Available online at www.sciencedirect.com
ScienceDirect
Neurocomputing 70 (2006) 241–251
NEUROCOMPUTING
www.elsevier.com/locate/neucom

Self-generation RBFNs using evolutional PSO learning

Hsuan-Ming Feng*

Department of Management Information, National Kinmen Institute of Technology, No. 1, University Rd., Kin-Ning Vallage Kinmen, 892, Taiwan, ROC

Received 29 January 2005; received in revised form 14 March 2006; accepted 14 March 2006
Communicated by T. Heskes
Available online 3 June 2006

Abstract

In this paper, a powerful evolutional particle swarm optimization (PSO) learning algorithm is developed that self-generates radial basis function neural networks (RBFNs) to deal with three nonlinear problems. In general, the simple but efficient PSO learning algorithm is adapted for solving complex and global optimization problems. With computer simulation, this paper illustrates, in detail, how the PSO algorithm can automatically tune the centers and spreads of each radial basis function, and the connection weights. Furthermore, in parallel with the free RBFNs parameters, the number of radial basis functions of the constructed RBFNs can be automatically minimized by choosing a special fitness function which takes this factor into account. Therefore, the proposed PSO allows high accuracy within a short training time when determining RBFNs with small numbers of radial basis functions. Simulation results demonstrate the proposed RBFNs' efficiency to stabilize an inverted pendulum, approximate a nonlinear function and approximate a discrete dynamic system.

Keywords: Particle swarm optimization; Radial basis function neural networks; Self-generation

1. Introduction

Artificial Neural networks (ANNs) are paralleled, adaptive, and generally non-linear machines built from a variety of learning technologies. ANNs have been widely used for solving some highly nonlinear and complex systems without any prior knowledge [5,13,19]. However, it is difficult for ANNs to configure optimal connection weights. Slow convergence and local minimization, are some other drawbacks. Radial basis function neural networks (RBFNs) have been applied to alleviate some of the limitations of ANNs [5,13,19]. RBFNs are a simple type of feed-forward neural network that contains only three layers: input, hidden, and output layers. Unlike other multilayer networks, data from the hidden layer to the output layer of the RBFNs is transformed with linear calculation. The generation of RBFNs can be considered a curve-fitting problem within a high-dimensional space. Correspondingly, the generalization of the approach is equivalent to the use of a multidimensional surface to interpolate the training data [12]. Because of the RBFNs' ability as a function approximator, its training rate is faster than that of a multilayer perceptron [21]. RBFNs have been successfully applied in many practical fields, especially in the application of functional approximation [1,6,11,22,23] and nonlinear control [4,18]. However, there still are some difficulties with building RBFNs. One of the main problems with RBFNs is determining the number of radial basis functions. In general, the choice of the number of radial basis functions is experience oriented and selected by a trial-and-error procedure. The process of obtaining the above-mentioned terms is time-consuming. Another important issue in designing an RBFN system is the selection of the free parameters for the radial basis functions (the centers, spreads, and connection weights). There are two major approaches to select proper parameters of the RBFNs. One approach is to use by human experts. However, this approach is too imprecise to deal with complex and ill-defined problems. The other approach is to use a gradient-type learning method [15,25]. Traditional trial-and-error and gradient-type learning strategies

*Tel.: +886 82 313540; fax: 886 82 313364.
E-mail address: hmfeng@kmit.edu.tw.

0925-2312/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2006.03.007

are difficult for designers when solving complex and high dimensional problems. In order to avoid trial-and-error methods and improve local optimal problems, we propose using an efficient particle swarm optimization (PSO) learning algorithm. This will allow the center and spreads of each radial basis function to self-generate. Consequently, the connection weight of the radial basis functions causes the extracted RBFNs system to have a robust ability to automatically approach the desired system response. The applied PSO method is used to find an appropriate RBFNs system, to balance the car-pole system and approximate two nonlinear functions, as well as, determine the proper number of radial basis functions.

Particle swarm optimization is an evolutionary computation technique, first introduced by Eberhart and Kennedy in 1995 [16,17], inspired by the social behavior of bird flocks or fish schools. PSO has been used it to solve various optimization problems [3,9,10,24,29]. In PSO searching space, each single solution acts as a bird and is called a particle. This algorithm simulates how natural creatures behave as a swarm; the individual particles are attracted stochastically toward their own previous best location and the swarm's best solution in multidimensional searching space. The computation of this swarm learning is dependent on two pieces of important information: every particle's best solution (up to the point of evaluation) and the swarm's best experience (i.e., the best point which has been found so far). Every particle has a fitness value, which is evaluated by the fitness function to be optimized, and a velocity which directs the trajectory of the particle. Researchers have used PSO to train neural networks and found that PSO-based ANNs have better training performance, faster convergence rate, and better predicting ability than other traditional ANNs [8,20,30].

2. RBFNs structure

RBFNs generally consist of three layers, referred to as the input, hidden, and output layers. The key is to select the radial basis functions of the hidden layers. Gaussian basis functions are the most frequently used radial basis functions with the following form,

$$\mathrm{HE}(x, c_i, \delta_i) = \exp\left[-\left(\frac{\|x - c_i\|^2}{2\delta_i^2}\right)\right], \tag{1}$$

where $\|x - c_i\|$ is the Euclidean distance between an input vector x and a center c_i and δ_i represent the deviation of the ith radial basis function. The output of the RBFNs is calculated by the weighed average method:

$$\mathrm{Out}(x) = \frac{\sum_{i=1}^{m} \mathrm{HE}_i(x) \bullet w_i}{\sum_{i=1}^{m} \mathrm{HE}_i(x)}, \tag{2}$$

where w_i is the ith weight between the hidden and output layers, m is the number of hidden nodes and $\mathrm{HE}_i(x)$ is the output of the ith hidden unit.

According to the above description, the contour of the radial basis function $\mathrm{HE}_i(x)$, which is defined by parameters $\{c_{i1}, \delta_{i1}; c_{i2}, \delta_{i2}, \ldots, c_{in}; \delta_{in}\}$ and the connection weights (w_i), determine the RBFNs. Thus, different parameters sets $R = \{c_{i1}, c_{i2}, \ldots, c_{in}; \delta_{i1}, \delta_{i2}, \ldots, \delta_{in}; w_i, 1 \leqslant i \leqslant m\}$ decide different RBFNs with different performance settings. If there are m radial basis functions to be constructed, total $m(2n + l)$ parameters need to be chosen for designing the best RBFNs. The parameter selection problem is formulated as a searching problem and a method based on a PSO evolutional learning method is applied to select a parameter set R in the searching space.

3. Parameter selection based on PSO

Here, the simple, but efficient, evolutional PSO algorithm with a special fitness function is used to automatically construct the proper RBFNs. Natural creatures sometimes behave as a swarm. One of the main streams of artificial life research is to observe how natural creatures act as a swarm and simulate the swarm models inside a computer. PSO employs the natural animal's behavior such as bird flocking, fish schooling, and swarm theory to yield the best of the characteristics among the population. PSO directly finds a very good solution within a swarm of particles. Each particle represents a possible solution to deal with the problem. A particle is treated as a point in a searching space and the status of a particle is characterized by its position and velocity [16,17]. The first positions and velocities of a PSO algorithm are randomly initialized within a population; then, at the next iteration, positions and velocities are adjusted based on the corresponding previous best information. At every learning cycle, each particle's position and velocity are updated by the two best pieces of data. The first best, Lbest, is the individual particle's best solution that it has achieved so far. The other, Gbest, is obtained by choosing the overall best value from all particles. For all flock members to share the knowledge of optimal solutions. At each iteration step, the velocity of the particle is modified according to the relative data of Lbest and Gbest. The new velocity for each particle is updated by the following equation:

$$\begin{aligned} v_{p,d}(k+1) = {} & SC \times v_{p,d}(k) + \varphi_1\alpha_1(\mathrm{Lbest}_{p,d}(k) - \Phi_{p,d}(k)) \\ & + \varphi_2\alpha_2(\mathrm{Gbest}_{p,d}(k) - \Phi_{p,d}(k)), \end{aligned} \tag{3}$$

Where $v_{p,d}$ is the corresponding velocity of the pth particle in the dth dimension space and $\Phi_{p,d}$ is the responding solution of the pth particle in the dth dimension. Here, p is the number of particles, k employs current state, $k+1$ describes the next state, α_1 and α_2 are acceleration constant,φ_1 andφ_2 are random numbers between 0 and 1, and SC is the scaling factor to control the learning rate.

Since the velocity of the particle is determined, the particle's solution will be modified at the next time step $(k+1)$.

$$\Phi_{p,d}(k+1) = \Phi_{p,d}(k) + v_{p,d}(k+1). \tag{4}$$

Based on formula (3) and (4), the direction of every particle will alter its trajectory and gradually move toward the direction of the best solution, Gbest. Every particle will also learn from the experience of their previous best, Lbest. Therefore, the computation of PSO is easy to implement.

The proposed selection method of radial basis functions is based on a special fitness function; this allows the PSO to simultaneously determine the suitable parameters of RBFNs and the proper number of radial basis functions. Thus, avoiding the disadvantage of conventional RBFNs, which require the number of radial basis functions to be defined in advance. The selecting method for the number of radial basis functions is described as follows (Fig. 1).

Let L_j and U_j be the lower and upper bounds of feasible radial basis functions of the jth particle, respectively. L_j and U_j are defined by an expert based on the complexity of the problem. When the number list γ_j, where $j = 1, 2, \ldots, p$, is randomly determined between the interval $[L_j, U_j]$, the corresponding γ_j number of radial basis functions for jth particle will be active. The active rule selection method is illustrated in Fig. 2. In this illustration, the number list is randomly generated and it denotes $\gamma_j = [3, 4, 2]$ indicating the remaining active radial basis functions for Particle1, Particle2 and Particle3 are 3, 4 and 2, respectively. Through this simple manner of selection, the number of working radial basis functions for every particle can be determined. Despite this being a random selection procedure, the remaining active radial basis functions for every particle have a high probability of resulting in different active numbers of radial basis functions (as in the previous number list example). This will cause operation trouble when the remaining vector length of Gbest and Lbest are not the same as the others. Based on this example, let the selected Particle1 and Particle3 be Gbest and Lbest, respectively. So, if we directly use the active number of radial basis functions to calculate formula (3), the learning process will be broken because of the different length of the working vector. To overcome this issue, Max selection and the other is Min selection will ensure the same vector length is achieved. In Max selection, the maximum from the number list γ_j (which, in this case, is known as 4) is initially selected. Then, the maximum value will be considered the active number of radial basis functions for all particles. The Min selection selects the minimum from γ_j and assigns the minimal value as the working number of radial basis functions in all the swarms. In this case, the active number is 2. Therefore, every particle contains the same numbers of radial basis functions which can execute the learning formula in Eqs. (3) and (4). The proposed simple selection method to determine the active number of radial basis functions is represented in Fig. 3. In the Max selecting scheme, theses particles, except the one with the Gbest and Lbest active vector length, will be regulated—by some unknown information—when the number of active radial basis functions of the two best is smaller than the other particles'. When this condition occurs, the additional information can be considered a disturbance. According to our experiments, the Max selection procedure has a better chance of reproducing the better mixed sharing information than the Min selection method.

Next, required solutions of radial basis functions must move into searching space to construct desired RBFNs with the proposed evolutional PSO algorithm. PSO, based on the mechanics of evolutional selections and knowledge-sharing mechanisms, is a powerful tool when confronted with global search problems [7,14]. Recently, some PSO-based intelligent tuning methods were developed to construct ANNs. PSO employs swarm intelligent theory to yield the best characteristics among the old population and utilize the previous best information to regenerate superior offspring. Normally, PSO works with the guide of

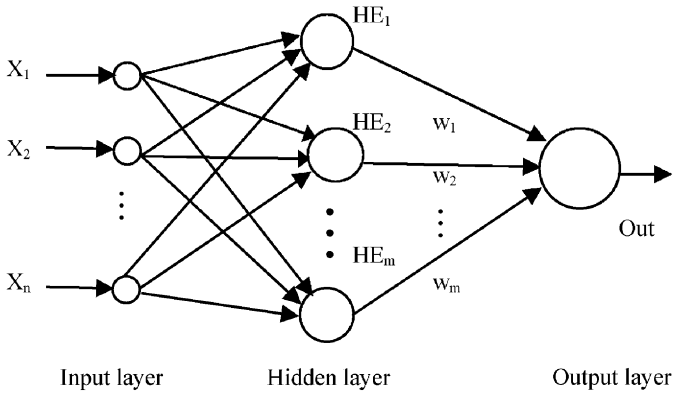

Fig. 1. The architecture of the RBFNs in this study.

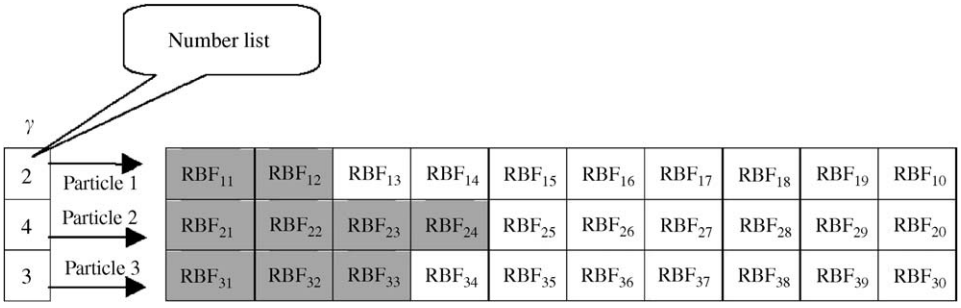

Fig. 2. An example for the selection of the active radial basis functions.

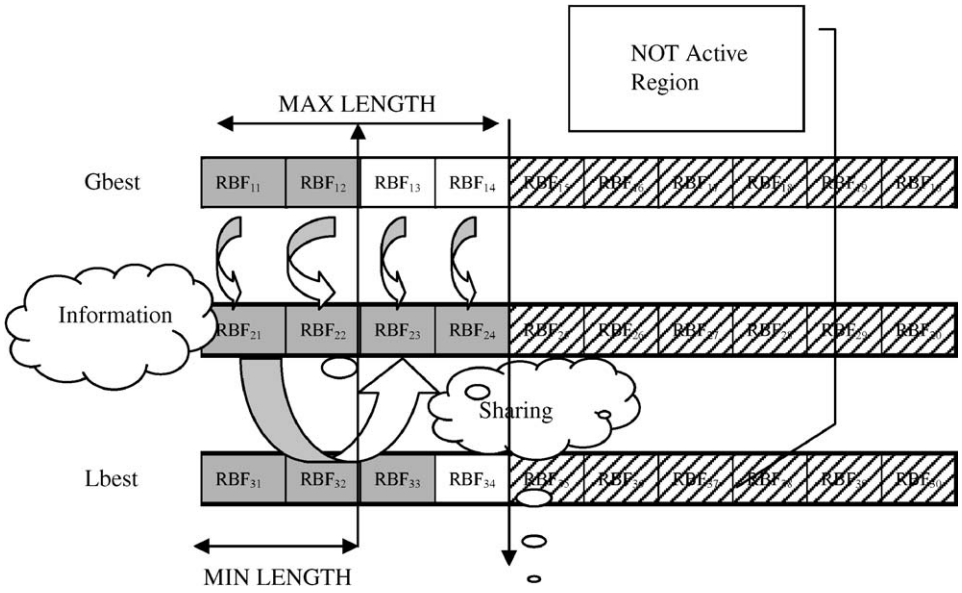

Fig. 3. The behavior of Min and Max Selection for the selection of active radial basis functions.

the proper fitness function to measure the performance of system response. Therefore, the learning stratagem differs from other learning methods requiring derivative information or complete knowledge of the considered problem. Due to PSO not relying on the characteristics of the considered problem, PSO is applicable to a wide range of complex and practical problems.

The goal of the PSO-based evolutional learning method is to choose appropriate RBFNs as well as use the fewest number of radial basis functions. In order to select a suitable parameter set and a proper number of radial basis functions using PSO, select $R_j^* = \{\gamma_j, c_{ij1}, c_{ij2}, \ldots, c_{in}, \delta_{ij1}, \delta_{ij2}, \ldots \delta_{ijn}, w_{ij}, 1 \leqslant i \leqslant m\}$ as a parameter set to form an RBFN (i.e. position of a particle); then choose a fitness function so that PSO can be used to search for a desired solution. To approach such a desired objective, the following fitness function is defined:

$$f(R_j) = g_1((m(R_j^*)) * g_2(J(R_j^*)), \tag{5}$$

where $m(R_j^*)$ is the required number of radial basis functions corresponding to the jth particle parameter set, (R_j^*) is based on the proposed radial basis functions selection method, and $J(R_j^*)$ is the performance measure with respect to the solution of the jth particle. Here, function g_1 and g_2 are utilized to evaluate each individual performance measure. Based on a desired multiple objective, we choose

$$g_1(m(R_j^*)) = \exp(-m(R_j^*)/\sigma_R), \tag{6}$$

$$g_2(m(R_j^*)) = 1/\mathrm{ERROR}(R_j^*), \tag{7}$$

where $\mathrm{ERROR}(R_j^*)$ means the actual error between the desired output and the actual output of the jth RBFNs, which is determined by the individual parameter set (R_j^*) corresponding to the same input. In our research, the ERROR function can be adaptive to define suitable mathematic equations to deal with different problems. $m(R_j^*)$ represents the active number of radial basis functions with respect to the individual R_j^* and σ_R is a constant defined by the designer. The shape of these fitness functions can be adjusted according to the choice of σ_R, so that the representation provides us with a flexible method to meet any objective. In this manner, the proposed PSO algorithm can find the best RBFNs which approach high accuracy (i.e. small ERROR) with fewer number of radial basis functions with the guidance of the proposed fitness function. Therefore, the RBFNs with very good performance can automatically be generated by the PSO method as compared to the time-consuming, traditional trial-and-error methods used to determining appropriate RBFNs parameters. Furthermore, the resulting RBFNs usually contains fewer radial basis functions leading to a reduction in computational complexity and memory load.

When a suitable fitness function $f(.)$ is given, the selection problem can be formulated as the following searching problem:

$$\underset{R_j^* \in \Re}{\mathrm{MAX}} f(R_j^*), \tag{8}$$

where R_j^* is a parameter set which represents the possible location of the jth particle (an RBFN) in searching space $(\Re)$, and $f(R_j^*)$ is the fitness value of the individual R_j^*. Here, two updating formulas (i.e., Eqs. (3) and (4)) can be used to select a parameter set R_j^* with a high fitness value in searching space according to the proposed fitness function. Choosing an appropriate fitness function is a key in solving the considered problem.

Self-generating RBFNs, by using evolutional PSO learning, with a proper number of radial basis functions

and appropriate parameter values of RBFNs can be formulated as follows:

1. Set the number of generations (G), the value of the scaling factor (SC), the constant value (α_1, α_2), the minimum and maximum number of radial basis functions (L, U), and the constant of the fintness function (σ_R).
2. Randomly generate initial particles R_j^* and the number list, γ_j, at $g = 0$. Each individual particle, with respect to the jth γ_j in the swarm, is represented as follows:

   $R_j^* = [\gamma_j R_j]$,

   where γ_j is randomly generated by following rule:

   $$\gamma_j = \text{round}\{L_j + (U_j - L_j)^* \text{rand}(\,)\}. \quad (9)$$

   Here, round is the round operator which generates $\gamma_j \in \{L_j, L_j + 1, \ldots, U_j\}$ and rand( ) is the uniformly distributed random number in [0, 1].
3. Construct the active RBFNs from the jth individual R_j^* by the respective γ_j list.
4. Find the Max or determine the Min value from the γ_j with respect to the Max Selection or the Min selection method. Regulate every particle's velocity and position value according to Formula (3) and (4).
5. Calculate the fitness value for each individual, and then compare each particle's evaluation value with the global best value (Gbest) and each personal best value (Lbest). Renew the global best value (Gbest) and the personal best value (Lbest). The personal best position of each particle can be updated by,

   $$\text{Lbest}_j(t+1) = \left\{ \begin{array}{ll} R_j^*(t+1) & \text{if } (f(R_j^*(t+1)) \geqslant f(R_j^*(t)) \\ R_j^*(t) & \text{if } (f(R_j^*(t+1) < f(R_j^*(t)) \end{array} \right\}. \quad (10)$$

   Then, the group's best position value (Gbset) found by any particle during the previous step is determined by

   $$\text{Gbset}(t+1) = \arg \underset{\text{Pbest}_i}{\text{MAX}} f(\text{Lbest}_j(t+1)) \quad (1 \leqslant j \leqslant p) \quad (11)$$

6. $g = g + 1$
7. If $G = g$, then go to step 8, otherwise go to step 3.
8. The global best solution will be selected to generate the final parameter set $R_j^* = [\gamma_j, R_j]$, then, the desired RBFNs system can be developed.

4. Illustrated examples

In order to demonstrate the efficiency of the proposed evolutional PSO learning method, different types of nonlinear problems are presented in this section.

Example 1. *Car-pole balance system*

The PSO self-generation method is applied to the design of an RBFN controller in the car-pole balance system. The control objective is to create an appropriate actuator force, F, to control the motion of the cart such that the pole can be balanced in the vertical position ($\theta = 0$). Let $x_1(t) = \theta$ (angle of the pole with respect to the vertical axis) and $x_2(t) = \dot{\theta}$ (angular velocity of the pole), then the state equation can be expressed as [26],

$$\dot{x}_1 = x_2, \quad (12)$$

$$\dot{x}_2 = H(x_1, x_2, F) = \frac{g \cdot \sin(x_1) + \cos(x_1)\left(\frac{-F - m \cdot 1 \cdot x_2^2 \sin(x_1)}{m+M}\right)}{1 \cdot \left(\frac{4}{3} - \frac{m \cos^2(x_1)}{m+M}\right)}, \quad (13)$$

g (acceleration due to the gravity) is 9.8 m/s^2, m_c (mass of cart) is 1.0 kg, m (mass of pole) is 0.1 kg, 1 (half length of pole) is 0.5 m, and F is the applied force in Newtons. In this problem, there are two input variables $x_1 = \theta$ and $x_2 = \dot{\theta}$, so the output of the RBFNs can be represented by:

$$Out(x) = \frac{\sum_{i=1}^{m} HE_i(\theta, \dot{\theta}) \cdot w_i}{\sum_{i=1}^{m} HE_i(\theta, \dot{\theta})} \quad (14)$$

The adopted RBFN controller is given 10 maximum rules at the start, therefore, 40 parameters $\{c_{i1}, c_{i2}; \delta_{i1}; w_i, 1 \leqslant i \leqslant 10\}$ are required to be efficiently chosen from the searching space. The number of the iterations is G: 30, the scaling factor is SC: 0.75, the α_1:1.5 and α_2:1.5, and the constant of σ_R for the fitness function is 6. It is assumed that no prior knowledge (from a human operator's point of view) about the car-pole balance (or inverted pendulum) system is available. The objective of this task is to simultaneously balance the angle of the pole toward 0 and select a small number of radial basis functions. In this example, the ERROR(R_j^*) is calculated with IAE(R_j^*), so the proposed fitness function is represented as follows:

$$f(R_j) = g_1((m(R_j^*)) * g_2(J(R_j^*)) = \exp(-m(R_j^*)/\sigma_R) * (1/\text{IAE}(R_j^*)), \quad (15)$$

where IAE denotes the integral absolute error which is defined in [28]. Therefore, the number of the radial basis functions and the integral absolute error between the desired position and the actual position forms the basis for (15). The goal of the PSO learning algorithm is to maximize the fitness function value, i.e. minimize the IAE and select a fewer number of radial basis functions. The following are results for the Max and Min selection type PSO learning algorithms, with the initial condition $\theta = 20$ and $\dot{\theta} = 0$ (shown in Fig. 4). Fig. 4(a)–(c) show the response of the pole angle, angle velocity and input force from $t = 0$–2.5 s. While Fig. 4(d)–(f) show the best fitness value, the integral absolute error and the number of radial basis functions against the iteration number for the proposed PSO self-generation RBFNs system, respectively. This experiment shows that the Max selection type PSO has a better performance than the Min selection type PSO. For another experiment, simulation results for 5 runs of the Max selection type PSO learning algorithm, with initial

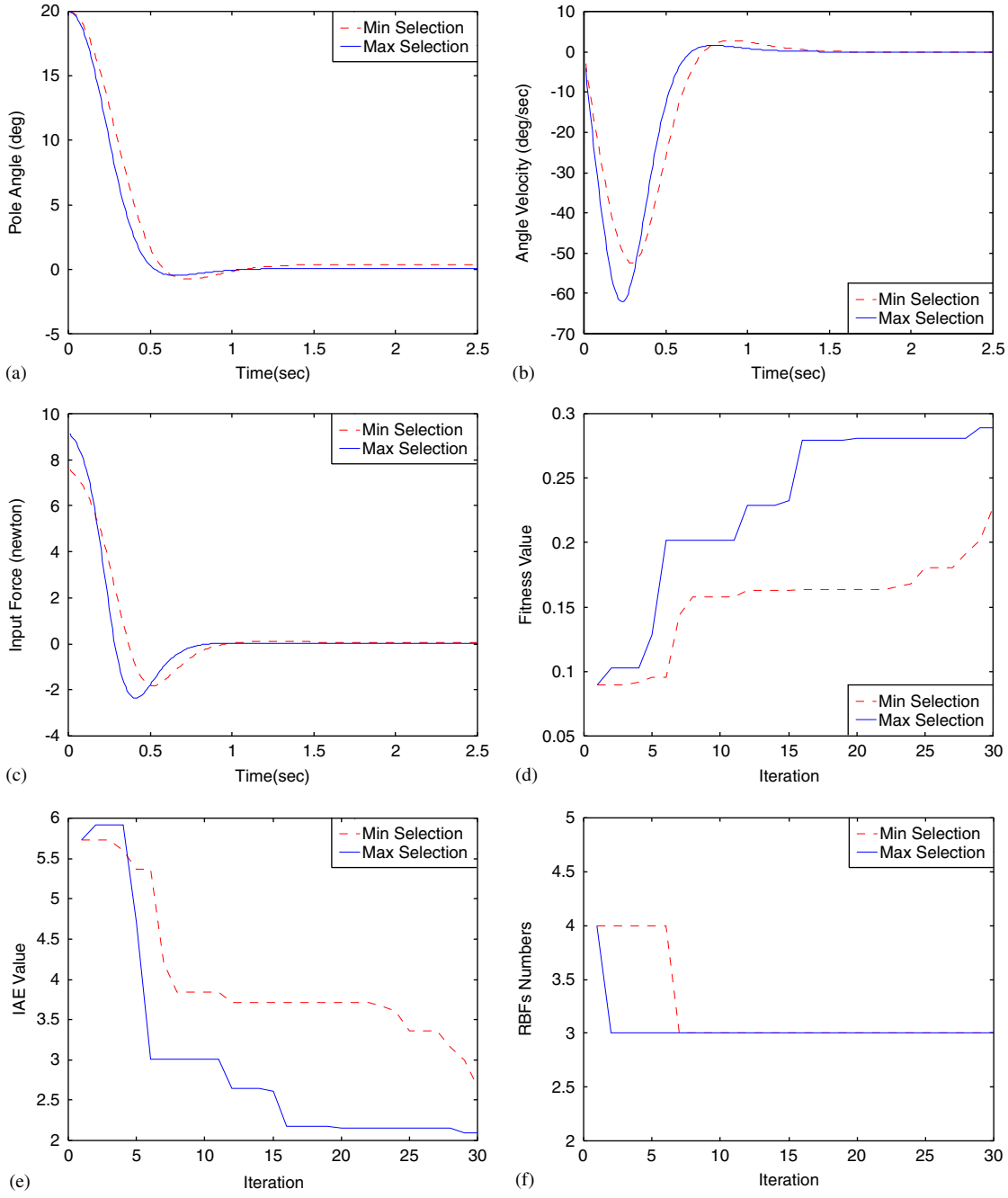

Fig. 4. Simulation results of the proposed PSO method with Max (solid line) and Min (dashed line) type selection in Example 1. (a) Pole angles, (b) pole velocities, (c) forces, (d) best fitness values against iteration, (e) IAE values against iteration, and (f) selected RBFs numbers against iteration.

conditions $\theta = 20$ and $\dot{\theta} = 0$, where carried out with results shown in Fig. 5. Fig. 5(a) represents the output of the pole position, while Fig. 5(b)–(d) show the fitness value, integral absolute error and number of radial basis functions versus iteration for PSO self-generation RBFNs system respectively. From this experiment, these collected dynamic behaviors, such as: fitness value, IAE, and the number of radial basis functions, show the character of the evolutional PSO method is learning in a heuristic and stochastic method. Each solution has slightly different parameters. Yet, with respect to approaching the high fitness value, the final responses of all RBFN controllers can quickly converge to an optimal solution. The selected parameter set for the best of the five runs is illustrated in Table 1. Performance comparison of the best result in five runs of the proposed Max type PSO method with the Chen's GA-based fuzzy controller system [2] is illustrated in Table 2. Simulations demonstrate that the PSO learning method has a shorter rise time, smaller maximal overshoot and smaller IAE values than previous results with the Chen's method. In addition, the PSO based RBFN controller uses only 3 radial basis functions to achieve

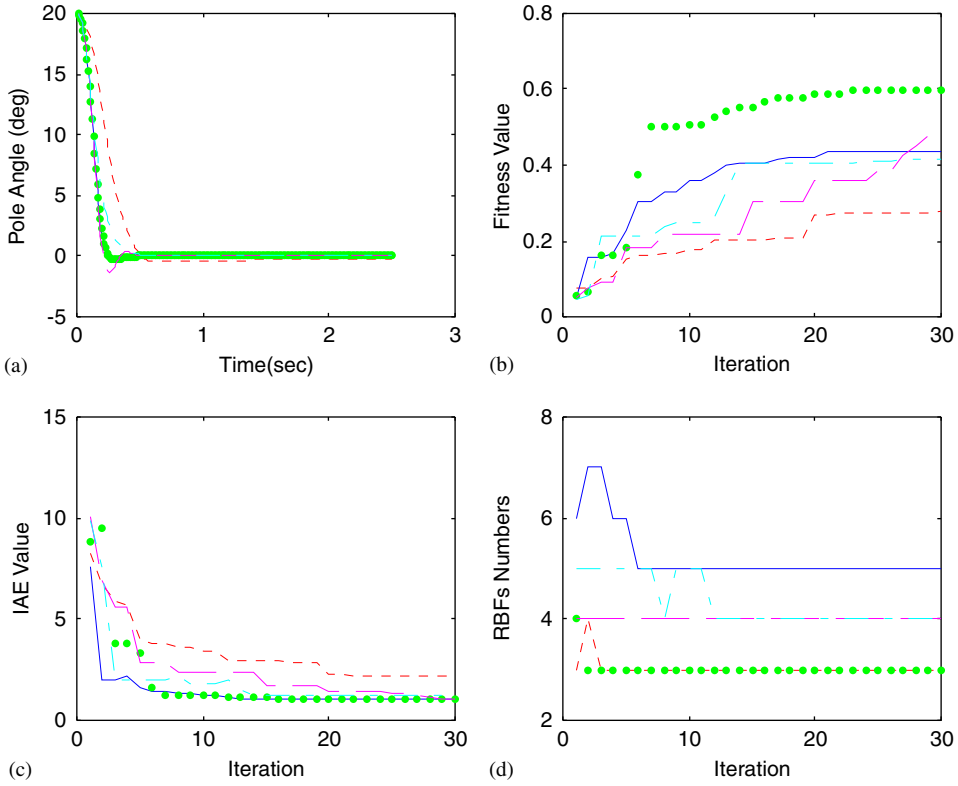

Fig. 5. Multiple runs with the proposed Max-type PSO method in Example 1. (a) Pole angles, (b) fitness values against iteration, (c) IAE value against iteration and (d) selected RBFs numbers against iteration.

Table 1
Data of the selected string for Example 1

i	c_{i1}	c_{i2}	δ_{i1}	w_i
1	−19.8000	39.600	7.6705	−9.9808
2	19.7039	39.6	1.4228	9.9869
3	14.6482	25.9667	19.8	−4.2865

Table 2
Performance comparison with different methods in Example 1

	Rise time	Maximal overshoot	IAE	Fuzzy rules/ RBFs numbers
Results with Ref. [2]	0.27	0.4125	2.6225	15
Best Max PSO	0.18	0.3138	1.0192	3

better result. The Max type PSO learning method can automatically design the RBFNs system with a fewer number of radial basis functions to control the car-pole balance system.

Example 2. *Non-linear modeling system*

In the second example, the RBFNs is denoted as an identifier for a nonlinear system and extracts the parameters of the radial basis functions with the learning method. The test plant to be identified is governed by the following two-dimensional equation [27]:

$$\mathrm{Out}(x) = \sin(\pi x_1) \cdot \sin(\pi x_2). \tag{16}$$

In this example, training data is uniformly distributed in the range of $x_1 \in [-1, 1]$ and $x_2 \in [0, 1]$. All desired input-output pairs are composed of 225 pieces of data. Here, Out(.) is a two-input and one-output function identified by the proposed RBFNs. When the range of the selected number of radial basis functions is between [4,15], the adopted RBFN is given 15 radial basis functions at the start. Therefore, 60 parameters $\{c_{i1}, c_{i2}, \delta_{i1}, w_i, 1 \leqslant i \leqslant 15\}$ are required to be efficiently chosen from the solution space. The number of iterations is G: 150, the scaling factor is SC: 0.75, the α_1:1.5 and α_2:1.5, and the constant for the fitness function is $(\sigma_R = 100)$. In this example, the definition of the proposed fitness function is as follows:

$$\begin{aligned} f(R_j) &= g_1((m(R_j^*)) * g_2(J(R_j^*)) \\ &= \exp(-m(R_j^*)/\sigma_R) * (1/\mathrm{MSE}(R_j^*)), \end{aligned} \tag{17}$$

where, MSE denotes the mean square error between the actual output and the desired output. Based on the Min and Max selection type PSO learning methods, simulation results of the proposed RBFNs identifier is illustrated in Fig. 6. Fig. 6(a) shows surface of generated RBFNs identifiers with Max selection type and the desired output

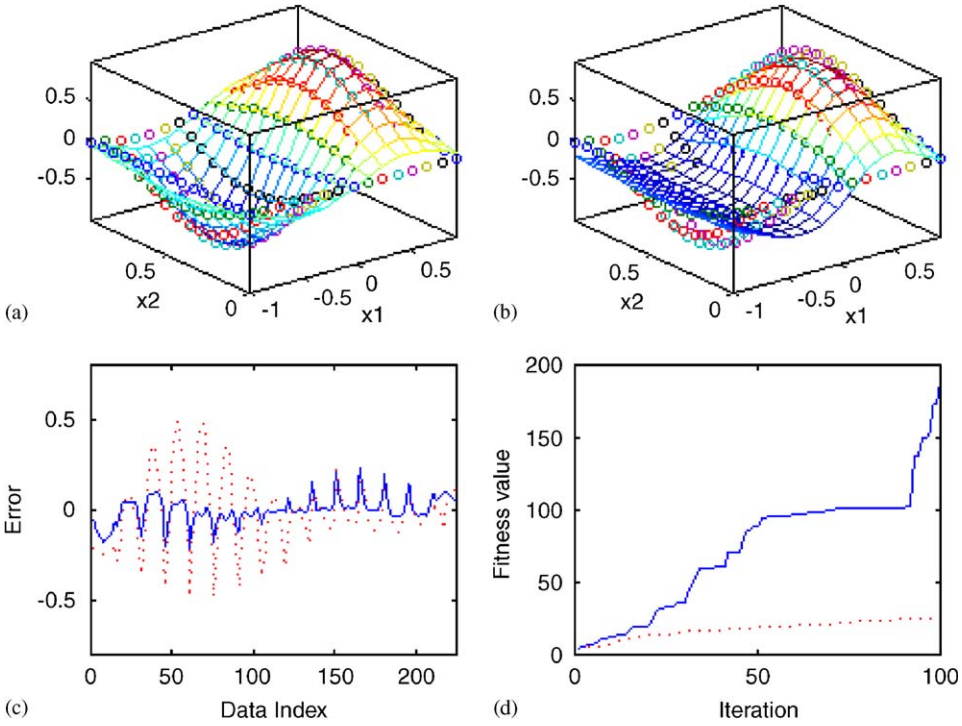

Fig. 6. Simulation results for Max- and Min-type PSO method in Example 2. (a) Max type (meshed surface) and training data (circles), (b) Min type (meshed surface) and training data (circles), (c) error against data index for Min type (dotted) and Max-type (solid) and (d) fitness value against iteration for Min type (dotted) and Max type (solid).

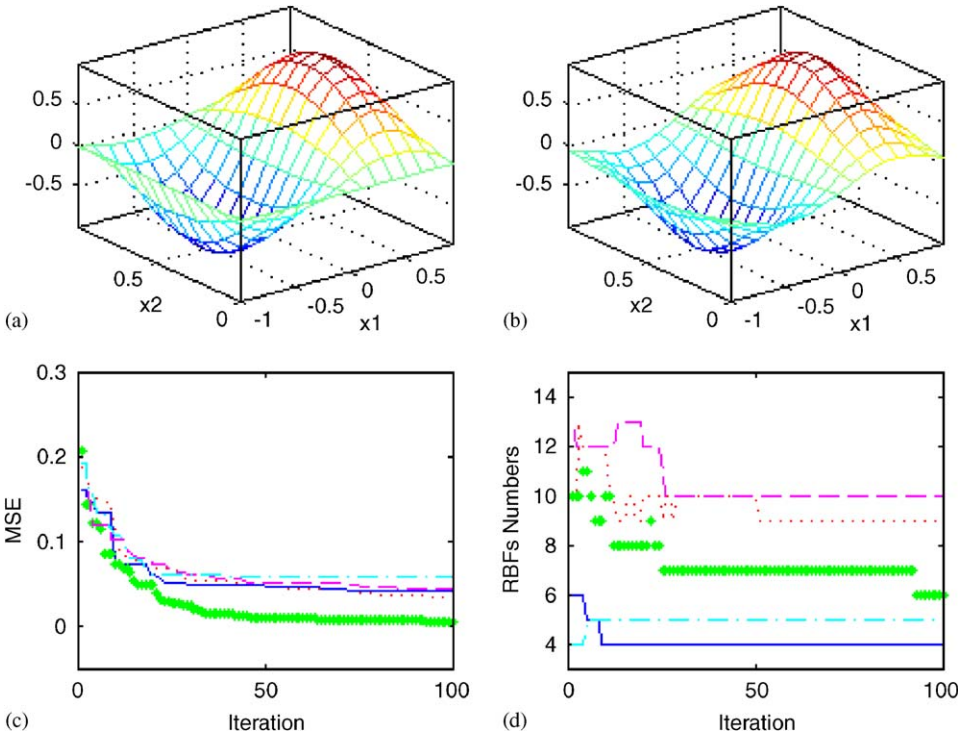

Fig. 7. Multiple runs with Max-type PSO method in Example 2. (a) Training data, (b) best output in five runs, (c) MSE values against iteration and (d) selected RBFs numbers against iteration.

of training data. Desired outputs of the training data are represented by circles and the output of the identification with Max selection is illustrated by the meshed surface. Same type simulation with the Min type PSO learning method is also shown in Fig. 6(b). In Fig. 6(c), it indicates differences between the desired training data and the

identified one. Simulation results show that the Max selection based PSO learning method can approach the desired data with greater accuracy than the Min selection type. From the plot of the fitness value against iteration in Fig. 6(d), it demonstrates previous discussed results. In the other experiment, 5 runs of the Max type PSO learning method are utilized to generate the RBFNs system. Proper RBFNs, with respect to five acceptable solutions, are selected and the simulation results are shown in Fig. 7. Fig. 7(a) and (b) show the desired output of the plant and the state response of the best generated RBFN identifiers, respectively. Fig. 7(c) and (d) show the MSE and the selected number of radial basis functions for each individual best particle in five runs, respectively. The parameter set of the best RBFNs system is shown in Table 3. Simulation results indicate that 6 radial basis functions are enough to identify this two-dimensional nonlinear model with small errors (i.e., the RBFNs approximate the identified plant more accurately). The performance comparison for the other modeling method with MSE is listed in Table 4. This experiment shows that the proposed PSO learning method is simpler and more efficient than the other fuzzy modeling system [16].

Table 3
Data of the selected string for Example 2

i	c_{i1}	c_{i2}	δ_{i1}	w_i
1	0.8576	0.0011	9.7568	5.0000
2	−0.8165	0.4994	7.7860	−5.0000
3	1.0000	0.1056	27.6202	4.9693
4	−1.0000	0.9998	27.8839	−4.9980
5	−0.4567	0.4927	0.2608	−5.0001
6	0.4767	0.5010	0.2713	4.9926

Table 4
Performance comparison for Example 2

	MSE	Fuzzy rules/RBFs numbers
Max PSO	0.0041	6
Wong's method [27]	0.0037	7

Example 3. *Modeling a discrete dynamic system*

In the final example, the discrete dynamic system, as described by [13], is

$$\mathrm{Out}(k+1) = \frac{\mathrm{Out}(k)u(k)}{1+\mathrm{Out}^2(k)} - \tan(u(k)), \tag{18}$$

where $\mathrm{Out}(k)$ and $u(k)$ are the state and action signal, respectively, at time step k. To train the RBFNs modeling system with the proposed PSO-based learning method, we choose inputs, $u(k)$, $k = 1$–101, as uniformly distributed random numbers between -1 and 1, and then use Eq. (18) to generate 100 training data pairs in the form $[\mathrm{Out}(k), \mathrm{Out}(k+1); u(k)]$. When the searching range of the available number of radial basis functions is [3,10], the

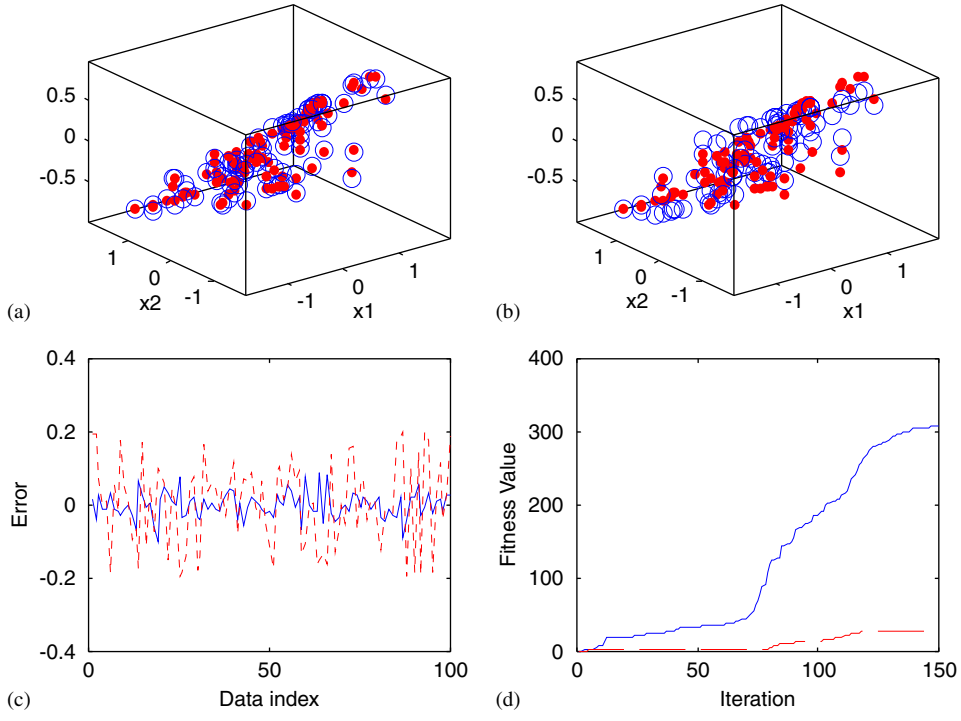

Fig. 8. Simulation results for Max-and Min-type PSO method in Example 3. (a) Max type (dotted point) and training data (circles), (b) Min type (dotted point) and training data (circles), (c) error against data index for Min type (doted) and Max type (solid), and (d) fitness value against iteration for Min type (doted) and Max type (solid).

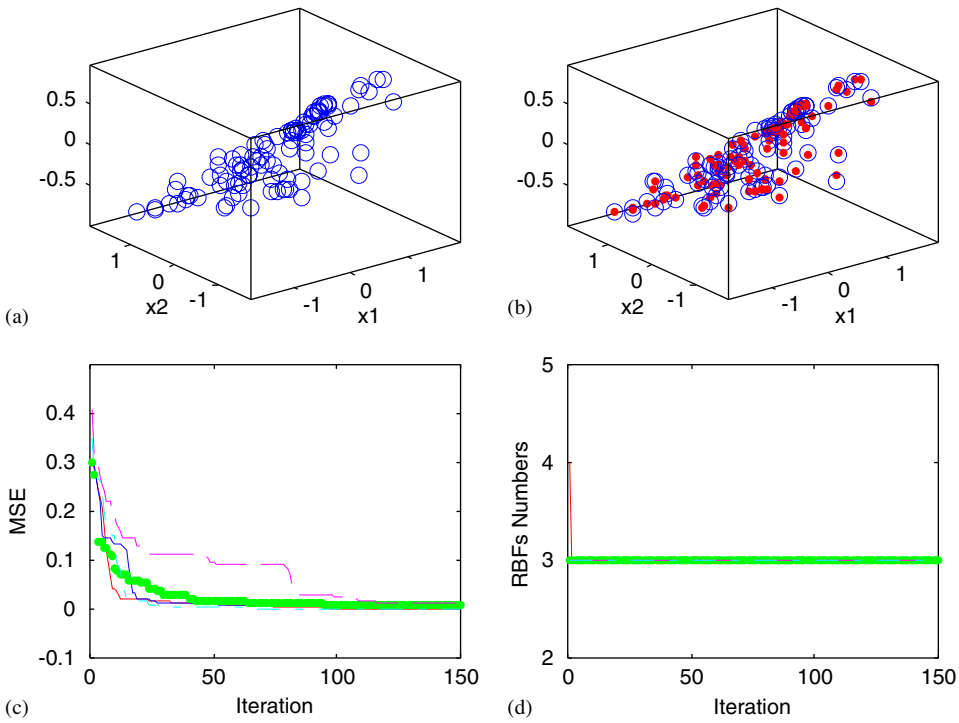

Fig. 9. Response of five runs with Max-type PSO method in Example 3. (a) training data, (b) best output (dotted point) and training data (circles), (c) MSE against iteration and (d) RBFs numbers against iterations.

adopted RBFNs identifier is given 10 radial basis functions to start. Therefore, 40 parameters $\{c_{i1}, c_{i2}; \delta_{i1}; w_i, 1 \leqslant i \leqslant 10\}$ are required to be efficiently chosen from the searching space. The number of iterations is G: 150, the scaling factor is SC: 0.7, the α_1:1.2 and α_2: 1.2, and the constant of the fitness function is ($\alpha_R = 3$). The definition of the fitness function is same as Eq. (17). Simulation results for the Max and Min selection-based PSO methods are respectively shown in Fig. 8(a) and (b), where circles show the desired output of training data and solid dots indicate the identified data of generated RBFNs identifiers. Differences between desired and identified data with the Max and Min selection-based PSO methods are shown in Fig. 8(c). The best fitness value against the iteration is illustrated in Fig. 8(d). It is clear that the Max selection-based PSO has a better result than the Min selection-based PSO. The multiple simulations, plotted in Fig. 9, Fig. 9(a) and (b), show the original training data (circles) with the best identified result (solid dots) of these multiple runs. Fig. 9(c) shows the individual responses for MSE, while Fig. 9(d) shows the selected number of the radial basis functions. The results show that the PSO-based RBFNs can efficiently represent the discrete dynamic function with only 3 radial basis functions. The selected parameter set for the best RBFN identifier is illustrated in Table 5 and data comparison with Jang's ANFIS modeling system is illustrated in Table 6. The result of the simulation with the proposed PSO learning algorithm demonstrates that it is achieving good performance while using less radial basis functions than presented in previous results.

Table 5
Data of the selected string for Example 3

i	c_{i1}	c_{i2}	δ_{i1}	w_i
1	0.1395	1.5	0.8234	−1.0002
2	−0.3306	−1.5	−0.9569	1.0001
3	−0.8442	−0.0347	−0.6534	−0.1803

Table 6
Performance comparison for Example 3

	MSE	RBFs numbers
Max PSO	0.001891	3
ANFIS	0.001801	9

5. Conclusion

In this paper, an evolutional PSO-based RBFNs system has been developed to solve non-linear control and modeling problems. PSO learning-based RBFNs can simultaneously select the proper number of radial basis functions and adjustable parameters of the RBFNs with a special fitness function. This learning approach is different from gradient-type RBFN learning methods that require the number of radial basis functions to be defined in advance. The solution to the non-linear car-pole balance, two-input and one-output nonlinear modeling, and two-dimensional discrete dynamic modeling problems illustrate the efficiency of the PSO learning method. Comparisons

with other ANNs or fuzzy systems show that the PSO-based method finds solutions with similar performance and uses less and in some cases uses far less RBFs.

Acknowledgements

The author wishes to thank the anonymous reviewers for their constructive and helpful comments on this paper.

References

[1] D.F. Akhmetov, Y. Dote, S.J. Ovaska, Fuzzy neural network with general parameter adaptation for modeling of nonlinear time-series, IEEE Trans. Neural Networks 12 (1) (2001) 148–152.
[2] C.C. Chen, C.-C. Wong, Self-generation rule-mapping fuzzy controller design using a genetic algorithm, IEE Proc.—Control Theory Appl. 149 (2002) 143–148.
[3] C.-Y. Chen, H.-M. Feng, F. Ye, Self grey particle swarm optimization patterns clustering algorithm, J. Grey Syst. 17 (2005) 327–336.
[4] S.-C. Chen, W.-L. Chen, Output regulation of nonlinear uncertain system with nonminimum phase via enhanced RBFN controller, IEEE Trans. Syst. Man Cybern. 33 (2) (2003) 265–270.
[5] S.W. Choi, D. Lee, J.H. Park, I.-B. Lee, Nonlinear regression using RBFN with linear submodels, Chemometr. Intell. Lab. Syst. 65 (2) (2003) 191–208.
[6] C.-C. Chuang, J.-T. Jeng, P.-T. Lin, Annealing robust radial basis function networks for function approximation with outliers, Neurocomputing 56 (2004) 123–129.
[7] M. Clerc, J. Kennedy, The particle swarm—explosion, stability, and convergence in a multidimensional complex space, IEEE Trans. Evolut. Comput. 6 (1) (2002) 58–73.
[8] Y. Da, G. Xiurun, An improved PSO-based ANN with simulated annealing technique, Neurocomputing 63 (2005) 527–533.
[9] H.-M. Feng, C.-Y. Chen, F. Ye, Evolutionary fuzzy particle swarm optimization vector quantization learning scheme in image compression, Expert Syst. Appl. 32 (1) (2006).
[10] Z.-L. Gaing, A Particle swarm optimization approach for optimum design of PID controller in AVR system, IEEE Trans. Energy Conver. 19 (2) (2004) 384–391.
[11] F. Girosi, T. Poggio, Networks and the best approximation property, Biol. Cybern. 63 (1990) 169–176.
[12] S. Haykin, Neural Networks: A Comprehensive Foundation, Prentice-Hall, New Jersey, 1999.
[13] J.S.R. Jang, C.T. Sun, E. Mizutani, Neuro-Fuzzy and Soft Computing, Prentice-Hall, New Jersey, 1997.
[14] C.-F. Juang, A Hybrid of genetic algorithm and particle swarm optimization for recurrent network design, IEEE Trans. Syst. Man Cybern. 34 (2) (2003) 997–1006.
[15] I.C. Kampolis, E.I. Karangelos, K.C. Giannakoglou, Gradient-assisted radial basis function networks: theory and applications, Appl. Math. Model. 28 (2) (2004) 197–209.
[17] J. Kennedy, R.C. Eberhart, Particle swarm optimization, in: Proceedings of the IEEE International Conference on Neural Networks, Perth, Australia, November 1995, pp. 1942–1948.
[16] J. Kennedy, The particle swarm: social adaptation of knowledge, in: Proceedings of the 1997 International Conference on Evolutionary Computation, Indianapolis, IN, April 1997, pp. 303–308.
[18] M.-J. Lee, Y.-K. Choi, An adaptive neurocontroller using RBFN for robot manipulators, IEEE Trans. Ind. Electron. 51 (3) (2004) 711–717.
[19] C.T. Lin, C.S.G. Lee, Neural Fuzzy Systems, Prentice-Hall, New Jersey, 1996.
[20] W.Z. Lu, H.Y. Fan, S.M. Lo, Application of evolutionary neural network method in predicting pollutant levels in downtown area of Hong Kong, Neurocomputing 51 (2003) 387–400.
[21] J. Moody, C.J. Darken, Fast learning in networks of locally-tuned processing units, Neural Comput. 1 (1989) 281–294.
[22] M.-D. Nam, T.-C. Thanh, Approximation of function and its derivatives using radial basis function networks, Appl. Math. Model. 27 (3) (2003) 197–220.
[23] J. Park, I.W. Sandberg, Approximation and radial basis function networks, Neural Comput. 5 (1993) 305–316.
[24] J. Robinson, Y. Rahmat-Samii, Particle swarm optimization in electromagnetics, IEEE Trans. Antennas Propag. 52 (2) (2004) 397–407.
[25] M.-T. Vakil-Baghmisheh, N. Pavešić, Training RBF networks with selective backpropagation, Neurocomputation 62 (2004) 39–64.
[26] L.X. Wang, A course in Fuzzy Systems and Control, Prentice-Hall, Englewood Cliffs, NJ, 1997.
[27] C.-C. Wong, C.-C. Chen, A hybrid clustering and gradient descent approach for fuzzy modeling, IEEE Trans. Syst. Man Cybern. 29 (6) (1999) 686–693.
[28] C.-C. Wong, C.-S. Fan, Rule mapping fuzzy controller design, Fuzzy Sets Syst. 108 (1999) 253–261.
[29] F. Ye, C.-Y. Chen, Alternative KPSO-clustering algorithm, Tamkang J. Sci. Eng. 8 (2) (2005) 165–174.
[30] C.K. Zhang, H.H. Shao, An ANN's evolved by a new evolutionary system and its application, in: Proceedings of the 39th IEEE Conference on Decision and Control, Sydney, Australia, December 2000, pp. 3562–3563.

Hsuan-Ming Feng received the BS degree in automatic control engineering from Feng-Chia University, Taichung, Taiwan, ROC, in 1992. He received MS and PhD degrees in Computer Science and Information Engineering from Tamkang University, Tamsui, Taipei Hsien, Taiwan, ROC, in 1994 and 2000, respectively. He was the assistant professor of Department of Management Information System, Yung-Ta Institute of Technology and Commerce from August, 2000 to July, 2003. Since August, 2003, he has been the assistant professor of Department of Management Information, National Kinmen Institute of Technology. His current research interests include fuzzy systems, neural networks, grey systems, optimal learning algorithms, image processing and data mining.