Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2013, Article ID 595639, 10 pages
http://dx.doi.org/10.1155/2013/595639

Research Article

Training ANFIS Model with an Improved Quantum-Behaved Particle Swarm Optimization Algorithm

Peilin Liu,[1,2] Wenhao Leng,[3] and Wei Fang[4]

[1] *School of Digital Media, Jiangnan University, Wuxi, Jiangsu 214122, China*
[2] *Department of Software, Wuxi Institute of Technology, Wuxi, Jiangsu 214122, China*
[3] *Department of Information Technology, China Ship Science Research Centre, Wuxi 214082, China*
[4] *Key Laboratory of Advanced Control for Light Industry (Ministry of Education, China), Jiangnan University, Wuxi, Jiangsu 214122, China*

Correspondence should be addressed to Peilin Liu; peilin_liu1972@sina.cn

Received 30 March 2013; Accepted 23 May 2013

Academic Editor: Yi-Kuei Lin

This paper proposes a novel method of training the parameters of adaptive-network-based fuzzy inference system (ANFIS). Different from the previous works which emphasized on gradient descent (GD) method, we present an approach to train the parameters of ANFIS by using an improved version of quantum-behaved particle swarm optimization (QPSO). This novel variant of QPSO employs an adaptive dynamical controlling method for the contraction-expansion (CE) coefficient which is the most influential algorithmic parameter for the performance of the QPSO algorithm. The ANFIS trained by the proposed QPSO with adaptive dynamical CE coefficient (QPSO-ADCEC) is applied to five example systems. The simulation results show that the ANFIS-QPSO-ADCEC method performs much better than the original ANFIS, ANFIS-PSO, and ANFIS-QPSO methods.

1. Introduction

Fuzzy systems (FSs) have been successfully applied in many areas, such as system modeling and controls. To ease the design and improve system performance, many neural or statistical learning approaches that automatically generate fuzzy rules have been proposed [1]. A fuzzy inference system employing fuzzy if-then rules can model the qualitative aspects of human knowledge and reasoning processes without using precise quantitative analyses. This fuzzy modeling or fuzzy identification, first explored systematically by Sugeno and Kang [2], has found numerous practical applications in control [3, 4], prediction, and inference [5, 6]. However, there are some basic aspects of this approach which are in need of better understanding. First, there are no standard methods for transforming human knowledge or experience into the rule base and data base of a fuzzy inference system. Second, there is a need for effective methods for tuning the membership functions (MF) so as to minimize the output error measure or maximize performance index.

This paper concentrates on a so-called adaptive-network-based fuzzy inference system (ANFIS), which can serve as a basis for constructing a set of fuzzy if-then rules with appropriate membership functions to generate the stipulated input-output pairs [7]. The ANFIS model is a reprehensive of adaptive fuzzy systems which are generated through training processes and are known as evolving fuzzy systems. It has been one of attractive researches focusing on fuzzy systems in recent years.

The TSK [2] is a fuzzy system with crisp functions and has been found to be efficient in complex applications [4]. It has been proved that, with proper number of rules, a TSK system is able to approximate every plant. As such, TSK systems are widely used in the ANFIS and play the advantage of good applicability since they can be interpreted as local linearization modeling and conventional linear techniques for state estimation and control.

The ANFIS has both the advantages of neural networks and fuzzy systems. However, training the parameters of the ANFIS model is one of the main issues encountered when

the model is applied to the real-world problems. Most of the training methods for the ANFIS are based on gradient descent (GD) approaches, where calculation of gradient in each step is tractable since the chain rule used may cause many local minima of the problem. The gradient methods are known to be local search approaches and their performances generally depend on initial values of parameters so that it is difficult for them to find the global optimal model parameters. Since the design of FSs can be reduced to an optimization problem, many researchers have proposed to design FSs by employing metaheuristics such as genetic algorithms (GAs) [8–10] and particle swarm optimization (PSO) [11–13]. However, GAs have always been complaint about their slow convergence speed, while PSO may encounter premature convergence at the later stage of the search process and is sensitive to neighborhood topology.

This paper explores the applicability of a variant of PSO, quantum-behaved particle swarm optimization (QPSO), to training of the ANFIS model. The QPSO algorithm was inspired by quantum mechanics [14–17]. Its iterative equation is very different from that of PSO and can lead QPSO to be globally convergent [18–20]. Besides, unlike PSO, QPSO needs no velocity vectors for particles and has fewer parameters to adjust, making it easier to implement. The QPSO algorithm has been shown to successfully solve a wide range of continuous optimization problems. Many empirical studies show that QPSO has stronger global search ability when solving various continuous optimization problems [21–27]. In order to make a further improvement of the QPSO, in this paper, we propose an adaptive dynamical control method for the contraction-expansion (CE) coefficient of the algorithm, which is the most influential algorithmic parameter that can be tuned to adjust the balance between the local search and global search of the particle. The improved QPSO algorithm is applied to ANFIS training and is tested on several example systems, with performance comparison between the improved QPSO, the original QPSO, and PSO algorithms.

The rest of the paper is organized as follows: in Section 2, we present a review of fuzzy systems and the ANFIS model. Section 3 presents the principle of QPSO. The improved QPSO approach is proposed in Section 4. Application of the QPSO and its improved version to ANFIS model training is described in Section 5. Section 6 presents the simulation results for five example systems using ANFIS model trained by the optimization algorithms. Finally, the paper is concluded in Section 7.

2. Fuzzy Systems and ANFIS Model

2.1. Fuzzy Systems. This subsection describes the fuzzy systems to be optimized through training processes in this study and their mathematical expressions. The TSK fuzzy model was originally proposed by Sugeno and Kang in an effort to formalize a systematic approach to generating fuzzy rules from an input-output data set [2]. The Takagi-Sugeno-Kang (TSK) fuzzy system can be of zero order or first order. The ith rule, which is denoted R_i, in a TSK fuzzy system is represented in the following form:

R_i: if $x_1(k)$ is A_{i1} and$\cdots$and $x_n(k)$ is A_{in}, then $y(k)$ is $f_i, i = 1, \ldots, r$,

where k is the number of time step, $x_1(k), \ldots, x_n(k)$ are input variables, $y(k)$ is the output variable of the system, A_{ij} is a fuzzy set, and f_i is the consequent part function. The fuzzy set A_{ij} uses a bell-shaped membership function given by

$$\mu(x) = \frac{1}{1 + \left(\left(x_j - c_{ij}\right)/a_{ij}\right)^{2b_{ij}}}, \tag{1}$$

where a_i, b_i, and c_i are the parameter set of the fuzzy set A_{ij}. Each of these parameters has a physical meaning: c_i determines the center of the corresponding membership function, a_i is the half width, and b_i (together with a_i) controls the slopes at the crossover points. In the inference engine, the fuzzy AND operation is implemented by the algebraic product in fuzzy theory. Thus, given an input dataset $x = (x_1, \ldots, x_n)$, the firing strength $\mu(x)$ of rule i is calculated by

$$\mu(\overline{x}) = \prod_{j=1}^{n} M_{ij}(x) = \prod_{j=1}^{n} \left\{ \frac{1}{1 + \left(\left(x_j - c_{ij}\right)/a_{ij}\right)^{2b_{ij}}} \right\}. \tag{2}$$

For zero-order and first-order TSK-type fuzzy systems, the consequent function f_i is set to a real value p_{i0} and a linear function of input variables $p_{i0} + \sum_{j=1}^{n} p_{ij}x_j$ for each i, respectively. If there are r rules in a fuzzy system, the output of the system, which is calculated by the weighted-average-defuzzification method, is given by

$$y = \frac{\sum_{i=1}^{r} \mu_i(\overline{x})\, f_i}{\sum_{i=1}^{r} \mu_i(x)_i}. \tag{3}$$

In this work, the proposed improved QPSO algorithm and other optimization algorithms will be used to optimize free parameters a_i, b_i, and c_i in each rule.

2.2. The Architecture of ANFIS. This subsection presents the architecture of ANFIS network. A detailed coverage of ANFIS can be found in [7]. The ANFIS network is a neurofuzzy network that was proposed by Jang in 1993 [6]. Since the ANFIS is an adaptive network, parts of its nodes are adaptive, which means that their outputs depend on the parameters belonging to these nodes. Two kinds of learning algorithms have been proposed to tune these parameters to optimize the approaching performance during the training period. For simplicity, the above-mentioned system is supposed to have two inputs and one output, and its rule base contains two fuzzy if-then rules of TSK fuzzy model. A typical TSK fuzzy model has the two rules that can be stated as

Rule 1: if x is A_1 and y is B_1, then $f_1 = p_1x + q_1y + r_1$,

Rule 2: if x is A_2 and y is B_2, then $f_2 = p_2x + q_2y + r_2$,

where x and y are the inputs of the ANFIS, A and B are the fuzzy sets, and f_i $(i = 1, 2)$ is a first-order polynomial and represents the outputs of the first-order TSK fuzzy inference system. In the above rules, p_i, q_i, and r_i $(i = 1, 2)$ are the parameters set, referred to as the consequent parameters.

The architecture of ANFIS is shown in Figure 1, and the node function in each layer is described below.

Layer 1. This layer is the layer of membership functions that contains adaptive nodes with node functions described as

$$O_i^1 = \mu_{Ai}(x), \quad O_i^1 = \mu_{B(i-2)}(y), \quad (i = 1, 2), \tag{4}$$

where x and y are the input nodes, A and B are the linguistic labels, $\mu(x)$ and $\mu(y)$ are the membership functions which usually adopts a bell shape with the maximum and minimum values equal to 1 and 0, respectively:

$$\mu(x) = \frac{1}{1 + \left((x - c_i)/a_i\right)^{2b_i}}, \tag{5}$$

where a_i, b_i, and c_i are the parameters set. When values of these parameters change, the bell-shaped functions vary accordingly, exhibiting various forms of membership functions on linguistic labels A and B. In fact, any continuous and piecewise differentiable functions, such as commonly used trapezoidal or triangular-shaped membership functions, are also qualified candidates for node functions in this layer. Parameters in this layer are referred to as premise parameters.

Layer 2. Every node in this layer is a fixed node, marked by a circle and labeled Π, with the node function to be multiplied by input signals to serve as output. Consider

$$O_i^2 = \omega_i = \mu_{Ai}(x) \cdot \mu_{Bi}(y), \quad (i = 1, 2). \tag{6}$$

The output ω_i represents the firing strength of a rule. The output of each node represents the firing strength of a rule. In fact, other T-norm operators that perform the generalized AND can be used as the node function in this layer.

Layer 3. Every node in this layer is a fixed node, marked by a circle and labeled N, with the node function to normalize the firing strength by calculating the ratio of the ith node firing strength to the sum of all rules' firing strength. Moreover,

$$O_i^3 = \varpi_i = \frac{\omega_i}{\sum \omega_i} = \frac{\omega_1}{\omega_1 + \omega_2}, \quad (i = 1, 2). \tag{7}$$

For convenience, the outputs of this layer are referred to as normalized firing strengths.

Layer 4. Every node in this layer is an adaptive node, marked by a square, with node function given by

$$O_i^4 = \varpi_i \cdot f_i = \varpi_i (p_i x + q_i y + r_i), \quad (i = 1, 2), \tag{8}$$

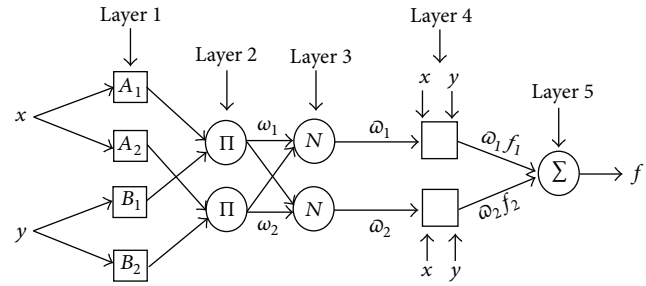

Figure 1: The architecture of ANFIS network.

where ϖ is the output of layer 3 and $\{p_i, q_i, r_i\}$ is the parameter set. The parameters in this layer are referred to as consequent parameters.

Layer 5. Every node in this layer is a fixed node, and the overall output can be expressed as linear combination of the consequent parameters. Consider

$$\begin{aligned} O_i^5 &= f_{\text{out}} = \sum_i \varpi_i \cdot f_i = \varpi_1 f_1 + \varpi_2 f_2 \\ &= \frac{\omega_1}{\omega_1 + \omega_2} f_1 = \frac{\omega_2}{\omega_1 + \omega_2} f_2 \\ &= (\varpi_1 x)\, p_1 + (\varpi_1 y)\, q_1 + (\varpi_1)\, r_1 + (\varpi_2 x)\, p_2 \\ &\quad + (\varpi_2 y)\, q_2 + (\varpi_2)\, r_2. \end{aligned} \tag{9}$$

2.3. Hybrid Learning Algorithm for ANFIS Model. It can be seen that there are two modifiable parameter sets, $\{a_i, b_i, c_i\}$ labeled as premise parameters and $\{p_i, q_i, r_i\}$ labeled as consequent parameters. The aim of the training procedure for this architecture is to tune the above two parameter sets to make the ANFIS output fit the training data. Each epoch of this hybrid learning procedure is composed of two passes: a forward pass and a backward pass. In the forward pass, the premise parameters are fixed, and the least squares estimation (LSE) is applied to identify consequent parameters. When the optimal parameters are found, the backward pass starts with the consequent parameters fixed, the error rate of output node back propagates from output end toward the input end, and the premise parameters are updated by the gradient descent (GD) method.

These methods update premise parameters by using GD or Kalman filtering and appear to be prone to trap into the local optima. In this paper, we propose a QPSO with adaptive dynamical contraction-expansion coefficient (QPSO-ADCEC) and employ this algorithm to train the parameters of ANFIS for the purpose of obtaining the global optimal solution.

3. Quantum-Behaved Particle Swarm Optimization

In the PSO with M individuals, each individual is treated as a volume-less particle in the D-dimensional space, with the current position vector and velocity vector of particle i at the

nth iteration represented as $X_{i,n} = (X_{i,n}^1, X_{i,n}^2, \ldots, X_{i,n}^D)$ and $V_{i,n} = (V_{i,n}^1, V_{i,n}^2, \ldots, V_{i,n}^D)$ [28, 29]. The particle moves according to the following equations:

$$V_{i,n+1}^j = wV_{i,n}^j + c_1 r_{i,n}^j \left(P_{i,n}^j - X_{i,n}^j\right) + c_2 R_{i,n}^j \left(G_{i,n}^j - X_{i,n}^j\right), \tag{10}$$

$$X_{i,n+1}^j = X_{i,n}^j + V_{i,n+1}^j, \tag{11}$$

for $i = 1, 2, \ldots, M$ and $j = 1, 2, \ldots, D$, where c_1 and c_2 are known as acceleration coefficients. The parameter w is known as the inertia weight which can be adjusted to balance the explorative search and the exploitive search of the particle. Vector $P_{i,n} = (P_{i,n}^1, P_{i,n}^2 \cdots P_{i,n}^D)$ is the best previous position (the position giving the best objective function value or fitness value) of particle i and called personal best (*pbest*) position, and vector $G_n = (G_n^1, G_n^2 \cdots G_n^D)$ is the position of the best particle among all the particles in the population and called global best (*gbest*) position. Without loss of generality, we consider the following maximization problem:

$$\begin{aligned} &\text{Minimize} \quad f(X) \\ &\text{s.t.} \quad\quad X \in S \subseteq R^D, \end{aligned} \tag{12}$$

where $f(X)$ is an objective function continuous almost everywhere and S is the feasible space. Accordingly, $P_{i,n}$ can be updated by

$$P_{i,n} = \begin{cases} X_{i,n} & \text{if } f\left(X_{i,n}\right) < f\left(P_{i,n-1}\right), \\ P_{i,n-1} & \text{if } f\left(X_{i,n}\right) \geq f\left(P_{i,n-1}\right), \end{cases} \tag{13}$$

and G_n can be found by $G_n = P_{g,n}$, where $g = \arg\max_{1 \leq i \leq M}[f(p_{i,n})]$. The parameters $r_{i,n}^j$ and $R_{i,n}^j$ are sequences of two different sequences of random numbers distributed uniformly within $(0, 1)$, which is denoted by $r_{i,n}^j, R_{i,n}^j \sim U(0, 1)$. Generally, the value of $V_{i,n}^j$ is restricted in the interval $[-V_{\max}, V_{\max}]$.

Trajectory analysis in [30] showed that convergence of the PSO algorithm may be achieved if each particle converges to its local attractor, $p_{i,n} = (p_{i,n}^1, p_{i,n}^2 \cdots p_{i,n}^D)$, defined at the coordinates

$$p_{i,n}^j = \frac{c_1 r_{i,n}^j p_{i,n}^j + c_2 R_{i,n}^j G_n^j}{c_1 r_{i,n}^j c_2 R_{i,n}^j} \quad 1 \leq j \leq D \tag{14}$$

or

$$p_{i,n}^j = \varphi_{i,n}^j \cdot p_{i,n}^j + \left(1 - \varphi_{i,n}^j\right) \cdot G_n^j, \tag{15}$$

where $\varphi_{i,n}^j = c_1 r_{i,n}^j / (c_1 r_{i,n}^j + c_2 R_{i,n}^j)$ with regard to the random numbers $r_{i,n}^j$ and $R_{i,n}^j$ in (10). In PSO, the acceleration coefficients c_1 and c_2 are generally set to be equal; that is, $c_1 = c_2$, and thus $\varphi_{i,n}^j$ is a sequence of uniformly distributed random numbers over $(0, 1)$. As a result, (15) can be restated as

$$\begin{aligned} p_{i,n}^j &= \varphi_{i,n}^j \cdot P_{i,n}^j + \left(1 - \varphi_{i,n}^j\right) \cdot G_n^j, \\ \phi_{i,n}^j &\sim U(0, 1). \end{aligned} \tag{16}$$

In QPSO, each single particle is treated as a spin-less one moving in quantum space. Thus state of the particle is characterized by wave function ψ, where $|\psi|^2$ is the probability density function of its position. Inspired by convergence analysis of the particle in PSO [30], it is assume that, at the nth iteration, particle i flies in the D-dimensional space with δ potential well centered at $p_{i,n}^j$ on the jth dimension ($1 \leq j \leq D$). Let $Y_{i,n+1}^j = |X_{i,n}^j - p_{i,n}^j|$; we can obtain the normalized wave function at iteration $n + 1$

$$\psi\left(Y_{i,n+1}^j\right) = \frac{1}{\sqrt{L_{i,n}^j}} \exp\left(-\frac{Y_{i,n+1}^j}{L_{i,n}^j}\right), \tag{17}$$

which satisfies the bound condition that $\psi(Y_{i,n+1}^j) \to 0$ as $Y_{i,n+1}^j \to \infty$. $L_{i,n}^j$ is the characteristic length of the wave function. By the statistical interpretation of wave function, the probability density function is given by

$$Q\left(Y_{i,n+1}^j\right) = \left|\psi\left(Y_{i,n}^j\right)\right|^2 = \frac{1}{L_{i,n}^j} \exp\left(-\frac{2Y_{i,n+1}^j}{L_{i,n}^j}\right), \tag{18}$$

and thus the probability distribution function is

$$F\left(Y_{i,n+1}^j\right) = 1 - \exp\left(-\frac{2Y_{i,n+1}^j}{L_{i,n}^j}\right). \tag{19}$$

Using Monte Carlo method, we can measure the jth component of position of particle i at the $(n + 1)$th iteration by

$$X_{i,n+1}^j = p_{i,n+1}^j \pm \frac{L_{i,n}^j}{2} \ln\left(\frac{1}{u_{i,n+1}^j}\right), \quad u_{i,n+1}^j \sim U(0, 1), \tag{20}$$

where $u_{i,n+1}^j$ is a sequence of random numbers uniformly distributed over $(0, 1)$. The value of $L_{i,n}^j$ is determined by

$$L_{i,n}^j = 2\alpha \cdot \left|X_{i,n}^j - C_{i,n}^j\right|, \tag{21}$$

where $C_n = (C_n^1, C_n^2, \ldots, C_n^D)$ is called mean best (*mbest*) position defined by the average of the *pbest* positions of all particles; namely,

$$C_n^j = \left(\frac{1}{M}\right) \sum_{i=1}^{M} p_{i,n}^j \quad (1 \leq j \leq D). \tag{22}$$

Thus the position of the particle updates according to the following equation:

$$X_{i,n+1}^j = p_{i,n}^j \pm \alpha \cdot \left|X_{i,n}^j - C_{i,n}^j\right| \cdot \ln\left(\frac{1}{u_{i,n+1}^j}\right). \tag{23}$$

The parameter α in (21) and (23) is called contraction-expansion (CE) coefficient, which can be adjusted to balance the local search and the global search of the algorithm during the optimization process. The current position of the particle in QPSO is thus updated according to (16) and (23).

The QPSO algorithm starts with the initialization of the particle's current positions and their *pbest* positions (setting $P_{i,0} = X_{i,0}$), followed by the iteration of updating the particle swarm. At each iteration, the *mbest* position of the particle swarm is computed and the current position of each particle is updated according to (16) and (23). Before each particle updates its current position, its fitness value is evaluated, and then its *pbest* position and the current *gbest* position are updated. In (23), the probability of using either operation "+" or operation "−" is equal to 0.5. The search procedure continues until the termination condition is met.

We outline the procedure of the QPSO algorithm as follows.

3.1. Procedure of the QPSO

Step 1. Initialize the population; that is, initialize the current position and personal best position of each particle.

Step 2. Execute the following steps.

Step 3. Compute mean best position C_n according to (22).

Step 4. Properly select the value of α.

Step 5. For each particle in the population, execute from Step 6 to Step 8.

Step 6. Evaluate the objective function value of the current position of the particle, that is, $f(X_{i,n})$.

Step 7. Update $P_{i,n}$ and G_n according to (13).

Step 8. Update each component of the particle's position according to (16) and (23).

Step 9. While the termination condition is not met, return to Step 2.

Step 10. Output the results.

4. The QPSO with Adaptively Dynamical CE Coefficient (QPSO-ADCEC)

The CE coefficient is the most influential algorithmic parameter for the search performance of the QPSO. In [19], the influence of the CE coefficient on particles' dynamical behaviour and the algorithmic performance were theoretically and empirically analyzed, and two control methods for the CE coefficient, linearly decreasing and fixed value methods, were investigated in depth on a suite of well-known benchmark functions. Here, we propose a novel control method for the parameter and thus propose a modified QPSO. The motivation of this proposal is to control the CE coefficient in an adaptive and dynamical way according to evolution process and the diversity of the particle swarm during the search process.

From experience it is well known that the convergence of the QPSO depends on the evolution speed of the fitness values of the particles. Therefore in order to handle complex and nonlinear optimization process, it would be helpful to include such factor into the CE coefficient design. In this paper, we propose a novel dynamically varying CE coefficient for the QPSO involving two factors, namely, the evolution speed factor and the aggregation degree factor. The evolution speed factor of the ith particle at the nth iteration is given by

$$h_{i,n} = \frac{\min\left(f\left(P_{i,n-1}\right), f\left(P_{i,n}\right)\right)}{\max\left(f\left(P_{i,n-1}\right), f\left(P_{i,n}\right)\right)}, \tag{24}$$

where $f(P_{i,n})$ is the fitness value of the personal best position of particle i at iteration n and n is the current iteration number. Note that the evolutionary speed factor lies in $(0, 1]$. This parameter reflects the run time history of the algorithm and the evolution speed of the particle. For the minimization problem described in (12), if the given optimization is a minimization one, $f(P_{i,n}) \leq f(P_{i,n-1})$ for each $n > 0$. Thus $h_{i,n}$ can be simply expressed as $h_{i,n} = f(P_{i,n})/f(P_{i,n-1})$. The smaller the value of $h_{i,n}$, the faster the decrease of $f(P_{i,n})$ and the faster the evolution speed of particle i. After certain number of iterations, the value of $h_{i,n}$ attains its maximum value indicating that $f(P_{i,n})$ is not improved any more, and the algorithm stagnates or finds the optimum solution. The aggregation degree factor s of particle i at the nth iteration is defined by

$$s_{i,n} = \frac{\min\left(f\left(B_n\right), \overline{f}_n\right)}{\max\left(f\left(B_n\right), \overline{f}_n\right)}, \tag{25}$$

where $\overline{f}_n$ is the average fitness value of all the particles at the nth iteration, that is;

$$\overline{f}_n = \frac{1}{M}\sum_{i=1}^{M} f\left(X_{i,n}\right), \tag{26}$$

where M is the swarm size. $f(B_n)$ represents the optimal value that the swarm found in the nth iteration. Note that $f(B_n)$ cannot be replaced by $f(G_n)$ since $f(G_n)$ denotes the optimal value that the entire swarm has found up to the nth iteration. It is obvious that the value of $s_{i,n}$ falls in $(0, 1]$. The aggregation can reflect not only the aggregation degree but also the diversity of the swarm in terms of the fitness values of the current position of the particles. The larger the value of $s_{i,n}$, the less the difference between $f(B_n)$ and $\overline{f}_n$, and thus the more the aggregation or the smaller the swarm diversity. In the case when all the $s_{i,n}$ are equal to 1, all the particles have the same identity with each other in terms of the fitness values of their current positions. The new CE coefficient for each particle can be written as

$$\alpha_{i,n} = \alpha_{\text{initial}} - \left(1 - h_{i,n}\right)\chi + s_{i,n}\lambda, \tag{27}$$

where α_{initial} is the initial value of the CE coefficient and χ and λ typically lie in the range $[0, 1]$. It is obvious that the value of inertia weight lies in between $(1 - \chi)$ and $(1 + \lambda)$. With this adaptively dynamical CE coefficient, the modified QPSO algorithm, called QPSO with adaptively dynamical CE coefficient (QPSO-ADCEC), can have strong ability to jump out of local search than the original QPSO with linearly decreasing or fixed CE coefficient.

5. Learning ANFIS Network by QPSO-ADCEC

This section presents how to employ the QPSO-ADCEC algorithm for learning the ANFIS parameters. The ANFIS has two types of parameters which need to be trained, that is, the premise parameters $\{a_i, b_i, c_i\}$ and the consequent parameters $\{p_i, q_i, r_i\}$. The premise parameters are estimated by the QPSO-ADCEC algorithm, and the consequent parameters are identified by the least squares estimation (LSE). Thus, the position of each particle in the QPSO-ADCEC represents a set of premise parameters. The fitness is defined as root mean squared error (RMSE) between actual output and desired output, which can be expressed by

$$\text{fitness} = \sqrt{\frac{\sum_{i=1}^{n} \left(f(i) - f_0(i)\right)^2}{n}}. \tag{28}$$

The ANFIS coupling with the QPSO-ADCEC algorithm is outlined as below.

Step 1. Initialize the swarm of particles such that the position of each particle are uniformly distributed within the search scope, and set the iteration number $n = 0$.

Step 2. Set the position of each particle $X_{i,n}$ as the premise parameters $\{a_i, b_i, c_i\}$, and identify consequent parameters $\{p_i, q_i, r_i\}$ with LSE. Then calculate fitness value of each particle, and set each particle's personal best position as $P_{i,0} = X_{i,0}$.

Step 3. Find out the mean value of all particles' personal best position C_n by using (22).

Step 4. For each particle in the population, execute from Step 5 to Step 8.

Step 5. Calculate each particle's fitness value, fitness$(X_{i,n})$, and then compare it with the fitness of its personal best position, fitness$(P_{i,n-1})$. If fitness$(X_{i,n})$ < fitness$(P_{i,n-1})$, then $P_{i,n} < X_{i,n}$, otherwise $P_{i,n} < P_{i,n-1}$.

Step 6. Compare the fitness value of each particle's personal best position fitness$(P_{i,n})$ with that of the global best position, fitness(G_{n-1}). If fitness$(P_{i,n})$ < fitness(G_{n-1}), then $G_n = P_{i,n}$, otherwise $G_n = G_{n-1}$.

Step 7. Calculate $h_{i,n}$ and $s_{i,n}$ for each particle according to (24) and (25). Then obtain the value of $\alpha_{i,n}$ by using (27).

Step 8. Update the position of each particle $X_{i,n}$ by using (16) and (23), where α is replaced by $\alpha_{i,n}$.

Step 9. If the termination condition is met, go to exit, otherwise go to Step 2, and set $n = n + 1$.

6. Simulation Results

To investigate the efficiency of the proposed method, five example systems were tested. The first two examples are the problem of identification of nonlinear systems. For the third example, ANFIS-QPSO-ADCEC was used as an identifier to identify a nonlinear component discrete control system. In the simulation study for the fourth example, we used the proposed method to predict a chaotic time series. The fifth example is to use the proposed ANFIS-QPSO-ADCEC method to predict the carbon dioxide in Box-Jenkins Furnace. For all the examples, besides the ANFIS-QPSO-ADCEC method, the ANFIS model trained with BP algorithm (ANFIS-BP), ANFIS with PSO (ANFIS-PSO), and ANFIS with QPSO (ANFIS-QPSO) were also tested and compared with the proposed ANFIS-QPSO-ADCEC. For the PSO, the inertia weight was set to be 0.73, and the acceleration coefficients were set as $c_1 = c_2 = 1.49$. For the QPSO algorithm, the CE coefficient was decreased linearly from 1.0 to 0.5 over

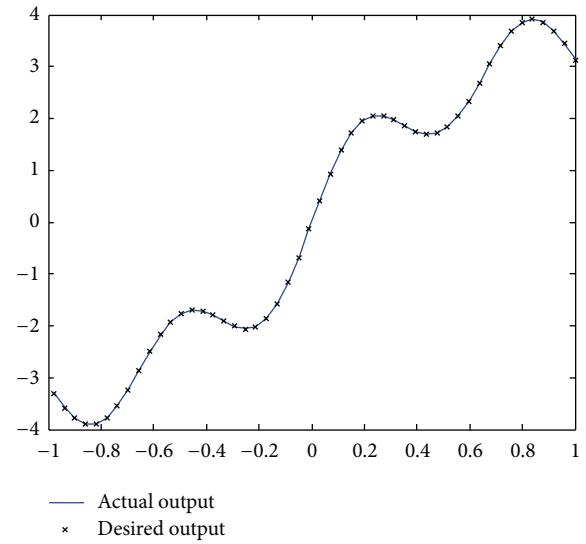

Figure 2: The actual output and desired output by ANFIS-QPSO-ADCEC for Example 1.

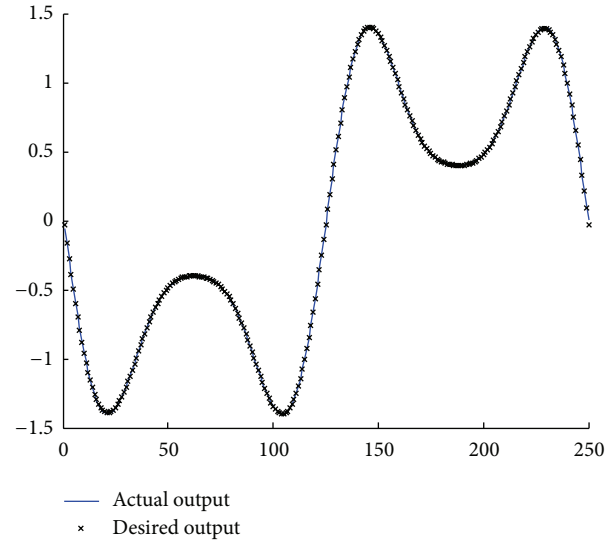

Figure 3: The actual output and desired output by ANFIS-QPSO-ADCEC for Example 2.

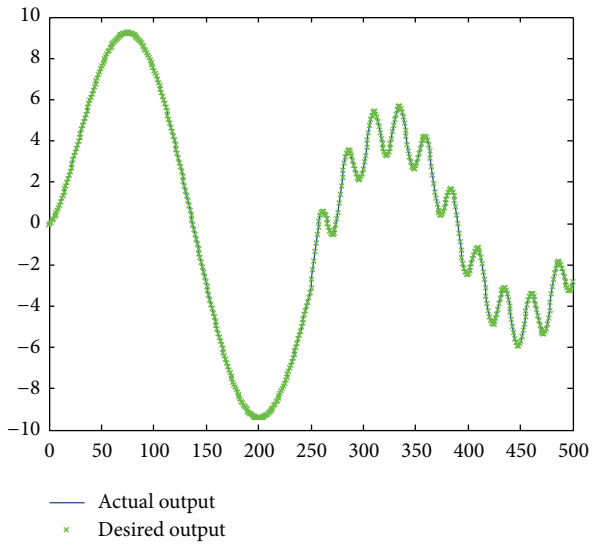

Figure 4: The actual output and desired output by ANFIS-QPSO-ADCEC for Example 3.

the course of search. For the QPSO-ADCEC, the initial CE coefficient $\alpha_{\text{initial}} = 0.5$, the value of χ and λ were set to be 0.6 and 0.4, respectively. These parameters were shown to result in good algorithmic performance in our preliminary experiments. For each of the PSO variants, the swarm size was set to be 50, and the maximum number of iterations was 2000.

Example 1 (nonlinear SISO System modeling [1]). In this example, the nonlinear plant to be identified by first-order TSK-type fuzzy system is described as

$$y = \sin(\pi x) + 0.8\sin(3\pi x) + 0.2\sin(5\pi x), \tag{29}$$

where x is the only input. The fuzzy system had five rules with the membership function of each rule being Gaussian function. Each member function had two parameters and there were 10 premise parameters, and 10 consequent parameters in the fuzzy system. Thus the dimensionality of the position of each particle is 10. We chose 100 input data which were randomly generated between −1 and 1, 50 of which were used as the training data and the remainder of which were employed as testing data.

Table 1 lists the training error and test error of each method, from which we can see that the training errors (trnRMSE) and test errors (testRMSE) of the ANFIS network trained by the QPSO-ADCEC, QPSO, and PSO are smaller than those of the ANFIS trained by BP algorithm. Among all the PSO variants, the QPSO-ADCEC was able to train the ANFIS model with the smallest training and test errors. Figure 2 visualizes results by using the ANFIS-QPSO-ADCEC for identification of the system and shows that the ANFIS model learned by the QPSO-ADCEC fits the nonlinear system very well.

Table 1: Results for Example 1.

	trnRMSE	testRMSE
ANFIS-BP	$1.665e-02$	$1.675e-02$
ANFIS-PSO	$9.796e-03$	$9.905e-03$
ANFIS-QPSO	$6.301e-03$	$7.620e-03$
ANFIS-QPSO-ADCEC	$1.486e-03$	$2.398e-03$

Table 2: Results for Example 2.

	trnRMSE	testRMSE
ANFIS-BP	$4.9757e-03$	$5.0329e-03$
ANFIS-PSO	$3.1639e-03$	$3.8402e-03$
ANFIS-QPSO	$2.9343e-03$	$3.2390e-03$
ANFIS-QPSO-ADCEC	$1.2611e-03$	$1.8366e-03$

Example 2. Nonlinear MISO System modeling is described as

$$\begin{aligned} y = {} & \sin(\pi x_1) + 0.8\sin(3\pi x_1) + 0.2\sin(5\pi x_1) \\ & + 0.6\sin(2\pi x_1) + 0.6\sin(4\pi x_2) \\ & + 0.1\sin(5\pi x_2) \\ & + 0.2\sin(3\pi x_2) + 0.3\sin(2\pi x_3) \\ & + 0.5\sin(\pi x_3). \end{aligned} \tag{30}$$

Equation (30) is three-input nonlinear function. Each variable had two membership functions so that the fuzzy system had eight fuzzy rules [1]. Each membership function was Gaussian with 2 parameters, and the fuzzy system had 12 premise parameters and 32 consequent parameters. Thus, the dimensionality of the position of each particle in QPSO-ADCEC was 12. From the grid points of the range $[-1, 1] \times [-1, 1] \times [-1, 1]$ within the input space of the above equation, 500 data pairs were obtained, where 250 data were used for training and the remaining 250 data for testing.

The results for this example are presented in Table 2, where it is shown by training and test errors that the proposed ANFIS method outperformed the other methods on this example system and ANFIS-BP performed worst among all the methods. Figure 3 also shows that the ANFIS model learned by the QPSO-ADCEC fitted the system fairly well.

Example 3 (online identification in control systems [31]). Here we use ANFIS-QPSO-ADCEC and the other methods to identify a nonlinear component in a control system. The plant under consideration is governed by the following difference equation:

$$y(k+1) = 0.3y(k) + 0.6y(k-1) + u(k). \tag{31}$$

We chose $y(k)$ and $u(k)$ as the inputs and $y(k+1)$ as the output. The input to the plant and the model was a sinusoid $u(k) = \sin(2\pi k/250)$, and the adaptation started at $k = 1$ and stopped at $k = 250$. As shown in Figure 4, the output of the model follows the output of the plant almost immediately even after the adaptation stopped at $k = 250$, and the $u(k)$

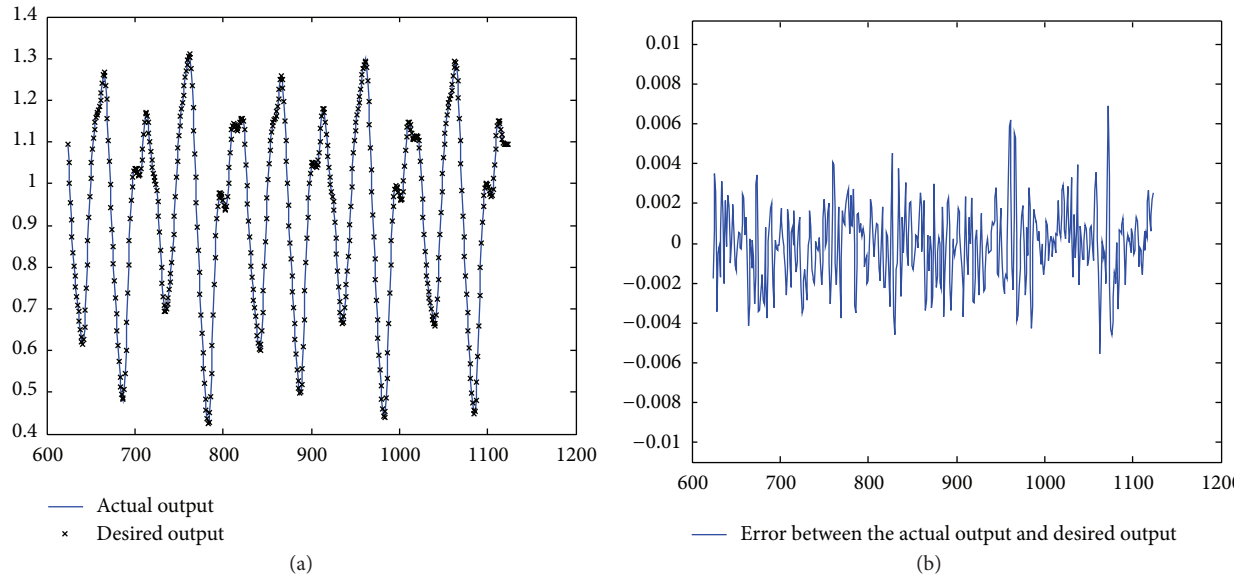

Figure 5: (a) The actual output and desired output; (b) errors between actual output and desired output by using the ANFIS-QPSO-ADCEC for Example 4.

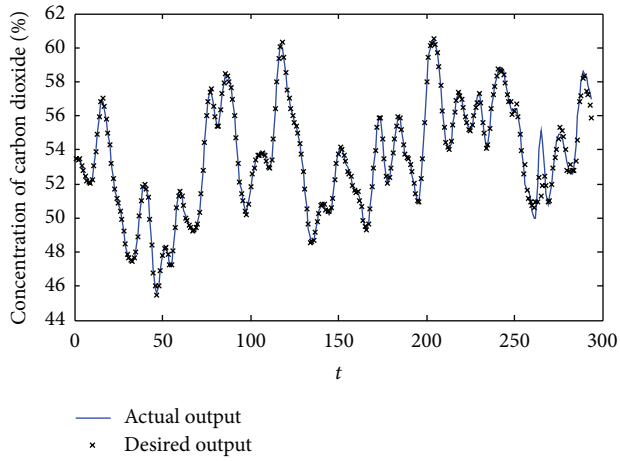

Figure 6: The actual output and desired output by using the ANFIS-QPSO-ADCEC for Example 5.

is changed to $u(k) = 0.5\sin(2\pi k/250) + 0.5\sin(2\pi k/25)$ after it stopped $k = 500$. Figure 4 also illustrates the results using the ANFIS-QPSO-ADCEC for identification. The training error and test error provided in Table 3 show the superiority of the ANFIS-QPSO-ADCEC to its competitors.

Example 4 (prediction of future values of a chaotic time series [31]). The simulation results for Examples 1 to 3 show that the ANFIS-QPSO-ADCEC can be used to model highly nonlinear functions effectively. In this example, we are to demonstrate how the proposed ANFIS-QPSO-ADCEC can be employed to predict future values of a chaotic time series. The time series used in our simulation is generated by the chaotic Mackey-Glass differential delay equation [22] defined by

$$x(t) = \frac{0.2x(t-\tau)}{1 + x^{10}(t-\tau)} - 0.1x(t). \quad (32)$$

Table 3: Results for Example 3.

	trnRMSE	testRMSE
ANFIS-BP	$1.2242e-02$	$1.3041e-02$
ANFIS-PSO	$5.0413e-03$	$1.0454e-03$
ANFIS-QPSO	$3.3281e-03$	$3.3476e-03$
ANFIS-QPSO-ADCEC	$2.1054e-03$	$2.3770e-03$

The prediction of future values of this time series is a benchmark problem which has been considered by a number of connectionist researchers (Lapedes and Farber [32], Moody [33], Jones et al. [34], Crowder [35], and Sanger [36]).

The initial conditions for $x(0)$ and τ are 1.2 and 17, respectively. We use four past data for this prediction, and the fuzzy system is generated as

$$[x(t-18), x(t-12), x(t-6), x(t) : x(t+6)]. \quad (33)$$

When t was varying from 118 to 1117, we generated 1000 data pairs for our data and applied the first 500 data pairs for training and the rest 500 data pairs for prediction. Table 4 provides the training error and test error of each method, showing again that the proposed QPSO-ADCEC algorithm performed the best in training the ANFIS model for this example. Figure 5 shows the errors between actual output and desired output by using ANFIS-QPSO-ADCEC.

Example 5 (prediction of carbon dioxide in Box-Jenkins Furnace [33]). Box-Jenkins Furnace data set contains 296 pairs of input and output data, where the input $x(t)$ is the flow of methane and the output $y(t)$ is the concentration of carbon dioxide dismissed from the furnace. In our experiment, we chose $x(t)$, $x(t-1)$, $x(t-2)$, $y(t-1)$, and $y(t-2)$ to be the input data and $y(t)$ to be the output of the system [37], thus obtain 294 pairs of input and output data. The first 244

Table 4: Results for Example 4.

	trnRMSE	testRMSE
ANFIS-BP	$2.550e-03$	$2.502e-03$
ANFIS-PSO	$2.073e-03$	$2.043e-03$
ANFIS-QPSO	$1.870e-03$	$1.777e-03$
ANFIS-QPSO-ADCEC	$1.3902e-03$	$1.3811e-03$

Table 5: Results for Example 5.

	trnRMSE	testRMSE
ANFIS-BP	$8.705e-02$	$1.047e+00$
ANFIS-PSO	$7.348e-02$	$1.394e-01$
ANFIS-QPSO	$3.984e-02$	$8.567e-02$
ANFIS-QPSO-ADCEC	$1.155e-02$	$5.820e-02$

pairs were used to training the ANFIS model and the last 50 pairs were employed to test the trained system.

Table 5 presents the training error and test error of each method, showing that the QPSO-ADCEC algorithm performed the best in training the ANFIS model and the trained system had the least test error. Figure 6 plotted the desired output and the actual output of the ANFIS model trained by the QPSO-ADCEC. It can be observed that the obtained system fitted the data very well.

7. Conclusion

In this paper, we proposed a novel variant of the QPSO algorithm for training the parameters of ANFIS. This improved QPSO, called QPSO-ADCEC, employs an adaptive dynamical varying CE coefficient which makes the QPSO have better global search ability. The proposed QPSO-ADCEC is applied to estimation of the premise parameters of the ANFIS model, and the LSE approach is used for optimizing the consequent parameters of the fuzzy system.

The effectiveness of the proposed ANFIS-QPSO-ADCEC method was verified by applying it to identification of nonlinear systems and predication of a chaotic system. The simulation results show that the proposed ANFIS-QPSO-ADCEC method had better performance than the ANFIS-PSO, ANFIS-QPSO, and the ANFIS trained with the gradient decent method due to the stronger global search ability of the QPSO-ADCEC algorithm.

Acknowledgments

This work was supported partly by the National Natural Science Foundation of China (Grant nos. 61170119, 61105128) and partly by Natural Science Foundation of Jiangsu Province, China (Grant no. BK2010143).

References

[1] C.-F. Juang, "A TSK-type recurrent fuzzy network for dynamic systems processing by neural network and genetic algorithms," *IEEE Transactions on Fuzzy Systems*, vol. 10, no. 2, pp. 155–170, 2002.

[2] M. Sugeno and G. T. Kang, "Structure identification of fuzzy model," *Fuzzy Sets and Systems*, vol. 28, no. 1, pp. 15–33, 1988.

[3] W. Pedrycz, *Fuzzy Control and Fuzzy Systems*, John Wiley & Sons, New York, NY, USA, 1989.

[4] M. Sugeno, Ed., *Industrial Applications of Fuzzy Control*, Elsevier, New York, NY, USA, 1985.

[5] A. Kandel, *Fuuy Expert Systems*, CRC Press, Boca Raton, Fla, USA, 1992.

[6] J.-S. R. Jang, "ANFIS: adaptive-network-based fuzzy inference system," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 23, no. 3, pp. 665–685, 1993.

[7] F. Hoffmann, D. Schauten, and S. Hölemann, "Incremental evolutionary design of TSK fuzzy controllers," *IEEE Transactions on Fuzzy Systems*, vol. 15, no. 4, pp. 563–577, 2007.

[8] E. G. Mansoori, M. J. Zolghadri, and S. D. Katebi, "SGERD: a steady-state genetic algorithm for extracting fuzzy classification rules from data," *IEEE Transactions on Fuzzy Systems*, vol. 16, no. 4, pp. 1061–1071, 2008.

[9] C.-F. Juang, "A TSK-type recurrent fuzzy network for dynamic systems processing by neural network and genetic algorithms," *IEEE Transactions on Fuzzy Systems*, vol. 10, no. 2, pp. 155–170, 2002.

[10] E. Araujo and L. D. S. Coelho, "Particle swarm approaches using Lozi map chaotic sequences to fuzzy modelling of an experimental thermal-vacuum system," *Applied Soft Computing Journal*, vol. 8, no. 4, pp. 1354–1364, 2008.

[11] A. M. El-Zonkoly, A. A. Khalil, and N. M. Ahmied, "Optimal tunning of lead-lag and fuzzy logic power system stabilizers using particle swarm optimization," *Expert Systems with Applications*, vol. 36, no. 2, pp. 2097–2106, 2009.

[12] K. Das Sharma, A. Chatterjee, and A. Rakshit, "A hybrid approach for design of stable adaptive fuzzy controllers employing Lyapunov theory and particle swarm optimization," *IEEE Transactions on Fuzzy Systems*, vol. 17, no. 2, pp. 329–342, 2009.

[13] J. Sun, B. Feng, and W. Xu, "Particle swarm optimization with particles having quantum behavior," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '04)*, pp. 325–331, June 2004.

[14] J. Sun, W. Xu, and B. Feng, "A global search strategy of Quantum-behaved Particle Swarm Optimization," in *Proceedings of the IEEE Conference on Cybernetics and Intelligent Systems*, pp. 111–116, December 2004.

[15] J. Sun, W. Xu, and B. Feng, "Adaptive parameter control for quantum-behaved particle swarm optimization on individual level," in *Proceedings of the IEEE Conference on Systems, Man and Cybernetics*, pp. 3049–3054, October 2005.

[16] J. Liu, W. Xu, and J. Sun, "Quantum-behaved Particle Swarm Optimization with mutation operator," in *Proceedings of the 17th IEEE International Conference on Tools with Artificial Intelligence (ICTAI '05)*, pp. 237–240, November 2005.

[17] J. Sun, X. Wu, V. Palade, W. Fang, C.-H. Lai, and W. Xu, "Convergence analysis and improvements of quantum-behaved particle swarm optimization," *Information Sciences*, vol. 193, pp. 81–103, 2012.

[18] J. Sun, W. Fang, X. Wu, V. Palade, and W. Xu, "Quantum-behaved particle swarm optimization: analysis of the individual particle's behavior and parameter selection," *Evolutionary Computation*, vol. 20, no. 3, pp. 349–393, 2012.

[19] W. Fang, J. Sun, Y. Ding, X. Wu, and W. Xu, "A review of quantum-behaved particle swarm optimization," *IETE Technical Review*, vol. 27, no. 4, pp. 336–347, 2010.

[20] J. Sun, C. H. Lai, and X. Wu, *Particle Swarm Optimization: Classical and Quantum Perspectives*, CRC Press, 2011.

[21] J. Sun, W. Chen, W. Fang, X. Wun, and W. Xu, "Gene expression data analysis with the clustering method based on an improved quantum-behaved Particle Swarm Optimization," *Engineering Applications of Artificial Intelligence*, vol. 25, no. 2, pp. 376–391, 2012.

[22] J. Sun, W. Fang, and W. Xu, "A quantum-behaved particle swarm optimization with diversity-guided mutation for the design of two-dimensional IIR digital filters," *IEEE Transactions on Circuits and Systems II*, vol. 57, no. 2, pp. 141–145, 2010.

[23] J. Sun, W. Fang, X. Wu, Z. Xie, and W. Xu, "QoS multicast routing using a quantum-behaved particle swarm optimization algorithm," *Engineering Applications of Artificial Intelligence*, vol. 24, no. 1, pp. 123–131, 2011.

[24] J. Sun, X. Wu, W. Fang, Y. Ding, H. Long, and W. Xu, "Multiple sequence alignment using the Hidden Markov Model trained by an improved quantum-behaved particle swarm optimization," *Information Sciences*, vol. 182, no. 1, pp. 93–114, 2012.

[25] J. Sun, W. Fang, V. Palade, X. Wu, and W. Xu, "Quantum-behaved particle swarm optimization with Gaussian distributed local attractor point," *Applied Mathematics and Computation*, vol. 218, no. 7, pp. 3763–3775, 2011.

[26] J. Sun, W. Fang, D. Wang, and W. Xu, "Solving the economic dispatch problem with a modified quantum-behaved particle swarm optimization method," *Energy Conversion and Management*, vol. 50, no. 12, pp. 2967–2975, 2009.

[27] J. Sun, J. Liu, and W. Xu, "Using quantum-behaved particle swarm optimization algorithm to solve non-linear programming problems," *International Journal of Computer Mathematics*, vol. 84, no. 2, pp. 261–272, 2007.

[28] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, vol. 1944, pp. 1942–1948, December 1995.

[29] Y. Shi and R. Eberhart, "Empirical study of particle swarm optimization," in *Proceedings of the Congress on Evolutionary Computation*, pp. 1945–1950, 1999.

[30] M. Clerc and J. Kennedy, "The particle swarm-explosion, stability, and convergence in a multidimensional complex space," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 1, pp. 58–73, 2002.

[31] M. C. Mackey and L. Glass, "Oscillation and chaos in physiological control systems," *Science*, vol. 197, no. 4300, pp. 287–289, 1977.

[32] A. S. Lapedes and R. Farber, "Nonlinear signal processing using neural networks: prediction and system modeling," Tech. Rep. LA-UR-87-2662, Los Alamos National Lab, Los Alamos, NM, USA, 1987.

[33] J. Moody, "Fast learning in multi-resolution hierarchies," in *Advances in Neural Information Processing Systems I*, D. S. Touretzky, Ed., 1, pp. 29–39, Morgan Kaufman, SanMateo, Calif, USA, 1989.

[34] R. D. Jones, Y. C. Lee, C. W. Barnes et al., "Function approximation and time series prediction with neural networks," in *Proceedings of the IEEE International Joint Conference on Neural Networks*, pp. 649–665, June 1990.

[35] R. S. Crowder, "Predicting the Mackey-Glass time series with cascade correlation learning," in *Proceedings of the Connectionist Models Summer School*, D. Touretzky, G. Hinton, and T. Sejnowski, Eds., pp. 117–123, Carnegie Mellon University, 1990.

[36] T. D. Sanger, "A tree-structured adaptive network for function approximation in high-dimensional spaces," *IEEE Transactions on Neural Networks*, vol. 2, no. 2, pp. 285–293, 1991.

[37] X.-N. Yu, F.-Z. Cheng, L.-L. Zhu, and L. Liu, "Fuzzy identification based on improved T-S fuzzy model and its application for thermal process," *Journal of System Simulation*, vol. 19, no. 3, pp. 505–509, 2007.