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

Cheng-Hung Chen*, *Student Member*, *IEEE*, Yong-Cheng Liu, Cheng-Jian Lin, *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. Finally, the proposed functional-link-based neural fuzzy network with cultural cooperative particle swarm optimization (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.**

I. Introduction

NEURAL fuzzy networks [1]-[7] have become a popular research topic. In the typical TSK-type neural fuzzy network [2]-[7], 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 [8] and the CANFIS [9] models. These models [8]-[9] 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) [10]-[11] in the consequent part of the fuzzy rules [12].

Training of the parameters is the main problem in designing a neural fuzzy system. Therefore, technologies that can be used to train the system parameters and find the global solution while optimizing the overall structure, are required. Hence, the proposed cultural cooperative particle swarm optimization (CCPSO) learning method, which combines the cooperative particle swarm optimization (CPSO) [13] and cultural algorithm (CA) [14], 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 cultural cooperative particle swarm optimization (CCPSO) for the functional-link-based neural fuzzy network (FLNFN) in several predictive applications. The proposed FLNFN model is based on our previous research [12]. 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 proposed CCPSO is a hybrid method which 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.

II. Structure of Functional-Link-Based Neural Fuzzy Networks

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. Figure 1 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} ... and x_i is A_{ij} ... and x_N is A_{Nj}

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

Y. C. Liu is with the Dept. of Computer Science and Information Engineering, Chaoyang University of Technology, Wufong Township Taichung County, Taiwan 41349, R.O.C.

C. J. Lin is with the Dept. of Electrical Engineering, National University of Kaohsiung, Kaohsiung, Taiwan 811, R.O.C.

*Corresponding author. (E-mail: chchen.ece93g@nctu.edu.tw)

978-1-4244-1819-0/08/$25.00©2008 IEEE

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

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.

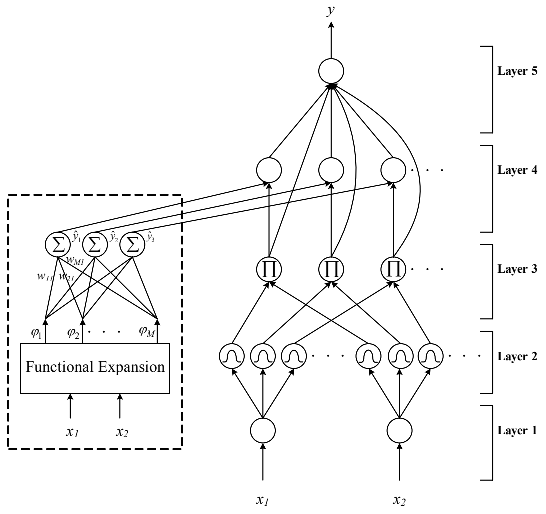

Fig. 1. Structure of proposed FLNFN model.

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:

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

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

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

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

Nodes in layer 3 receive one-dimensional membership degrees of the associated rule from the nodes of a set in layer 2. Here, the product operator described above is adopted to perform the precondition part of the fuzzy rules. As a result, the output function of each inference node is

$$u_j^{(3)} = \prod_i u_{ij}^{(2)} \quad (4)$$

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, as shown in Fig. 1. For such a node,

$$u_j^{(4)} = u_j^{(3)} \cdot \sum_{k=1}^{M} w_{kj}\phi_k \quad (5)$$

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 two-dimensional 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,

$$y = u^{(5)} = \frac{\sum_{j=1}^{R} u_j^{(4)}}{\sum_{j=1}^{R} u_j^{(3)}} = \frac{\sum_{j=1}^{R} u_j^{(3)} \left(\sum_{k=1}^{M} w_{kj}\phi_k \right)}{\sum_{j=1}^{R} u_j^{(3)}} = \frac{\sum_{j=1}^{R} u_j^{(3)} \hat{y}_j}{\sum_{j=1}^{R} u_j^{(3)}} \quad (6)$$

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

III. Learning Algorithms for the FLNFN Model

This section describes the proposed cultural cooperative particle swarm optimization (CCPSO) method. Before the CCPSO method is designed, cooperative particle swarm optimization (CPSO) [13] that differs from the traditional PSO is introduced.

The traditional PSO uses one swarm of particles defined by the P-dimension vectors to evolve. The CPSO method can change traditional PSO into P swarms of one-dimension vectors, such that each swarm represents a dimension of the original problem. Figures 2 (a)-(b) show 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-dimension vector, the vector is split into its components so that P swarms (of I particles each) optimize a one-dimension 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 one-dimension 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.

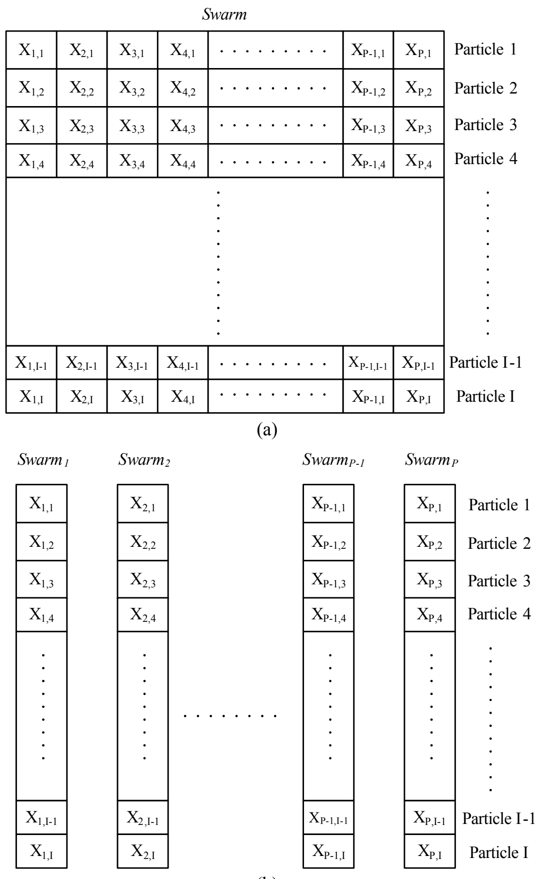

Fig. 2. Framework of the (a) PSO and (b) CPSO.

The CCPSO method is characteristic of the cooperative particle swarm optimization and cultural algorithm. Figure 3 shows the framework of the proposed CCPSO learning method, which is based on a CPSO all of whose parameters are simultaneously tuned using the belief space of the CA. The CCPSO method can strengthen the global search capability. If 50-dimension vectors are used in the original PSO, then the vectors in CCPSO can be changed into 50 swarms of one-dimension vectors. In the original PSO, the particle can exhibit 50 variations in each generation, whereas the CCPSO offers 50x50=2500 different combinations in each generation. Additionally, each position of the CCPSO can be adjusted not only using the belief space which 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.

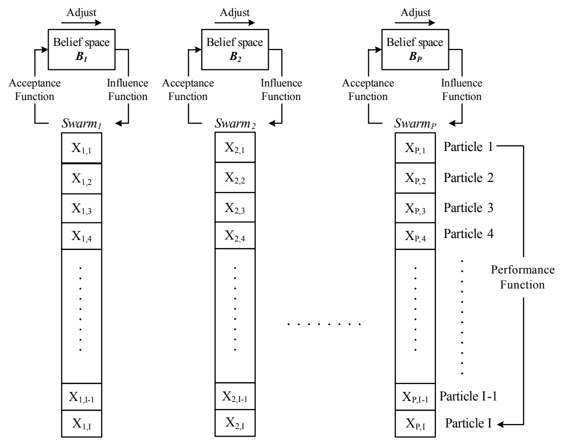

Fig. 3. Framework of proposed CCPSO learning method.

The detailed flowchart of the proposed CCPSO method is presented in Fig. 4. The foremost step in CCPSO is the coding of the neural fuzzy network into a particle. Figure 5 shows an example of the coding of parameters of neural fuzzy network into a particle where *i* and *j* represent the *i*th input variable and the *j*th rule, respectively. In this study, a Gaussian membership function is adopted with variables that represent the mean and deviation of the membership function. Figure 5 represents the neural fuzzy network given by Eq. (1), 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 *j*th rule node. In this study, a real number represents the position of each particle.

The learning process is described step-by-step below.

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, …, P represents the pth swarm, i=1, 2, …, 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, …, 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.

$$F = \sqrt{\frac{1}{D}\sum_{d=1}^{D}(y_d - \bar{y}_d)^2} \quad (7)$$

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 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.

$$L_{p,i}(t+1)=\begin{cases} x_{p,i}(t), & \text{if } F(x_{p,i}(t)) < F(L_{p,i}(t)) \\ L_{p,i}(t), & \text{if } F(x_{p,i}(t)) \geq F(L_{p,i}(t)) \end{cases}. \quad (8)$$
$$G_p(t+1)=\arg\min_{L_{p,i}} F(L_{p,i}(t+1)), \qquad 1\leq i\leq I$$

■ 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 order of increasing fitness. Then, the paragon of each $Swarm_p$ is put into 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.

$$N_{accepted}=n\%\cdot I+\frac{n\%}{t}\cdot I \quad (9)$$

where $n\%$ is a parameter that is set by user, and must specify the top performing 20% [15]; 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 $BI_p=[l_p,u_p]=\{x \mid l_p \leq x \leq u_p, x\in\Re\}$, 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 below.

$$\begin{aligned} l_p &= \begin{cases} x_{p,i} & \text{if } x_{p,i}\leq l_p \\ l_p & \text{otherwise} \end{cases} \\ u_p &= \begin{cases} x_{p,i} & \text{if } x_{p,i}\geq u_p \\ u_p & \text{otherwise} \end{cases} \end{aligned} . \quad (10)$$

■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 Eq. (11). 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 velocity and position of each particle to generate the each new $Swarm_p$ using Eqs. (12) and (13).

$$x_{p,i}(t)=\begin{cases} x_{p,i}(t)+\left| Rand()\cdot(u_p-l_p)\right| & \text{if } x_{p,i} < l_p \\ x_{p,i}(t)-\left| Rand()\cdot(u_p-l_p)\right| & \text{if } x_{p,i} > u_p \end{cases} \quad (11)$$

$$\begin{aligned} v_{p,i}(t+1)=w\cdot v_{p,i}(t)&+c_1\cdot Rand()\cdot[L_{p,i}(t+1)-x_{p,i}(t)] \\ &+c_2\cdot Rand()\cdot[G_p(t+1)-x_{p,i}(t)] \end{aligned} \quad (12)$$

$$x_{p,i}(t+1)=x_{p,i}(t)+v_{p,i}(t+1) \quad (13)$$

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 $v_{p,i}(t)$.

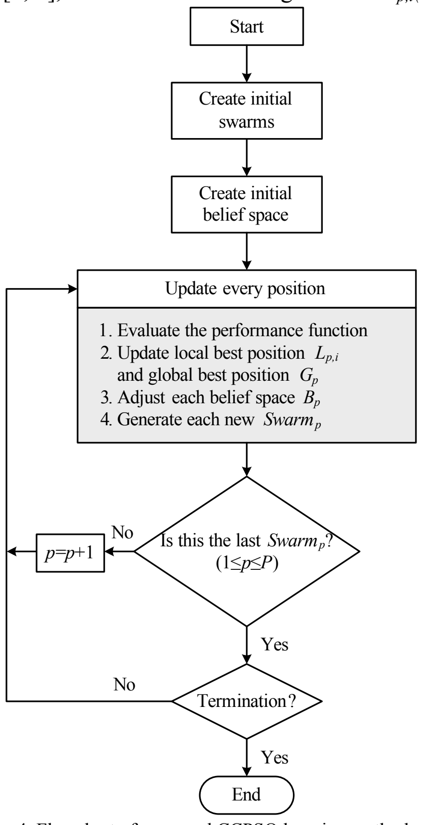
Start
Create initial swarms
Create initial belief space
Update every position
1. Evaluate the performance function
2. Update local best position $L_{p,i}$ and global best position G_p
3. Adjust each belief space B_p
4. Generate each new $Swarm_p$
No
$p=p+1$
Is this the last $Swarm_p$? ($1\leq p\leq P$)
Yes
No
Termination?
Yes
End

Fig. 4. Flowchart of proposed CCPSO learning method.

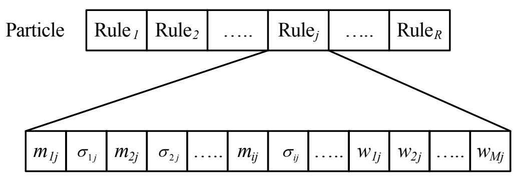

Fig. 5. Coding FLNFN model into a particle in the proposed CCPSO.

IV. Experimental Results

This section discusses two examples that were considered to evaluate the FLNFN model with the CCPSO learning method. The first example involves predicting a chaotic time series [16], and the second example involves forecasting the number of sunspots [17].

A. Example 1: Prediction of chaotic time series

The Mackey-Glass chaotic time series *x(t)* was generated using the following delay differential equation;

$$\frac{dx(t)}{dt} = \frac{0.2x(t-\tau)}{1+x^{10}(t-\tau)} - 0.1x(t)\,. \qquad (14)$$

Crowder [16] extracted 1000 input-output data pairs {x, yd} using four past values of *x(t)*:

$$[x(t-18), x(t-12), x(t-6), x(t); x(t+6)] \qquad (15)$$

where τ=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+Δt)*, where *Δt* is a time interval into the future. The first 500 pairs (from *x(1)* to *x(500)*) were the training data set, 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 CCPSO method. 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. The swarm sizes were set to 50. The learning proceeded for 1000 generations, and was performed fifty times. In this example, three fuzzy rules are applied. They are as follows.

Rule1:

IF x_1 is $\mu(0.452959,-5.36833)$ and x_2 is $\mu(-0.10799,0.768855)$
and x_3 is $\mu(-0.850613,-3.60999)$ and x_4 is $\mu(1.09886,0.495632)$

$$\begin{aligned}\text{THEN } \hat{y}_1 = {} & 2.20613 + 0.580829x_1 + 0.391061\cos(\pi x_1) + 0.332886\sin(\pi x_1) \\ & - 4.68232x_2 - 5.05388\cos(\pi x_2) + 1.73753\sin(\pi x_2) \\ & - 0.656754x_3 + 1.71626\cos(\pi x_3) + 0.0923789\sin(\pi x_3) \\ & + 4.93925x_4 - 0.416084\cos(\pi x_4) + 1.45935\sin(\pi x_4) \\ & + 0.990628x_1 x_2 x_3 x_4\end{aligned}$$

Rule2:

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 $\mu(1.01531,0.524704)$

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

Rule3:

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 $\mu(-0.152534,0.74255)$

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

where $\mu(m_{ij}, \sigma_{ij})$ represents a Gaussian membership function with mean m_{ij} and deviation σ_{ij} in the *i*th input variable and the *j*th rule. The final RMS error of the prediction output is about 0.008424. Figure 6(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. Figure 6(b) plots the prediction errors between the proposed model and the desired output.

In this example, the performance of the FLNFN model with the CCPSO learning method was compared to that of other methods. In the PSO [18] and CPSO [13], the swarm sizes were set to 50. 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. Three rules are set to construct the fuzzy model. The learning proceeded for 1000 generations, and was performed fifty times. Figures 6(c) plots the prediction errors of particle swarm optimization [18]. Figures 6(d) plots the prediction errors of cooperative particle swarm optimization [13]. Figures 6(e) plots the prediction errors of differential evolution [19]. Figures 6(f) plots the prediction errors of genetic algorithm. Figure 7 plots the learning curves of the best performance of the FLNFN model with CCPSO, PSO [18], CPSO [13], differential evolution (DE) [19] and genetic algorithm (GA) [20] learning methods. The proposed CCPSO method yields better prediction results than the other methods. Table 1 compares the best performance of the CCPSO was compared with those of PSO [18], CPSO [13], differential evolution (DE) [19], and GA [20]. Table 2 lists the generalization capabilities of other methods [21]-[23]. The generalization capabilities were measured by using each model to predict 500 points immediately following the training data set. The results show that the proposed FLNFN-CCPSO method offers a smaller RMS error than other methods.

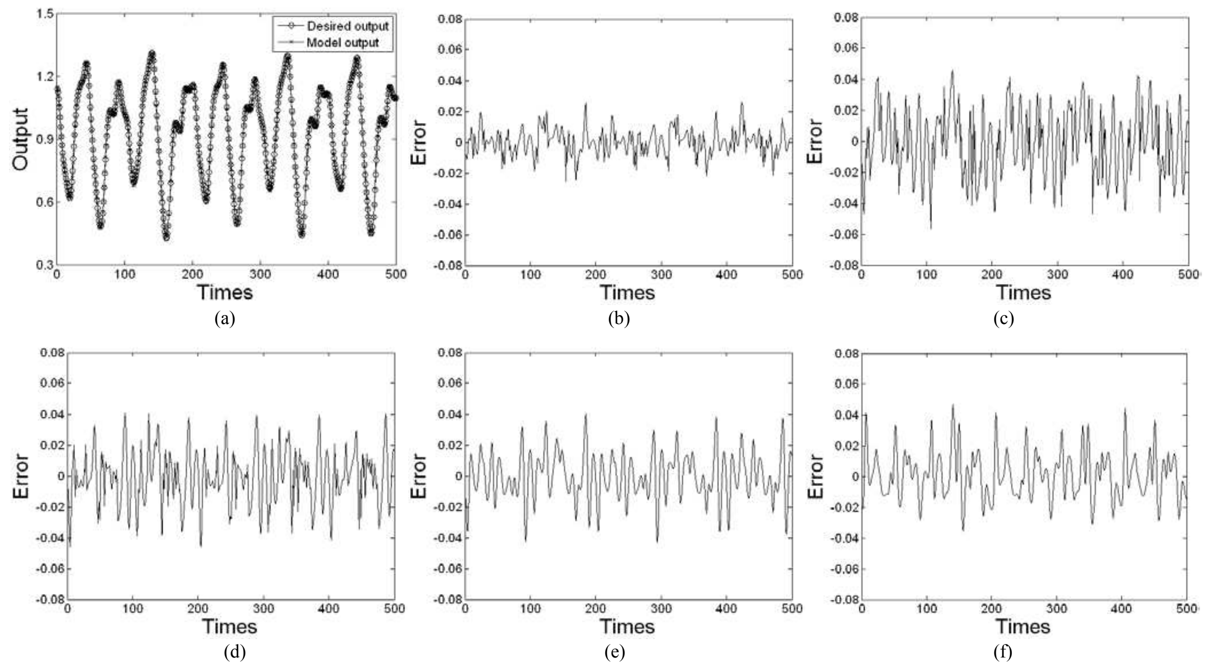

Fig. 6. (a) Prediction results of the proposed method. (b) Prediction errors of the proposed method. (c) Prediction errors of particle swarm optimization [18]. (d) Prediction errors of cooperative particle swarm optimization [13]. (e) Prediction errors of differential evolution [19]. (f) Prediction errors of genetic algorithm [20].

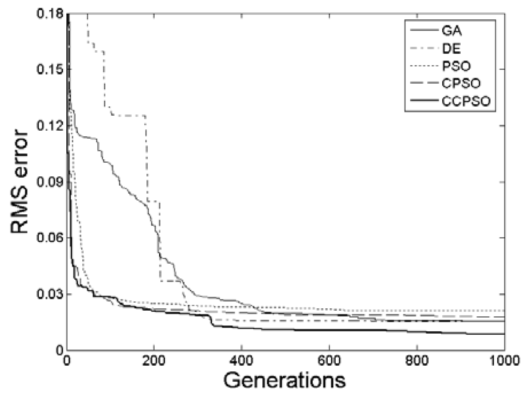

Fig. 7. Learning curves of best performance of proposed method, PSO [18], CPSO [13], DE [19] and GA[20].

B. Example 2: Forecast of the number of sunspots

The number of sunspots varied nonlinearly from 1700 to 2004, in non-stationary, and non-Gaussian cycles that are difficult to predict [17]. 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 $y_1^d(t)$ 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 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 *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. The swarm sizes were set to 50. The learning proceeded for 1000 generations, and was performed fifty times. In this example, three fuzzy rules are applied.

The final RMS error of the forecast output is about 10.337347. Figure 8(a) presents the forecast outputs for years 1703 to 2004, using 151 training data from years 1703 to 1853. Figure 8(b) plots the forecast errors between the proposed model and the desired output.

In this example, as in example 1, the performance of the FLNFN model with CCPSO learning method was compared with that of other methods. In PSO [18] and CPSO [13], the parameters are the same as in example 1. Three rules are used to construct the fuzzy model. The learning proceeded for 1000 generations, and was performed fifty times. Figures 8(c) plots the forecast errors of particle swarm optimization [18]. Figures 8(d) plots the forecast errors of cooperative particle swarm optimization [13]. Figures 8(e) plots the forecast errors of differential evolution [19]. Figures 8(f) plots the forecast errors of GA [20]. Figure 9 plots the learning curves of 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 3 presents the best RMS errors of training and forecasting for CCPSO, PSO [18], CPSO [13], DE [19], and GA [20] learning methods. Table 4 lists the generalization capabilities of other methods [22]-[23]. As presented in Table

3 and Table 4, the proposed FLNFN-CCPSO method outperforms the other methods.

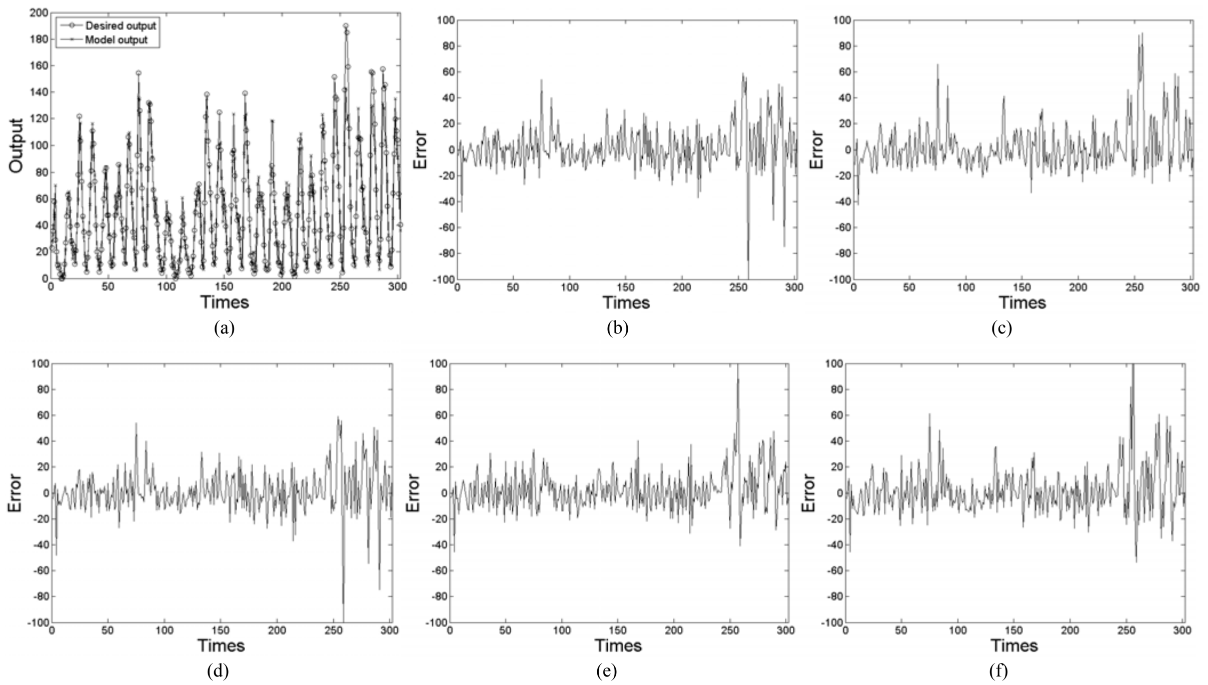

Fig. 8. (a) Forecast results of the proposed method. (b) Forecast errors of the proposed method. (c) Forecast errors of particle swarm optimization [18]. (d) Forecast errors of cooperative particle swarm optimization [13]. (e) Forecast errors of differential evolution [19]. (f) Forecast errors of GA [20].

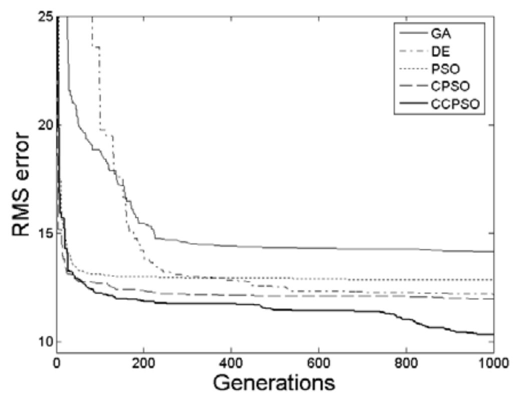

Fig. 9. Learning curves of best performance of proposed method, PSO [18], CPSO [13], DE [19] and GA[20].

Table 1: Comparison of best performance of CCPSO, PSO, CPSO, DE, and GA in Example 1.

	CCPSO	PSO	CPSO	DE	GA
RMS error (training)	**0.0083**	0.0210	0.0175	0.0162	0.0162
RMS error (predicting)	**0.0084**	0.0211	0.0176	0.0163	0.0163

Table 2: Comparison of performance of various existing models.

METHOD	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 [22]	0.26
SEFC [23]	0.032

Table 3: Comparison of best performance of CCPSO, PSO and CPSO, DE, and GA in Example 2.

	CCPSO	PSO	CPSO	DE	GA
RMS error (training)	**10.34**	12.85	11.98	12.19	14.14
RMS error (forecasting)	**14.73**	17.70	17.45	16.08	19.73

Table 4: Comparison of performance of various existing models.

METHOD	RMS error (training)	RMS error (forecasting)
FLNFN-CCPSO	**10.3363**	**14.7275**
GA-FLC [22]	12.27	19.81
SEFC [23]	11.05	15.05

V. 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.

ACKNOWLEDGMENT

This research was sponsored in part by the Ministry of Economic Affairs, Taiwan, R.O.C., under Grant 96-EC-17-A-02-S1-032, in part by the National Science Council, Taiwan, R.O.C., under Grant NSC 95-2221-E-009-180, in part by the National Science Council, Taiwan, R.O.C., under Grant NSC 95-2221-E-390-040-MY2, and in part by the Industrial Technology Research Institute, Taiwan, R.O.C., under Grant 6301XS3210.

REFERENCES

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

[2] T. Takagi and M. Sugeno, "Fuzzy identification of systems and its applications to modeling and control," *IEEE Trans. on Syst., Man, Cybern.*, vol. 15, pp. 116-132, 1985.

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

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

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

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

[7] 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.

[8] H. Takagi, N. Suzuki, T. Koda, and Y. Kojima, "Neural networks designed on approximate reasoning architecture and their application," *IEEE Trans. Neural Networks*, vol. 3, pp. 752-759, Sept. 1992.

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

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

[11] 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. on Syst., Man, and Cybern.*, vol. 29, Apr. 1999.

[12] C. H. Chen, C. T. Lin, and C. J. Lin, "A functional-link-based fuzzy neural network for temperature control," *2007 IEEE Symposium on Foundations of Computational Intelligence*, Honolulu, Hawaii, USA, pp. 53-58, April 1-5, 2007.

[13] 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.

[14] R. G. Reynolds, "An introduction to cultural algorithms," *Proc. 3rd Annual Conference on Evolutionary Programming*, A.V. Sebald and L.J. Fogel (eds.), World Scientific, River Edge, NJ, USA, pp. 131- 139, 1994.

[15] S. M. Saleem, *Knowledge-based solution to dynamic optimization problems using cultural algorithms*, PhD thesis, Wayne State University, Detroit, Michigan, 2001.

[16] R. S. Cowder, "Predicting the Mackey-glass time series with cascade-correlation learning," *Proceedings of the 1990 Connectionist Models Summer School*, pp.117-123, 1990.

[17] S. H. Ling, Frank 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. Industrial Electornic.*, vol. 50, no. 4, pp. 793-799, Aug. 2003.

[18] J. Kennedy and R. Eberhart, "Particle swarm optimization," *Proc. IEEE Int. Conf. on Neural Networks*, pp. 1942-1948,1995.

[19] R. Storn, "System design by constraint adaptation and differential evolution," *IEEE Trans. on Evolutionary Computation*, vol. 3, no. 1, pp. 22-34, Apr. 1999.

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

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

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

[23] 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., Part B*, vol. 30, no. 2, pp. 290-302, Apr. 2000.