS, starting at initial positions $(x, y, \phi) =$ (a) (40, 20, $-30°$), (b) (70, 20, $-30°$), and (c) (80, 20, $150°$), after the training process has been terminated. We compare the performance of our method with those of other existing methods [4], [27], [50]. The comparison results are tabulated in Table V. According to these results, the proposed NFIS-SEELA outperforms various existing models.

VI. CONCLUSION AND FUTURE WORKS

This study proposed an efficient SEELA for NFISs. The major novelty of the proposed SEELA is that it is based on evolutionary algorithms that can determine the number of fuzzy rules and adjust the NFIS parameters. The experimental results demonstrated that the proposed SEELA method can obtain a smaller rms error than the generally used PSO and CPSO.

Three advanced topics for the proposed SEELA method should be addressed in future research. First, the computational complexity of the SEELA method should be decreased in SSE. The crowding distance operator [51] presented in the nondominated sorting genetic algorithm II (NSGA-II) can be adopted for like fast nondominated sorting in the solution space according to each objective function. Second, the fitness function of the SEELA method uses only the error term to evaluate the performance of fuzzy systems, and does not address the number of fuzzy rules. Ishibuchi *et al.* [17]–[20] proposed multiobjective optimization methods to simultaneously evaluate the number of fuzzy rules and the performance of fuzzy systems. Third, there are nine parameters in the SEELA method that influence the accuracy and complexity of the final NFIS and training duration. These parameters should be automatically selected using an effective method in the future. Khosla *et al.* [52] presented a systematic method based on the Taguchi approach reasoning scheme for identifying the strategy parameters for the evolutionary algorithm. The Taguchi approach provides systematic, simple, and efficient methodology using fractional factorial design to study a large number of parameters with only a few well-defined experimental sets.

REFERENCES

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

[2] N. Kasabov, *Foundations of Neural Networks, Fuzzy Systems and Knowledge Engineering*. Cambridge, MA: MIT Press, 1996.

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

[4] 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. 1997.

[5] C. F. Juang and C. T. Lin, "An online self-constructing neural fuzzy inference network and its applications," *IEEE Trans. Fuzzy Syst.*, vol. 6, no. 1, pp. 12–31, Feb. 1998.

[6] 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.

[7] C. J. Lin and C. C. Chin, "Prediction and identification using wavelet-based recurrent fuzzy neural networks," *IEEE Trans. Syst., Man, Cybern.*, vol. 34, no. 5, pp. 2144–2154, Oct. 2004.

[8] O. Cordon, F. Herrera, F. Hoffmann, and L. Magdalena, *Genetic Fuzzy Systems*. Singapore: World Scientific, 2001.

[9] O. Cordon, F. Gomide, F. Herrera, F. Hoffmann, and L. Magdalena, "Ten years of genetic fuzzy systems: Current framework and new trends," *Fuzzy Sets Syst.*, vol. 141, no. 1, pp. 5–31, 2004.

[10] F. Herrera, "Genetic fuzzy systems: Status, critical considerations and future directions," *Int. J. Comput. Intell. Res.*, vol. 1, no. 1, pp. 59–67, 2005.

[11] S. H. Stewart, S. Taylor, J. M. Baker, F. Hoffmann, and G. Pfister, "Evolutionary design of a fuzzy knowledge base for a mobile robot," *Int. J. Approx. Reason.*, vol. 17, no. 4, pp. 447–469, Nov. 1997.

[12] A. Parodi and P. Bonelli, "A new approach to fuzzy classifier systems," in *Proc. 5th Int. Conf. Genetic Algorithms*, 1993, pp. 223–230.

[13] H. Ishibuchi, T. Nakashima, and T. Murata, "Performance evaluation of fuzzy classifier systems for multi-dimensional pattern classification problems," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 29, no. 5, pp. 601–618, Oct. 1999.

[14] O. Cordon, M. J. del Jesus, F. Herrera, and M. Lozano, "MOGUL: A methodology to obtain genetic fuzzy rule-based systems under the iterative rule learning approach," *Int. J. Intell. Syst.*, vol. 14, no. 11, pp. 1123–1153, 1999.

[15] A. Gonzalez and R. Perez, "SLAVE: A genetic learning system based on an iterative approach," *IEEE Trans. Fuzzy Syst.*, vol. 7, no. 2, pp. 176–191, Apr. 1999.

[16] L. Castillo, A. Gonzalez, and R. Perez, "Including a simplicity criterion in the selection of the best rule in a genetic fuzzy learning algorithm," *Fuzzy Sets Syst.*, vol. 120, no. 2, pp. 309–321, 2001.

[17] 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.

[18] H. Ishibuchi, T. Murata, and I. B. Turksen, "Single-objective and two-objective genetic algorithms for selecting linguistic rules for pattern classification problems," *Fuzzy Sets Syst.*, vol. 89, no. 2, pp. 135–150, 1997.

[19] H. Ishibuchi, T. Nakashima, and T. Murata, "Three-objective genetics-based machine learning for linguistic rule extraction," *Inf. Sci.*, vol. 136, no. 1–4, pp. 109–133, 2001.

[20] H. Ishibuchi and Y. Nojima, "Analysis of interpretability-accuracy tradeoff of fuzzy systems by multiobjective fuzzy genetics-based machine learning," *Int. J. Approx. Reason.*, vol. 44, no. 1, pp. 4–31, 2007.

[21] S. Mitra and S. K. Pal, "Fuzzy multi-layer perceptron, inferencing and rule generation," *IEEE Trans. Neural Netw.*, vol. 6, no. 1, pp. 51–63, Jan. 1995.

[22] S. Mitra and S. K. Pal, "Fuzzy self-organization, inferencing, and rule generation," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 26, no. 5, pp. 608–620, Sep. 1996.

[23] S. Mitra and Y. Hayashi, "Neurofuzzy rule generation: Survey in soft computing framework," *IEEE Trans. Neural Netw.*, vol. 11, no. 3, pp. 748–768, May 2000.

[24] S. Mitra, K. M. Konwar, and S. K. Pal, "Fuzzy decision tree, linguistic rules and fuzzy knowledge-based network: Generation and evaluation," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 32, no. 4, pp. 328–339, Nov. 2002.

[25] S. K. Pal, S. Mitra, and P. Mitra, "Rough-fuzzy MLP: Modular evolution, rule generation, and evaluation," *IEEE Trans. Knowl. Data Eng.*, vol. 15, no. 1, pp. 14–25, Jan./Feb. 2003.

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

[27] 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, Cybern.*, vol. 30, no. 2, pp. 290–302, Apr. 2000.

[28] C. J. Lin and Y. J. Xu, "A self-adaptive neural fuzzy network with group-based symbiotic evolution and its prediction applications," *Fuzzy Sets Syst.*, vol. 157, no. 8, pp. 1036–1056, Apr. 2006.

[29] G. R. Harik, F. G. Lobo, and D. E. Goldberg, "The compact genetic algorithm," *IEEE Trans. Evol. Comput.*, vol. 3, no. 4, pp. 287–297, Nov. 1999.

[30] C. J. Lin and Y. J. Xu, "A hybrid evolutionary learning algorithm for TSK-type fuzzy model design," *Math. Comput. Model.*, vol. 43, no. 5/6, pp. 563–581, Mar. 2006.

[31] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. IEEE Int. Conf. Neural Netw.*, Perth, W.A., Australia, vol. 4, Nov./Dec.1995, pp. 1942–1948.

[32] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proc. 6th Int. Symp. Micro Mach. Hum. Sci.*, Nagoya, Japan, Oct. 4–6, 1995, pp. 39–43.

[33] Z. L. Gaing, "A particle swarm optimization approach for optimum design of PID controller in AVR system," *IEEE Trans. Energy Convers.*, vol. 19, no. 2, pp. 384–391, Jun. 2004.

[34] H. Yoshida, K. Kawata, Y. Fukuyama, S. Takayama, and Y. Nakanishi, "A particle swarm optimization for reactive power and voltage control considering voltage security assessment," *IEEE Trans. Power Syst.*, vol. 15, no. 4, pp. 1232–1239, Nov. 2000.

[35] M. A. Abido, "Optimal design of power-system stabilizers using particle swarm optimization," *IEEE Trans. Energy Convers.*, vol. 17, no. 3, pp. 406–413, Sep. 2002.

[36] C. F. Juang, "A hybrid of genetic algorithm and particle swarm optimization for recurrent network design," *IEEE Trans. Syst., Man Cybern.*, vol. 34, no. 2, pp. 997–1006, Apr. 2004.

[37] R. Mendes, P. Cortez, M. Rocha, and J. Neves, "Particle swarms for feedforward neural network training," in *Proc. 2002 Int. Joint Conf. Neural Netw.*, Honolulu, HI, vol. 2, pp. 1895–1899.

[38] F. Van Den Bergh and A. P. Engelbrecht, "A cooperative approach to particle swarm optimization," *IEEE Trans. Evol. Comput.*, vol. 8, no. 3, pp. 225–239, Jun. 2004.

[39] C. H. Chen, C. T. Lin, and C. J. Lin, "A functional-link-based fuzzy neural network for temperature control," in *Proc. 2007 IEEE Symp. Found. Comput. Intell.*, Honolulu, HI, Apr. 1–5, pp. 53–58.

[40] J. C. Patra, R. N. Pal, B. N. Chatterji, and G. Panda, "Identification of nonlinear dynamic systems using functional link artificial neural networks," *IEEE Trans. Syst., Man, Cybern.*, vol. 29, no. 2, pp. 254–262, Apr. 1999.

[41] R. G. Reynolds, "An introduction to cultural algorithms," in *Proc. 3rd Annu. Conf. Evol. Program.*, A. V. Sebald and L. J. Fogel, Eds. River Edge, NJ: World Scientific, 1994, pp. 131–139.

[42] X. Jin and R. G. Reynolds, "Using knowledge-based evolutionary computation to solve nonlinear constraint optimization problems: A cultural algorithm approach," in *Proc. IEEE Congr. Evol. Comput.*, Washington, DC, 1999, vol. 3, pp. 1672–1678.

[43] A. Silva, A. Neves, and E. Costa, "An empirical comparison of particle swarm and predator prey optimization," *Lecture Notes Comput. Sci.*, vol. 2464, pp. 103–110, 2002.

[44] M. Mansour, S. F. Mekhamer, and N. E.-S. El-Kharbawe, "A modified particle swarm optimizer for the coordination of directional overcurrent relays," *IEEE Trans. Power Del.*, vol. 22, no. 3, pp. 1400–1410, Jul. 2007.

[45] S. M. Saleem, "Knowledge-based solution to dynamic optimization problems using cultural algorithms," Ph.D. dissertation, Wayne State Univ., Detroit, MI, 2001.

[46] R. S. Cowder, "Predicting the Mackey–Glass time series with cascade-correlation learning," in *Proc. 1990 Connectionist Models Summer School*, pp. 117–123.

[47] S. H. Ling, F. H. F. Leung, H. K. Lam, Y. S. Lee, and P. K. S. Tam, "A novel genetic-algorithm-based neural network for short-term load forecasting," *IEEE Trans. Ind. Electron.*, vol. 50, no. 4, pp. 793–799, Aug. 2003.

[48] D. Nguyen and B. Widrow, "The truck backer-upper: An example of self-learning in neural network," *IEEE Control Syst. Mag.*, vol. 10, no. 3, pp. 18–23, Apr. 1990.

[49] C. J. Lin, "A GA-based neural fuzzy system for temperature control," *Fuzzy Sets Syst.*, vol. 143, no. 2, pp. 311–333, Apr. 2004.

[50] H. Nomura, I. Hayashi, and N. Wakami, "A learning method of fuzzy inference rules by descent method," in *Proc. IEEE Conf. Fuzzy Syst.*, San Diego, CA, Mar. 1992, pp. 203–210.

[51] K. Deb, A. Pratap, S. Agrawal, and T. Meyarian, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, Apr. 2002.

[52] A. Khosla, S. Kumar, and K. K. Aggarwal, "Identification of strategy parameters for particle swarm optimizer through Taguchi method," *J. Zhejiang Univ. Sci.*, vol. 7, no. 12, pp. 1989–1994, 2006.

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.

From April 1996 to July 1999, he was an Associate Professor in the Department of Electronic Engineering, Nan-Kai College, Nantou, Taiwan. From August 1999 to January 2005, he was an Associate Professor in the Department of Computer Science and Information Engineering, Chaoyang University of Technology, where he was a Full Professor from February 2005 to July 2007 and the Chairman from 2001 to 2005, and the Library Director of the Poding Memorial Library from 2005 to 2007. From August 2007 to July 2008, he was a Professor in the Department of Electrical Engineering, National University of Kaohsiung. He is currently a Full Professor in the Department of Computer Science and Information Engineering, National Chin-Yi University of Technology, Taiping City, Taiwan. From 2002 to 2005, he was the Associate Editor of the *International Journal of Applied Science and Engineering*. His current research interests include soft computing, pattern recognition, intelligent control, image processing, bioinformatics, and field-programmable gate array (FPGA) design. He has authored or coauthored more than 150 papers published in referred journals and conference proceedings.

Prof. Lin is a member of the Phi Tau Phi, 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 a member of the Executive Committees of the TAAI from 2003 to 2008 and the CFSA from 2007 to 2008. He has received several honors and awards, including the 2006 Outstanding Paper Award of the 11th Conference on Artificial Intelligence and Applications, the 2007 Outstanding Paper Award of 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).

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 Chaoyang University of Technology, Wufong, 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.

He is currently with the Department of Electrical and Control Engineering, National Chiao-Tung University. His current research interests include fuzzy systems, neural networks, evolutionary algorithms, intelligent control, and pattern recognition.

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 the Founding Dean of the Computer Science College from 2005 to 2007, and is currently the Provost of Academic Affairs and the Chair Professor of Electrical and Control Engineering. He is also with the Brain Research Center, University System of Taiwan, Hsinchu. He is the author of *Neural Fuzzy Systems* (Prentice-Hall) and *Neural Fuzzy Control Systems With Structure and Parameter Learning* (World Scientific). He has authored or coauthored over 110 journal papers published, including about 80 IEEE Transaction papers. His current research interests include intelligent technology, soft computing, brain–computer interfaces, intelligent transportation systems, robotics and intelligent sensing, and nano-bio-information technologies and cognitive science (NBIC).

Dr. Lin was a member of the Board of Governors (BoG) of the IEEE Systems, Man, Cybernetics Society (SMCS) from 2003 to 2005. He is currently a BoG member of the IEEE Circuits and Systems Society (CASS). From 2003 to 2005, he was the IEEE Distinguished Lecturer. He is also the Deputy-Editor-in-Chief (EIC) of the IEEE TRANSACTIONS OF CIRCUITS AND SYSTEMS, PART II. He was the Program Chair of the 2006 IEEE International Conference on Systems, Man, and Cybernetics, Taipei, Taiwan. From 2004 to 2005, he was the President of the BoG of the Asia Pacific Neural Networks Assembly (APNNA). He has been receiving the Outstanding Research Award by the National Science Council (NSC), Taiwan, since 1997 to present, and received the Outstanding Professor Award by the Chinese Institute of Engineering (CIE) in 2000 and the 2002 Taiwan Outstanding Information-Technology Expert Award. He was also elected to be one of 38th Ten Outstanding Rising Stars in Taiwan (2000). He is a member of the Tau Beta Pi, the Eta Kappa Nu, and the Phi Kappa Phi honorary societies.