An Adaptive-PSO-Based Self-Organizing RBF Neural Network

Hong-Gui Han, *Senior Member, IEEE*, Wei Lu, Ying Hou, and Jun-Fei Qiao, *Member, IEEE*

***Abstract*— In this paper, a self-organizing radial basis function (SORBF) neural network is designed to improve both accuracy and parsimony with the aid of adaptive particle swarm optimization (APSO). In the proposed APSO algorithm, to avoid being trapped into local optimal values, a nonlinear regressive function is developed to adjust the inertia weight. Furthermore, the APSO algorithm can optimize both the network size and the parameters of an RBF neural network simultaneously. As a result, the proposed APSO-SORBF neural network can effectively generate a network model with a compact structure and high accuracy. Moreover, the analysis of convergence is given to guarantee the successful application of the APSO-SORBF neural network. Finally, multiple numerical examples are presented to illustrate the effectiveness of the proposed APSO-SORBF neural network. The results demonstrate that the proposed method is more competitive in solving nonlinear problems than some other existing SORBF neural networks.**

***Index Terms*— Adaptive particle swarm optimization (APSO), nonlinear system modeling, radial basis function neural networks (RBFNNs), self-organizing.**

I. Introduction

RADIAL basis function neural networks (RBFNNs) have been extensively used for modeling and controlling nonlinear systems due to their universal approximation ability and structural simplicity [1]–[3]. It has been proved that RBFNNs can approximate any continuous function if a sufficient number of RBF neurons are provided [4]. To achieve the required approximation accuracy, various learning algorithms for RBFNNs have been proposed to adjust the model parameters (i.e., the coordinates of RBF centers, the widths of RBF neurons, and the output weights) and to determine the network size (the number of RBF neurons) [5], [6]. A brief literature review of works on parameter adjustment and network size determination is given in the following.

Gradient-based and evolutionary methods to adjust the RBF network parameters have been proposed in [7] and [8]. Among the gradient-based methods, back-propagation (BP) algorithms are the most popular [9]. However, such approaches may take a long time to reach convergence and have very limited searching ability to find the global minimum of a cost function [10]. Compared with BP algorithms, recursive least squares (RLS) algorithms have a faster convergence speed [11]. However, the RLS algorithms involve more complicated mathematical operations and require more computational resources than the BP algorithms [12]. To avoid the above problems, Jiang and Zhang [13] proposed a variable-length sliding window blockwise least squares (VLSWBLS) algorithm, which outperforms the RLS with forgetting factors. Peng *et al.* [14] introduced a continuous forward algorithm (CFA) to optimize the parameters of RBFNNs. This CFA method can improve the performance with significantly lower memory usage and computational complexity. Qiao and Han [15] proposed a forward-only computation (FOC) algorithm to adjust the parameters rather than employing the traditional forward and backward computation. The FOC algorithm simplified the training process, resulting in lower computational complexity. Although the CFA, FOC and VLSWBLS algorithms have lower computational complexity than the conventional optimization techniques, they focus more on selecting the network size than adjusting the parameters. Recently, an improved second-order (ISO) algorithm has been proposed to train RBFNNs in [16]. Using this ISO algorithm, a high accuracy and a fast training speed can be obtained by increasing the dimensions of variables during the training process. However, as the network size increases, the ISO algorithm performs similar to the Levenberg–Marquardt method. Unlike the gradient-based methods, evolutionary algorithms perform robustly for training RBFNNs. A significant advantage of the evolutionary algorithms lies in their global optimization ability [17], [18]. The results in [19] and [20] show that the evolutionary algorithms can escape from local minima, in contrast to the existing gradient-based methods. The disadvantages, however, are that these methods may cause overfitting and have higher computational complexity, especially when the searching space is huge. Moreover, the analysis of the algorithm's convergence is challenging, and sometimes, there is no guarantee of convergence [21], [22].

The construction of RBFNNs is an optimization problem with both parameters and network size to be determined simultaneously [23], [24]. The design of network size, relating to the computational complexity and the generalization of RBFNNs, is an important issue [25]. To design a compact RBFNN, many strategies and algorithms have been developed to generate or prune the hidden neurons based on the instantaneous training information. For example, Huang *et al.* [26] proposed

Manuscript received December 21, 2015; revised March 18, 2016, June 17, 2016, September 15, 2016, and October 5, 2016; accepted October 8, 2016. This work was supported in part by the National Science Foundation of China under Grant 61622301, Grant 61203099, and Grant 61225016, and in part by the Beijing Municipal Education Commission Science and Technology Development Program under Grant KZ201410005002 and Grant km201410005001.

H.-G Han is with the Faculty of Information Technology and Beijing Key Laboratory of Computational Intelligence and Intelligent System, Beijing University of Technology, 100124, Beijing, China. (e-mail: Rechardhan@sina.com).

W. Lu, Y. Hou and J.-F Qiao are with the Faculty of Information Technology, Beijing University of Technology, 100124, Beijing, China.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2016.2616413

a sequential learning algorithm, known as the growing and pruning RBF (GAP-RBF) algorithm. Based on the GAP-RBF algorithm, a more advanced model termed the GGAP-RBF algorithm was suggested in [27]. Both the GAP-RBF and GGAP-RBF algorithms utilize a pruning and growing strategy that uses the "significance" of a hidden neuron and links it to the learning accuracy. The resulting structure of the RBF network is relatively compact, with less computational time. However, both the GAP-RBF and GGAP-RBF algorithms require a complete set of samples for the training process. In general, it is not possible for designers to have *a priori* knowledge of the training samples before implementation [28]. Recently, a flexible structure RBFNN (FS-RBFNN) was developed based on neuron activity and mutual information in [29]. Experimental results indicate that the FS-RBFNN algorithm could produce an RBFNN with a compact structure. Su *et al.* [30] introduced a new structure for RBFNNs that can successfully model symbolic interval-valued data using the interval competitive agglomeration clustering algorithm. Furthermore, a sequential projection-based metacognitive learning algorithm for an RBF network (PBL-McRBFN) was proposed in [31]. This PBL-McRBFN algorithm was inspired by human metacognitive learning principles and the past knowledge of samples. In addition, readers may find some more information and discussion on self-organizing RBF (SORBF) neural networks in [32] and [33]. Notice that most of these SORBF neural networks cannot handle the parameter adjustment problem under dynamic network structures [16].

To optimize the parameters and the network size of an RBFNN simultaneously, particle swarm optimization (PSO) has been widely used, because it can find solutions much faster than other evolutionary algorithms, such as the genetic algorithm, differential evolution, and so on [34]. For example, Feng [35] proposed an SORBF neural network based on the PSO algorithm to speed up the learning process. Alexandridis *et al.* [36] developed a novel algorithm for training RBFNNs using fuzzy means and PSO algorithms. The results show that the proposed SORBF neural network exhibits a higher prediction accuracy and smaller network structure with low computational complexity. Moreover, a nonlinear time-varying evolutionary PSO (NTVE-PSO) algorithm was proposed for constructing RBFNNs in [37]. This NTVE-PSO algorithm was adopted to determine the optimal parameters and network size of an RBFNN simultaneously for time series prediction problems. The simulation results illustrate that the NTVE-PSO-based RBFNN has a better performance in terms of both the forecast accuracy and the computational efficiency than the other PSO-based RBFNNs [37]. Similar studies on this topic can be found in [38]–[40]. While applying PSO to RBF model optimization, most of the existing works separate the evaluation criteria from the behavior of parameter adjustment and network size determination. Therefore, how to build a unified criterion for SORBF neural networks remains an open problem for further research [41], [42]. In addition to these issues related to algorithm development and implementation, the theoretical analysis of algorithm convergence seems quite challenging [43], [44].

In this paper, a novel self-organizing method based on an adaptive PSO (APSO) algorithm is developed to construct RBFNNs with improved performance. The advantages of the proposed APSO-based SORBF (APSO-SORBF) neural network are twofold. First, a nonlinear regressive function is employed to enhance the quality of the results and the searching speed. Second, the proposed APSO algorithm is used to optimize both the parameters and the size of the RBFNNs. A convergence analysis of the APSO-SORBF neural network is given, and the effectiveness is verified by simulations.

The main contributions of this paper are summarized as follows.

1) An APSO algorithm is developed. One of the key factors of APSO is the generation of a high-quality solution by employing a nonlinear regressive function. Then, this APSO algorithm, which can adjust the inertia weights to enhance the searching ability of the particles, has a higher solution accuracy than conventional PSO algorithms.
2) To optimize both the parameters and the network size simultaneously, a unified criterion is developed to self-organize RBFNNs. The main objective of the proposed APSO-SORBF neural network is to achieve a higher solution accuracy with a compact network size. The illustrative results confirm its superiority to other SORBF neural networks.
3) Since the convergence of the neural networks is important for successful application, the convergence of the APSO-SORBF neural network has been proved and demonstrated theoretically and experimentally.

The remainder of this paper is organized as follows. Section II briefly introduces the RBFNN and the PSO algorithm. Section III details the APSO algorithm and the APSO-SORBF neural network. Section IV gives an analysis on the convergence for the learning algorithm of building APSO-SORBF neural networks. Section V presents our simulation results, and we conclude this paper in Section VI.

II. Problem Formulation and Preliminaries

A. RBF Neural Networks

RBFNNs are motivated by the locally tuned responses in biologic neurons. Generally speaking, an RBFNN consists of three layers: the input layer, the hidden layer, and the output layer. For example, an RBFNN with multiple-input single-output is shown in Fig. 1. The inputs to the hidden layer are the linear combinations of scalar weights and input vector, where the scalar weights are usually assigned to be unity values. The incoming vectors are mapped by the RBFs in each hidden neuron. The output layer produces the final output through the linear combination of the outputs of hidden neurons. The network output can be obtained by

$$y(t) = \sum_{k=1}^{K} w_k(t)\phi_k(t) \tag{1}$$

where $w_k(t)$ is the output weight between the kth hidden neuron and the output neuron at time t, $k = 1, 2, \ldots, K$; K is the number of hidden neurons; $\phi_k(t)$ is the output of the

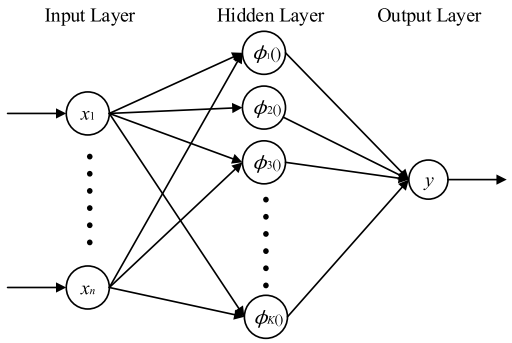

Fig. 1. Multiple-input single-output RBFNN.

kth hidden neuron, which is usually defined by a normalized Gaussian function

$$\phi_k(t) = e^{\|\mathbf{x}(t)-\boldsymbol{\mu}_k(t)\|/\sigma_k^2(t)} \tag{2}$$

where $\mathbf{x}(t) = [x_1(t), x_2(t), \ldots, x_n(t)]^{\mathrm{T}}$ is the input of the network, and $\mathbf{x} \in \Re^{n\times 1}$; $\mu_k(t)$ denotes the center vector of the kth hidden neuron; $||\mathbf{x}(t) - \mu_k(t)||$ is the Euclidean distance between $\mathbf{x}(t)$ and $\mu_k(t)$; $\sigma_k(t)$ is the width of the kth hidden neuron. In the training process, the parameters (i.e., the center and width of the hidden neurons and the output weight) and the number of hidden neurons will be tuned using the training samples.

B. PSO Algorithm

PSO is a population-based optimization technique, in which the population is referred as a swarm. For the minimization problem, a swarm consists of a number of particles. Each particle has a position represented by a vector

$$\mathbf{a}_i(t) = [a_{i,1}(t), a_{i,2}(t), \ldots a_{i,D}(t)] \tag{3}$$

where D is the dimensionality of the searching space, and $i = 1, 2, \ldots, s$, with s being the swarm size. Each particle has a velocity, which is represented by

$$\mathbf{v}_i(t) = [v_{i,1}(t), v_{i,2}(t), \ldots v_{i,D}(t)]. \tag{4}$$

During the movement, the best previous position of each particle is recorded as $\mathbf{p}_i(t) = [p_{i,1}(t), p_{i,2}(t), \ldots, p_{i,D}(t)]$, and the best position obtained by the swarm is denoted as $\mathbf{g}(t) = [g_1(t), g_2(t), \ldots, g_D(t)]$. Based on $\mathbf{p}_i(t)$ and $\mathbf{g}(t)$, the new velocity of each particle is updated by

$$\begin{aligned} v_{i,d}(t+1) = {} & \omega v_{i,d}(t) + c_1 r_1 (p_{i,d}(t) - a_{i,d}(t)) \\ & + c_2 r_2 (g_d(t) - a_{i,d}(t)) \end{aligned} \tag{5}$$

where $i = 1, 2, \ldots, s$; t represents the tth iteration in the displacement process; $d \in D$ represents the dth dimension in the searching space; ω is the inertia weight; c_1 and c_2 are the acceleration constants; r_1 and r_2 are the random values uniformly distributed in [0, 1]. It is noted that the inertia weight is used to control the impact of the previous velocities on the current velocity. Then, the new position of the ith particle is calculated by

$$a_{i,d}(t+1) = a_{i,d}(t) + v_{i,d}(t+1). \tag{6}$$

Meanwhile, the best previous position $\mathbf{p}_i(t+1)$ is computed using the following expression:

$$\mathbf{p}_i(t+1) = \begin{cases} \mathbf{p}_i(t), & \text{if } f(\mathbf{a}_i(t+1)) \geq f(\mathbf{p}_i(t)) \\ \mathbf{a}_i(t+1), & \text{otherwise} \end{cases} \tag{7}$$

where $f(\cdot)$ is the fitness function reflecting the solution quality. The global best position vector $\mathbf{g}(t+1)$ is given by

$$\mathbf{g}(t+1) = \arg\min_{\mathbf{p}_i}(f(\mathbf{p}_i(t+1))), \quad 1 \leq i \leq s. \tag{8}$$

The displacement process of the PSO algorithm will be repeated until specified termination conditions are satisfied. Since the initial positions of particles are randomly generated, their distribution may be uneven over the searching space. For example, if the inertia weight ω is large enough, the algorithm has good global searching ability. However, when ω is relatively small, the algorithm has good local searching performance. Therefore, the performance of PSO could be improved by setting the value of ω appropriately for a tradeoff between its local and global searching abilities.

III. APSO-SORBF Neural Network

In this section, we will provide a detailed introduction to the APSO algorithm, which is used to train RBFNNs. Then, the APSO-SORBF neural network is developed, with the aim of determining both the optimal parameters and a compact network structure to achieve better performance.

A. APSO Algorithm

In PSO, diversity plays an important role in improving the effectiveness of evolution [45]–[49]. The main reason for premature convergence might be a lack of the diversity [50]. To increase the diversity, the inertia weight should be adjusted according to the situation of the particles [55], [56]. Moreover, the flight of the particle is not a simple linear process, and the training process of RBFNNs is a complex nonlinear process. In this case, a nonlinear adaptive strategy based on the diversity is introduced, which is used to train the inertia weights. In APSO, the diversity is defined as

$$S(t) = f_{\min}(\mathbf{a}(t))/f_{\max}(\mathbf{a}(t)) \tag{9}$$

and

$$\begin{cases} f_{\min}(\mathbf{a}(t)) = \mathrm{Min}(f(\mathbf{a}_i(t))) \\ f_{\max}(\mathbf{a}(t)) = \mathrm{Max}(f(\mathbf{a}_i(t))) \end{cases} \tag{10}$$

where $f(\mathbf{a}_i(t))$ is the fitness value of the ith particle, $i = 1, 2, \ldots, s$, and $f_{\min}(\mathbf{a}(t))$ and $f_{\max}(\mathbf{a}(t))$ are the minimum and maximum fitness values of the swarm at time t, respectively. This diversity $S(t)$ is employed to describe the movement characteristics of the particles. Then, to adjust the inertia weights, such that the searching ability can be balanced, a nonlinear regressive function is introduced as follows:

$$\gamma(t) = (L - S(t))^{-t} \tag{11}$$

where $L \geq 2$ is a predefined constant.

TABLE I
APSO ALGORITHM

Initialize the parameters (c_1, c_2, v_{max}), the best previous position $\mathbf{p}_i$ and the global best position $\mathbf{g}$ of the swarm
$\mathbf{p}_i(1)=\mathbf{a}_i(1)$, $\mathbf{g}(1)=\arg\min(\mathbf{p}_i(1))$, $i=1, 2, \ldots, s$,
for t=1 to the maximum iteration
for i=1 to s
Evaluate the diversity of particle $S(t)$, %Eq. (9)
Compute the regressive function $\gamma(t)$, %Eq.(11)
Calculate the change ratios of particles $A_i(t)$, %Eq. (12)
Calculate the inertia weight $\omega_i(t)$, %Eq. (13)
Update the velocity $\mathbf{v}_i(t+1)$, %Eq.(5)
Update the position $\mathbf{a}_i(t+1)$, %Eq.(6)
end
Update the best previous position $\mathbf{p}_i(t+1)$, %Eq. (7)
Update the global best position $\mathbf{g}(t+1)$, %Eq.(8)
End

In addition, to obtain suitable velocities, the change ratio between the ith particle and the best particle of the swarm is given by

$$A_i(t) = f(\mathbf{g}(t))/f(\mathbf{a}_i(t)) \tag{12}$$

where $f(\mathbf{g}(t))$ is the best fitness value of the swarm.

Based on the above analysis, the adaptive strategy of inertia weight is defined as

$$\omega_i(t) = \gamma(t)(A_i(t) + c) \tag{13}$$

where $\omega_i(t)$ is the inertia weight of the ith particle at time t, and $c \geq 0$ is a predefined constant to improve the global searching ability of the particles.

After combination with the nonlinear regressive function, the proposed APSO algorithm is summarized in Table I.

The key concept of (11) is to balance the local and global searching abilities. Some remarks are given in the following.

Remark 1: Adaptive searching methods are widely used in PSO variants [36]–[39]. However, in contrast to the existing methods, the inertia weight ω of the proposed APSO is changed according to a nonlinear regressive function, as shown in (11), which has not been well documented in the existing literature.

Remark 2: To improve PSO performance, the inertia weight ω was adjusted using adaptive searching methods, requiring some additional parameters, in [36]–[39]. However, in contrast to the existing methods, the inertia weight ω of the proposed APSO is changed according to a nonlinear regressive function, as shown in (11), which can be solved using the fitness of particles. Moreover, this nonlinear regressive function is adjusted by the diversity of the swarm, in which $S(t)$ and $\gamma(t)$ will be decreased if the particles move together and increased if the particles become dispersed. This adjustment helps the particles easily escape from local optima.

Remark 3: It is a major obstacle for PSO to find the global minimal solution instead of falling into local minimal solutions during the adjustment process. The proposed APSO algorithm can improve the balance between the exploration and exploitation in terms of: 1) A_i, which is defined by the ratio of the best fitness value and the individual fitness value of the particles and 2) the constant c. The results in [48] indicated that a relatively large inertia weight is better for a global search. Thus, the constant c is employed to increase the initial values of the inertia weights, such that the global searching ability can be enhanced.

B. APSO-SORBF Neural Network

This section describes the APSO-SORBF neural network, which can adjust both the network size and parameters during the training process. In the particle initialization stage, let the position of the ith particle be represented as

$$\mathbf{a}_i = \left[\boldsymbol{\mu}_{i,1}^T, \sigma_{i,1}, w_{i,1}, \boldsymbol{\mu}_{i,2}^T, \sigma_{i,2}, w_{i,2} \cdots \boldsymbol{\mu}_{i,K_i}^T, \sigma_{i,K_i}, w_{i,K_i}\right] \tag{14}$$

where $\boldsymbol{\mu}_{i,k}$, $\sigma_{i,k}$, and $w_{i,k}$ are the center, width, and output weight of the kth hidden neuron in the ith particle, respectively; K_i is the network size (number of hidden neurons); D_i is the dimension of the ith particle satisfying $D_i = (2+n)K_i$, $i = 1, 2, \ldots, s$, with n being the number of input variable.

In the APSO-SORBF neural network, the fitness value of each particle represents the accuracy of the network. In this paper, to optimize both the parameters and the network size of RBFNNs, the fitness value of each particle is designed by considering both the error criterion and the network size. The smaller the fitness value, the better is the ith solution. The expression of the fitness value is given as follows:

$$f(\mathbf{a}_i(t)) = E_i(t) + \alpha K_i(t) \tag{15}$$

where $f(\mathbf{a}_i(t))$ is the fitness value of the ith particle, K_i is the number of hidden neurons, and $E_i(t)$ is the root-mean-square error (RMSE) of the neural network. Furthermore, $E_i(t)$ is given by

$$E_i(t) = \sqrt{\frac{1}{T}\sum_{t=1}^{T}(y(t) - y_d(t))^2} \tag{16}$$

where T is the number of data pairs, $y(t)$ and $y_d(t)$ are the network output and the desired output at time t, respectively, and $\alpha > 0$ is the balance factor.

Then, the APSO algorithm is used to achieve the optimal particle based on (5) and (6). The dimensions of other particles will be updated by changing the network size

$$K_i = \begin{cases} K_i - 1 & \text{if } (K_{\text{best}} < K_i) \\ K_i + 1 & \text{if } (K_{\text{best}} \geq K_i) \end{cases} \tag{17}$$

where K_i is the network size of the ith particle ($i = 1, 2, \ldots, s$), and K_{best} is the network size of the optimal particle.

To demonstrate the adjustment process for both the parameters and the network size in the APSO-SORBF neural network, Fig. 2 shows an example with three particles. The initial number of hidden neurons is assumed to be $K_1 = 2$, $K_2 = 4$, and $K_3 = 5$, and the second particle is assumed to be the optimal particle ($K_{\text{best}} = K_2$). In the adjustment process, when the number of hidden neurons in the particles is larger than for the optimal particle (e.g., K_3 is larger than K_2), the hidden neurons in these particles need to be reduced. In contrast, when the number of hidden neurons in the particles is less than for the optimal particle (e.g., K_1 is less than K_2), the number of

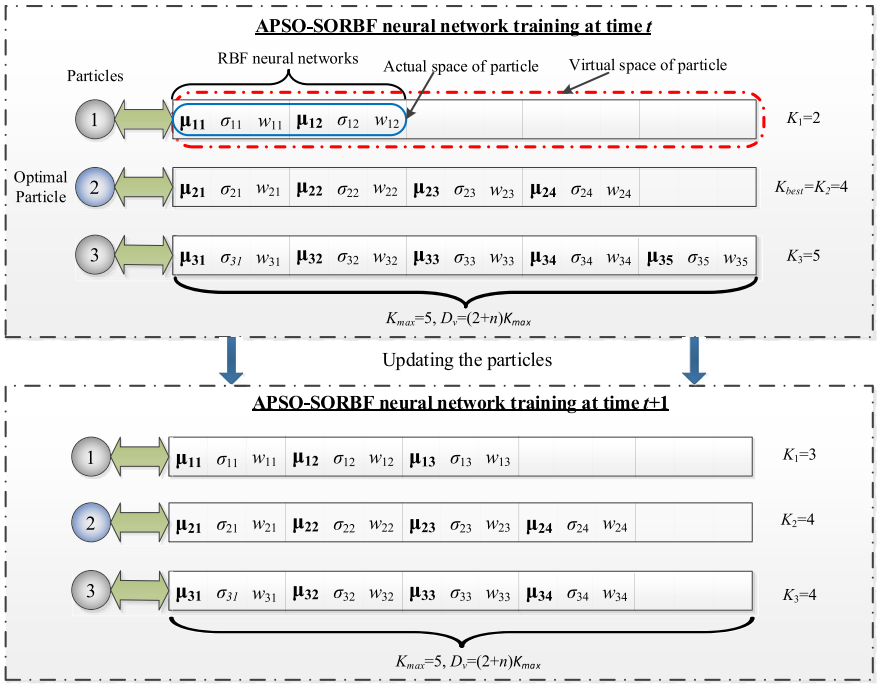

Fig. 2. Example of the APSO-SORBF neural network training process.

hidden neurons will be increased. The network size is then updated by (17). The particles will gradually approach the optimal particle during the iteration process, and the optimal RBFNN will be obtained.

However, in the training process, if the dimensions of each particle are not the same, the particle velocities cannot be updated. To solve this problem, the maximum dimension criterion was introduced in [35], i.e., all particles share the same virtual space D_v, which is given by the maximum dimension of the existing particles. In the updating process, for all particles, if virtual space D_v is larger than the actual space, the remaining positions of the virtual space will be randomly initialized. The remaining positions of the virtual space will be empty after the updating process is finished. Based on the above analysis, the procedure for the APSO-SORBF neural network is summarized in Table II.

Certain remarks should be emphasized.

Remark 4: In the proposed APSO-SORBF neural network, the fitness value of each particle is designed based on the error criterion and the network size. The displacement process of particles is particularly suitable for optimizing the RBFNN. As a result, the APSO-SORBF neural network can adjust both the parameters and network size during the training process. Thus, the APSO-based optimization can be used as the basis for an integrated RBF network training methodology.

Remark 5: To share information between solutions with different network sizes, the dimensions of all particles are set equal to the virtual space dimension D_v. Thus, the speeds and the positions of all particles can be updated in the displacement process.

IV. Convergence Analysis

For the proposed APSO-SORBF neural network, the convergence of the algorithm is an important issue and needs careful investigation. In this section, the analysis of convergence is given to guarantee the successful application of the proposed APSO-SORBF neural network in detail. Furthermore, one can get a better understanding of the APSO-SORBF neural network through this analysis.

TABLE II
APSO-SORBF Neural Network

Initialize the parameters (c_1, c_2, v_{max}). Generate an RBF neural network for every particle, and initialize the position of the particles, $\mathbf{p}_i(1)$=$\mathbf{a}_i(1)$, $\mathbf{g}(1)$=min($\mathbf{p}_i(1)$), i=1, 2, …, s.
for t=1 to the maximum iteration
  for i=1 to s
    Calculate the dimension of the particle $D_i(t)$
    Evaluate the RMSE values %Eq.(16)
    Calculate the fitness value $f(\mathbf{a}_i(t))$ %Eq.(15)
    Evaluate the diversity of particle $S(t)$, %Eq.(9)
    Compute the regressive function $\gamma(t)$, %Eq.(11)
    Calculate the differences of the particles $A_i(t)$, %Eq.(12)
    Calculate the inertia weight $\omega_i(t)$, %Eq.(13)
    Update the velocity $\mathbf{v}_i(t+1)$, %Eq.(5)
    Update the position $\mathbf{a}_i(t+1)$, %Eq.(6)
  end
    Update the individual best position $\mathbf{p}_i(t+1)$, %Eq.(7)
    Update the global best position $\mathbf{g}(t+1)$, %Eq.(8)
    Update the network size K_i %Eq.(17)
End

A. Convergence Analysis of APSO Algorithm

For the APSO algorithm, it can be observed from (5) and (6) that the position and the velocity of each particle are updated independently. Thus, without loss of generality, some variables are introduced

$$\beta_1 = c_1 r_1, \quad \beta_2 = c_2 r_2, \quad \beta = \beta_1 + \beta_2. \tag{18}$$

Moreover, to analyze the convergence of the APSO algorithm, the following basic assumptions are made in this paper.

Assumption 1: The best previous position $\mathbf{p}_i(t)$ and the global best position $\mathbf{g}(t)$ satisfy the condition $\{\mathbf{p}_i(t), \mathbf{g}(t)\} \in \Gamma$, where Γ is the searching space, and $i = 1, 2, \ldots, s$. The best previous position $\mathbf{p}_i(t)$ and the global best position $\mathbf{g}(t)$ have a lower bound.

Assumption 2: There exist "optimal" constant parameters $\mathbf{p}^*$ and $\mathbf{g}^*$ for $\mathbf{p}_i$ and $\mathbf{g}$, respectively.

Assumption 3: There exist $A_i(t) \geq 0$, and a constant c satisfying the condition $-A_i(t) < c < \gamma^{-1}(t) - A_i(t)$.

Assumption 4: There exist $c_1 r_1 > 0$, $c_2 r_2 > 0$, and the parameter β satisfying the condition $0 < \beta < 2(1 + \omega_i(t))$.

Theorem 1: Suppose that Assumptions 1–4 are satisfied. Then, given $\beta_1 \geq 0$, $\beta_2 \geq 0$, the particle position $\mathbf{a}_i(t)$ will converge to $(\beta_1\mathbf{p}^* + \beta_2\mathbf{g}^*)/(\beta_1 + \beta_2)$.

Proof: According to (18), by substituting (5) for (6), the following nonhomogeneous recurrence equation is obtained:

$$a_{i,d}(t+1) = (1 + \omega_i(t) - \beta)a_{i,d}(t) - \omega_i(t)a_{i,d}(t-1) + \beta_1 p_{i,d}(t) + \beta_2 g_d(t) \quad (19)$$

where $i = 1, 2, \ldots, s$ and $d = 1, 2, \ldots, D_v$. Equation (19) can be expressed as the matrix form

$$\begin{bmatrix} a_{i,d}(t+1) \\ a_{i,d}(t) \\ 1 \end{bmatrix} = \varphi\ (t) \begin{bmatrix} a_{i,d}(t) \\ a_{i,d}(t-1) \\ 1 \end{bmatrix} \quad (20)$$

where the coefficient matrix $\varphi\ (t)$ is

$$\varphi\ (t) = \begin{bmatrix} 1 + \omega_i(t) - \beta & -\omega_i(t) & \beta_1 p_{i,d}(t) + \beta_2 g_d(t) \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \quad (21)$$

Then, the characteristic polynomial of matrix $\varphi\ (t)$ is

$$(\lambda - 1)(\lambda^2 - (1 + \omega_i(t) - \beta)\lambda + \omega_i(t)) = 0 \quad (22)$$

and the eigenvalues of $\varphi\ (t)$ are

$$\lambda_1 = 1, \quad \lambda_2 = \frac{1 + \omega_i(t) - \beta + \sqrt{(1 + \omega_i(t) - \beta)^2 - 4\omega_i(t)}}{2}$$
$$\lambda_3 = \frac{1 + \omega_i(t) - \beta - \sqrt{(1 + \omega_i(t) - \beta)^2 - 4\omega_i(t)}}{2}. \quad (23)$$

Hence, the particle position can be expressed as in [51]

$$a_{i,d}(t) = k_1 + k_2\lambda_2^t + k_3\lambda_3^t \quad (24)$$

where k_1, k_2, and k_3 are constants.

Obviously, there are two parameters $\omega_i(t)$ and β that affect the eigenvalues. The convergence condition of the APSO algorithm is $\max(|\lambda_2|, |\lambda_3|) < 1$. That is

$$\frac{1}{2}|1 + \omega_i(t) - \beta \pm \sqrt{(1 + \omega_i(t) - \beta)^2 - 4\omega_i(t)}| < 1. \quad (25)$$

Consider two cases: 1) $(1 + \omega_i(t) - \beta)^2 - 4\omega_i(t) < 0$ and 2) $(1 + \omega_i(t) - \beta)^2 - 4\omega_i(t) \geq 0$. In case 1), both eigenvalues λ_2 and λ_3 are complex numbers

$$|\lambda_2|^2 = |\lambda_3|^2 = \frac{1}{4}\|1 + \omega_i(t) - \beta \pm \sqrt{(1 + \omega_i(t) - \beta)^2 - 4\omega_i(t)}\|^2 = \omega_i(t). \quad (26)$$

Indeed, $\max(|\lambda_2|, |\lambda_3|) < 1$ requires only $\omega_i(t) < 1$ in case 1). Moreover, case 1) requires that $\omega_i(t) > 0$ and $(1 + \omega_i(t) - 2\sqrt{\omega_i(t)}) < \beta < (1 + \omega_i(t) + 2\sqrt{\omega_i(t)})$. Then, the convergent conditions in case 1) are

$$\begin{cases} 0 < \omega_i(t) < 1 \\ (1 + \omega_i(t) - 2\sqrt{\omega_i(t)}) < \beta < (1 + \omega_i(t) + 2\sqrt{\omega_i(t)}). \end{cases} \quad (27)$$

Meanwhile, in case 2), both eigenvalues λ_2 and λ_3 are the real numbers. The conditions of case 2) are equivalent to $\omega_i(t) \geq 0$, and $\beta \leq (1 + \omega_i(t) - 2\sqrt{\omega_i(t)})$ or $\beta \geq (1 + \omega_i(t) + 2\sqrt{\omega_i(t)})$. If $\beta \leq (1 + \omega_i(t) - 2\sqrt{\omega_i(t)})$, then $\max(|\lambda_2|, |\lambda_3|) < 1$ implies

$$\frac{1}{2}[1 + \omega_i(t) - \beta + \sqrt{(1 + \omega_i(t) - \beta)^2 - 4\omega_i(t)}] < 1. \quad (28)$$

Thus, $0 < \beta \leq 2(1 + \omega_i(t) - 2\sqrt{\omega_i(t)})$ and $\omega_i(t) < 1$. If $\beta \geq (1 + \omega_i(t) + 2\sqrt{\omega_i(t)})$, $\max(|\lambda_2|, |\lambda_3|) < 1$ implies

$$\frac{1}{2}[1 + \omega_i(t) - \beta - \sqrt{(1 + \omega_i(t) - \beta)^2 - 4\omega_i(t)}] > -1. \quad (29)$$

Thus, $(1 + \omega_i(t) + 2\sqrt{\omega_i(t)}) < \beta < 2(1 + \omega_i(t))$ and $\omega_i(t) < 1$. Then, the convergent conditions in case 2) are

$$\begin{cases} 0 \leq \omega_i(t) < 1 \\ (1 + \omega_i(t) + 2\sqrt{\omega_i(t)}) < \beta < 2(1 + \omega_i(t)). \end{cases} \quad (30)$$

Combining cases 1) and 2), the convergent conditions of the APSO algorithm are

$$\begin{cases} 0 \leq \omega_i(t) < 1 \\ 0 < \beta < 2(1 + \omega_i(t)). \end{cases} \quad (31)$$

According to (9) and (11), we have

$$\begin{cases} 0 < \gamma\ (t) < 1 \\ 0 < A_i(t) \leq 1. \end{cases} \quad (32)$$

It follows from (32) and Assumption 3 that there exists $0 \leq \omega_i(t) < 1$ in the learning process of the APSO algorithm. Based on Assumption 4, the necessary and sufficient conditions to guarantee the convergence of the APSO algorithm are easily obtained. Then, according to (24), the convergent value of the particle position can be calculated as

$$\lim_{t\to\infty} a_{i,d}(t) = k_1. \quad (33)$$

Let $t = 0$, $t = 1$, and $t = 2$ in (20). Then, the value of k_1 can be calculated as

$$\lim_{t\to\infty} a_{i,d}(t) = \lim_{t\to\infty} (\beta_1 p_{i,d}(t) + \beta_2 g_d(t))/(\beta_1 + \beta_2). \quad (34)$$

According to [52, Th. 1], it follows from Assumptions 1 and 2 that:

$$\lim_{t\to\infty} p_{i,d}(t) = p_d^*, \quad \lim_{t\to\infty} g_d(t) = g_d^* \quad (35)$$

where $\mathbf{p}^* = [p_1^*, p_2^*, \cdots, p_{D_v}^*]$ and $\mathbf{g}^* = [g_1^*, g_2^*, \cdots, g_{D_v}^*]$.

Thus, we have

$$\lim_{t\to\infty} \mathbf{a}_i(t) = (\beta_1 \mathbf{p}^* + \beta_2 \mathbf{g}^*)/(\beta_1 + \beta_2) \tag{36}$$

where $\mathbf{a}_i(t) = [a_{i,1}(t), a_{i,2}(t), \ldots, a_{i,D_v}(t)]$, $i = 1, 2, \ldots, s$.

This reasoning completes the proof of Theorem 1. ■

Theorem 2: Suppose that Assumptions 1–4 are satisfied. Then, given $\beta_1 \geq 0$, $\beta_2 \geq 0$, the particle velocity $\mathbf{v}_i(t)$ will converge to $\mathbf{0}$.

Proof: According to (5), the following equation holds:

$$v_{i,d}(t+1) - (1 + \omega_i(t) - \beta)v_{i,d}(t) + \omega_i(t)v_{i,d}(t-1) = 0. \tag{37}$$

Then, the characteristic polynomial of the coefficient matrix is

$$(\lambda^2 - (1 + \omega_i(t) - \beta)\lambda + \omega_i(t)) = 0 \tag{38}$$

and the eigenvalues are

$$\lambda_4 = \frac{1 + \omega_i(t) - \beta + \sqrt{(1 + \omega_i(t) - \beta)^2 - 4\omega_i(t)}}{2}$$
$$\lambda_5 = \frac{1 + \omega_i(t) - \beta - \sqrt{(1 + \omega_i(t) - \beta)^2 - 4\omega_i(t)}}{2}. \tag{39}$$

Meanwhile, the particle velocity can be expressed as

$$v_{i,d}(t) = k_4\lambda_4 + k_5\lambda_5 \tag{40}$$

where k_4 and k_5 are constants.

Since Assumptions 1–4 are valid, according to the analysis of Theorem 1, we have

$$\lim_{t\to\infty} v_{i,d}(t) = 0. \tag{41}$$

Thus, we can conclude that

$$\lim_{t\to\infty} \mathbf{v}_i(t) = \mathbf{0} \tag{42}$$

where $\mathbf{v}_i(t) = [v_{i,1}(t), v_{i,2}(t), \ldots, v_{i,D_v}(t)]$.

This reasoning completes the proof of Theorem 2. ■

Remark 6: Theorems 1 and 2 present the convergence analysis of the APSO algorithm. Two main properties of APSO algorithm about the particles' position and velocity are proposed. It is noted that Theorems 1 and 2 verify that each particle in the APSO algorithm will converge to the best position.

Remark 7: The convergence analysis of the particles' position and the velocity in Theorems 1 and 2 leads to some conditions on the parameters of APSO algorithm, namely, $-A_i(t) < c < \gamma^{-1}(t) - A_i(t)$ and $0 < \beta < 2(1 + \omega_i(t))$. These conditions can be used for the parameter selection in the APSO algorithm.

B. Convergence Analysis of APSO-SORBF Neural Network

According to the above description, a simple and efficient APSO algorithm with a special fitness function is employed to automatically construct RBFNNs. The goal of the APSO algorithm is to construct appropriate RBFNNs. After the APSO algorithm returns a set of particles, the parameters and size of the APSO-SORBF neural network corresponding to each particle will be obtained using (3).

To provide the theoretical basis for the applications, this section presents the convergence analysis of the APSO-SORBF neural network based upon the convergence analysis of the APSO algorithm. The convergence analysis of the APSO-SORBF neural network is summarized in Theorem 3.

Theorem 3: Under the same conditions as in Theorems 1 and 2, if the bounds of the predefined maximum velocity are adjusted dynamically and $v_{\max}(t) < 2|E_i(t)|/(2 + n)K_i(t)$, then the APSO-SORBF neural network is convergent, and $E_i(t) \to 0$ as $t \to \infty$, $i = 1, 2, \ldots, s$.

Proof: Consider the following Lyapunov function:

$$V_i(t) = \frac{1}{2}E_i^2(t). \tag{43}$$

According to (16), the change in the Lyapunov function between two steps is

$$\Delta V_i(t) = V_i(t+1) - V_i(t) = \frac{1}{2}\left[E_i^2(t+1) - E_i^2(t)\right]. \tag{44}$$

In addition, the error change is denoted as

$$E_i(t+1) = E_i(t) + \Delta E_i(t). \tag{45}$$

Then, the strictly differential formula of RMSE is

$$\Delta E_i(t) = \frac{\partial E_i(t)}{\partial \mathbf{w}_i(t)}[\Delta \mathbf{w}_i(t)]^T + \frac{\partial E_i(t)}{\partial \bar{\boldsymbol{\mu}}_i(t)}[\Delta \bar{\boldsymbol{\mu}}_i(t)]^T + \frac{\partial E_i(t)}{\partial \boldsymbol{\sigma}_i(t)}[\Delta \boldsymbol{\sigma}_i(t)]^T \tag{46}$$

where $\mathbf{w}_i(t) = [w_{i,1}(t), w_{i,2}(t), \ldots, w_{i,Ki}(t)]$, $\bar{\boldsymbol{\mu}}_i(t) = [\mu_{i,1}(t), \mu_{i,2}(t), \ldots, \mu_{i,Ki}(t)]$, and $\sigma_i(t) = [\sigma_{i,1}(t), \sigma_{i,2}(t), \ldots, \sigma_{i,Ki}(t)]$ are the three adjusted parameters in RBFNNs. The above parameter updating rules are as follows:

$$\Delta \mathbf{w}_i(t) = -2E_i(t)\frac{\partial E_i(t)}{\partial \mathbf{w}_i(t)} = \mathbf{v}_{i,w}(t)$$
$$\Delta \bar{\boldsymbol{\mu}}_i(t) = -2E_i(t)\frac{\partial E_i(t)}{\partial \bar{\boldsymbol{\mu}}_i(t)} = \mathbf{v}_{i,\boldsymbol{\mu}}(t)$$
$$\Delta \boldsymbol{\sigma}_i(t) = -2E_i(t)\frac{\partial E_i(t)}{\partial \boldsymbol{\sigma}_i(t)} = \mathbf{v}_{i,\sigma}(t) \tag{47}$$

where $\mathbf{v}_i(t) = [\mathbf{v}_{i,\mu}(t), \mathbf{v}_{i,\sigma}(t), \mathbf{v}_{i,w}(t)]$ is the velocity of the ith particle. According to the above analysis, the factorization of (44) is described as

$$\Delta V_i(t) = 2E_i(t)\Delta E_i(t) + \Delta E_i^2(t) = 2E_i^2(t)\Omega(t)(\Omega(t) - 1) \tag{48}$$

where

$$\Omega(t) = |\partial E_i(t)/\partial \mathbf{w}_i(t)|^2 + |\partial E_i(t)/\partial \bar{\boldsymbol{\mu}}_i(t)|^2 + |\partial E_i(t)/\partial \boldsymbol{\sigma}_i(t)|^2. \tag{49}$$

Since the bounds of the predefined maximum velocity are adjusted dynamically, and $v_{\max}(t) < 2|E_i(t)|/((2 + n)K_i(t))^{1/2}$, the following conditions can be obtained:

$$\left|\frac{\partial E_i(t)}{\partial \mathbf{w}_i(t)}\right|^2 = \left|\frac{\mathbf{v}_{i,w}(t)}{2E_i(t)}\right|^2 < \frac{K_i(t)}{(2+n)K_i(t)}$$
$$\left|\frac{\partial E_i(t)}{\partial \bar{\boldsymbol{\mu}}_i(t)}\right|^2 = \left|\frac{\mathbf{v}_{i,\mu}(t)}{2E_i(t)}\right|^2 < \frac{nK_i(t)}{(2+n)K_i(t)}$$
$$\left|\frac{\partial E_i(t)}{\partial \boldsymbol{\sigma}_i(t)}\right|^2 = \left|\frac{\mathbf{v}_{i,\sigma}(t)}{2E_i(t)}\right|^2 < \frac{K_i(t)}{(2+n)K_i(t)} \tag{50}$$

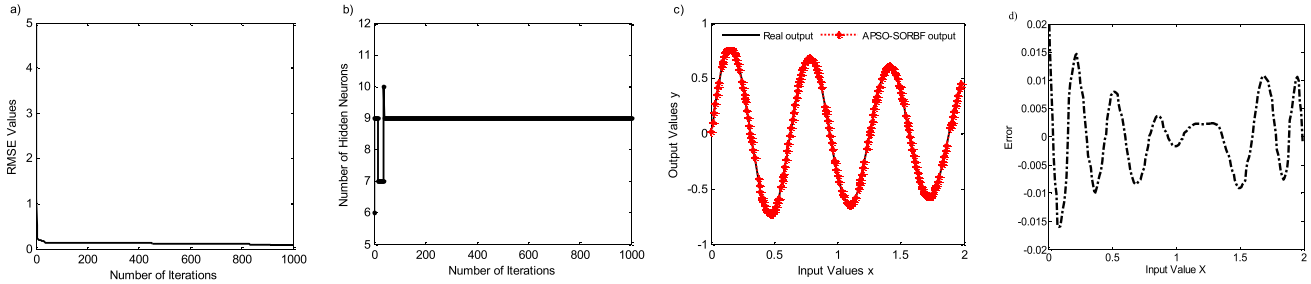

Fig. 3. Experimental results of the proposed APSO-SORBF neural network for function approximation. (a) RMSE values against iterations. (b) Number of hidden neurons against iterations. (c) Approximation results from the APSO-SORBF neural network. (d) Approximation error versus input values.

where $K_i(t)$ is the number of hidden neurons at time t for the ith particle. Furthermore, it follows from Theorems 1 and 2 that:

$$\left|\frac{\partial E_i(t)}{\partial \mathbf{w}_i(t)}\right|^2 + \left|\frac{\partial E_i(t)}{\partial \bar{\boldsymbol{\mu}}_i(t)}\right|^2 + \left|\frac{\partial E_i(t)}{\partial \boldsymbol{\sigma}_i(t)}\right|^2 < 1 \tag{51}$$

which leads to

$$\Delta V_i(t) < 0. \tag{52}$$

Thus, $E_i(t)$ is bounded for $t \geq t_0$. Moreover, through the Lyapunov-like lemma, it is implied that

$$\lim_{t\to\infty} E_i(t) = 0. \tag{53}$$

Therefore, $E_i(t) \to 0$ as $t \to \infty$, $i = 1, 2, \ldots, s$. Thus, the convergence of the proposed APSO-SORBF neural network is proved. ■

Remark 8: It can be observed that the bounds of velocity are adaptively adjusted to ensure the convergence of the APSO-SORBF neural network, with no need for manual setting. Considering both sides of the predefined maximum velocity, $E_i(t)$ decreases and $K_i(t)$ moves to an "optimal" value during the training process of the APSO-SORBF neural network, and the bounds are dynamically reduced. Thus, the particles tend to eventually stop.

Remark 9: Based on the above discussion, the convergence of the APSO-SORBF neural network can be maintained in both the parameter-adjusting phase and the network size-adjusting phase. Therefore, the APSO algorithm can indeed guarantee the convergence of the APSO-SORBF neural network, which is among the most distinguishing features required for any successful application.

V. Simulation and Discussion

In practical applications, it is a time-consuming process to choose the proper size of an RBFNN. A manual size selection process requires many experiments, and the result may lack generality, while the proposed APSO-SORBF neural network can adjust the parameters and network size dynamically. In this section, to verify the effectiveness of the proposed APSO-SORBF neural network, comparisons are made with some well-known methods in five examples: example A addresses function approximation, examples B–D focus on nonlinear system modeling or predicting problems, and example E is concerned with a real-world problem of predicting the effluent total phosphorus (TP) in a wastewater treatment process (WWTP). For simplicity, the initial parameters of the APSO-SORBF neural network in these examples are the same: $r_1 = r_2 = 1.49$, $L = 2.1$, $c = 2$, and $\alpha = 0.03$. All the examples were programmed in MATLAB version 2010 and run on a PC with a clock speed of 2.6 GHz and 4 GB RAM, under a Microsoft Windows 8.0 environment.

A. Function Approximation

In this example, the APSO-SORBF neural network is applied to approximate the following function:

$$y = 0.8e^{-0.2x}\sin(10x). \tag{54}$$

This function approximation is a benchmark problem used in [16], [26], and [33] to test many popular algorithms. There are 300 training patterns with x-coordinates uniformly distributed in the range [0, 2], and the testing data set consists of 200 patterns with x-coordinates randomly generated in the same range [0, 2].

Fig. 3 shows the results of the proposed APSO-SORBF neural network: the training RMSE, the number of hidden neurons, the testing output, and the approximation error. The results are compared with GAP-RBF [26], SORBF [33], error correction [16], PSO-RBF [35], adaptive improved PSO-based RBF (AI-PSO-RBF) [48], and stability adaptive inertia weight PSO-based RBF (SAIW-PSO-RBF) [58] algorithms. All of the algorithms use the same training data sets and test partitions. Table III shows the comparison of the mean and deviation (Dev.) of testing RMSE, the number of hidden neurons, the testing time, and the training time. This information shows that the APSO-SORBF neural network can self-organize the network structure. Meanwhile, the results show that the final structure of the APSO-SORBF neural network is the most compact (nine hidden neurons). In addition, the APSO-SORBF neural network produces the best mean and Dev. of testing RMSE (generalization performance) for this function approximation problem. Moreover, the proposed APSO-SORBF neural network needs less training time (869.6 s) than the AI-PSO-RBF neural network or the SAIW-PSO-RBFNN. Moreover, the proposed APSO-SORBF neural network needs the least testing time (0.0039 s) of all algorithms.

Remark 10: In all five examples, the final network structure is the steadiest over 50 trials of simulations. To make the

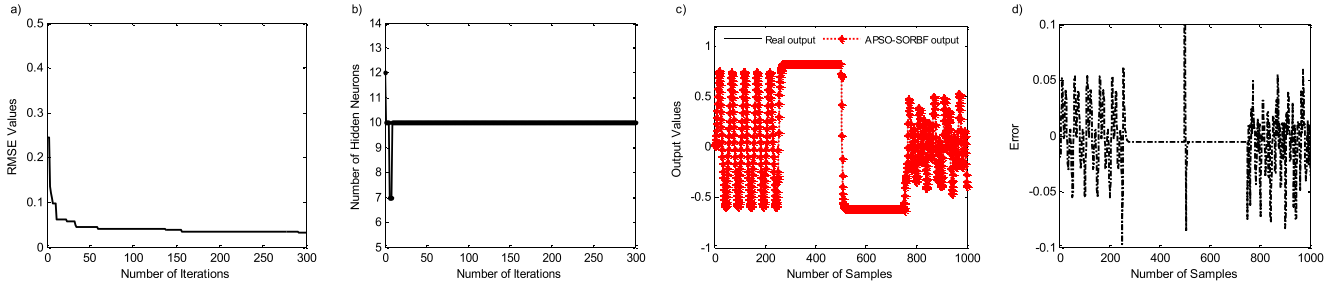

Fig. 4. Experimental results of the proposed APSO-SORBF neural network for nonlinear system identification. (a) RMSE values against iterations. (b) Number of hidden neurons against iterations. (c) Identification results for the APSO-SORBF neural network. (d) Identification error versus sample number.

TABLE III
COMPARISON OF DIFFERENT ALGORITHMS FOR THE FUNCTION APPROXIMATION

Algorithm	Testing RMSE		No. of hidden neurons	Training time(s)	Testing time(s)
	Mean	Dev.			
APSO-SORBF	0.0133	0.0056	9	869.6	0.0039
GAP-RBF[26]	0.0415*	0.0087*	19*	—	0.0087*
SORBF[33]	0.0248*	0.0067	12*	—	0.0044*
ErrCor[16]	0.0141*	—	20*	—	—
PSO-RBF[35]	0.0368	0.0164	13	832.5	0.0054
AI-PSO-RBF[48]	0.0295	0.0073	11	1269.8	0.0042
SAIW-PSO-RBF[58]	0.0197	0.0026	11	1057.2	0.0046

* The results are also listed in the original papers.

TABLE IV
COMPARISON OF DIFFERENT ALGORITHMS FOR THE NONLINEAR SYSTEM IDENTIFICATION

Algorithm	Testing RMSE		No. of hidden neurons	Training time(s)	Testing time(s)
	Mean	Dev.			
APSO-SORBF	0.0153	0.0073	11±1	1251.4	0.0047
GGAP-RBF[27]	0.2229	0.0165	14±1	—	0.0068
HBF-GP[28]	0.0626*	—	12±3*	—	—
FS-RBFNN[29]	0.1622*	—	13±1*	—	0.0054*
PSO-RBF[35]	0.2564	0.0126	11±1	1245.1	0.0049
AI-PSO-RBF[48]	0.1536	0.0109	12±2	1491.1	0.0051
SAIW-PSO-RBF[58]	0.0934	0.0113	11±2	1369.8	0.0056

* The results are listed in the original papers.

comparisons meaningful, each algorithm was run 50 times, and the presented performance is the average value of 50 trials.

B. Nonlinear System Identification

The nonlinear system is given by

$$y(t+1) = 0.72y(t) + 0.025y(t-1)u(t-1) + 0.01u^2(t-2) + 0.2u(t-3) \quad (55)$$

where there are only two input values, $y(t)$ and $u(t)$, and $y(t+1)$ the output. The nonlinear system has been used in [16], [26], and [33] to demonstrate the performance of neural networks. The training inputs were sequenced uniformly over the interval $[-2, 2]$ for approximately half of the training time, while a single sinusoid signal generated by $1.05 \times \sin(t/45)$ was used for the remaining training time. The sizes of the training and testing sample were 1000. The check input signal was used to determine the identification results for the testing signal

$$u(t) = \begin{cases} \sin(\pi t/25), & 0 < t < 250 \\ 1.0, & 250 \le t < 500 \\ -1.0, & 500 \le t < 750 \\ 0.3\sin(\pi t/25) + 0.1(\pi t/32) & \\ \quad +0.6(\pi t/10), & 750 \le t < 1000. \end{cases} \quad (56)$$

To evaluate the performance of the APSO-SORBF neural network, the results are compared with the results of five other SORBF neural networks: the GGAP-RBF [27], hyper basis functions with growing and pruning [28], FS-RBFNN [29], PSO-RBF [35], AI-PSO-RBF [48], and SAIW-PSO-RBF [58] algorithms.

Fig. 4 records the RMSE values, the number of hidden neurons in the training process, the outputs, and the error between the desired output and the APSO-SORBF output. A number of indices are selected to reflect the performance: the number of preserved hidden neurons, the mean and Dev. of the testing RMSE, the training time, and the testing time. Table IV exhibits the detailed results of the different algorithms.

Fig. 4 shows that the proposed APSO-SORBF neural network can modify both the structure and parameters automatically. Regardless of the initial structures, the final structure of APSO-SORBF is more compact than the structures obtained by the other algorithms. The mean and Dev. of the testing RMSE are the smallest among all the algorithms. Compared with the other algorithms, the APSO-SORBF neural network requires the least testing time and training time (except for PSO-RBF). Moreover, this example shows that the APSO-SORBF neural network has better identification ability than any of the other neural networks.

C. Mackey-Glass Time Series Prediction

The Mackey-Glass time series prediction problem has been recognized as one of the benchmark problems for assessing

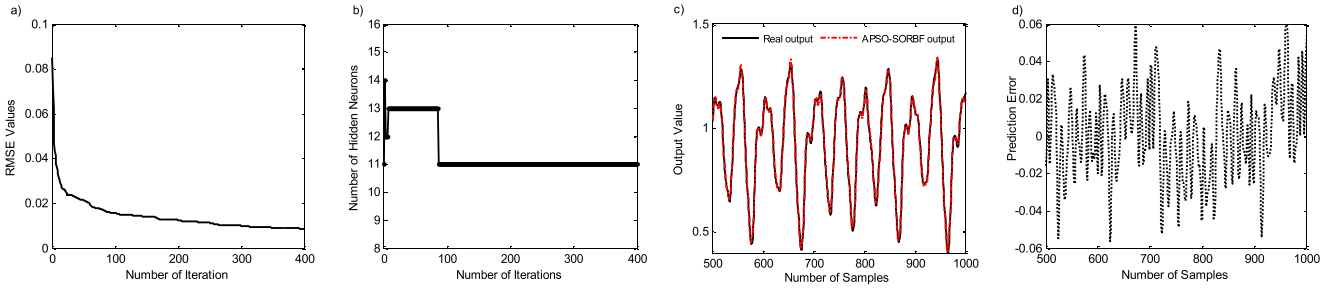

Fig. 5. Experimental results of the proposed APSO-SORBF neural network for Mackey-Glass time series prediction. (a) RMSE values against iterations. (b) Number of hidden neurons against iterations. (c) Predicting results for the APSO-SORBF neural network. (d) Prediction error versus sample number.

the performance of learning algorithms [15], [26], [42]. The time series prediction is generated by the following discrete equation:

$$x(t+1) = (1-a)x(t) + bx(t-\tau)/(1+x^{10}(t-\tau)) \quad (57)$$

where $a = 0.1$, $b = 0.2$, and $\tau = 17$, and the initial condition $x(0) = 1.2$. The value $x(t+\Delta t)$ is predicted from the previous values $\{x(t), x(t-\Delta t), \ldots, x(t-(l-1)\Delta t)\}$. In this paper, the prediction model is given by

$$x(t+\Delta t) = f(x(t), x(t-6), x(t-12), x(t-18)). \quad (58)$$

For the training phase, 500 data points were acquired from $t = 1$–500 (the time step is 1) and used as the input and output sample data. As shown by the training results in Fig. 5, the prediction capability of the trained APSO-SORBF neural network can be illustrated using another 500 data points ranging from $t = 501$–1000. The testing results in Fig. 5 demonstrate that the performance of the APSO-SORBF neural network is good because most of the identification errors are in the range $[-0.05, 0.05]$.

To evaluate the performance of the proposed APSO-SORBF neural network, we conducted comparisons with the GGAP-RBF [27], GAP-RBF [26], multi-innovation RLS with quantum PSO RBF (MRLS-QPSO-RBF) [42], PSO-RBF [35], AI-PSO-RBF [48], and SAIW-PSO-RBF [58] algorithms. Table V shows that, compared with the other algorithms, the APSO-SORBF neural network achieves the most compact network (11 hidden neurons), the lowest testing time, and the lowest training time in this example. Moreover, the APSO-SORBF neural network has better prediction and generalization performance than the other algorithms (except for GGAP-RBF [27]).

TABLE V
COMPARISON OF DIFFERENT ALGORITHMS FOR THE MACKEY-GLASS TIME SERIES PREDICTION

Algorithm	Testing RMSE		No. of hidden neurons	Training time(s)	Testing time(s)
	Mean	Dev.			
APSO-SORBF	0.0135	0.0095	11	832.7	0.0039
GGAP-RBF[27]	0.0121*	—	12*	—	—
GAP-RBF[26]	0.0321*	—	19*	—	—
MRLS-QPSO-RBF[42]	0.0168*	—	25*	—	—
PSO-RBF [35]	0.0208	0.0249	12	859.6	0.0047
AI-PSO-RBF[48]	0.0189	0.0132	11	1183.6	0.0043
SAIW-PSO-RBF[58]	0.0166	0.0145	11	1135.9	0.0053

* The results are listed in the original papers.

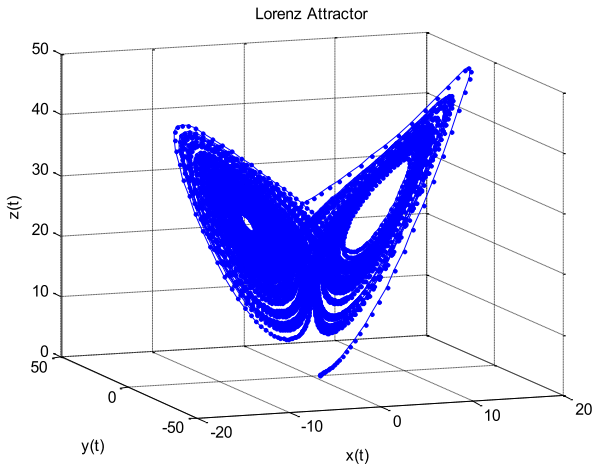

Fig. 6. Lorenz attractor.

D. Lorenz Time Series Prediction

The Lorenz time series system is a mathematical model for atmospheric convection, which is also widely used as a benchmark in many applications [42]. As a 3-D and highly nonlinear system, the Lorenz system is governed by

$$\begin{cases} dx(t)/dt = a_1 y(t) - a_1 x(t) \\ dy(t)/dt = a_2 x(t) - x(t)z(t) - y(t) \\ dz(t)/dt = x(t)y(t) - a_3 z(t) \end{cases} \quad (59)$$

where a_1, a_2, and a_3 are the system parameter $a_1 = 10$, $a_2 = 28$, and $a_3 = 8/3$; $x(t)$, $y(t)$, and $z(t)$ are the 3-D space vectors of Lorenz system. Fig. 6 gives the trajectory of this Lorenz system.

In this example, the fourth-order Runge–Kutta approach with a step size 0.01 is adopted to generate the Lorenz samples, and only the Y-dimension samples $y(t)$ are used for the time series prediction. For 5000 data samples generated from $y(t)$, the first 2000 samples were taken as training data, and the last 3000 samples were used to check the proposed model. The results are displayed in Fig. 7. The data in Fig. 7 show that the proposed APSO-SORBF neural

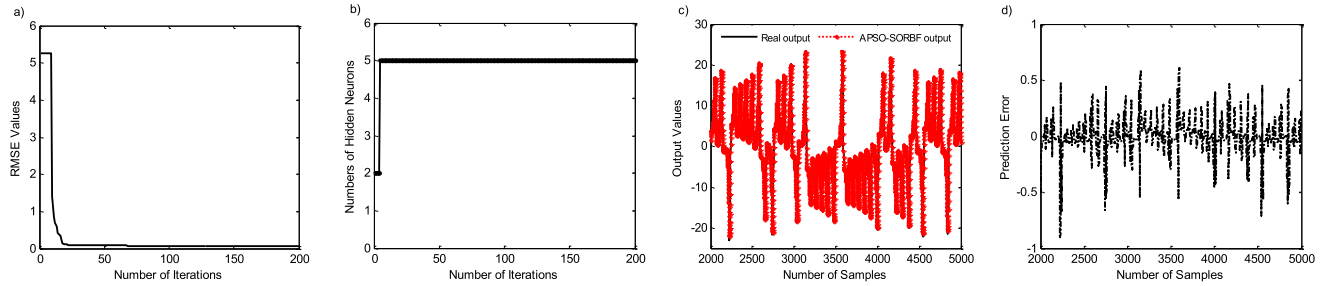

Fig. 7. Experimental results of the proposed APSO-SORBF neural network for Lorenz time series prediction. (a) RMSE values against iterations. (b) Number of hidden neurons against iterations. (c) Prediction results for the APSO-SORBF neural network. (d) Prediction error versus sample number.

TABLE VI
COMPARISON OF DIFFERENT ALGORITHMS FOR THE LORENZ TIME SERIES PREDICTION

Algorithm	Testing RMSE		No. of hidden neurons	Training time(s)	Testing time(s)
	Mean	Dev.			
APSO-SORBF	0.1726	0.054	5	1157.1	0.0069
GAP-RBF[26]	2.3294*	—	70*	—	—
MRLS-QPSO-RBF[42]	0.1822*	—	9*	—	—
PSO-RBF[35]	0.2673	0.095	6	1234.8	0.0076
AI-PSO-RBF[48]	0.2017	0.058	6	2095.5	0.0076
SAIW-PSO-RBF[58]	0.1981	0.073	5	1881.4	0.0072

* The results are listed in the original papers.

network can predict the Lorenz time series system. Moreover, the results are compared with those of GAP-RBF [26], MRL-QPSO-RBF [42], PSO-RBF [35], AI-PSO-RBF [48], and SAIW-PSO-RBF [58] algorithms. The detailed comparison is shown in Table VI.

Table VI shows that both MRLS-QPSO-RBFNN and APSO-SORBF neural network have comparable network sizes and performance. However, the APSO-SORBF neural network can achieve better performance than the MRLS-QPSO-RBFNN. Moreover, the proposed APSO-SORBF neural network has better mean testing RMSE than that of the MRLS-QPSO-RBFNN, which is also an effective approach to track this highly nonstationary Lorenz time series in [42]. In addition, the APSO-SORBF neural network achieves the least testing time and training time than the other algorithms.

E. Effluent TP Prediction in WWTP

The effluent TP is an important parameter to evaluate the performance of WWTP [53]. However, due to the biological characteristics of the activated sludge process, the values of the effluent TP are hard-to-measure. The availability of the effluent TP is often associated with expensive capital and maintenance costs [54]. In this experiment, the proposed APSO-SORBF neural network is used to predict the values of the effluent TP. The easy-to-measure process variables include: influent TP, temperature, oxidation reduction potential, dissolved oxygen, total soluble solid, and pH, which were selected as the input

TABLE VII
COMPARISON OF DIFFERENT ALGORITHMS FOR THE EFFLUENT TP PREDICTION

Algorithm	Testing RMSE		No. of hidden neurons	Training time(s)	Testing time(s)
	Mean	Dev.			
APSO-SORBF	0.0127	0.0025	12	782.7	0.0120
GGAP-RBF[27]	0.0356	0.0085	18	—	0.0630
PSO-RBF[35]	0.1602	0.0915	14	895.2	0.0055
AI-PSO-RBF[48]	0.0191	0.0052	12	1234.8	0.0290
SAIW-PSO-RBF[58]	0.0159	0.0049	12	1158.4	0.0410

variables of the APSO-SORBF neural network. The data were collected from 1/6/2014 to 1/5/2015 of a real WWTP in Beijing, China. After deleting the abnormal data, 2720 samples were obtained and normalized, where 1700 samples were used as the training data, while the remaining 1020 samples were used as the testing data. The performance of APSO-SORBF neural network is compared with GAP-RBF [26], PSO-RBF [35], AI-PSO-RBF [48], and SAIW-PSO-RBF [58] algorithms. The results are shown in Fig. 8 and the details of the comparison are recorded in Table VII. In Fig. 8(a), the real values of the effluent TP and the predicting values of different algorithms are displayed. The prediction errors of different algorithms are shown in Fig. 8(b). It can be seen from Fig. 8 that the proposed APSO-SORBF neural network has comparable prediction performance and the prediction error stay in the interval $[-0.2, 0.5]$.

Table VII shows that the proposed APSO-SORBF neural network has less mean testing RMSE and more compact structure than the other algorithms. Moreover, the testing time and the training time of the APSO-SORBF neural network are least than that of the other algorithms. The above results further show that the APSO algorithm is more suitable to optimize RBFNNs than the AI-PSO and the SAIW-PSO algorithms. The proposed APSO-RBFNN is a more suitable and effective method than the other SORBF neural networks for predicting the effluent TP values.

Remark 11: In all examples, the proposed APSO-SORBF neural network achieves the least testing time than the other algorithms. The training time of APSO-SORBF neural network is more than that of PSO-RBFNN in examples *A* and *B* due to the simple feature of nonlinear dynamics. However, for

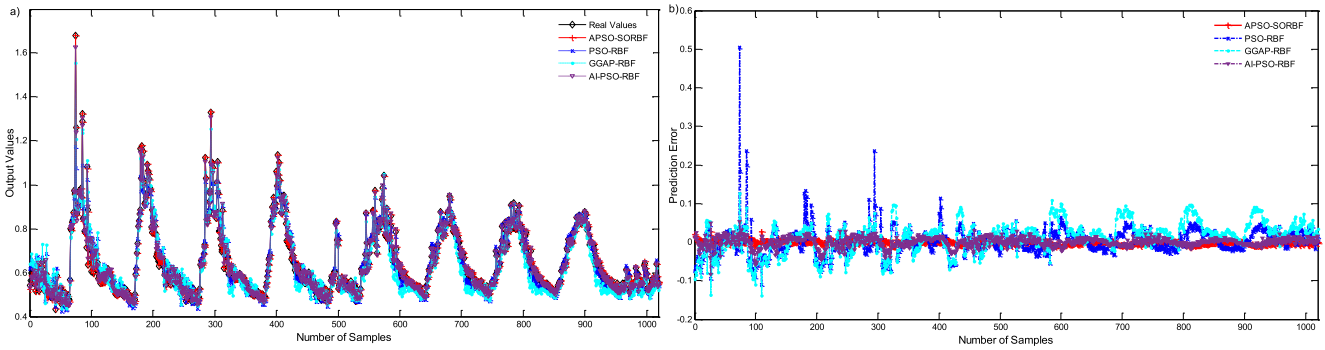

Fig. 8. Experimental results for effluent TP prediction. (a) Prediction results for the different algorithms versus sample number. (b) Prediction error versus sample number.

the more complex and practical examples (examples C–E), the proposed APSO-SORBF neural network achieves the least training time than the other algorithms.

VI. Conclusion

An SORBF neural network for modeling uncertain nonlinear dynamics is presented in this paper. The proposed algorithm is technically built based on the APSO strategy, which performs favorably in terms of both effectiveness and efficiency. It enables the proposed APSO-SORBF neural network to optimize the parameters and network size simultaneously during the learning processes. First, the APSO algorithm can adjust the inertia weights by employing a nonlinear regressive function, enhancing the searching ability of the particles. Meanwhile, to obtain the optimal parameters and size for RBFNNs, a unified criterion is designed for the proposed APSO-SORBF neural network. The experimental results demonstrate that the proposed APSO-SORBF neural network is more effective in solving nonlinear learning problems than some other existing SORBF neural networks.

Some theoretical results on the algorithm convergence of our proposed APSO-SORBF neural network are given in this paper. Therefore, this paper can be regarded as an essential step for real-world applications. Our simulation results over a number of simulated time series and an engineering modeling task demonstrate the good potential of the proposed techniques in real-world applications. Such applications include the dynamical estimate of some important parameters in WWTP systems and the adaptive control of some uncertain nonlinear systems.

Acknowledgment

The authors would like to thank Prof. G. Feng from Hong Kong City University, and the Associate Professor D. Wang from La Trobe University, for their helps in reading the manuscript, providing valuable comments, and improving the linguistic quality of this paper. They would also like to thank the Editor-in-Chief and the reviewers for their valuable comments and suggestions, which helped improve this paper greatly.

References

[1] Y. W. Wong, K. P. Seng, and L.-M. Ang, "Radial basis function neural network with incremental learning for face recognition," *IEEE Trans. Syst. Man Cybern. B*, vol. 41, no. 4, pp. 940–949, Aug. 2011.

[2] H.-G. Han, X.-L. Wu, and J.-F. Qiao, "Real-time model predictive control using a self-organizing neural network," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 24, no. 9, pp. 1425–1436, Sep. 2013.

[3] S.-C. Huang and B.-H. Do, "Radial basis function based neural network for motion detection in dynamic scenes," *IEEE Trans. Cybernetics*, vol. 44, no. 1, pp. 114–125, Jan. 2014.

[4] J. Park and I. W. Sandberg, "Universal approximation using radial-basis-function networks," *Neural Comput.*, vol. 3, no. 2, pp. 246–257, Mar. 1991.

[5] G.-B. Huang, L. Chen, and C.-K. Siew, "Universal approximation using incremental constructive feedforward networks with random hidden nodes," *IEEE Trans. Neural Netw.*, vol. 17, no. 4, pp. 879–892, Jul. 2006.

[6] P. Singla, K. Subbarao, and J. L. Junkins, "Direction-dependent learning approach for radial basis function networks," *IEEE Trans. Neural Netw.*, vol. 18, no. 1, pp. 203–222, Jan. 2007.

[7] A. Feraz and S. A. William, "Predicting single-neuron activity in locally connected networks," *Neural Comput.*, vol. 24, no. 10, pp. 2655–2677, Oct. 2012.

[8] C. Arteaga and I. Marrero, "Universal approximation by radial basis function networks of Delsarte translates," *Neural Netw.*, vol. 46, pp. 299–305, Oct. 2013.

[9] R. Zhang, Z. B. Xu, G. B. Huang, and D. H. Wang, "Global convergence of online BP training with dynamic learning rate," *IEEE Trans. Neural Netw. Learn.Systems*, vol. 23, no. 2, pp. 330–341, Feb. 2012.

[10] Z. B. Xu, R. Zhang, and W. F. Jing, "When does online bp training converge?" *IEEE Trans. Neural Netw.*, vol. 20, no. 10, pp. 1529–1539, Oct. 2009.

[11] B. Chen, S. Zhao, P. Zhu, and J. C. Príncipe, "Quantized kernel recursive least squares algorithm," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 24, no. 9, pp. 1484–1491, Sep. 2013.

[12] M. S. Al-Batah, N. A. M. Isa, K. Z. Zamli, and K. A. Azizli, "Modified recursive least squares algorithm to train the hybrid multilayered perceptron (HMLP) network," *Appl. Soft Comput.*, vol. 10, no. 1, pp. 236–244, Jan. 2010.

[13] J. Jiang and Y. M. Zhang, "A novel variable-length sliding window blockwise least-squares algorithm for on-line estimation of time-varying parameters," *Int. J. Adapt. Control Signal Process.*, vol. 18, no. 6, pp. 505–521, Aug. 2004.

[14] J. X. Peng, K. Li, and G. W. Irwin, "A novel continuous forward algorithm for RBF neural modelling," *IEEE Trans. Autom. Control*, vol. 52, no. 1, pp. 117–122, Jan. 2007.

[15] J. F. Qiao and H. G. Han, "Identification and modeling of nonlinear dynamical systems using a novel self-organizing RBF-based approach," *Automatica*, vol. 48, no. 8, pp. 1729–1734, Aug. 2014.

[16] T. Xie, H. Yu, J. Hewlett, P. Rozycki, and B. Wilamowski, "Fast and efficient second-order method for training radial basis function networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 4, pp. 609–619, Apr. 2012.

[17] S. Kiranyaz, T. Ince, A. Yildirim, and M. Gabbouj, "Evolutionary artificial neural networks by multi-dimensional particle swarm optimization," *Neural Netw.*, vol. 22, no. 10, pp. 1448–1462, Dec. 2009.

[18] O. Buchtala, M. Klimek, and B. Sick, "Evolutionary optimization of radial basis function classifiers for data mining applications," *IEEE Trans. Syst. Man B (Cybernetics)*, vol. 35, no. 5, pp. 928–947, Oct. 2005.

[19] D. Wu *et al.*, "Prediction of parkinson's disease tremor onset using a radial basis function neural network based on particle swarm optimization," *Int. J. Neural Syst.*, vol. 20, no. 2, pp. 109–116, Apr. 2010.

[20] P. A. Gutierrez, C. Hervas-Martinez, and F. J. Martinez-Estudillo, "Logistic regression by means of evolutionary radial basis function neural networks," *IEEE Trans. Neural Netw.*, vol. 22, no. 2, pp. 246–263, Feb. 2011.

[21] T. H. Oong and N. A. M. Isa, "Adaptive evolutionary artificial neural networks for pattern classification," *IEEE Trans. Neural Netw.*, vol. 22, no. 11, pp. 1823–1836, Nov. 2011.

[22] P. Larrañaga, H. Karshenas, C. Bielza, and R. Santana, "A review on evolutionary algorithms in Bayesian network learning and inference tasks," *Inf. Sci.*, vol. 233, no. 1, pp. 109–125, Jun. 2013.

[23] D. S. Yeung, W. W. Y. Ng, D. Wang, E. C. C. Tsang, and X.-Z. Wang, "Localized generalization error model and its application to architecture selection for radial basis function neural network," *IEEE Trans. Neural Netw.*, vol. 18, no. 5, pp. 1294–1305, Sep. 2007.

[24] V. Tomenko, "Online dimensionality reduction using competitive learning and radial basis function network," *Neural Netw.*, vol. 24, no. 5, pp. 501–511, Jun. 2011.

[25] J. Lefebvre and T. J. Perkins, "Neural population densities shape network correlations," *Phys. Rev. E*, vol. 85, no. 2, p. 021914, Feb. 2012.

[26] G.-B. Huang, P. Saratchandran, and N. Sundararajan, "An efficient sequential learning algorithm for growing and pruning RBF (GAP-RBF) networks," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 34, no. 6, pp. 2284–2292, Dec. 2004.

[27] G.-B. Huang, P. Saratchandran, and N. Sundararajan, "A generalized growing and pruning RBF (GGAP-RBF) neural network for function approximation," *IEEE Trans. Neural Netw.*, vol. 16, no. 1, pp. 57–67, Jan. 2005.

[28] N. Vuković and Z. Miljković, "A growing and pruning sequential learning algorithm of hyper basis function neural network for function approximation," *Neural Netw.*, vol. 46, pp. 210–226, Oct. 2013.

[29] H.-G. Han, Q.-L. Chen, and J.-F. Qiao, "An efficient self-organizing RBF neural network for water quality prediction," *Neural Netw.*, vol. 24, no. 7, pp. 717–725, Sep. 2011.

[30] S.-F. Su, C.-C. Chuang, C. W. Tao, J.-T. Jeng, and C.-C. Hsiao, "Radial basis function networks with linear interval regression weights for symbolic interval data," *IEEE Trans. Syst. Man Cybernetics B*, vol. 42, no. 1, pp. 69–80, Feb. 2012.

[31] G. S. Babu and S. Suresh, "Sequential projection-based metacognitive learning in a radial basis function network for classification problems," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 24, no. 2, pp. 194–206, Feb. 2013.

[32] L. Zhang, K. Li, H. He, and G. W. Irwin, "A new discrete-continuous algorithm for radial basis function networks construction," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 24, no. 11, pp. 1785–1798, Nov. 2013.

[33] H. Han, W. Zhou, J. Qiao, and G. Feng, "A direct self-constructing neural controller design for a class of nonlinear systems," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 6, pp. 1312–1322, Jun. 2015.

[34] J. Tian, M. Li, F. Chen, and N. Feng, "Learning subspace-based RBFNN using coevolutionary algorithm for complex classification tasks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 1, pp. 47–61, Jan. 2016.

[35] H. M. Feng, "Self-generation RBFNs using evolutional PSO learning," *Neurocomputing*, vol. 70, nos. 1–3, pp. 241–251, Dec. 2006.

[36] A. Alexandridis, E. Chondrodima, and H. Sarimveis, "Radial basis function network training using a nonsymmetric partition of the input space and particle swarm optimization," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 24, no. 2, pp. 219–230, Feb. 2013.

[37] C.-M. Lee and C.-N. Ko, "Time series prediction using RBF neural networks with a nonlinear time-varying evolution PSO algorithm," *Neurocomputing*, vol. 73, nos. 1–3, pp. 449–460, Dec. 2009.

[38] H. Quan, D. Srinivasan, and A. Khosravi, "Particle swarm optimization for construction of neural network-based prediction intervals," *Neurocomputing*, vol. 127, no. 15, pp. 172–180, Mar. 2014.

[39] V. Fathi and G. A. Montazer, "An improvement in RBF learning algorithm based on PSO for real time applications," *Neurocomputing*, vol. 111, no. 2, pp. 169–176, Jul. 2013.

[40] M. Han, J. Fan, and J. Wang, "A dynamic feedforward neural network based on Gaussian particle swarm optimization and its application for predictive control," *IEEE Trans. Neural Netw.*, vol. 22, no. 9, pp. 1457–1468, Sep. 2011.

[41] L. Zhang, K. Li, and E.-W. Bai, "A new extension of newton algorithm for nonlinear system modelling using RBF neural networks," *IEEE Trans. Autom. Control*, vol. 58, no. 11, pp. 2929–2933, Nov. 2013.

[42] H. Chen, Y. Gong, and X. Hong, "Online modeling with tunable RBF network," *IEEE Trans. Cybernetics*, vol. 43, no. 3, pp. 935–947, Jun. 2013.

[43] Q. Fan, J. M. Zurada, and W. Wu, "Convergence of online gradient method for feedforward neural networks with smoothing $L_1/2$ regularization penalty," *Neurocomputing*, vol. 131, no. 5, pp. 208–216, May 2014.

[44] M. A. Costa, A. P. Braga, and B. R. de Menezes, "Convergence analysis of sliding mode trajectories in multi-objective neural networks learning," *Neural Netw.*, vol. 33, pp. 21–31, Sep. 2012.

[45] Y. V. Pehlivanoglu, "A new particle swarm optimization method enhanced with a periodic mutation strategy and neural networks," *IEEE Trans. Evol. Comput.*, vol. 17, no. 3, pp. 436–452, Jun. 2013.

[46] L. M. Zhang, Y. G. Tang, C. C. Hua, and X. P. Guan, "A new particle swarm optimization algorithm with adaptive inertia weight based on Bayesian techniques," *Appl. Soft Comput.*, vol. 28, pp. 138–149, Mar. 2015.

[47] B. Jiao, Z. Lian, and X. Gu, "A dynamic inertia weight particle swarm optimization algorithm," *Chaos Solitons & Fractals*, vol. 37, no. 3, pp. 698–705, Aug. 2008.

[48] A. Nickabadi, M. M. Ebadzadeh, and R. Safabakhsh, "A novel particle swarm optimization algorithm with adaptive inertia weight," *Appl. Soft Comput.*, vol. 11, no. 4, pp. 3658–3670, Jun. 2011.

[49] H. Wang, "Diversity enhanced particle swarm optimization with neighborhood search," *Inf. Sci.*, vol. 223, pp. 119–135, Feb. 2013.

[50] O. M. Nezami, A. Bahrampour, and P. Jamshidlou, "Dynamic diversity enhancement in particle swarm optimization (DDEPSO) algorithm for preventing from premature convergence," *Proc. Comput. Sci.*, vol. 24, no. 1, pp. 54–65, Jan. 2013.

[51] M. A. Semenov and D. A. Terkel, "Analysis of convergence of an evolutionary algorithm with self-adaptation using a stochastic lyapunov function," *Evol. Comput.*, vol. 11, no. 4, pp. 363–379, Dec. 2003.

[52] M. Jiang, Y. P. Luo, and S. Y. Yang, "Stochastic convergence analysis and parameter selection of the standard particle swarm optimization algorithm," *Inf. Process. Lett.*, vol. 102, no. 1, pp. 8–16, Apr. 2007.

[53] R. Barat, T. Montoya, A. Seco, and J. Ferrer, "Modelling biological and chemically induced precipitation of calcium phosphate in enhanced biological phosphorus removal systems," *Water Res.*, vol. 45, no. 12, pp. 3744–3752, Jun. 2011.

[54] Y. Ouyang, T. D. Leininger, and J. Hatten, "Real-time estimation of TP load in a Mississippi Delta stream using a dynamic data driven application system," *J. Environ. Manage.*, vol. 122, pp. 37–41, Jun. 2013.

[55] M. Hu, T. Wu, and J. D. Weir, "An adaptive particle swarm optimization with multiple adaptive methods," *IEEE Trans. Evol. Comput.*, vol. 17, no. 5, pp. 705–720, Oct. 2013.

[56] G. Xu, "An adaptive parameter tuning of particle swarm optimization algorithm," *Appl. Math. Comput.*, vol. 219, no. 9, pp. 4560–4569, Jan. 2013.

[57] H. Wang, W. Wang, and Z. J. Wu, "Particle swarm optimization with adaptive mutation for multimodal optimization," *Appl. Math. Comput.*, vol. 221, pp. 296–305, Sep. 2013.

[58] M. Taherkhani and R. Safabakhsh, "A novel stability-based adaptive inertia weight for particle swarm optimization," *Appl. Soft Comput.*, vol. 38, pp. 281–295, Jan. 2016.

Hong-Gui Han (M'11–SM'15) received the B.E. degree in automatic from the Civil Aviation University of China, Tianjin, China, in 2005, and the M.E. and Ph.D. degrees from the Beijing University of Technology, Beijing, China, in 2007 and 2011, respectively.

He has been with the Beijing University of Technology since 2011, where he is currently a Professor. His current research interests include neural networks, fuzzy systems, intelligent systems, modeling and control in process systems, and civil and environmental engineering.

Prof. Han is a member of the IEEE Computational Intelligence Society. He is also a Reviewer of the IEEE TRANSACTIONS ON FUZZY SYSTEMS, the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, the IEEE TRANSACTIONS ON CYBERNETICS, the IEEE TRANSACTIONS ON CONTROL SYSTEMS TECHNOLOGY, *Control Engineering Practice*, and *Journal of Process Control*.

Wei Lu received the B.E. degree in automatic control from Heze University, Heze, China, in 2015. She is currently pursuing the M.E. degree with the Beijing Key Laboratory of Computational Intelligence and Intelligent System, Faculty of Information Technology, Beijing University of Technology, Beijing, China.

Her current research interests include neural networks, intelligent systems, modeling, and control in process systems.

Ying Hou received the B.E. and M.E. degrees in control engineering from the Beijing University of Technology, Beijing, China, in 2004 and 2007, respectively, where she is currently pursuing the Ph.D. degree with the Beijing Key Laboratory of Computational Intelligence and Intelligent System, Faculty of Information Technology.

Her current research interests include neural networks, self-adaptive/learning systems, modeling, control, and optimization in process systems.

Jun-Fei Qiao (M'11) received the B.E. and M.E. degrees in control engineering from Liaoning Technical University, Fuxin, China, in 1992 and 1995, respectively, and the Ph.D. degree from Northeast University, Shenyang, China, in 1998.

From 1998 to 2000, he was a Post-Doctoral Fellow with the School of Automatics, Tianjin University, Tianjin, China. He joined the Beijing University of Technology, Beijing, China, where he is currently a Professor. He is the Director of the Intelligence Systems Laboratory, Beijing, China. His current research interests include neural networks, intelligent systems, self-adaptive/learning systems, and process control systems.

Prof. Qiao is a member of the IEEE Computational Intelligence Society. He is also a Reviewer for more than 20 international journals, such as the IEEE TRANSACTIONS ON FUZZY SYSTEMS and the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS.