A Hybrid of Cooperative Particle Swarm Optimization and Cultural Algorithm for Neural Fuzzy Networks

Conference Paper *in* IEEE International Conference on Fuzzy Systems · July 2008

DOI: 10.1109/FUZZY.2008.4630371 · Source: IEEE Xplore

4 authors, including:

Cheng-Hung Chen
National Formosa University
67 PUBLICATIONS **1,020** CITATIONS

SEE PROFILE

Cheng-Jian Lin
National Chin-Yi University of Technology
276 PUBLICATIONS **5,702** CITATIONS

SEE PROFILE

Chin-Teng Lin
National Chiao Tung University
698 PUBLICATIONS **19,231** CITATIONS

SEE PROFILE

A Hybrid of Cooperative Particle Swarm Optimization and Cultural Algorithm for Neural Fuzzy Networks and Its Prediction Applications

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

***Abstract*—This study presents an evolutionary neural fuzzy network, designed using the functional-link-based neural fuzzy network (FLNFN) and a new evolutionary learning algorithm. This new evolutionary learning algorithm is based on a hybrid of cooperative particle swarm optimization and cultural algorithm. It is thus called cultural cooperative particle swarm optimization (CCPSO). The proposed CCPSO method, which uses cooperative behavior among multiple swarms, can increase the global search capacity using the belief space. Cooperative behavior involves a collection of multiple swarms that interact by exchanging information to solve a problem. The belief space is the information repository in which the individuals can store their experiences such that other individuals can learn from them indirectly. The proposed FLNFN model uses functional link neural networks as the consequent part of the fuzzy rules. This study uses orthogonal polynomials and linearly independent functions in a functional expansion of the functional link neural networks. The FLNFN model can generate the consequent part of a nonlinear combination of input variables. Finally, the proposed FLNFN with CCPSO (FLNFN-CCPSO) is adopted in several predictive applications. Experimental results have demonstrated that the proposed CCPSO method performs well in predicting the time series problems.**

***Index Terms*—Chaotic time series, cultural algorithm, functional-link network, neural fuzzy network, particle swarm optimization, prediction.**

I. Introduction

PREDICTION has been widely studied for many years as time series analysis [1], [2]. Traditionally, prediction is based on a statistical model that is either linear or nonlinear [3]. Recently, several studies have adopted neural fuzzy networks to predict time series [4]–[6]. Researchers have discussed that the network paradigm is a very useful model for predicting time series and especially for predicting nonlinear time series.

Neural fuzzy networks [7]–[13] have become a popular research topic. They bring the low-level learning and computational power of neural networks into fuzzy systems and bring the high-level human-like thinking and reasoning of fuzzy systems to neural networks. In the typical TSK-type neural fuzzy network [8]–[13], which is a linear polynomial of input variables, the model output is approximated locally by the rule hyperplanes. However, the traditional TSK-type neural fuzzy network does not take full advantage of the mapping capabilities that may be offered by the consequent part. Introducing a nonlinear function, especially a neural structure, to the consequent part of the fuzzy rules has yielded the NARA [14] and the CANFIS [15] models. These models [14], [15] use multilayer neural networks in the consequent part of the fuzzy rules. Although the interpretability of the model is reduced, the representational capability of the model is significantly improved. However, the multilayer neural network has such disadvantages as slower convergence and greater computational complexity. Therefore, we proposed the functional link neural fuzzy network (FLNFN), which uses the functional link neural network (FLNN) [16], [17] in the consequent part of the fuzzy rules [18]. The FLNN is a single-layer neural structure that is capable of forming arbitrarily complex decision regions by generating nonlinear decision boundaries. Additionally, using functional expansion effectively increases the dimensionality of the input vector and the hyperplanes that are generated by the FLNN provide a good discrimination capability in input data space.

Training of the parameters is the main problem in designing a neural fuzzy system. Backpropagation (BP) training is commonly adopted to solve this problem. It is a powerful training technique that can be applied to networks with a forward structure. Since the steepest descent approach is used in BP training to minimize the error function, the algorithms may reach the local minima very quickly and never find the global solution.

The aforementioned disadvantages lead to suboptimal performance, even for a favorable neural fuzzy network topology. Therefore, technologies that can be used to train the system parameters and find the global solution while optimizing the overall structure, are required. Accordingly, a new optimization algorithm, called particle swarm optimization (PSO), appears to be better than the backpropagation algorithm. It is an evolutionary computation technique that was developed by Kennedy and Eberhart in 1995 [19], [20]. The underlying motivation for the development of PSO algorithm is the social behavior of animals, such as bird flocking, fish schooling and swarm theory. PSO has been successfully applied to many optimization problems, such as control problems [21]–[23] and feedforward neural network design [24]–[28]. However, PSO suffers from the burden of many dimensions, such that its performance falls

Manuscript received May 25, 2007; revised September 22, 2007. Current version published December 22, 2008. This work was supported in part by the National Science Council, Taiwan, R.O.C., 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. This paper was recommended by Associate Editor X. Guan.

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.-H. Chen and C.-T. Lin are with the Department of Electrical and Control Engineering, National Chiao-Tung University, Hsinchu 300, Taiwan, R.O.C.

Digital Object Identifier 10.1109/TSMCC.2008.2002333

as the dimensionality of the search space increases. Therefore, Bergh *et al.* [29] proposed a cooperative approach that employs cooperative behavior, called CPSO, which uses multiple swarms to improve upon traditional PSO. However, the CPSO still uses the formula (the local best position of each particle and global best position in the swarm) of the traditional PSO to evolve. The trajectory of each particle in the search space is adjusted according to the local best position of the particle and the global best position in the same search space, but it is unable to yield high diversity of particles to increase search space. That is, it is lacking enough capability to satisfy the requirements of exploration [30], [31]. Therefore, the CPSO may find a suboptimal solution. Additionally, the cultural algorithm [32], [33] can exploit the information of specific belief space to guide the feasible search space and it can also change the direction of each individual in solution space. Hence, the proposed cultural cooperative particle swarm optimization (CCPSO) learning method, which combines the cooperative particle swarm optimization and cultural algorithm, to increase global search capacity, is proposed herein to avoid trapping in a suboptimal solution and to ensure that a nearby global optimal solution can be found.

This study presents an efficient CCPSO for the FLNFN in several predictive applications. The proposed FLNFN model is based on our previous research [18]. The FLNFN model, which combines a neural fuzzy network with a functional link neural network, is designed to improve the accuracy of functional approximation. The consequent part of the fuzzy rules that corresponds to an FLNN comprises the functional expansion of input variables. The orthogonal polynomials and linearly independent functions are adopted as functional link neural network bases. The proposed CCPSO is a hybrid method that combines cooperative particle swarm optimization and cultural algorithms. The 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 FLNFN-CCPSO method are as follows: 1) the consequent of the fuzzy rules involves a nonlinear combination of input variables. This study uses a functional link neural network to the consequent part of the fuzzy rules. The functional expansion in the FLNFN model can yield the consequent part of a nonlinear combination of input variables; 2) the proposed CCPSO with cooperative behavior among multiple swarms can accelerate the search and increase global search capacity using the belief space; and 3) as demonstrated in Section V, the FLNFN-CCPSO method is a more effective controller than the other methods.

The rest of this paper is organized as follows. Section II describes the basic concept of particle swarm optimization and cultural algorithm. Section III presents the structure of the functional-link-based neural fuzzy network. Next, Section IV presents the cultural cooperative particle swarm optimization method. Section V presents the results of the simulation of several predictive applications. Section VI draws conclusions.

II. Particle Swarm Optimization and Cultural Algorithm

This section describes basic concepts concerning particle swarm optimization and the cultural algorithm. The specialization property of particle swarm optimization and cultural algorithm is consistent with the learning property of the neural fuzzy network. Therefore, the development of a neural fuzzy network based on particle swarm optimization and the cultural algorithm is valuable.

A. Particle Swarm Optimization

In 1995, Kennedy and Eberhart introduced the particle swarm optimization algorithm (PSO) [19], [20] in the field of social and cognitive behavior. The PSO is a population-based optimization approach, in which the population is called a swarm. Furthermore, each swarm consists of many particles. In the PSO, the trajectory of each particle in the search space is adjusted by dynamically altering the velocity of each particle. Each particle has a velocity vector $\vec{v}_i$ and a position vector $\vec{x}_i$, which represents a possible solution. Then, the particles move rapidly around and search the solution space using the moving velocity of each particle. Each of these particle positions is scored to obtain a fitness value, based on how to define the solution of the problem. The local best position (Lbest) of each particle and the global best position (Gbest) in the swarm are used to yield a new velocity for each particle

$$\vec{v}_i(k+1) = \omega \times \vec{v}_i(k) + \phi_1 \times \text{rand}() \times (\text{Lbest} - \vec{x}_i(k)) + \phi_2 \times \text{rand}() \times (\text{Gbest} - \vec{x}_i(k)) \quad (1)$$

where ω, ϕ_1, and ϕ_2 are called the coefficient of the inertia term, the cognitive term, and the society term, respectively. The term $\vec{v}_i$ is limited to the range $\pm\vec{v}_{\max}$. If the velocity violates this limit, then it is set to the actual limit.

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

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

Fig. 1 presents the concept of the updated velocity using (1) and (2).

B. Cultural Algorithm

Cultural algorithms [32], [33] involve acquiring the belief space from the evolving population space and then exploiting that information to guide the search. Fig. 2 presents the cultural algorithm components. Cultural algorithms can be described in terms of two basic components—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 cultural algorithms, 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

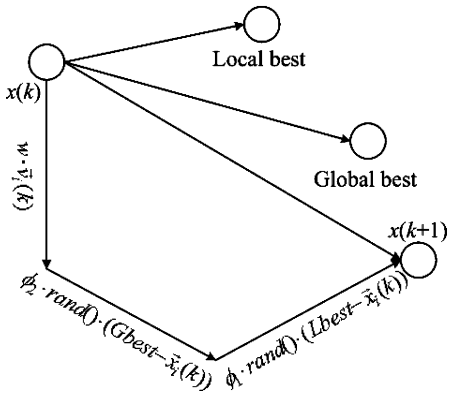

Fig. 1. Diagram of the updated velocity in the PSO.

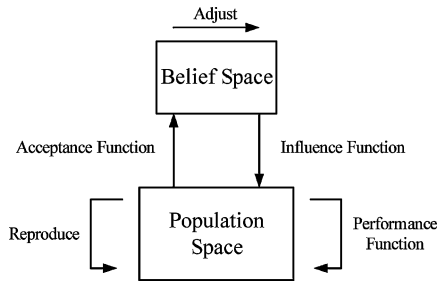

Fig. 2. Framework of cultural algorithm.

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 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).

III. Structure of Functional-Link-Based Neural Fuzzy Network

This section describes the structure of functional link neural networks and the structure of the FLNFN model. In functional link neural networks, the input data usually incorporate high-order effects, and thus, artificially increase the dimensions of the input space. Accordingly, the input representation is enhanced and linear separability is achieved in the extended space. The FLNFN model adopted the functional link neural network generating complex nonlinear combination of input variables as the consequent part of the fuzzy rules. The rest of this section details these structures.

A. Functional Link Neural Networks

The functional link neural network is a single-layer network in which the need for hidden layers is eliminated. While the input variables generated by the linear links of neural networks are linearly weighted, the functional link acts on an element of input variables by generating a set of linearly independent functions, which are suitable orthogonal polynomials for a

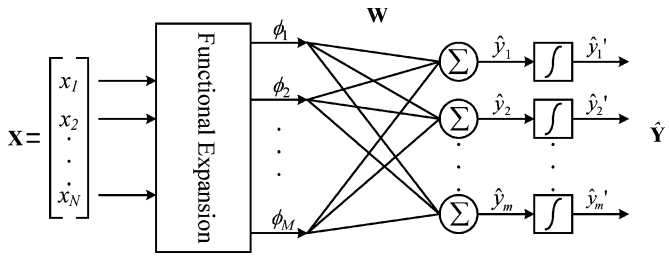

Fig. 3. Structure of the FLNN.

functional expansion, and then evaluating these functions with the variables as the arguments. Therefore, the FLNN structure considers trigonometric functions. For example, for a 2-D input $\mathbf{X} = [x_1, x_2]^T$, the enhanced data are obtained by using trigonometric functions as functional expansion $\mathbf{\Phi} = [1, x_1, \sin(\pi x_1), \cos(\pi x_1), \ldots, x_2, \sin(\pi x_2), \cos(\pi x_2), \ldots]^T$. Thus, the input variables can be separated in the enhanced space [16]. In the FLNN structure with reference to Fig. 3, a set of basis functions $\mathbf{\Phi}$ and a fixed number of weight parameters $\mathbf{W}$ represent $f_{\mathbf{W}}(x)$. The theory behind the FLNN for multidimensional function approximation has been discussed elsewhere [17] and is analyzed next.

Consider a set of basis functions $\mathbf{B} = \{\phi_k \in \mathbf{\Phi}(A)\}_{k \in \mathbf{K}}$, $\mathbf{K} = \{1, 2, \ldots\}$ with the following properties; 1) $\phi_1 = 1$; 2) the subset $\mathrm{B}_j = \{\phi_k \in \mathbf{B}\}_{k=1}^{M}$ is a linearly independent set, meaning that if $\sum_{k=1}^{M} w_k \phi_k = 0$, then $w_k = 0$ for all $k = 1, 2, \ldots, M$; and 3) $\sup_j \left[\sum_{k=1}^{j} \|\phi_k\|_A^2\right]^{1/2} < \infty$.

Let $\mathbf{B} = \{\phi_k\}_{k=1}^{M}$ be a set of basis functions to be considered, as shown in Fig. 3. The FLNN comprises M basis functions $\{\phi_1, \phi_2, \ldots, \phi_M\} \in \mathbf{B}$. The linear sum of the jth node is given by

$$\hat{y}_j = \sum_{k=1}^{M} w_{kj} \phi_k(\mathbf{X}) \tag{3}$$

where $\mathbf{X} \in \mathrm{A} \subset \Re^N, \mathbf{X} = [x_1, x_2, \ldots, x_N]^T$ is the input vector, and $\mathbf{W}_j = [w_{1j}, w_{2j}, \ldots, w_{Mj}]$ is the weight vector associated with the jth output of the FLNN. $\hat{\mathbf{y}}_j$ denotes the local output of the FLNN structure and the consequent part of the jth fuzzy rule in the FLNFN model. Thus, (3) can be expressed in matrix form as $\hat{\mathbf{y}}_j = \mathbf{W}_j \mathbf{\Phi}$, where $\mathbf{\Phi} = [\phi_1(x), \phi_2(x), \ldots, \phi_M(x)]^T$ is the basis function vector, which is the output of the functional expansion block. The m-dimensional linear output may be given by $\hat{\mathbf{y}} = \mathbf{W}\mathbf{\Phi}$, where $\hat{\mathbf{y}} = [\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_m]^T$, m denotes the number of functional link bases, which equals the number of fuzzy rules in the FLNFN model, and $\mathbf{W}$ is a $(m \times M)$-dimensional weight matrix of the FLNN given by $\mathbf{W} = [\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_m]^T$. The jth output of the FLNN is given by $\hat{y}_j' = \rho(\hat{y}_j)$, where the nonlinear function $\rho(\cdot) = \tanh(\cdot)$. Thus, the m-dimensional output vector is given by

$$\hat{\mathbf{Y}} = \rho(\hat{y}) = f_{\mathbf{W}}(x) \tag{4}$$

where $\hat{\mathbf{Y}}$ denotes the output of the FLNN.

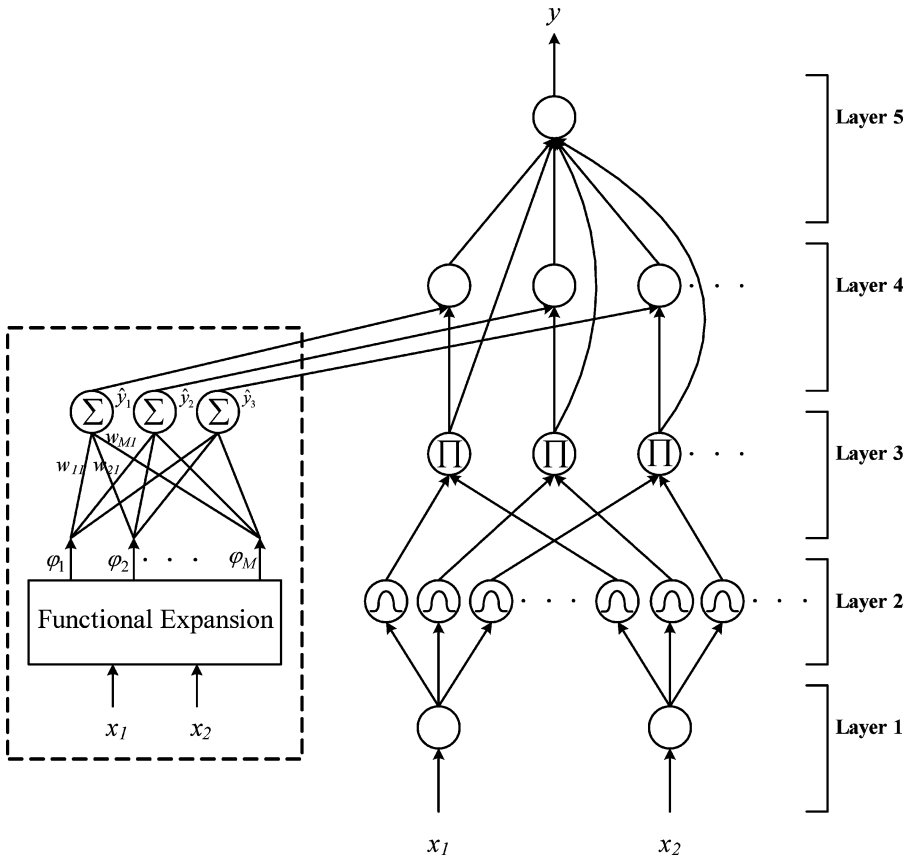

Fig. 4. Structure of the proposed FLNFN model.

B. Structure of the FLNFN Model

This subsection describes the FLNFN model, which uses a nonlinear combination of input variables (FLNN). Each fuzzy rule corresponds to a sub-FLNN, comprising a functional link. Fig. 4 presents the structure of the proposed FLNFN model. The FLNFN model realizes a fuzzy if-then rule in the following form.

Rule_j: IF x_1 is $A_{1j}$and x_2 is $A_{2j}\ldots$ and $x_i$is $A_{ij}\ldots$ and $x_N]$is A_{Nj}

$$\text{THEN } \hat{y}_j = \sum_{k=1}^{M} w_{kj}\phi_k = w_{1j}\phi_1 + w_{2j}\phi_2 + \ldots + w_{Mj}\phi_M \tag{5}$$

where 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 Gaussian membership function; N is the number of input variables; w_{kj} is the link weight of the local output; ϕ_k is the basis trigonometric function of input variables; M is the number of basis function, and Rule_j is the jth fuzzy rule.

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

No computation is performed in layer 1. Each node in this layer only transmits input values to the next layer directly

$$u_i^{(1)} = x_i. \tag{6}$$

Each fuzzy set A_{ij} is described here by a Gaussian membership function. Therefore, the calculated membership value in layer 2 is

$$u_{ij}^{(2)} = \exp\left(-\frac{[u_i^{(1)} - m_{ij}]^2}{\sigma_{ij}^2}\right) \tag{7}$$

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.

Nodes in layer 3 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 precondition part of the fuzzy rules. As a result, the output function of each inference node is

$$u_j^{(3)} = \prod_i u_{ij}^{(2)} \tag{8}$$

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

Nodes in layer 4 are called consequent nodes. The input to a node in layer 4 is the output from layer 3, and the other inputs are calculated from a functional link neural network that has not used the function $\tanh(\cdot)$, as shown in Fig. 4. For such a node

$$u_j^{(4)} = u_j^{(3)} \times \sum_{k=1}^{M} w_{kj}\phi_k \tag{9}$$

where w_{kj} is the corresponding link weight of functional link neural network and ϕ_k is the functional expansion of input variables. The functional expansion uses a trigonometric polynomial basis function, given by $[x_1 \sin(\pi x_1) \cos(\pi x_1) x_2 \sin(\pi x_2) \cos(\pi x_2)]$ 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 functional link neural network depend on the number of fuzzy rules of the FLNFN model.

The output node in layer 5 integrates all of the actions recommended by layers 3 and 4 and acts as a defuzzifier with

$$y = u^{(5)} = \frac{\sum_{j=1}^{R} u_j^{(4)}}{\sum_{j=1}^{R} u_j^{(3)}} = \frac{\sum_{j=1}^{R} u_j^{(3)} \left(\sum_{k=1}^{M} w_{kj} \phi_k \right)}{\sum_{j=1}^{R} u_j^{(3)}} = \frac{\sum_{j=1}^{R} u_j^{(3)} \hat{y}_j}{\sum_{j=1}^{R} u_j^{(3)}} \tag{10}$$

where R is the number of fuzzy rules, and y is the output of the FLNFN model.

IV. Learning Algorithms for the FLNFN Model

This section describes the proposed CCPSO method. Before the CCPSO method is designed, CPSO [29] that differs from the traditional PSO is introduced.

The traditional PSO uses one swarm of particles defined by the P-dimensional vectors to evolve. The CPSO method can change traditional PSO into P swarms of 1-D vectors, such that each swarm represents a dimension of the original problem. Fig. 5(a) and (b) shows the framework of the traditional PSO and CPSO method. The key point is that, instead of using one swarm (of I particles) to find the optimal P-dimensional vector, the vector is split into its components so that P swarms (of I particles each) optimize a 1-D vector. Notably, the function that is being optimized still requires a P-dimension vector to be evaluated. However, if each swarm represents only a single dimension of the search space, it cannot directly compute the fitness of the individuals of a single population considered in isolation. A context vector is required to provide a suitable context in which the individuals of a population can be evaluated. To calculate the fitness for all particles in swarm, the other P-1 components in the context vector keep constant values, while the pth component of the context vector is replaced, in turn, by each particle from the pth swarm. 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 local best position and the global best position of the traditional PSO to evolution process. Therefore, the CPSO may fall into a suboptimal solution. The CCPSO learning method, which combines the cooperative particle swarm optimization and the cultural algorithm to increase the global search capacity, is proposed to avoid trapping in a suboptimal solution and to ensure the ability to search for a near-global optimal solution.

The CCPSO method is characteristic of the cooperative particle swarm optimization and cultural algorithm. Fig. 6 shows the framework of the proposed CCPSO learning method, which

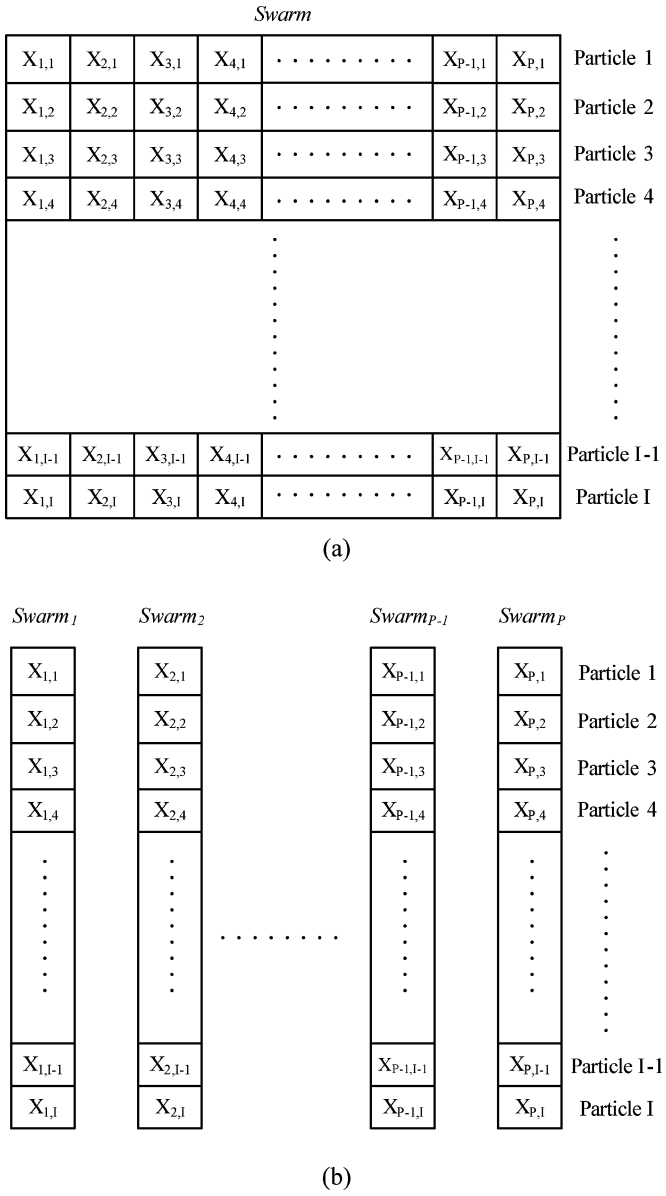

Fig. 5. Framework of (a) PSO and (b) CPSO.

is based on a CPSO all of whose parameters are simultaneously tuned using the brief space of the culture algorithm (CA). The CCPSO method can strengthen the global search capability. If 50-dimensional vectors are used in the original PSO, then the vectors in CCPSO can be changed into 50 swarms of 1-D vectors. In the original PSO, the particle can exhibit 50 variations in each generation, whereas the CCPSO offers $50 \times 50 = 2500$ different combinations in each generation. Additionally, each position of the CCPSO can be adjusted not only using the belief space that stores the paragons of each swarm, but also by searching around the local best solution and the global best solution. In the aforementioned scheme, the proposed CCPSO method can avoid falling into a suboptimal solution and ensure that the approximate global optimal solution can be found.

The detailed flowchart of the proposed CCPSO method is presented in Fig. 7. The foremost step in CCPSO is the coding of

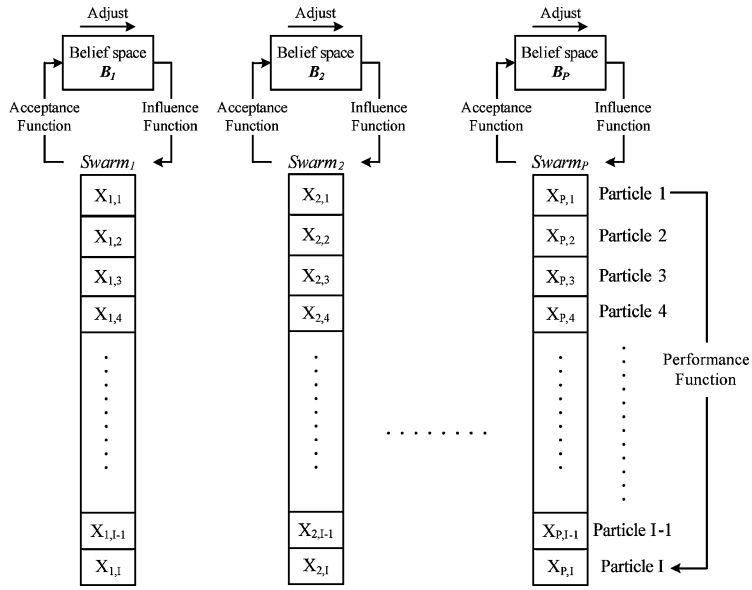

Fig. 6. Framework of the proposed CCPSO learning method.

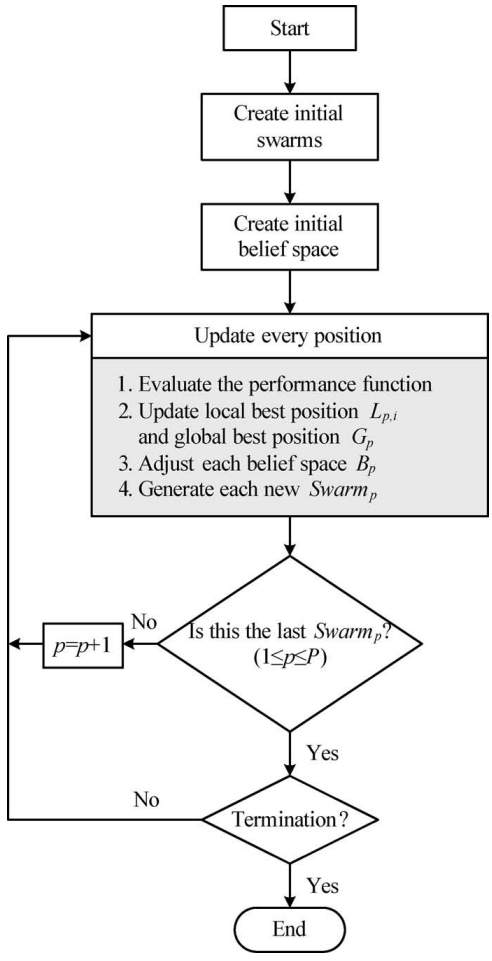

Fig. 7. Flowchart of the proposed CCPSO learning method.

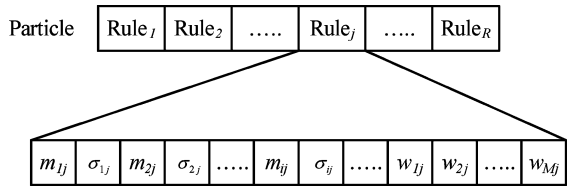

Fig. 8. Coding FLNFN model into a particle in the proposed CCPSO.

the neural fuzzy network into a particle. Fig. 8 shows an example of the coding of parameters of neural fuzzy network into a particle, 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. 8 represents the neural fuzzy network given by (5), 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 particle.

The learning process is described step-by-step as follows.

Step 1: Create initial swarms

Before the CCPSO method is applied, every position $x_{p,i}(t)$ must be created randomly in the range [0, 1], where $p = 1, 2, \ldots, P$ represents the pth swarm, $i = 1, 2, \ldots, I$ represents the ith particle, and t denotes the tth generation.

Step 2: Create initial belief space

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 space, B_p ($p = 1, 2, \ldots, P$). Each initial B_p is defined as an empty set.

Step 3: Update every position

Step 3.1: Evaluate the performance function of each $Particle_i$

The fitness function is used to evaluate the performance function of each particle. The fitness function is defined as follows

$$F = \sqrt{\frac{1}{D}\sum_{d=1}^{D}(y_d - \overline{y}_d)^2} \tag{11}$$

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

Step 3.2: Update local best position $L_{p,i}$ and global best position G_p

The local best 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 global best position G_p is generated by the whole local best position. In *Step 3.2*, the first step updates the local best position. Compare the fitness value of each current particle with that of its local best position. If the fitness value of the current particle exceeds those of its local best position, then the local best position is replaced with the position of the current particle. The second step updates the global best position. Compare the fitness value of all particles in their local best positions with that of the particle in the global best position. If the fitness value of the particle in the local best position is better than those of the particles in the global best position, then the global best position is replaced with the current local best position

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

Step 3.3: Adjust each belief space B_p using an acceptance function

The first part of *Step 3.3* 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 as follows. The number of accepted particles decreases as the number of generations increases

$$N_{\text{accepted}} = n\% \times I + \frac{n\%}{t} \times I \quad (13)$$

where $n\%$ is a parameter that is set by the user, and must specify the top performing 20% [34]; I is the number of particles, and t represents the tth generation. The second step adjusts B_p. The interval of belief space BI_p is defined as$BI_p = [l_p, u_p] = \{x | l_p \leq x \leq u_p, x \in \Re\}$, where l_p is the lower bound on belief space B_p and u_p is the upper bound on 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:

$$l_p = \begin{cases} x_{p,i}, & \text{if } x_{p,i} \leq l_p \\ l_p, & \text{otherwise} \end{cases}$$
$$u_p = \begin{cases} x_{p,i}, & \text{if } x_{p,i} \geq u_p \\ u_p, & \text{otherwise.} \end{cases} \quad (14)$$

Step 3.4: Generate each new *Swarm*$_p$ using $l_p, u_p, L_{p,i}$, and G_p

In *Step 3.4*, 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 solution space, not easily being trapped at a local optimum. Then, the second step updates the velocity and position of each particle to generate the each new *Swarm*$_p$ using (16) and (17)

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

$$v_{p,i}(t+1) = w \times v_{p,i}(t) + c_1 \times \text{Rand}() \times [L_{p,i}(t+1) - x_{p,i}(t)] + c_2 \times \text{Rand}() \times [G_p(t+1) - x_{p,i}(t)] \quad (16)$$

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

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

V. Experimental Results

This section discusses three examples that were considered to evaluate the FLNFN model with the CCPSO learning method. The first example involves predicting a chaotic signal that has been described in [7]; the second example involves predicting a chaotic time series [4], and the third example involves forecasting the number of sunspots [5].

A. Example 1: Prediction of a Chaotic Signal

In this example, an FLNFN model with a CCPSO learning method (FLNFN-CCPSO) was used to predict a chaotic signal. The classical time series prediction problem is a one-step-ahead prediction that has been described in [7]. The following equation describes the logistic function:

$$x(k+1) = ax(k)(1 - x(k)). \quad (18)$$

The behavior of the time series generated by this equation depends critically on parameter a. If $a < 1$, then the system has a single fixed point at the origin, and from a random initial value between [0, 1], the time series collapses to a constant value. For $a > 3$, the system generates a periodic attractor. At $a \geq 3.6$, the system becomes chaotic. In this example, a was set to 3.8. The first 60 pairs (from $x(1)$ to $x(60)$), with initial value$x(1) = 0.001$, were the training dataset, while the remaining 100 pairs (from $x(1)$ to $x(100)$), with initial value $x(1) = 0.9$, were the testing dataset used to validate the proposed method.

In this example, several particles will be found to minimize the fitness value using the proposed FLNFN-CCPSO method. The learning stage involved parameter learning using the CCPSO method. The coefficient ω was set to 0.4. The cognitive coefficient c_1 1.6, and the society coefficient c_2 was set to 2. The swarm sizes were set to 50. The learning proceeded for 1000 generations, and was repeated 50 times. After 1000 generations, the final average rms error of the predicted output is about 0.002285. In this example, three fuzzy rules are adopted. They

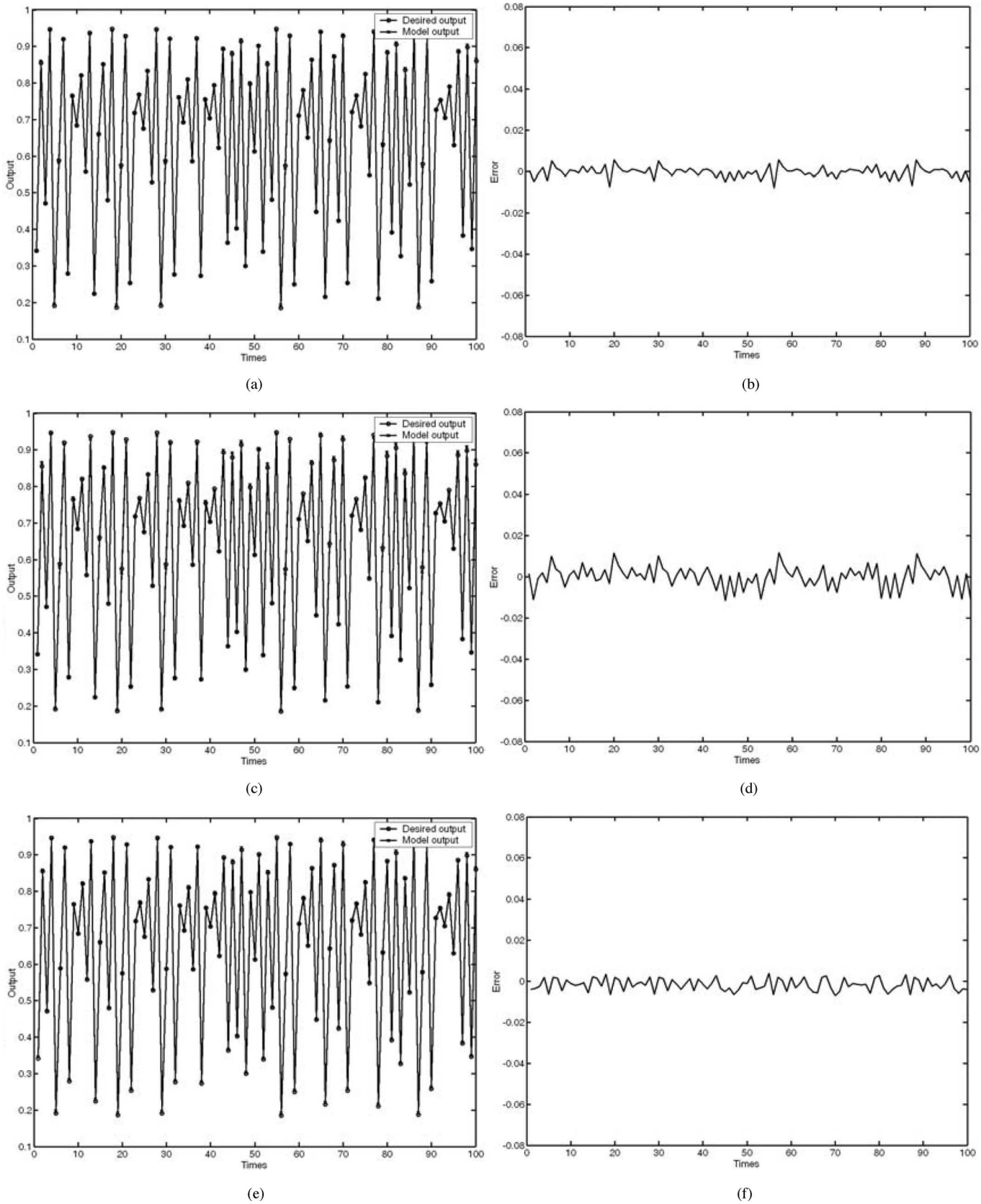

Fig. 9. (a) Predictions of the proposed method. (b) Prediction errors of the proposed method. (c) Predictions of the PSO [19]. (d) Prediction errors of the PSO. (e) Predictions of the CPSO [29]. (f) Prediction errors of the CPSO.

are shown as follows.

$$\text{Rule}_1\text{: IF } x \text{ is } \mu(0.763596,\ 19.0781)$$
$$\text{THEN } \hat{y}_1 = -0.846419 + 0.840237x + 0.0103279\cos(\pi x) + 1.61874\sin(\pi x) - 0.364635x$$
$$\text{Rule}_2\text{: IF } x \text{ is } \mu(0.235112, 0.307009)$$
$$\text{THEN } \hat{y}_1 = 0.145784 - 0.961044x - 0.146496\cos(\pi x) + 0.857966\sin(\pi x) + 6.62004x$$
$$\text{Rule}_3\text{: IF } x \text{ is } \mu(0.771367, 0.351594)$$
$$\text{THEN } \hat{y}_1 = 0.727383 + 0.625871x + 1.00717\cos(\pi x) + 2.25003\sin(\pi x) + 0.178136x$$

where $\mu(m_{ij}, \sigma_{ij})$ represents a Gaussian membership function with mean m_{ij} and deviation σ_{ij} in the ith input variable and the jth rule. Fig. 9(a) plots the predictions of the desired output and the model output in 1000 generations of learning. The solid line represents the desired output of the time series, and the notation "$*$" represents the output of the FLNFN-CCPSO method. Fig. 9(b) presents the prediction errors of the proposed method. The experimental results demonstrate the perfect predictive capability of theFLNFN-CCPSO method.

In this example, PSO [19] and CPSO [29] were applied to the same problem to show the effectiveness and efficiency of the FLNFN model with the CCPSO learning method. In the PSO and CPSO, the swarm sizes were set to 50. The coefficient ω 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. Three rules were applied to construct the fuzzy model. In the PSO [19] and CPSO [29], learning proceeded for 1000 generations, and was performed 50 times.

The performance of the FLNFN model with CCPSO learning was compared with the performance of other methods. First, the performance of the FLNFN-CCPSO method was compared with that of the PSO [19]. Fig. 9(c) plots the results predicted using PSO. Fig. 9(d) presents the prediction errors of the PSO. Second, CPSO [29] is adopted to solve the predictive problem. Fig. 9(e) and (f) plots the results and the errors of the CPSO. As presented in Fig. 9, the results predicted by the FLNFN model with the CCPSO learning method are better than those predicted by other methods.

Fig. 10 plots the learning curves of the best performance of the FLNFN model with the CCPSO learning method, PSO [19], and CPSO [29]. This figure indicates that the proposed method converges quickly and yields a lower rms error than the other methods. Computer simulations indicated that the proposed method outperforms other methods. The best performance of the CCPSO was compared with that of the PSO [19] and CPSO [29]. Table I compares the results. The comparison indicates that the rms error of training and predicting for the FLNFN-CCPSO method is better than those obtained using other methods.

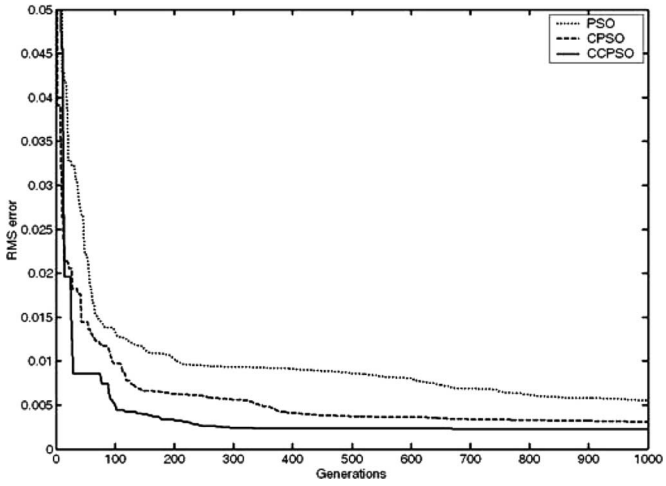

Fig. 10. Learning curves of the proposed method, PSO [19] and CPSO [29].

TABLE I
COMPARISON OF THE BEST PERFORMANCE OF THE CCPSO, PSO, AND CPSO IN EXAMPLE 1

	CCPSO	PSO[19]	CPSO[29]
RMS error (training)	**0.002285**	0.005239	0.003423
RMS error (predicting)	**0.002717**	0.005514	0.003958

B. Example 2: Prediction of Chaotic Time Series

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

$$\frac{dx(t)}{dt} = \frac{0.2x(t-\tau)}{1 + x^{10}(t-\tau)} - 0.1x(t). \tag{19}$$

Crowder [4] extracted 1000 input–output data pairs $\{x, y^d\}$ using four past values of $x(t)$

$$[x(t-18),\ x(t-12),\ x(t-6),\ x(t); x(t+6)] \tag{20}$$

where $\tau = 17$ and $x(0) = 1.2$. Four inputs to the FLNFN-CCPSO method, corresponded to these values of $x(t)$, and one output was $x(t + \Delta t)$, where Δt is a 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.

The learning stage entered parameter learning through the CCPSO method. The coefficient ω 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. The swarm sizes were set to 50. The learning proceeded for 1000 generations, and was performed 50 times. In this example, three fuzzy rules are applied. They are as follows

$$\text{Rule}_1\text{: IF } x_1 \text{ is } \mu(0.452959, -5.36833) \text{ and}$$
$$x_2 \text{ is } \mu(-0.10799, 0.768855) \text{ and}$$
$$x_3 \text{ is } \mu(-0.850613, -3.60999) \text{ and}$$
$$x_4 \text{ is } \mu(1.09886, 0.495632)$$
$$\text{THEN } \hat{y}_1 = 2.20613 + 0.580829x_1 + 0.391061\cos(\pi x_1) + 0.332886\sin(\pi x_1) - 4.68232x_2 - 5.05388\cos(\pi x_2) + 1.73753\sin(\pi x_2)$$

$$
\begin{aligned}
&- 0.656754x_3 + 1.71626\cos(\pi x_3)\\
&+ 0.0923789\sin(\pi x_3) + 4.93925x_4\\
&- 0.416084\cos(\pi x_4) + 1.45935\sin(\pi x_4)\\
&+ 0.990628x_1x_2x_3x_4
\end{aligned}
$$

Rule$_2$: IF x_1 is $\mu(-0.596747, -0.896165)$ and
x_2 is $\mu(0.841226,\ 1.1499)$ and
x_3 is $\mu(0.20028, 0.310169)$ and
x_4 is $\mu(1.01531,\ 0.524704)$

$$
\begin{aligned}
\text{THEN } \hat{y}_1 &= 0.683119 + 0.649552x_1\\
&+ 1.74121\cos(\pi x_1) - 4.32156\sin(\pi x_1)\\
&+ 0.200504x_2 - 2.74432\cos(\pi x_2)\\
&+ 1.18918\sin(\pi x_2) + 0.519391x_3\\
&+ 0.641173\cos(\pi x_3) + 3.17329\sin(\pi x_3)\\
&- 0.22503x_4 + 0.524293\cos(\pi x_4)\\
&+ 0.685239\sin(\pi x_4)\\
&- 0.127742x_1x_2x_3x_4
\end{aligned}
$$

Rule$_3$: IF x_1 is $\mu(1.03417, 0.919468)$ and
x_2 is $\mu(-0.115958, 1.69308)$ and
x_3 is $\mu(-0.114371, 1.1357)$ and
x_4 is $\mu(-0.152534, 0.74255)$

$$
\begin{aligned}
\text{THEN } \hat{y}_1 &= 0.58632 - 1.28024x_1\\
&- 0.180169\cos(\pi x_1) - 0.470873\sin(\pi x_1)\\
&- 0.530146x_2 - 0.597328\cos(\pi x_2)\\
&+ 0.156929\sin(\pi x_2) + 0.176057x_3\\
&+ 0.0405789\cos(\pi x_3) + 1.09262\sin(\pi x_3)\\
&+ 0.353992x_4 - 0.437468\cos(\pi x_4)\\
&- 1.09654\sin(\pi x_4)\\
&+ 0.479358x_1x_2x_3x_4
\end{aligned}
$$

where $\mu(m_{ij}, \sigma_{ij})$ represents a Gaussian membership function with mean m_{ij} and deviation σ_{ij} in the ith input variable and the jth rule. The final rms error of the prediction output is about 0.008424. Fig. 11(a) plots the prediction outputs of the chaotic time series from $x(501)$ to $x(1000)$, when 500 training data from $x(1)$ to $x(500)$ were used. Fig. 11(b) plots the prediction errors between the proposed model and the desired output.

In this example, as in Example 1, the performance of the FLNFN model with the CCPSO learning method was compared to that of other methods. In the PSO [19] and CPSO [29], the parameters are the same as in Example 1. Three rules are set to construct the fuzzy model. The learning proceeded for 1000 generations, and was performed 50 times. Fig. 11(c) and (d) plots the predictions and the prediction errors of the PSO [19]. Fig. 11(e) and (f) plots the predictions and the prediction errors of the CPSO [29]. Fig. 11(g) and (h) plots the predictions and the prediction errors of differential evolution (DE) [35]. Fig. 11(i) and (j) plots the predictions and the prediction errors of the genetic algorithm (GA). Fig. 12 plots the learning curves of the best performance of the FLNFN model with CCPSO, PSO [19], CPSO [29], DE [35], and GA [38] learning methods. The proposed CCPSO method yields better prediction results than the other methods. Table II compares the best performance

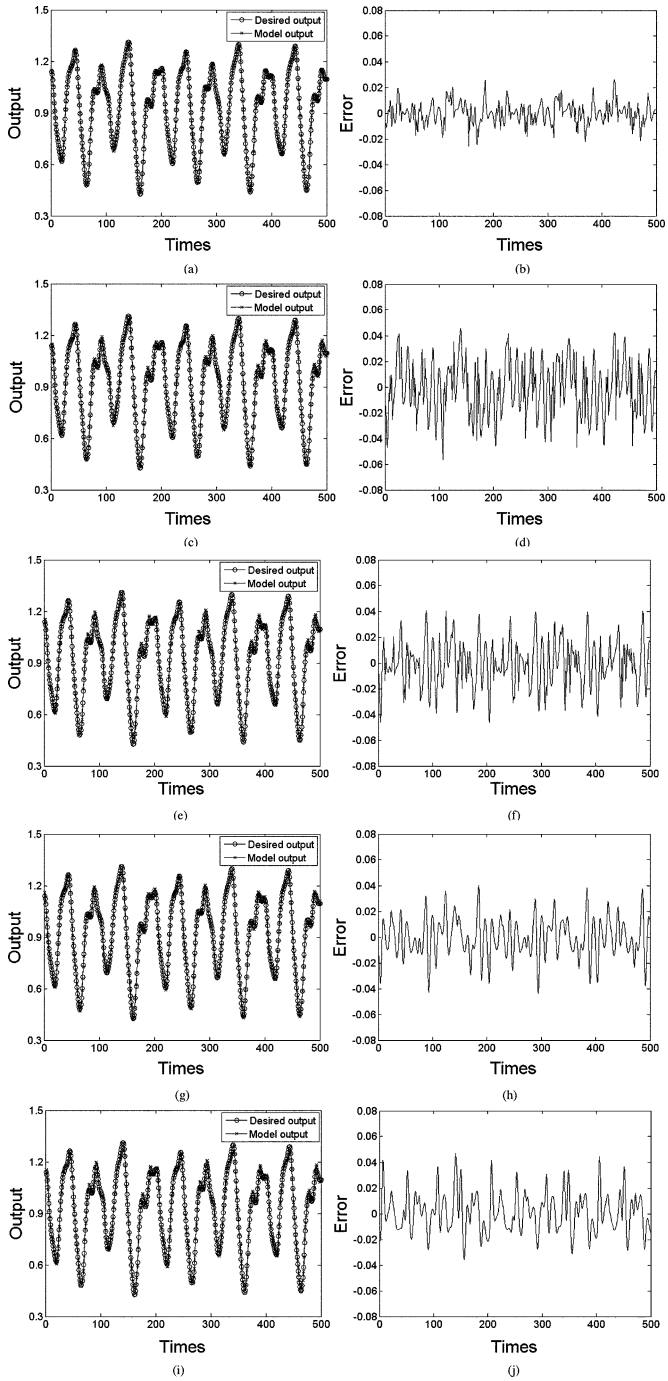

Fig. 11. (a) Prediction results of the proposed method. (b) Prediction errors of the proposed method. (c) Prediction results of the PSO [19]. (d) Prediction errors of the PSO. (e) Prediction results of the CPSO [29]. (f) Prediction errors of the CPSO. (g) Prediction results of the DE [35]. (h) Prediction errors of the DE. (i) Prediction results of the GA [38]. (j) Prediction errors of the GA.

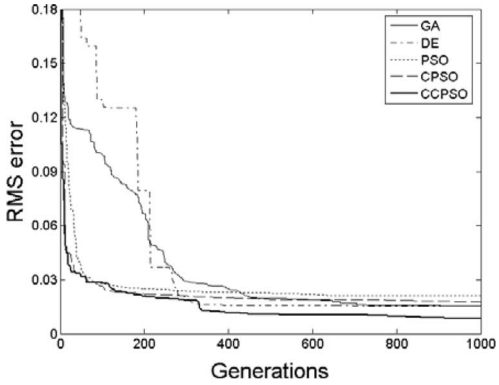

Fig. 12. Learning curves of the best performance of the proposed method, PSO [19], CPSO [29], DE [35], and GA [38].

TABLE II
COMPARISON OF THE BEST PERFORMANCE OF THE CCPSO, PSO, CPSO, DE, AND GA IN EXAMPLE 2

	CCPSO	PSO[19]	CPSO[29]	DE[35]	GA[39]
RMS error (training)	**0.008274**	0.020977	0.017527	0.016165	0.016176
RMS error (predicting)	**0.008424**	0.021054	0.017667	0.016258	0.016341

TABLE III
COMPARISON OF THE PERFORMANCE OF VARIOUS EXISTING MODELS

Method	$\text{RMSE}_{\text{Prediction}}$
FLNFN-CCPSO	**0.008274**
Back-propagation NN	0.02
Six-order polynomial	0.04
Cascaded-correlation	0.06
Auto regressive model	0.19
Linear predictive	0.55
GA-FLC [37]	0.26
SEFC [38]	0.032

of the CCPSO with those of the PSO [19], CPSO [29], DE [35], and GA [38]. Table III lists the generalization capabilities of the other methods [4], [36], [37]. The generalization capabilities were measured by using each model to predict 500 points immediately following the training dataset. The results show that the proposed FLNFN-CCPSO method offers a smaller rms error than the other methods.

C. Example 3: 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 [5]. In this example, the FLNFN model with the CCPSO learning method was used to forecast the number of sunspots The inputs x_i of the FLNFN-CCPSO method 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 $y_1^d(t)$ represents 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 FLNFN-CCPSO method while the number of sunspots of all 302 years (from 1703 to 2004) was used to test the FLNFN-CCPSO method.

The learning stage involved parameter learning by the CCPSO method. The coefficient ω 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. The swarm sizes were set to 50. The learning proceeded for 1000 generations, and was performed 50 times. In this example, three fuzzy rules are applied. They are as follows:

Rule 1: IF x_1 is $\mu(0.845312,\ 0.508771)$ and

x_2 is $\mu(0.90418,\ 0.451389)$ and

x_3 is $\mu(-0.0895866,\ 1.06449)$

$$\begin{aligned}\text{THEN } \hat{y}_1 = {} & -3.35896 - 0.436238x_1 + 1.4272\cos(\pi x_1) \\ & - 0.417788\sin(\pi x_1) + 2.19244x_2 \\ & - 0.32409\cos(\pi x_2) + 0.2113\ \sin(\pi x_2) \\ & - 1.36183x_3 - 0.480986\cos(\pi x_3) \\ & + 2.59738\ \sin(\pi x_3) - 0.361671(x_1x_2x_3)\end{aligned}$$

Rule 2: IF x_1 is $\mu(1.44016,\ 0.616583)$ and

x_2 is $\mu(0.314697,\ 7.34735)$ and

x_3 is $\mu(2.38597,\ 1.31093)$

$$\begin{aligned}\text{THEN } \hat{y}_1 = {} & 1.88537 + 1.78931x_1 \\ & + 1.73373\cos(\pi x_1) + 2.86658\sin(\pi x_1) \\ & + 4.53188x_2 - 3.75512\cos(\pi x_2) \\ & - 7.18406\ \sin(\pi x_2) + 0.868682x_3 \\ & - 0.541793\cos(\pi x_3) + 1.52449\ \sin(\pi x_3) \\ & + 0.763891(x_1x_2x_3)\end{aligned}$$

Rule 3: IF x_1 is $\mu(0.115385,\ 0.93777)$ and

x_2 is $\mu(0.326872,\ 1.02448)$ and

x_3 is $\mu(0.984958,\ 0.403378)$

$$\begin{aligned}\text{THEN } \hat{y}_1 = {} & 1.56458 + 0.703153x_1 \\ & + 0.0115128\cos(\pi x_1) - 0.119185\sin(\pi x_1) \\ & - 0.0263568x_2 - 0.681762\cos(\pi x_2) \\ & + 0.478785\ \sin(\pi x_2) + 7.0577x_3 \\ & - 0.808627\cos(\pi x_3) + 0.462158\ \sin(\pi x_3) \\ & + 10.6957(x_1x_2x_3)\end{aligned}$$

where $\mu(m_{ij}, \sigma_{ij})$ represents a Gaussian membership function with mean m_{ij} and deviation σ_{ij} in the ith input variable and the jth rule. The final rms error of the forecast output is about 10.337347. Fig. 13(a) presents the forecast outputs for years 1703–2004, using 151 training data from years 1703 to 1853. Fig. 13(b) plots the forecast errors between the proposed model and the desired output.

In this example, as in Examples 1 and 2, the performance of the FLNFN model with the CCPSO learning method was

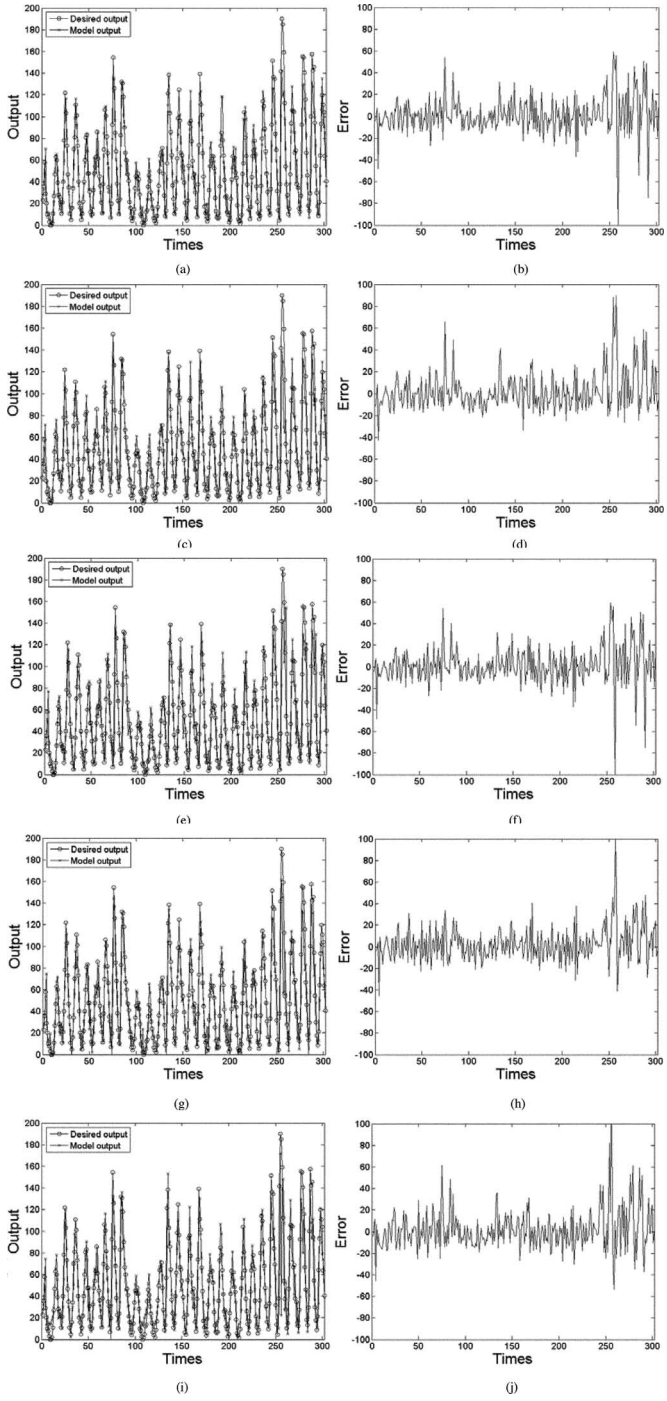

Fig. 13. (a) Forecast results of the proposed method. (b) Forecast errors of the proposed method. (c) Forecast results of the PSO [19]. (d) Forecast errors of the PSO. (e) Forecast results of the CPSO [29]. (f) Forecast errors of the CPSO. (g) Forecast results of the DE [35]. (h) Forecast errors of the DE. (i) Forecast results of the GA. (j) Forecast errors of the GA [38].

compared with that of the other methods. In PSO [19] and CPSO [29], the parameters are the same as in Examples 1 and 2. Three rules are used to construct the fuzzy model. The learning proceeded for 1000 generations, and was performed 50 times. Fig. 13(c) and (d) plot the forecast results and the forecast errors of the PSO [19]. Fig. 13(e) and (f) plots the forecast results and the forecast errors of the CPSO [29]. Fig. 13(g) and (h) plots the forecast results and the forecast errors of the DE [35]. Fig. 13(i) and (j) plots the forecast results and the forecast errors of the GA. Fig. 14 plots the learning curves of the best performance of the FLNFN model with CCPSO, PSO, CPSO, DE, and GA learning. The proposed CCPSO learning method yields better forecast results than the other methods. Table IV presents the best rms errors of training and forecasting for the CCPSO, PSO [19], CPSO [29], DE [35], and GA [38] learning methods. Table V lists the generalization capabilities of other methods [36], [37]. As presented in Tables IV and V, the proposed FLNFN-CCPSO method outperforms the other methods.

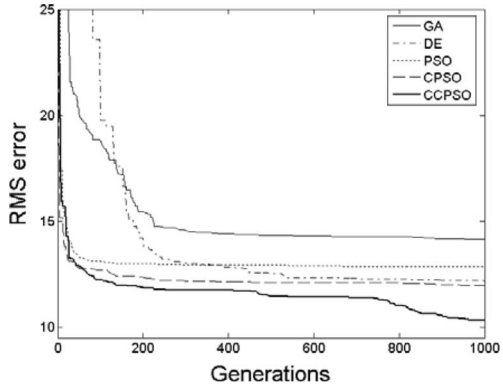

Fig. 14. Learning curves of the best performance of the proposed method, PSO [19], CPSO [29], DE [33], and GA [38].

TABLE IV
COMPARISON OF THE BEST PERFORMANCE OF CCPSO, PSO, AND CPSO IN EXAMPLE 3

	CCPSO	PSO[19]	CPSO[26]	DE[33]	GA[39]
RMS error (training)	**10.336261**	12.853411	11.983985	12.188084	14.143368
RMS error (forecasting)	**14.727455**	17.695794	17.446382	16.075044	19.726994

TABLE V
COMPARISON OF THE PERFORMANCE OF VARIOUS EXISTING MODELS

Method	RMS error (training)	RMS error (forecasting)
FLNFN-CCPSO	**10.3363**	**14.7275**
GA-FLC [37]	12.27	19.81
SEFC [38]	11.05	15.05

VI. CONCLUSION

This study proposes an efficient cultural cooperative particle swarm optimization learning method for the functional-link-based neural fuzzy network in predictive applications. The FLNFN model can generate the consequent part of a nonlinear combination of input variables. The proposed CCPSO method with cooperative behavior among multiple swarms increases the global search capacity using the belief space. The advantages of the proposed FLNFN-CCPSO method are as follows. 1) The consequent of the fuzzy rules is a nonlinear combination of input variables. This study uses the functional link neural network to the consequent part of the fuzzy rules. The functional expansion in the FLNFN model can yield the consequent part of a nonlinear combination of input variables; 2) the proposed CCPSO with cooperative behavior among multiple swarms can accelerate the search and increase global search capacity using the belief

space. The experimental results demonstrate that the CCPSO method can obtain a smaller rms error than the generally used PSO and CPSO for solving time series prediction problems.

Although the FLNFN-CCPSO method can perform better than the other methods, there is an advanced topic to the proposed FLNFN-CCPSO method. In this study, the number of rules is predefined. In future studies, it would be better if the proposed method has the ability to determine the number of fuzzy rules.

REFERENCES

[1] J. E. Box and G. M. Jenkins, *Time Series Analysis, Forecasting and Control*. San Francisco, CA: Holden Day, 1970.

[2] H. Tong, *Non-linear Time Series: A Dynamical System Approach*. London, U.K.: Oxford Univ. Press, 1990.

[3] M. Li, K. Mehrotra, C. Mohan, and S. Ranka, "Sunspot numbers forecasting using neural networks," in *Proc. IEEE Int. Conf.*, Sep. 1990, vol. 1, pp. 524–529.

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

[5] 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.

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

[8] T. Takagi and M. Sugeno, "Fuzzy identification of systems and its applications to modeling and control," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-15, no. 1, pp. 116–132, Jan./Feb. 1985.

[9] 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.

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

[11] C. Li and C. Y. Lee, "Self-organizing neuro-fuzzy system for control of unknown plants," *IEEE Trans. Fuzzy Syst.*, vol. 11, no. 1, pp. 135–150, Feb. 2003.

[12] F. Sun, Z. Sun, L. Li, and H. X. Li, "Neuro-fuzzy adaptive control based on dynamic inversion for robotic manipulators," *Fuzzy Sets Syst.*, vol. 134, pp. 117–133, 2003.

[13] P. P. Angelov and D. P. Filev, "An approach to online identification of Takagi–Sugeno fuzzy models," *IEEE Trans. Syst., Man, Cybern.*, vol. 34, no. 1, pp. 484–498, Feb. 2004.

[14] H. Takagi, N. Suzuki, T. Koda, and Y. Kojima, "Neural networks designed on approximate reasoning architecture and their application," *IEEE Trans. Neural Netw.*, vol. 3, no. 5, pp. 752–759, Sep. 1992.

[15] E. Mizutani and J.-S.R. Jang, "Coactive neural fuzzy modeling," in *Proc. Int. Conf. Neural Netw.*, 1995, pp. 760–765.

[16] Y. H. Pao, *Adaptive Pattern Recognition and Neural Networks*. Reading, MA: Addison-Wesley, 1989.

[17] 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.

[18] 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, 2007, pp. 53–58.

[19] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. IEEE Int. Conf. Neural Netw.*, 1995, pp. 1942–1948.

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

[21] 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.

[22] 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.

[23] 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.

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

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

[26] H. M. Feng, "Self-generation RBFNs using evolutional PSO learning," *Neurocomputing*, vol. 70, no. 1–3, pp. 241–251, Dec. 2006.

[27] Y. Song, Z. Chen, and Z. Yuan, "New chaotic PSO-based neural network predictive control for nonlinear process," *IEEE Trans. Neural Netw.*, vol. 18, no. 2, pp. 595–601, Mar. 2007.

[28] X. Cai, N. Zhang, G. K. Venayagamoorthy, and D. C. Wunsch, "Time series prediction with recurrent neural networks trained by a hybrid PSO-EA algorithm," *Neurocomputing*, vol. 70, no. 13–15, pp. 2342–2353, Aug. 2007.

[29] 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.

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

[31] M. Mansour, S. F. Mekhamer, and N. El-Sherif 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.

[32] 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.

[33] 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, pp. 1672–1678.

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

[35] R. Storn, "System design by constraint adaptation and differential evolution," *IEEE Trans. Evol. Comput.*, vol. 3, no. 1, pp. 22–34, Apr. 1999.

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

[37] 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.

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

Cheng-Jian Lin (S'93–M'95) received the B.S. degree in electrical engineering from Tatung University, Taipei, Taiwan, R.O.C., 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, Taichung, Taiwan, where from February 2005 to July 2007, he was a full Professor. Currently, he is a full Professor in the Department of Electrical Engineering, National University of Kaohsiung, Kaohsiung, Taiwan. From 2001 to 2005, he served as the Chairman of the Department of Computer Science and Information Engineering, Chaoyang University of Technology, where, from 2005 to 2007, he served as the Library Director of Poding Memorial Library. He is the author or coauthor of more than 150 papers published in referred journals and conference proceedings. His current research interests include soft computing, pattern recognition, intelligent control, image processing, bioinformatics, and field-programmable gate array design.

Prof. Lin is a member of the Phi Tau Phi. He is also a member of the Chinese Fuzzy Systems Association, the Chinese Automation Association, the Taiwanese Association for Artificial Intelligence (TAAI), the Institute of Electronics, Information, and Communication Engineers, and the IEEE Computational Intelligence Society. He is an Executive Committee Member of the TAAI. He has served as the Associate Editor of the *International Journal of Applied Science and Engineering* from 2002 to 2005.

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

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 National Chiao-Tung University (NCTU), Hsinchu, Taiwan, R.O.C., 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 is currently the Provost of Academic Affairs and the Chair Professor of electrical and control engineering. He has served as the Founding Dean of the Computer Science College of NCTU from 2005 to 2007. He is the author or coauthor of more than 110 journal papers, including about 80 IEEE Transactions papers. He is the author of a textbook *Neural Fuzzy Systems* (Prentice-Hall, 1996) and *Neural Fuzzy Control Systems with Structure and Parameter Learning* (World Scientific, 1994). His current research interests include intelligent technology, soft computing, brain–computer interface, intelligent transportation systems, robotics and intelligent sensing, and nanobioinformation technologies and cognitive science.

Prof. Lin is a member of Tau Beta Pi, Eta Kappa Nu, and Phi Kappa Phi honorary societies. He was a Member of the Board of Governors BoG of the IEEE Systems, Man, Cybernetics Society (SMCS) from 2003 to 2005, and is the current BoG member of the IEEE Circuits and Systems Society (CASS). He was the IEEE Distinguished Lecturer from 2003 to 2005. He also serves as the Deputy Editor-in-Chief (EIC) of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS, PART II now. He was the Program Chair of the 2006 IEEE International Conference on Systems, Man, and Cybernetics held in Taipei. He was the President of the Board of Government of the Asia Pacific Neural Networks Assembly from 2004 to 2005. He has been the recipient of several awards including the Outstanding Research Award granted by the National Science Council (NSC), Taiwan, since 1997 to present, the Outstanding Professor Award granted by the Chinese Institute of Engineering 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 in 2000.