Available online at www.sciencedirect.com
ScienceDirect
Neurocomputing 71 (2007) 297–310
NEUROCOMPUTING
www.elsevier.com/locate/neucom

The design of neuro-fuzzy networks using particle swarm optimization and recursive singular value decomposition

Cheng-Jian Lin*, Shang-Jin Hong

Department of Computer Science and Information Engineering, Chaoyang University of Technology, 168 Gifong E. Rd., Wufong, Taichung County 413, Taiwan

Received 27 December 2005; received in revised form 30 August 2006; accepted 15 December 2006
Communicated by A. Abraham
Available online 22 February 2007

Abstract

In this paper, a neuro-fuzzy network with novel hybrid learning algorithm is proposed. The novel hybrid learning algorithm is based on the fuzzy entropy clustering (FEC), the modified particle swarm optimization (MPSO), and the recursive singular value decomposition (RSVD). The FEC is used to partition the input data for performing structure learning. Then, we adopt the MPSO to adjust the antecedent parameters of fuzzy rules. Two strategies in the MPSO, called the effective local approximation method (ELAM) and the multi-elites strategy (MES), are proposed to improve the performance of the traditional PSO. Moreover, we will apply RSVD to obtain the optimal consequent parameters of fuzzy rules. The proposed hybrid learning algorithm achieves superior performance in learning speed and learning accuracy than those of some traditional genetic methods.
© 2007 Elsevier B.V. All rights reserved.

Keywords: Neuro-fuzzy networks; Fuzzy entropy; Particle swarm optimization; Function approximation; Singular value decomposition (SVD)

1. Introduction

Neuro-fuzzy networks have been demonstrated to be successful [10,17–20,23–25,27]. It combines the semantic transparency of rule-based fuzzy systems and the learning capability of neural networks. Neuro-fuzzy networks have two typical types which are Mamdani-type and TSK-type models. For Mamdani-type neuro-fuzzy networks [19,27], the minimum fuzzy implication is used in fuzzy reasoning. Meanwhile, for TSK-type neuro-fuzzy networks (TNFNs) [10,17,20,23], the consequence of each rule is a function input variable. The general adopted function is a linear combination of input variables plus a constant term. Many researchers [10,17] have been shown that a TNFN achieves superior performance in network size and learning accuracy than that of a Mamdani-type neuro-fuzzy network.

The back-propagation (BP) learning algorithm [18] is widely used for training neuro-fuzzy networks by means of error propagation via variation calculus. However, the BP learning algorithm is a powerful training technique that can be applied in networks with feed-forward structure to transform them into adaptive systems. But the algorithm may reach the local minima and the global solution may never be found because the steepest descent optimization technique is used in BP training to minimize the error function. In addition, the performance of the BP learning algorithm depends on the initial values of the model parameters, and for different network topologies one has to derive new mathematical expressions for each network layer. About this, the advent of evolutionary computation has inspired new designs and models. In contrast to traditional computation systems, which may be good at accurate and exact computation but have brittle operations, evolutionary computation provides a more robust and efficient approach for solving complex real-world problems [2,6,28]. For this reason, many researchers use genetic algorithms (GAs) for learning of fuzzy models. In

*Corresponding author.
E-mail address: cjlin@mail.cyut.edu.tw (C.-J. Lin).

0925-2312/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2006.12.016

the literature [9,11,16], several GA-based approaches have appeared and have better candidates than BP algorithm.

Recently, a new algorithm, called particle swarm optimization (PSO), was proposed. It is an evolutionary computation technique developed by Kennedy and Eberhart in 1995 [14]. The underlying motivation for the development of PSO algorithm was social behavior of animals such as bird flocking, fish schooling, and swarm theory. The PSO begins with a random population and searches for optima by updating the population. The advantages of PSO are that it has no evolution operators such as crossover and mutation and it need not adjust too many free parameters. Moreover, each potential solution is also assigned a randomized velocity. The potential solutions, called particles, are then "flown" through the problem space. Compared with the GA, the PSO has some attractive characteristics. First, the PSO has memory. The knowledge of good solutions in the PSO is retained by all particles; whereas in the GA, the previous knowledge of the problem is destroyed when the population is changed. Second, the PSO has constructive cooperation between particles. The particles in the swarm share information between them. Successful applications of the PSO for several optimization problems, like control problems [1,7,29] and feed-forward neural network design [12,21]. Therefore, we introduce a modified PSO (MPSO) to determine the antecedent parameters of fuzzy rule.

In this paper, we propose a novel hybrid learning algorithm for neuro-fuzzy networks. The construction of neuro-fuzzy networks involves two phases: structure learning and parameter learning. In structure learning, the input data set is partitioned into a set of clusters using fuzzy entropy clustering (FEC). Membership functions associated with each cluster are defined according to statistical means and variances of the data points that were included in the cluster. Then a fuzzy rule is extracted from each cluster to form a fuzzy rule base. In parameter learning, most neuro-fuzzy networks use BP to refine parameters of the system. However, BP suffers from the problems of local minima and lower convergence rate. To decrease the size of the search space and speed up the convergence, we propose a MPSO to adjust the antecedent parameters of fuzzy rules. In the MPSO, two strategies, called effective local approximation method (ELAM) and multi-elites strategy (MES), are proposed to improve the performance of the traditional PSO. Moreover, we adopt the recursive singular value decomposition (RSVD) to determine the optimal consequent parameters of fuzzy rules. The proposed hybrid learning algorithm has the following advantages: (1) Using the MPSO to find the global optimal solution is easier than the BP method. (2) Determining the initial particles of MPSO by FEC method can reduce blind search for parameters of fuzzy rules. (3) The proposed hybrid learning algorithm achieves superior performance in learning speed and learning accuracy.

This paper is organized as follows. Section 2 describes the TSK-type fuzzy model. Overview of the PSO is described in Section 3. In Section 4, we will describe the proposed hybrid learning algorithm for TNFNs. Section 5 presents the simulation results. Finally, conclusions are given in Section 6.

2. Structure of a TNFN

A fuzzy model is a knowledge-based system characterized by a set of rules, which models the relationship among control input and output. The reasoning process is defined by means of the employed aggregation operators, the fuzzy connectives and the inference method. The fuzzy knowledge base contains the definition of fuzzy sets stored in the fuzzy database and a collection of fuzzy rules, which constitute the fuzzy rule base. Fuzzy rules are defined by their antecedents and consequents, which relates an observed input state to a desired control action. Most fuzzy systems employ the inference method proposed by Mamdani in which the consequence parts are defined by fuzzy sets [18]. A Mamdani-type fuzzy rule has the form:

$$\begin{aligned}&\text{IF } x_1 \text{ is } A_{1j}(m_{1j},\sigma_{1j}) \text{ and } x_2 \text{ is } A_{2j}(m_{2j},\sigma_{2j})\ldots\text{and}\\&\quad x_n \text{ is } A_{nj}(m_{nj},\sigma_{nj}) \text{ THEN } y' \text{ is } B_j\ (m_j,\sigma_j),\end{aligned} \tag{1}$$

where m_{ij} and σ_{ij} represent a Gaussian membership function with mean and deviation with ith dimension and jth rule node. The consequences B_j of jth rule is aggregated into one fuzzy set for the output variable y'. The crisp action is obtained through defuzzification, which calculates the centroid of the output fuzzy set. Besides the more common fuzzy inference method proposed by Mamdani, Takagi, Sugeno and Kang introduced a modified inference scheme [23]. The first two parts of the fuzzy inference process, fuzzifier the inputs and applying the fuzzy operator are exactly the same. A Takagi–Sugeno–Kang (TSK)-type fuzzy model employs different implication and aggregation methods than the standard Mamdani controller. Instead of using fuzzy sets the conclusion part of a rule, is a linear combination of the crisp inputs:

$$\begin{aligned}&\text{IF } x_1 \text{ is } A_{1j}(m_{1j},\sigma_{1j}) \text{ and } x_2 \text{ is } A_{2j}(m_{2j},\sigma_{2j})\ldots\text{and}\\&\quad x_n \text{ is } A_{nj}(m_{nj},\sigma_{nj}) \text{ THEN } y' = w_{0j} + w_{1j}x_1 + \cdots + w_{nj}x_n.\end{aligned} \tag{2}$$

Since the consequence of a rule is crisp, the defuzzification step becomes obsolete in the TSK inference scheme. Instead, the control output is computed as the weighted average of the crisp rule outputs, which is computationally less expensive then calculating the center of gravity.

In this paper, the structure of the TNFN is shown in Fig. 1, where n and R are, respectively, the number of input dimensions and the number of rules. It is a five-layer network structure. The functions of the nodes in each layer are described as follows:

Layer 1 (*Input node*): No function is performed in this layer. The node only transmits input values to layer 2.

$$u_i^{(1)} = x_i, \quad i = 1\ldots n. \tag{3}$$

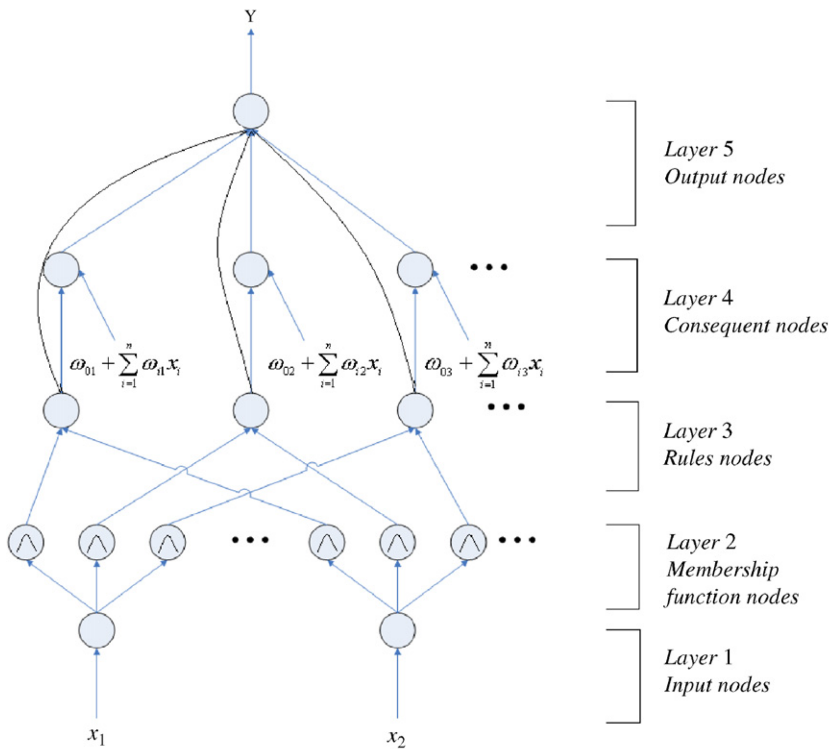

Y
Layer 5
Output nodes
Layer 4
Consequent nodes
$\omega_{01}+\sum_{i=1}^{n}\omega_{i1}x_i$
$\omega_{02}+\sum_{i=1}^{n}\omega_{i2}x_i$
$\omega_{03}+\sum_{i=1}^{n}\omega_{i3}x_i$
Layer 3
Rules nodes
Layer 2
Membership function nodes
Layer 1
Input nodes
x_1
x_2

Fig. 1. Structure of the TNFN model.

Layer 2 (*Membership function node*): Nodes in this layer correspond to one linguistic label of the input variables in layer 1; that is, the membership value specifying the degree to which an input value belongs to a fuzzy set is calculated in this layer. For an external input x_i, the following Gaussian membership function is used:

$$u_{ij}^{(2)} = \exp\left(-\frac{[u_i^{(1)} - m_{ij}]^2}{\sigma_{ij}^2}\right), \quad j = 1 \ldots R, \tag{4}$$

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

Layer 3 (*Rule node*): The output of each node in this layer is determined by the fuzzy AND operation. Here, the product operation is utilized to determine the firing strength of each rule. The function of each rule is

$$u_j^{(3)} = \prod_i u_{ij}^{(2)}. \tag{5}$$

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

$$u_j^{(4)} = u_j^{(3)}\left(w_{0j} + \sum_{i=1}^{n} w_{ij}x_i\right), \tag{6}$$

where the summation is over all the inputs and w_{ij} are the corresponding parameters of the consequent part. The w_{ij} is any real value. If $w_{ij} = 0$, $I > 0$, the TNFN model in this case will be called the zero-order TNFN model.

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

$$y = u^{(5)} = \frac{\sum_{j=1}^{R} u_j^{(4)}}{\sum_{j=1}^{R} u_j^{(3)}} = \frac{\sum_{j=1}^{R} u_j^{(3)}(w_{0j} + \sum_{i=1}^{n} w_{ij}x_i)}{\sum_{j=1}^{R} u_j^{(3)}}. \tag{7}$$

3. An overview of PSO

PSO is a recently invented high performance optimizer that possesses several highly desirable attributes, including the fact that the basic algorithm is very easy to understand and implement. It is similar to GAs and evolutionary algorithms, but requires less computational memory and fewer lines of code. The PSO conducts search using a population of particles which correspond to individuals in GA. Each particle has a velocity vector $\vec{v}_i$ and a position vector $\vec{x}_i$ to represent a possible solution.

Consider an optimization problem that requires the simultaneous optimization of variables. A collection or swarm of particles are defined, where each particle is assigned a random position in the N-dimensional problem space so that each particle's position corresponds to a candidate solution to the optimization problem. Then the particles fly rapidly over and search the space with the moving velocity of each particle. The PSO has a simple rule. Each particle has three choices in evolution: (1) Insist on oneself. (2) Move towards the optimum itself at present. Each particle remembers its own personal best position that it has ever found, called the local best. (3) Move towards the best solution of the population has met. Each particle also knows the best position found by any particle in the swarm, called the global best. The PSO reaches a balance among these three choices.

At each time step, each of these particle positions is scored to obtain a fitness value based on how well it solves the problem. Using the local best position (*Lbest*) and the global best position (*Gbest*), a new velocity for each particle is updated by

$$\vec{v}_i(k+1) = \omega * \vec{v}_i(k) + \phi_1 * rand() * (Lbest - \vec{x}_i(k)) + \phi_2 * rand() * (Gbest - \vec{x}_i(k)), \tag{8}$$

where ω, ϕ_1 and ϕ_2 are called the coefficient of inertia, cognitive and society study respectively. The *rand*() is uniformly distributed random numbers in [0,1]. The term $\vec{v}_i$ is limited to the range $\pm\vec{v}_{\max}$. If the velocity violates this limit, it will be set at its proper limit. The concept of the updated velocity is illustrated in Fig. 2.

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

$$\vec{x}_i(k+1) = \vec{x}_i(k) + \vec{v}_i(k+1). \tag{9}$$

When every particle is updated, the fitness value of each particle is calculated again. If the fitness value of the new particle is higher than those of local best, then the local best will be replaced with the new particle. If the fitness value of the new particle is higher than those of global best, then the global best will be also replaced with the new particle. We repeat above updating process step by step, the whole population evolves toward the optimum solution. The detailed flowchart is shown in Fig. 3.

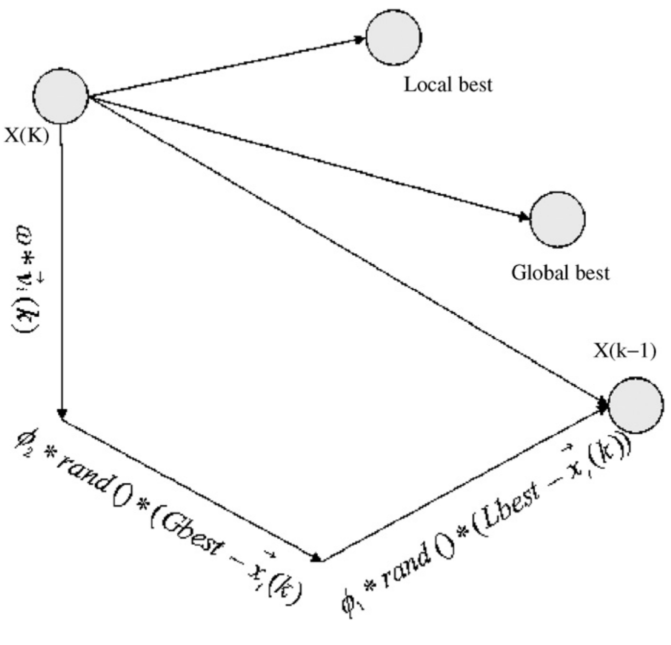

Local best
X(K)
Global best
X(k−1)
$\omega * \vec{v}_i(k)$
$\phi_2 * rand() * (Gbest - \vec{x}_i(k)$
$\phi_1 * rand() * (Lbest - \vec{x}_i(k))$

Fig. 2. The diagram of the updated velocity in the PSO.

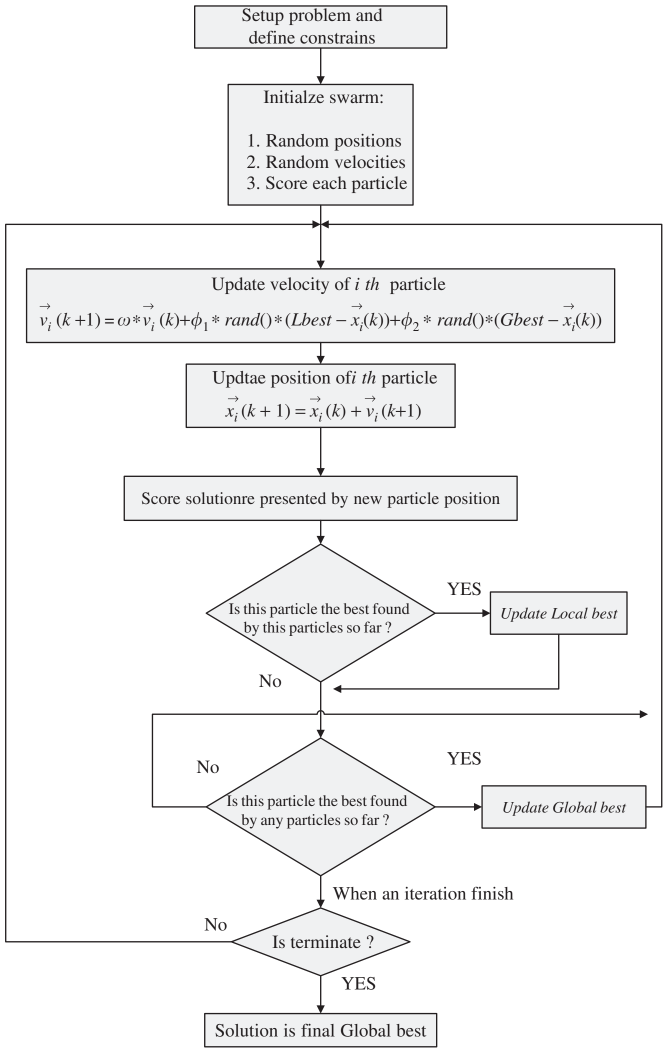

Setup problem and define constrains
Initialze swarm:
1. Random positions
2. Random velocities
3. Score each particle
Update velocity of i th particle
$\vec{v}_i(k+1) = \omega * \vec{v}_i(k) + \phi_1 * rand() * (Lbest - \vec{x}_i(k)) + \phi_2 * rand() * (Gbest - \vec{x}_i(k))$
Updtae position ofi th particle
$\vec{x}_i(k+1) = \vec{x}_i(k) + \vec{v}_i(k+1)$
Score solutionre presented by new particle position
Is this particle the best found by this particles so far ?
YES
Update Local best
No
No
YES
Is this particle the best found by any particles so far ?
Update Global best
When an iteration finish
No
Is terminate ?
YES
Solution is final Global best

Fig. 3. PSO typical flowchart illustrates the steps and update equations.

4. A novel hybrid learning algorithm

In recently years, many researchers [12,13] have presented some hybrid methods for solving various problems. Juang [12] propose a hybrid of the GA and the PSO for recurrent network design. Kumar et al. [13] use the GA method to construct TSK fuzzy rules. In this paper, we propose a novel hybrid learning algorithm. The hybrid learning algorithm consists of the FEC, the MPSO and the RSVD. The structure learning depends on the FEC to obtain the number of fuzzy rules and the initial values of the antecedent parameters. Meanwhile, the parameter learning is based on the MPSO and the RSVD for adjusting the shape of the membership functions and the consequent parameters of fuzzy rules.

The proposed hybrid method improves the convergence speed and determines a more suitable solution for various problems. The flowchart of the hybrid learning algorithm is

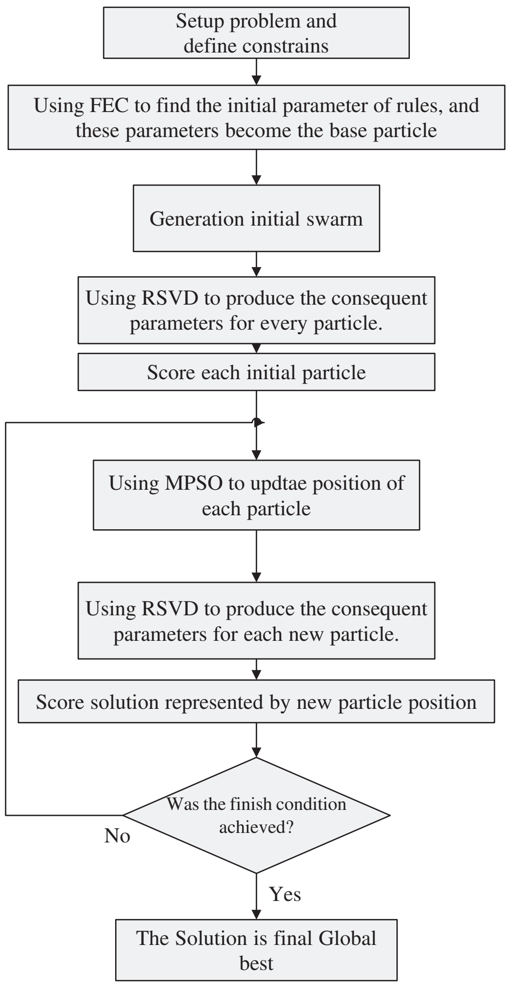

Setup problem and define constrains
Using FEC to find the initial parameter of rules, and these parameters become the base particle
Generation initial swarm
Using RSVD to produce the consequent parameters for every particle.
Score each initial particle
Using MPSO to updtae position of each particle
Using RSVD to produce the consequent parameters for each new particle.
Score solution represented by new particle position
Was the finish condition achieved?
No
Yes
The Solution is final Global best

Fig. 4. The simple flowchart of the novel learning algorithm.

shown in Fig. 4 and the detailed process is described as follows.

4.1. The FEC

The FEC is proposed by Tran and Wagner [26]. In this paper, we use FEC to partition the training data into a set of cluster, and then each fuzzy rule is extracted from each cluster to form a fuzzy base.

For clustering, the most important requirement is to find a suitable measure of clusters and objective function methods are allowed the most precise formulation of clustering criterion. Let's consider the following function:

$$H_n(U,\theta;X)=\sum_{i=1}^{C}\sum_{j=1}^{N}u_{ij}d(x_j,\theta_i)+n\sum_{i=1}^{C}\sum_{j=1}^{N}u_{ij}\log u_{ij}, \tag{10}$$

where C is the number of clusters, N is the number of data, $n>0$ is a weight factor, $u_{it}\in[0,1]$ is the membership value of vector x_{ij} in cluster i, θ_i is a prototype of cluster i, and $d(x_j,\theta_i)$ is defined the distance between x_j and θ_i. The number of classes (C) is set in advance. In each experiment, the C is determined by trial and error. We assume that the matrices U satisfy the following conditions:

$$\sum_{i=1}^{C}u_{ij}=1\ \forall t \quad \text{and} \quad 0<\sum_{i=1}^{C}u_{ij}<N \tag{11}$$

which mean that each x_t belongs to C clusters, no cluster is empty when $2\leqslant C<N$. The second term in Eq. (10) is the negative of the following function $E(U)$ multiplied by n:

$$E(U)=-\sum_{i=1}^{C}\sum_{j=1}^{N}u_{ij}\log u_{ij}. \tag{12}$$

The fuzzy entropy function $E(U)$ expresses the average degree of non-membership of members in a fuzzy set. The function $E(U)$ is maximum when $u_{ij}=1/C\ \forall i$, and minimum when $u_{ij}=0$ or 1. On the other hand, the first term in Eq. (10) also needs to be minimized and obtain a good partition for X. Therefore, we minimize Lagrangian form $H_n^*(U,\theta;X)$

$$H_n^*(U,\theta;X)=\sum_{i=1}^{C}\sum_{j=1}^{N}u_{ij}d(x_j,\theta_i)+n\sum_{i=1}^{C}\sum_{j=1}^{N}u_{ij}\log u_{ij}+\sum_{i=1}^{C}k_i\left(\sum_{j=1}^{N}u_{ij}-1\right). \tag{13}$$

Then, we obtain following function:

$$\bar{u}_{ij}=\left\{\sum_{k=1}^{C}[\mathrm{e}^{d^2(x_j,\theta_i)}/\mathrm{e}^{d^2(x_j,\theta_k)}]^{1/n}\right\}^{-1}. \tag{14}$$

The fuzzy mean vector is formed

$$\bar{u}_i=\sum_{j=1}^{N}\bar{u}_{ij}x_j\Big/\sum_{j=1}^{N}\bar{u}_{ij}. \tag{15}$$

The weight factor $n>0$ is called the degree of fuzzy entropy, which determines the partition of X. The FEC has following characteristics: (a) If n approximates infinity, then u_{ij} approximates $1/C$. On the other hand, each feature vector is equally assigned to C clusters, so we have only one single cluster. (b) When n approaches zero, the u_{ij} approximates zero or one.

Moreover, the two well-known cluster validity criteria: the partition coefficient V_{PC} and the partition entropy V_{PE} are used for FEC. The V_{PC} is an increasing function and the V_{PE} is a decreasing function. A suitable value of n in Eq. (14) will be chosen when $V_{\mathrm{PE}}=V_{\mathrm{PC}}$ [26].

4.2. The MPSO

Two strategies, called the ELAM and the MES, are proposed for improving the performance of the traditional PSO. For the traditional PSO and GA, the initial swarm and population are randomly generated; hence that need a very long time to converge. For this reason, we wish that a swarm consists of some approximation optimization solutions initially. Because all solutions in swarm approximate optimization solutions, the search space will be focused in the beginning swarm generated. Therefore, the learning speed will be improved in the beginning learning.

An ELAM is proposed to obtain a better initial swarm. First, we set the base particle using the initial parameters of fuzzy rules which are generated by the FEC, and the fitness value of particle is calculated. The structure of a particle is shown in Fig. 5. The parameters m_{ij} and σ_{ij} represent a Gaussian membership function with mean and deviation with ith dimension and jth rule node. The deviation is $x_{ij(\text{Max})} - x_{ij(\text{Min})}, \mu_{ij}(x_{ij}) \in [0.4\ \ 0.6]$.

We add a small perturbation to each position of base particle to generate new particles. The fitness values of new particles are calculated. If the fitness values of the new particles are higher than the fitness value of the base particle, then the new particles will be reserved in the swarm. This process is repeated until all particles of the swarm are produced. When the swarm is generated initially, all particles are around the base particle and the cluster (i.e., fuzzy rule) is near approximation optimization solution. The pseudocode about the ELAM is shown in Fig. 6. The L and Δ_l represent the length of a particle and a very small random value. In this paper, the region of Δ_l is the 10% domain space of every position.

Recently, many researches [3,5] focus on global solution validation measurement in evolution computation. In [3,5], the elite's selection is to estimate many factors, not only single information. The concept is introduced to the PSO, it can prevent that the swarm tends to the global best too early in the searching process. Because the global best found early in the searching process may be a poor local minima. Therefore, we propose a MES for searching the global best of the PSO.

m_{11}	σ_{11}	m_{12}	σ_{12}	$\cdots$	m_{ij}	σ_{ij}	$\cdots$	m_{nJ}	σ_{nJ}

Fig. 5. The structure of a particle.

```
For$i$ =1 to $N$        % swarm size is $N$
    If $i$ =1
        Particle $_{first}$ is given from FEC.
        Measure the fitness value of Particle $_{first}$ $(F_1)$.
    Else
        Do
            For j=1 to L     % particle size is L
                Position$_j$ of Particle$_i$ = Position$_j$ of Particle $_{first}$ $\pm\Delta$.
            End
            Measure the fitness value of Particle$_i$ $(F_i)$.
        While $(F_i < F_1)$
    End
End
```

Fig. 6. The pseudocode of effective local approximation method (ELAM).

We define a growth rate δ for each particle. When the fitness value of a particle of ith iteration is higher than that of a particle of $(i-1)$th iteration, the δ will be increased. When the fitness value of a particle of ith iteration is higher than that of a particle of $(i-1)$th iteration, the growth rate will be increased. If the growth rate is always increasing in the process of evolution, it means that the direction of adjusting parameters is advantageous to find the optimal solution. Therefore, when we set the particle to be the global best, it can attract other particles toward its leading way. After the local best of every particle is determined in each generation, we move the local best which has higher fitness value than the global best into the candidate area. Then the global best will be replaced by the local best with the highest growth rate δ. Therefore, the fitness value of the new global best is always higher than the old global best. The pseudocode about MES is shown in Fig. 7.

4.3. The RSVD

After the antecedent parameters of fuzzy rules are decided by MPSO, we will determine the optimal consequent parameters of fuzzy rules. The consequent parameters of fuzzy rules can be found by singular value decomposition (SVD) when the antecedent parameters of fuzzy rules, the training inputs and outputs are given. The SVD is one of the most useful and powerful tools of numerical linear algebra and has found successful applications in various areas such as statistical analysis, image and signal processing. The traditional SVD needs very large memory space and time consumption. We adopt the RSVD [15] for adjusting the consequent parameters of fuzzy rules. RSVD only requires the decomposition of a small matrix in

```
For $i$ =1 to $G$   % total generation is $G$
    IF $i$<>G
        For $j$ =1 to $N$      % swarm size is $N$
            If (the fitness value of particle$_j$ in $i$th iteration > that of particle$_j$ in
                ( $i$-1)th iteration)
                    $\delta_j = \delta_j + 1$;
            End
            Update Local best$_j$.
                If (the fitness of Local best$_j$ > that of Global best $_{now}$)
                    Choose Local best$_j$ put into candidate area.
                End
        End
        Calculate $\delta$ of every candidate, and record the candidate of $\delta_{\max}$.
        Update the Global best to become the candidate of $\delta_{\max}$.
    Else
        Update the Global best to become the particle of highest fitness value.
    End
End
```

Fig. 7. The pseudocode of multi-elites strategy (MES)

each generation, leading to less time and space requirements.

We assumes that $(\vec{x}_k, q_k)$ be the kth training pattern, $\vec{x}_k$ is input data and q_k is desired output. By Eq. (7), the q_k is written

$$q_k = u^{(5)} = \frac{\sum_{j=1}^{R} u_j^{(4)}}{\sum_{j=1}^{R} u_j^{(3)}} = \frac{\sum_{j=1}^{R} u_j^{(3)}(w_{0j} + \sum_{i=1}^{n} w_{ij}x_i)}{\sum_{j=1}^{R} u_j^{(3)}}. \quad (16)$$

And we can rewrite to another form by Eq. (16)

$$\begin{aligned} q_k = {} & (v_{k1}\omega_{01} + v_{k1}\omega_{11}x_1 + \cdots + v_{k1}\omega_{n1}x_n) \\ & + \cdots + (v_{k2}\omega_{02} + v_{k2}\omega_{12}x_1 + \cdots + v_{k2}\omega_{n2}x_n) \\ & + \cdots + (v_{kj}\omega_{0j} + v_{kj}\omega_{1j}x_1 + \cdots + v_{kj}\omega_{nj}x_n) \\ & + \cdots + (v_{kR}\omega_{0R} + v_{kR}\omega_{1R}x_1 + \cdots + v_{kR}\omega_{nR}x_n), \end{aligned} \quad (17)$$

where

$$v_{kj} = \frac{u_j^{(3)}}{\sum_{j=1}^{R} u_j^{(3)}}.$$

For all T training patterns, they are denoted a matrix form

$$Q = [q_1 q_2 ... q_T]^{\mathrm{T}}, \quad (18)$$

$$A = \begin{bmatrix} (v_{11}, v_{11}x_{11}, v_{11}x_{12}, \ldots, v_{11}x_{1n}), (v_{12}, v_{12}x_{11}, \ldots, v_{12}x_{1n}), \ldots, (v_{1R}, \ldots, v_{1R}x_{1n}) \\ \vdots \\ (v_{t1}, v_{t1}x_{t1}, v_{t1}x_{t2}, \ldots, v_{t1}x_{tn}), (v_{t2}, v_{t2}x_{t1}, \ldots, v_{t2}x_{tn}), \ldots, (v_{tR}, \ldots, v_{tR}x_{tn}) \\ \vdots \\ (v_{T1}, v_{T1}x_{T1}, v_{T1}x_{T2}, \ldots, v_{T1}x_{Tn}), (v_{T2}, v_{T2}x_{T1}, \ldots, v_{T2}x_{Tn}), \ldots, (v_{TR}, \ldots, v_{TR}x_{Tn}) \end{bmatrix} = \begin{bmatrix} \vec{a}_1^{\mathrm{T}} \\ \vdots \\ \vec{a}_t^{\mathrm{T}} \\ \vdots \\ \vec{a}_T^{\mathrm{T}} \end{bmatrix}, \quad (19)$$

$$W = [(\omega_{01}\ \omega_{11} \cdots \omega_{n1}), (\omega_{02}\ \omega_{12} \cdots \omega_{n2}) \cdots (\omega_{0R}\ \omega_{1R} \cdots \omega_{nR})]^{\mathrm{T}}, \quad (20)$$

where Q, A, and W are matrices of $T \times 1$, $T \times ((n+1)*R)$, and $((n+1)*R) \times 1$, respectively. We minimize the following equation:

$$J(W) = \|Q - AW\| \quad (21)$$

to obtain the optimal solution W^* using the RSVD.

In RSVD, training patterns are considered one by one from first pattern until the last pattern. For each pattern, we want to find the optimal $W(t)$ such that

$$J(W(t)) = \|Q(t) - A(t)W(t)\|, \quad t = 1 \ldots T \quad (22)$$

is minimized. Note that

$$A(t) = \begin{bmatrix} \vec{a}_1^{\mathrm{T}} \\ \vec{a}_2^{\mathrm{T}} \\ \vdots \\ \vec{a}_t^{\mathrm{T}} \end{bmatrix}, \quad Q(t) = \begin{bmatrix} q_1 \\ q_2 \\ \vdots \\ q_t \end{bmatrix}. \quad (23)$$

Minimizing Eq. (22) is equivalent to minimize

$$\hat{J}(W(t)) = \|Q(t) - \Sigma'(t)V^{\mathrm{T}}(t)W(t)\|, \quad (24)$$

where $Q'(t)$, $\Sigma'(t)$, $V^{\mathrm{T}}(t)$ and $W(t)$ satisfy

$$\begin{cases} A(1) = U(1)\Sigma(1)V^{\mathrm{T}}(1), \\ \begin{bmatrix} \Sigma'(t-1)V^{\mathrm{T}}(t-1) \\ \vec{a}_t \end{bmatrix} = U(t)\Sigma(t)V^{\mathrm{T}}(t), \quad t \geqslant 2, \end{cases} \quad (25)$$

$$\Sigma(t) = \begin{bmatrix} \Sigma'(t) \\ 0 \end{bmatrix}, \quad t \geqslant 1, \quad (26)$$

$$\begin{cases} U^{\mathrm{T}}(1)Q(1) = \begin{bmatrix} Q'(1) \\ Q''(1) \end{bmatrix}, \\ U^{\mathrm{T}}(t)\begin{bmatrix} Q'(t-1) \\ q_t \end{bmatrix} = \begin{bmatrix} Q'(t) \\ Q''(t) \end{bmatrix}, \quad t \geqslant 2. \end{cases} \quad (27)$$

The $Q'(T)$, $\Sigma'(T)$, and $V^{\mathrm{T}}(T)$ are obtained by the pseudocode as shown in Fig. 8.

We want to find the optimal $W^*(T)$ which minimizes

$$\hat{J}(Y(T)) = \|Q'(T) - \Sigma'(T)V^{\mathrm{T}}(T)W(T)\|,$$

where

$$Y(T) = V^{\mathrm{T}}(T)W(T). \quad (28)$$

Eq. (28) is minimized by $Y^*(N)$ such that

$$Q'(T) - \Sigma'(T)Y^*(T) = 0 \quad (29)$$

```
For $t$ =1 to $T$
    If t =1
        Calculate U(1), Σ(1), and V(1) by Eq.(25.1);
        get Σ'(1), and $Q'(1)$ by Eq.(26) (27.1);
    Else
        Calculate U($t$), Σ($t$),and V($t$) by Eq.(25.2);
        get Σ'($t$), and $Q'(t)$ by Eq.(26) (27.2);
    End
End
```

Fig. 8. The pseudocode of the RSVD about getting $Q'(T)$, $\Sigma'(T)$, and $V^{\mathrm{T}}(T)$.

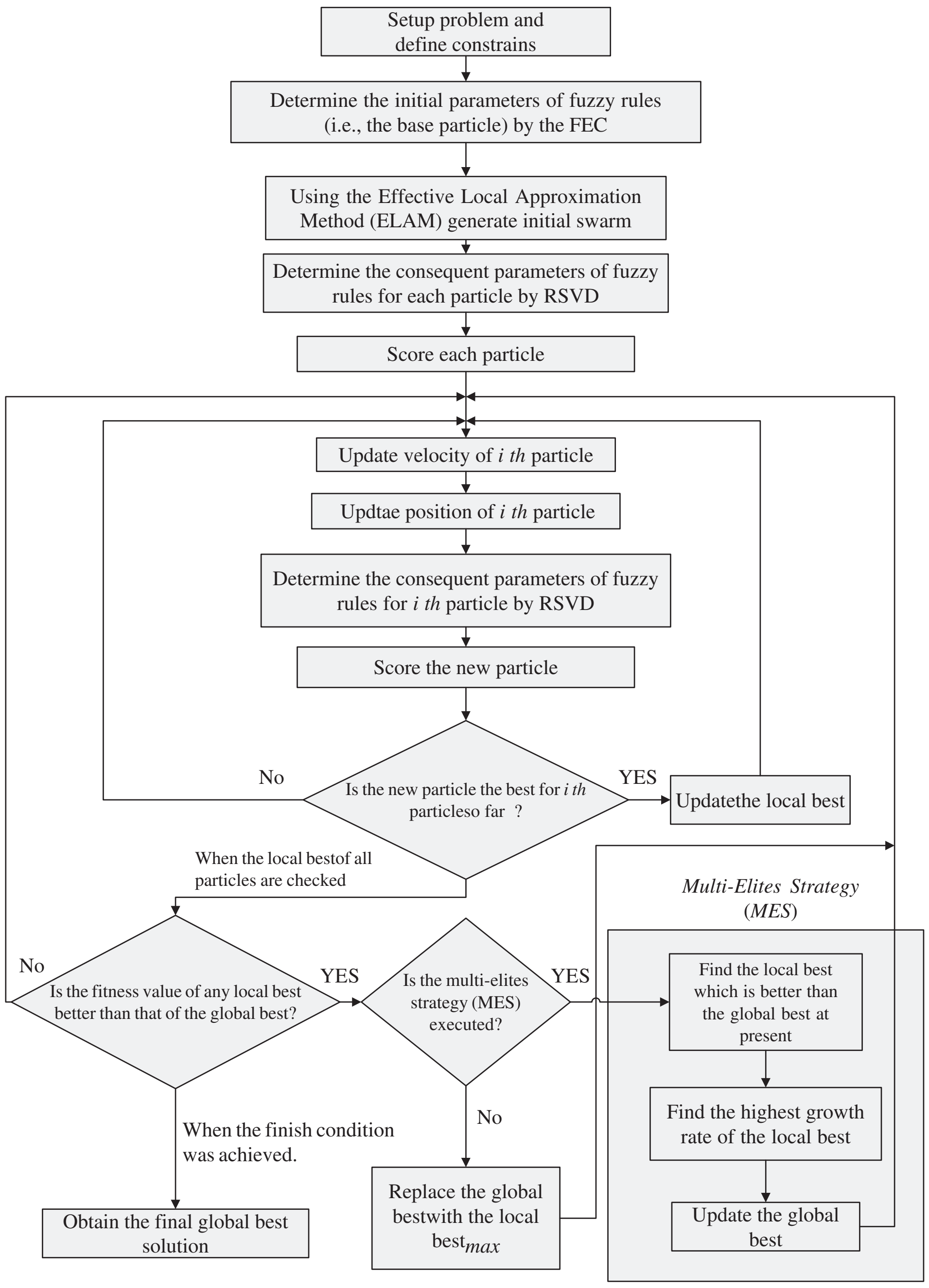

Setup problem and define constrains
Determine the initial parameters of fuzzy rules (i.e., the base particle) by the FEC
Using the Effective Local Approximation Method (ELAM) generate initial swarm
Determine the consequent parameters of fuzzy rules for each particle by RSVD
Score each particle
Update velocity of i th particle
Updtae position of i th particle
Determine the consequent parameters of fuzzy rules for i th particle by RSVD
Score the new particle
Is the new particle the best for i th particleso far ?
No
YES
Updatethe local best
When the local bestof all particles are checked
Multi-Elites Strategy (MES)
No
Is the fitness value of any local best better than that of the global best?
YES
Is the multi-elites strategy (MES) executed?
YES
Find the local best which is better than the global best at present
Find the highest growth rate of the local best
Update the global best
No
When the finish condition was achieved.
Replace the global bestwith the local best_max
Obtain the final global best solution

Fig. 9. The follow chart of hybrid learning algorithm.

Suppose $\Sigma'(T)$ we got is a $h' \times ((n+1)*R)$ diagonal matrix with each component $\Sigma'(T)_{ij}$ being

$$\Sigma'(T)_{ij} = \begin{cases} 0 & \text{if } i \neq j, \\ e'_i & \text{otherwise,} \end{cases} \tag{30}$$

where $h' \leqslant ((n+1)*R)$. The $Q'(T)$ and $Y^*(T)$ are represented by

$$Q'(T) = [q'_1\ q'_2\ \dots\ q'_{h'}]^{\mathrm{T}},$$
$$Y^*(T) = [y^*_1\ y^*_2\ \dots\ y^*_{(n+1)*R}]^{\mathrm{T}}. \tag{31}$$

Then, we obtain

$$y^*_i = \begin{cases} \dfrac{q_i}{e'_i} & \text{if } i \leqslant h', \\ 0 & \text{if } h' < i \leqslant ((n+1)*R). \end{cases} \tag{32}$$

The optimal solution $W^*(T)$ is

$$W^*(T) = V(T)Y^*(T). \tag{33}$$

When the parameters of consequent are decided, the particle can be scored to obtain a fitness value for solving problem. The process of the novel learning algorithm is repeated until finish condition is achieved. The complete follow chart is shown in Fig. 9.

5. Experimental results

In this section, the novel hybrid learning algorithm is applied to TNFN design and compare with the GA [16] and the PSO [14] and the combination of the FEC and the MPSO (called FECMPSO). The three methods (i.e., GA [16], PSO [14], and FECMPSO) are used to adjust the antecedent and consequent parameters of fuzzy rules in TNFN.

We use three different simulations for all methods. The first simulation uses the example given by Narendra and Parthasarathy [22]. The second simulation predicts the chaotic time series [4], and the third example approximates a piecewise function [30]. In our simulations, the population size is set to 50, the parameters ω, ϕ_1 and ϕ_2 of the PSO and the MPSO are set to 0.4, 2, and 2, and the crossover and mutation probabilities of the GA are set to 0.5 and 0.3, respectively. All the programs were developed using MATLAB 6.1 software and each problem was simulated on a Pentium III 1 GHz desktop computer.

Example 1. (*Identification of nonlinear dynamic system*) The first example used for identification is described by the

Table 1
The performance comparison with various existing methods

	GA method	PSO method	FECMPSO method	Hybrid learning algorithm
RMSE (ave.)	0.085	0.023	0.019	0.000336
RMSE (best)	0.018	0.011	0.008	0.000103

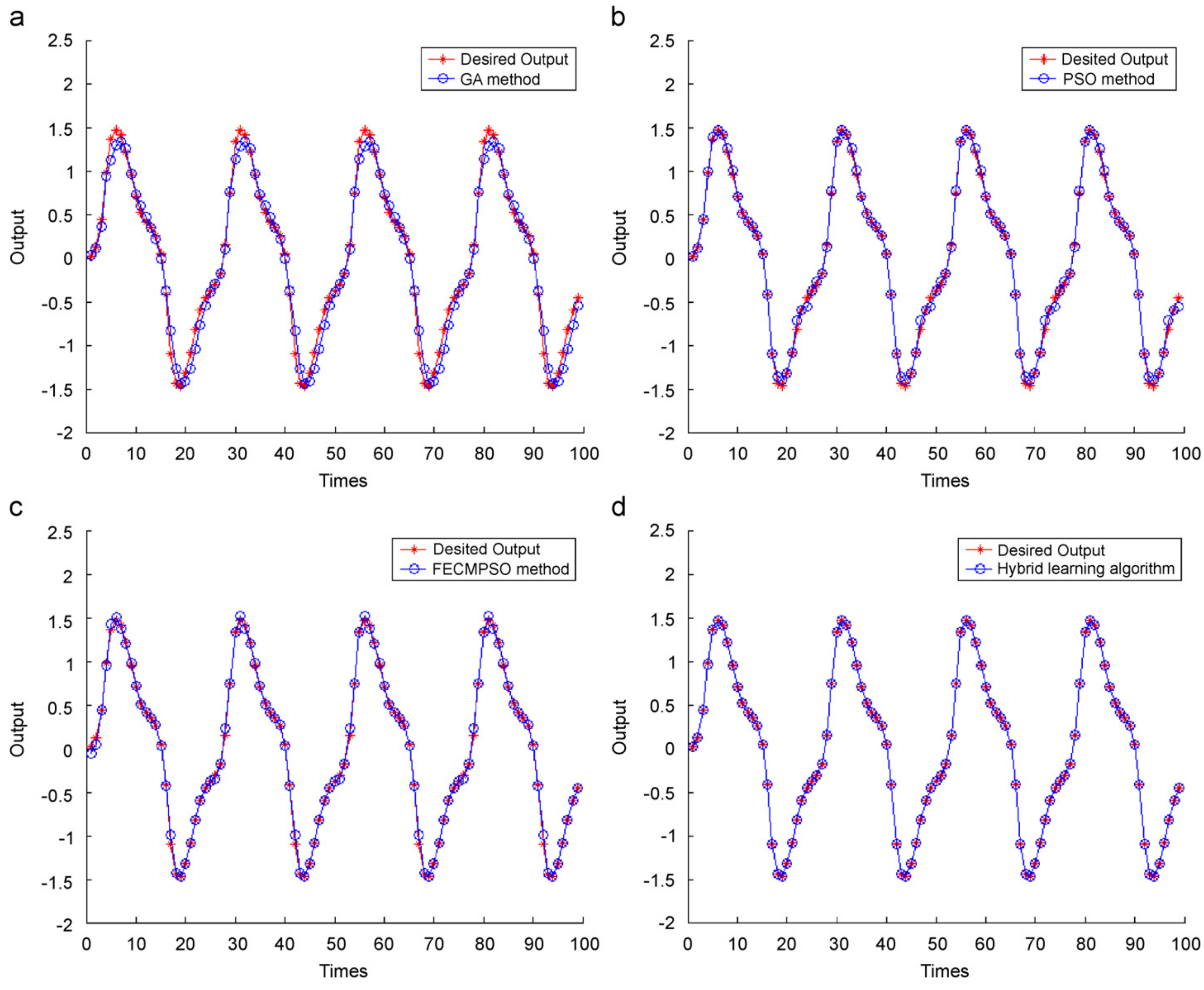


Fig. 10. Results of the desired output and (a) GA, (b) PSO, (b) FECMPSO and (d) the hybrid learning algorithm.

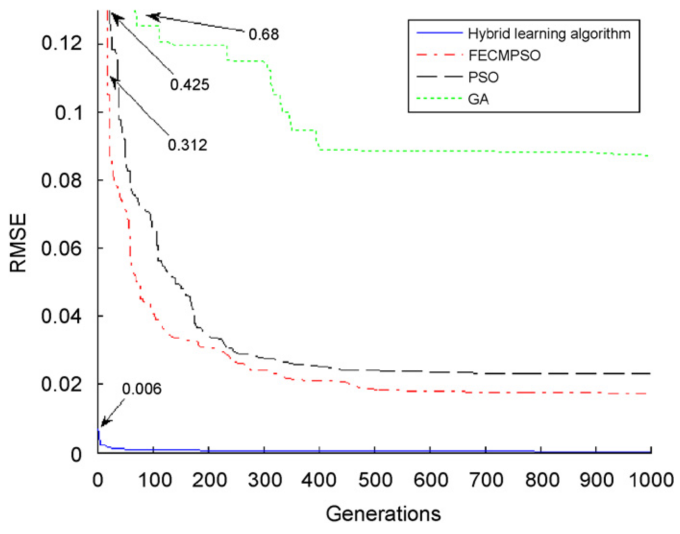


Fig. 11. The learning curves of the hybrid learning algorithm, GA, PSO and MPSO.

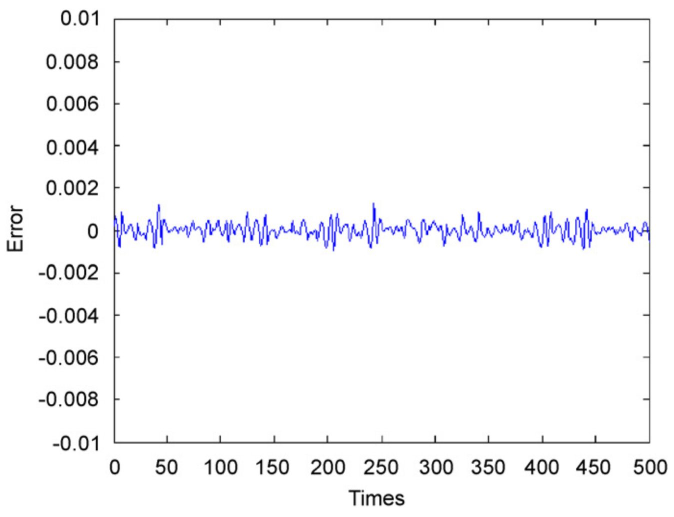


Fig. 13. The prediction errors of the hybrid learning algorithm.

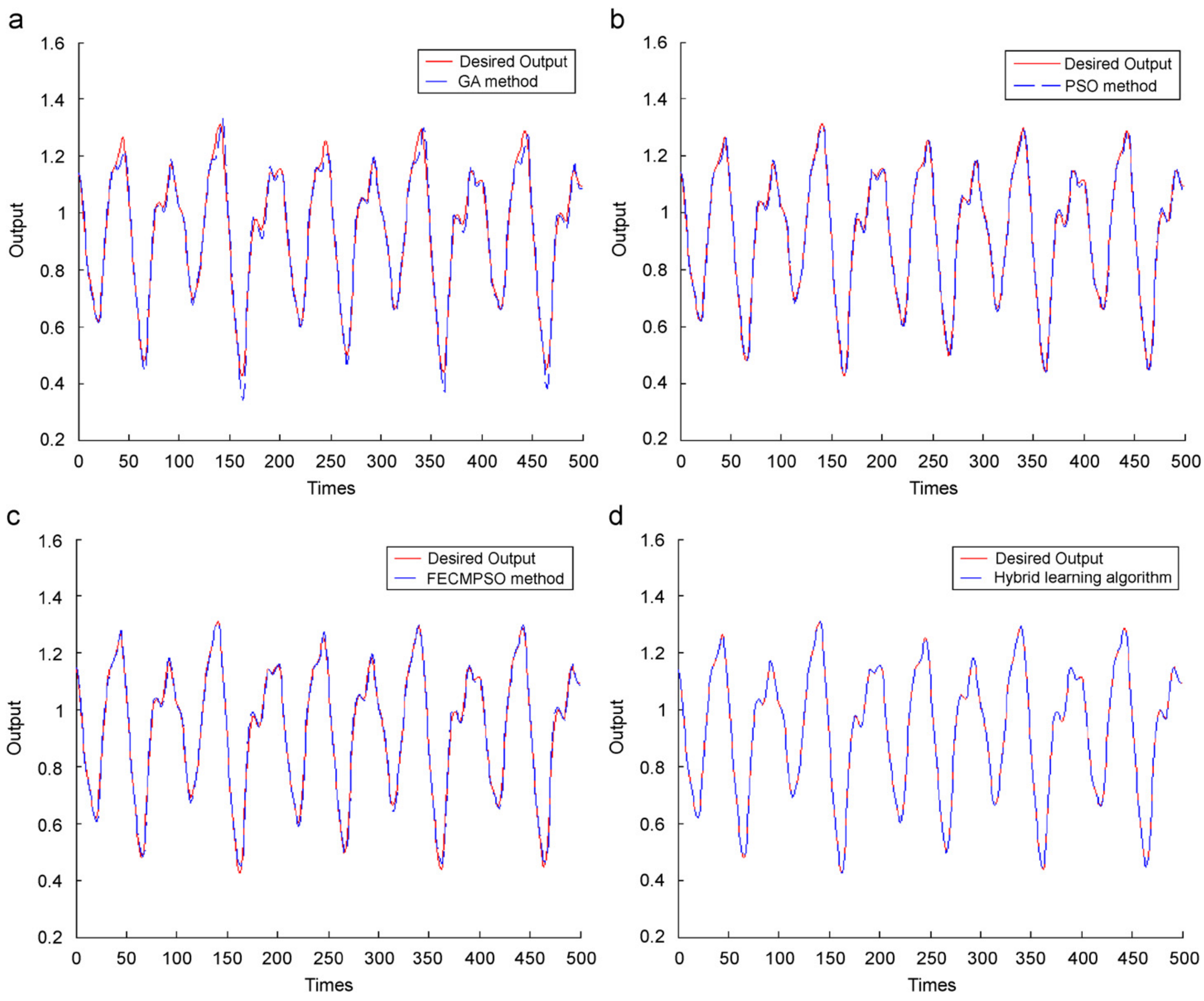


Fig. 12. The prediction results of (a) the GA method, (b) the PSO, (c) the MPSO and (d) the hybrid learning algorithm.

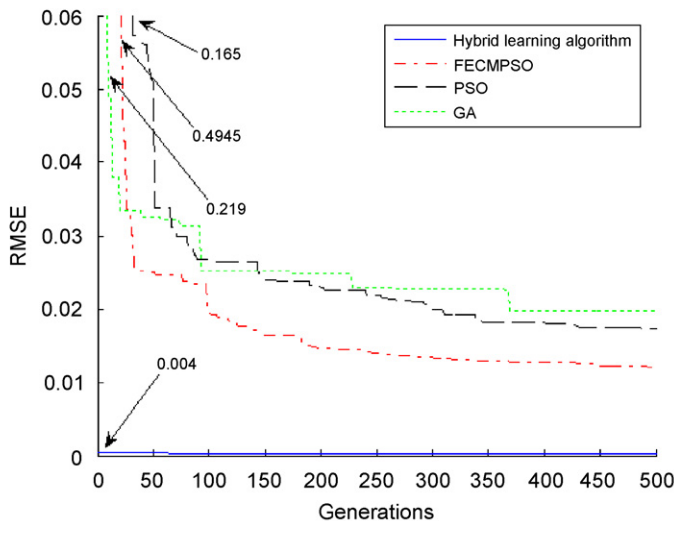


Fig. 14. The learning curves of the GA method, the PSO, the MPSO and proposed hybrid learning algorithm for prediction problem.

Table 2
The performance comparison with various existing methods

	GA method	PSO method	FECMPSO method	Hybrid learning algorithm
RMSE (ave.)	0.019	0.012	0.010	0.000243
RMSE (best)	0.011	0.006	0.006	0.000234

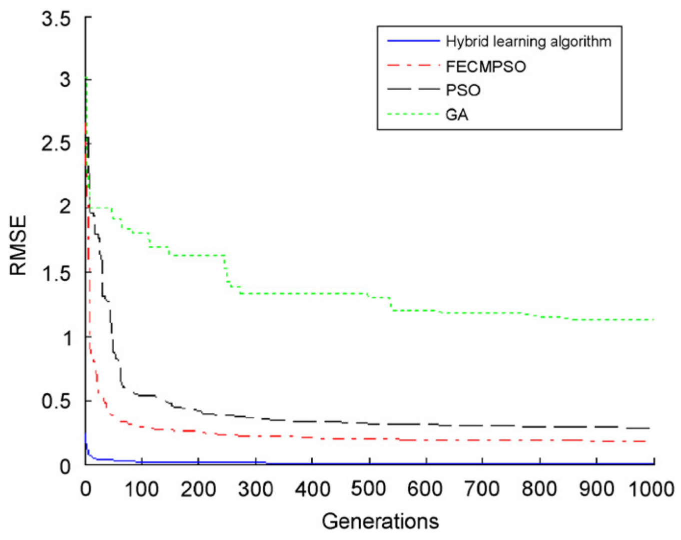


Fig. 15. The learning curves of the GA, the PSO, the MPSO, and the hybrid learning algorithm for piecewise problem.

difference equation:

$$y(k+1)=\frac{y(k)}{1+y^2(k)}+u^3(k). \tag{34}$$

The output of this equation depends nonlinearly on both its past value and the input, but the effects of the input and output values do not additive. The 100 training input patterns are randomly generated in the interval [−2,2], and the 100 testing input patterns are also generated by the same method. Evolution progressed for 1000 generations and repeated 50 trials. In each trial, we used a set of different initial swarms. To show the effectiveness and efficiency of the proposed hybrid learning algorithm, a TNFN using the GA, PSO and MPSO methods are applied to the same problem. After using the FEC for performing input space partition, we obtain five fuzzy rules. The final fuzzy rules of the TFM using the hybrid learning algorithm are described as follows:

Rule 1: *IF* x_1 *is* $A_{11}(1.1818, 2.0947)$ *and* x_2 *is* $A_{21}(1.0165, 0.9485)$
THEN $y' = -8.5416 + 2.5689x_1 - 0.7287x_2$
Rule 2: *IF* x_1 *is* $A_{12}(-1.1893, 0.8832)$ *and* x_2 *is* $A_{22}(0.5600, 1.9776)$
THEN $y' = -0.2095 - 0.5733x_1 - 1.7868x_2$
Rule 3: *IF* x_1 *is* $A_{13}(-0.1193, 2.4693)$ *and* x_2 *is* $A_{23}(0.4798, 1.7983)$
THEN $y' = -0.8614 - 1.8437x_1 + 16.4487x_2$
Rule 4: *IF* x_1 *is* $A_{14}(-0.2011, 2.1739)$ *and* x_2 *is* $A_{24}(-0.8542, 1.2091)$
THEN $y' = 17.1529 - 1.8978x_1 + 1.9976x_2$
Rule 5: *IF* x_1 *is* $A_{15}(-1.5298, 1.6492)$ *and* x_2 *is* $A_{25}(-1.7533, 12.7969)$
THEN $y' = -18.1112 - 4.9666x_1 + 1.2256x_2$

After 1000 generations, the final average best root mean square error (RMSE) of the output approximated 0.000336. Fig. 10(a)–(d) show the outputs of the four methods for the input $u(k) = \sin(2\pi k/25)$. As shown in Fig. 10(a)–(d), the identification ability of the hybrid learning algorithm is better than those of the GA, PSO and FECMPSO methods. In Table 1, we observe that the average and best RMSE of the PSO are smaller than that of GA, but are still larger than that of the proposed hybrid learning algorithm. Fig. 11 shows the RMSE curves of the four methods. We can find that the hybrid learning algorithm obtains a lowest RMSE value than the others.

Example 2. (*Prediction of the chaotic time series*) The Mackey–Glass chaotic time series $x(t)$ in consideration here is generated from the following delay differential equation:

$$\frac{dx(t)}{dt}=\frac{0.2x(t-\tau)}{1+x^{10}(t-\tau)}-0.1x(t). \tag{35}$$

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

$$[x(t-18), x(t-12), x(t-6), x(t); x(t+6)], \tag{36}$$

where $\tau = 17$ and $x(0) = 1.2$. There are four inputs to model, corresponding to these values of $x(t)$, and one output representing the value $x(t+\Delta t)$, where Δt is a time prediction into the future. The first 500 pairs (from $x(1)$ to $x(500)$) are the training data set, while the remaining 500 pairs (from $x(501)$ to $x(1000)$) are the testing data set used

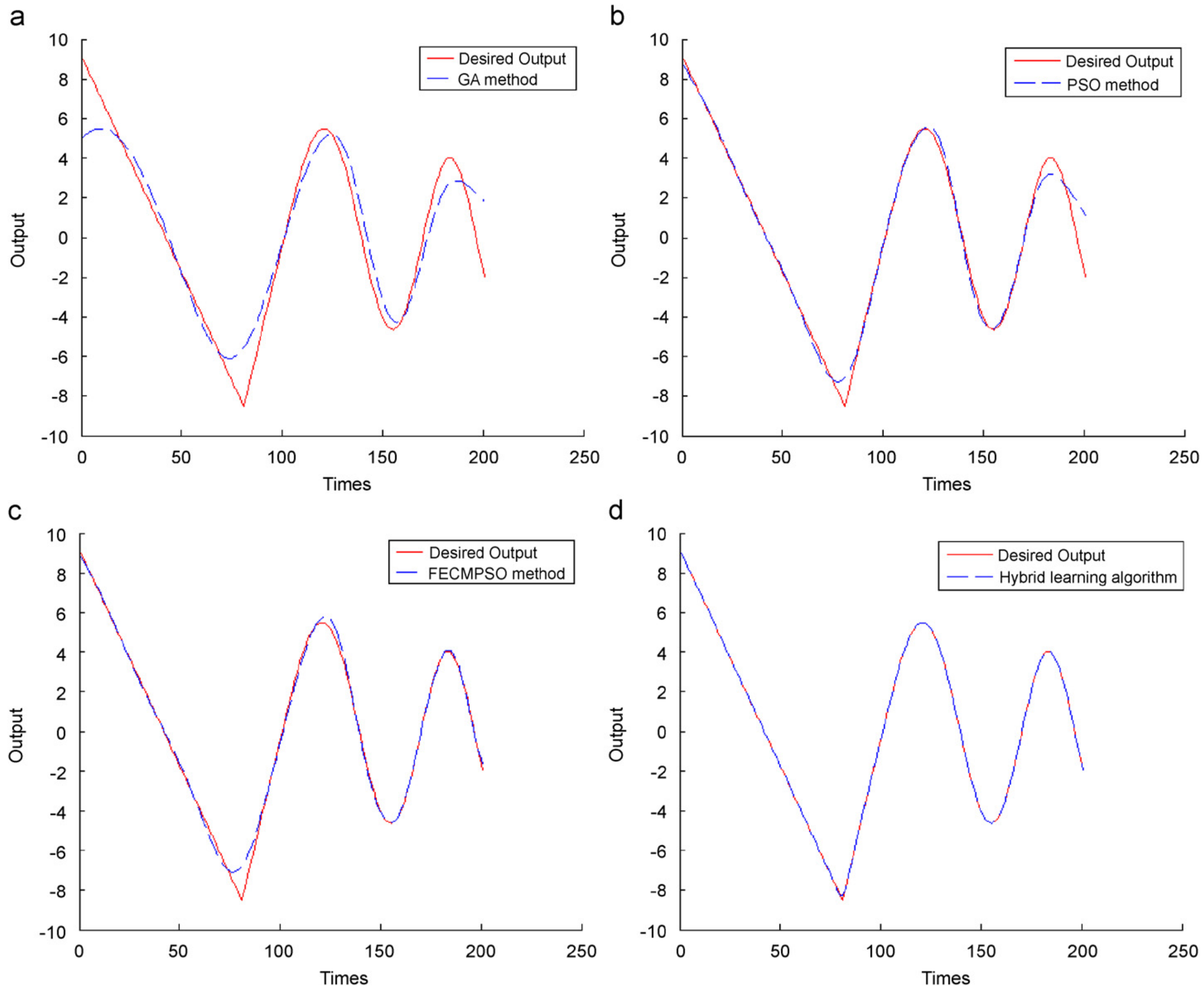


Fig. 16. The results of approximation using (a) the GA method, (b) the PSO method, (c) the MPSO method, and (d) the hybrid learning algorithm.

Table 3
The performance comparison with various existing methods

	GA method	PSO method	FECMPSO method	Hybrid learning algorithm
RMSE (ave.)	1.13	0.28	0.18	0.006
RMSE (best)	0.72	0.12	0.11	0.002

for validating the proposed method. After using the FEC for performing input space partition, we obtain six fuzzy rules. The average best RMSE of the prediction output approximates 0.000243 after 500 generations.

In this example, we compared the proposed hybrid learning algorithm with those of other methods. Fig. 12(a)–(d) show the prediction results of the GA, PSO, FECMPSO and the hybrid learning algorithm. We observe that the prediction result of the hybrid learning algorithm is better than those of other methods. The prediction errors of the proposed hybrid learning algorithm are shown in Fig. 13. Fig. 14 shows the RMSE curves of the four models. In this figure, we find that the proposed method converges quickly and obtains a lower RMSE than other models.

Table 2 shows the comparison results of the prediction performance among other methods. The RMSE obtained by hybrid learning algorithm is very smaller than all methods.

Example 3. (*Approximation of the piecewise function*) The piecewise function is studied by Zhang [30] and Xu [8] and is defined as

$$f(x) = \begin{cases} -2.186x - 12.864, & -10 \leqslant x < -2, \\ 4.246x, & -2 \leqslant x < 0, \\ 10e^{-0.05x-0.5}\sin[(0.03x+0.7)x], & 0 \leqslant x \leqslant 10 \end{cases} \tag{37}$$

over the domain $D = [-10, 10]$. The piecewise function is continuous and can be analyzed. However, traditional

Table 4
The average computation time of three examples (Unit:sec)

	GA method [16]	PSO method [14]	FECMPSO method	Hybrid learning algorithm
Identification of nonlinear dynamic system	374.547 (1000 generations)	352.735 (1000 generations)	421.75 (1000 generations)	302.37 (300 generations) 1007.9 (1000 generations)
Prediction of the chaotic time series	1048.14 (500 generations)	1014.788 (500 generations)	1069.047 (500 generations)	937.45 (250 generations) 1874.9 (500 generations)
Approximation of the piecewise function	918.797 (1000 generations)	904.609 (1000 generations)	1027.032 (1000 generations)	832.34 (300 generations) 2774.46 (1000 generations)

analytical tools are inadequate and often fail. This failure may be due to two reasons, namely, the wide-band information hidden at the turning points and the amalgamation of linearity and nonlinearity.

In this example, 200 training input patterns and 200 testing input patterns are uniformly generated from Eq. (37). There are seven fuzzy rules generated by the FEC in this example. The RMSE curve is shown in Fig. 15. Fig. 16(a)–(d) show the outputs of the function f and all methods. The solid line represents the output of function f, and the dotted line represents the approximation of various methods.

The results comparing our model with other models are tabulated in Table 3. The simulation results demonstrate that the hybrid learning algorithm obtains very good approximation capability than other methods.

We find that the proposed hybrid learning algorithm has a better performance than the GA, the PSO, and the FECPSO from the three different simulations. In addition, the learning speed of the FECMPSO is faster than that of the PSO in the beginning learning process while the RMSE of the FECMPSO is lower than that of the PSO in the ending learning process.

In addition to the comparison of learning performance, the time consumed of every learning method for three examples is also shown in Table 4. We can see that the average computation time of the proposed hybrid learning algorithm is longer than the other methods, owing to the use of more complex matrix computation of the RSVD. However, our algorithm can converge quicker and obtain a better performance than the other methods for about 200–300 generations. In the three examples, the proposed method only consumes 302.37 and 832.34 s for 300 generations in the first and third examples and 937.45 s for 250 generations in the second example (see the fifth column of Table 4). Therefore, the proposed hybrid learning algorithm with the FEC, the MPSO, and the RSVD can improve the learning performance efficiently. In the future work, we will continue to improve our method by reducing its complex computation.

6. Conclusion

In this paper, a novel hybrid learning algorithm for TNFNs has developed. We use fuzzy entropy clustering (FEC) to generate base particles and propose the modified PSO (MPSO) to improve effectively the performance of the traditional PSO. The MPSO is used to find the optimal antecedent parameters of fuzzy rules. In addition, we also use the RSVD to determine the optima consequent parameters of fuzzy rules. The RSVD need fewer computation time and space requirements than the traditional SVD. In experimental results, we show that the hybrid learning algorithm has a better learning performance than the traditional GA and PSO methods.

References

[1] M.A. Abido, Optimal design of power-system stabilizers using particle swarm optimization, IEEE Trans. Energy Convers. 17 (3) (2002) 406–413.
[2] T. Bäck, H.P. Schwefel, An overview of evolutionary algorithms for parameter optimization, Evol. Comput. 1 (1) (1993) 1–23.
[3] S. Bandyopadhyay, S.K. Pal, B. Aruna, Multiobjective gas, quantitative indices, and pattern classification, IEEE Trans. Syst. Man Cybern. Part B 34 (5) (2004).
[4] R.S. Cowder III, in: D. Touretzky, G. Hinton, T. Sejnowski (Eds.), Predicting the Mackey–Glass Time Series with Cascade-Correlation Learning, 1990, pp. 117–123.
[5] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, IEEE Trans. Evol. Comput. 6 (2) (2002).
[6] D.B. Fogel, Evolutionary Computation: Toward a New Philosophy of Machine Intelligence, IEEE Press, Piscataway, NJ, 1995.
[7] Z.L. Gaing, A particle swarm optimization approach for optimum design of PID controller in AVR system, IEEE Trans. Energy Convers. 19 (2) (2004) 384–391.
[8] D.W.C. Ho, P.A. Zhang, J. Xu, Fuzzy wavelet networks for function learning, IEEE Trans. Fuzzy Syst. 9 (2001) 200–211.
[9] A. Homaifar, E. McCormick, Simultaneous design of membership functions and rule sets for fuzzy controllers using genetic algorithms, IEEE Trans. Fuzzy Syst. 3 (9) (1995) 129–139.
[10] J.-S.R. Jang, ANFIS: adaptive-network-based fuzzy inference system, IEEE Trans. Syst. Man Cybern. 23 (1993) 665–685.

[11] C.F. Juang, A TSK-type recurrent fuzzy network for dynamic systems processing by neural network and genetic algorithms, IEEE Trans. Fuzzy Syst. 10 (2) (2002) 155–170.

[12] C.F. Juang, A hybrid of genetic algorithm and particle swarm optimization for recurrent network design, IEEE Trans. Syst. Man Cybern. Part B 34 (2) (2004) 997–1006.

[13] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proceedings of the IEEE International Conference on Neural Networks, 1995, pp. 1942–1948.

[14] A. Kumar, D.P. Agrawal, S.D. Joshi, A GA-based method for constructing TSK fuzzy rules from numerical data, in: The 12th IEEE International Conference on Fuzzy Systems, vol. 1 May 2003, pp. 25–28.

[15] S.J. Lee, C.S. Ouyang, A neuro-fuzzy system modeling with self-constructing rule generation and hybrid SVD-based learning, IEEE Trans. Fuzzy Syst. 11 (3) (2003) 341–353.

[16] M.A. Lee, H. Takagi, Integrating design stages of fuzzy systems using genetic algorithms, in: Proceedings of the IEEE International Conference on Fuzzy Systems, vol. 1, New York, April 1993, pp. 612–617.

[17] C.J. Lin, C.C. Chin, Prediction and identification using wavelet-based recurrent fuzzy neural networks, IEEE Trans. Syst. Man Cybern. (Part:B) 34 (5) (2004) 2144–2154.

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

[19] C.J. Lin, C.T. Lin, An ART-based fuzzy adaptive learning control network, IEEE Trans. Fuzzy Syst. 5 (4) (1997) 477–496.

[20] F.J. Lin, C.H. Lin, P.H. Shen, Self-constructing fuzzy neural network speed controller for permanent-magnet synchronous motor drive, IEEE Trans. Fuzzy Syst. 9 (5) (2001) 751–759.

[21] R. Mendes, P. Cortez, M. Rocha, J. Neves, Particle swarms for feedforward neural network training, in: The 2002 International Joint Conference on Neural Networks, 2002, pp. 1895–1899.

[22] K.S. Narendra, K. Parthasarathy, Identification and control of dynamical systems using neural networks, IEEE Trans. Neural Networks 1 (1) (1990) 4–27.

[23] T. Takagi, M. Sugeno, Fuzzy identification of systems and its applications to modeling and control, IEEE Trans. Syst. Man Cybern. SMC-15 (1985) 116–132.

[24] H. Takagi, N. Suzuki, T. Koda, Y. Kojima, Neural networks designed on approximate reasoning architecture and their application, IEEE Trans. Neural Networks 3 (5) (1992) 752–759.

[25] G.G. Towell, J.W. Shavlik, Extracting refined rules from knowledge-based neural networks, Mach. Learn. 13 (1993) 71–101.

[26] D. Tran, M.Wagner, Fuzzy entropy clustering, in: The Ninth IEEE International Conference on Fuzzy Systems, vol. 1, 2000, pp. 152–157.

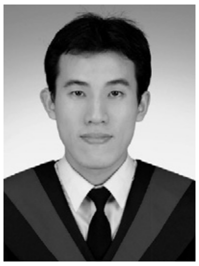
[27] L.X. Wang, J.M. Mendel, Generating fuzzy rules by learning from examples, IEEE Trans. Syst. Man Cybern. 22 (6) (1992) 1414–1427.

[28] X. Yao (Ed.), Evolutionary Computation: Theory and Applications, World Scientific, Singapore, 1999.

[29] H. Yoshida, K. Kawata, Y. Fukuyama, S. Takayama, Y. Nakanishi, A particle swarm optimization for reactive power and voltage control considering voltage security assessment, IEEE Trans. Power Syst. 15 (4) (2000) 1232–1239.

[30] Q. Zhang, A. Benveniste, Wavelet networks, IEEE Trans. Neural Networks (1992) 889–898.

Cheng-Jian Lin received the B.S. degree in electrical engineering from Ta-Tung University, Taiwan, ROC, in 1986 and the M.S. and Ph.D. degrees in electrical and control engineering from the National Chiao-Tung University, Taiwan, ROC, in 1991 and 1996. From April 1996 to July 1999, he was an Associate Professor in the Department of Electronic Engineering, Nan-Kai College, Nantou, Taiwan, ROC. Since August 1999, he has been with the Department of Computer Science and Information Engineering, Chaoyang University of Technology. Currently, he is a Professor of Computer Science and Information Engineering Department, Chaoyang University of Technology, Taichung, Taiwan, ROC. He served as the chairman of Computer Science and Information Engineering Department from 2001 to 2005. His current research interests are neural networks, fuzzy systems, pattern recognition, intelligence control, bioinformatics, and FPGA design. He has published more than 90 papers in the referred journals and conference proceedings. Dr. Lin served as the Associate Editor of International Journal of Applied Science and Engineering from 2002 to 2006. Dr. Lin is a member of the Phi Tau Phi. He is also a member of the Chinese Fuzzy Systems Association (CFSA), the Chinese Automation Association, the Taiwanese Association for Artificial Intelligence (TAAI), and the IEEE Computational Intelligence Society. He is an executive committee member of the Taiwanese Association for Artificial Intelligence (TAAI).



Shang-Jin Hong received the B.S. and M.S. degree in Computer Science and Information Engineering from Chaoyang University of Technology, Taiwan, ROC, in 2003 and 2005. His research interests are neural networks, fuzzy systems, pattern recognition, and intelligent control. He is a member of the Phi Tau Phi.