Proceedings of the 2009 IEEE International Conference on Systems, Man, and Cybernetics
San Antonio, TX, USA - October 2009

Delay Nonlinear System Predictive Control On MPSO+DNN

Min Han
School of Electronic and Information Engineering
Dalian University of Technology
Dalian, P. R. China
minhan@dlut.edu.cn

Jianchao Fan
School of Electronic and Information Engineering
Dalian University of Technology
Dalian, P. R. China
fjchao@student.dlut.edu.cn

***Abstract*—This paper presents a novel dynamic neural network (DNN) predictive control strategy based on modified particle swarm optimization (PSO) for long time delay nonlinear process. The proposed dynamic NN structure could approximate to the actual system model and obtain the pure delay time exactly. An improved version of the original PSO is put forward to train the parameters of NN to enhance the convergence and accuracy. The effectiveness of the proposed control scheme is demonstrated by simulation as well as a test on an experiment on the actual pH Neutralization Process.**

***Keywords*—delay system, model predictive control, dynamic NN, PSO**

I. INTRODUCTION

Most industrial processes in reality are nonlinear and long time delay. Proportional Integral Derivative (PID) controllers are the most popular ones in industry [1]. But the pure delay time may cause the control single response to the output long time ago, so that such controllers may produce large error or even be out of control [2]. Thus, in this case, the Smith predictor is a well-known dead time compensator for stable processes with large time delays. When the model is accurate, the closed-loop characteristic will be delay free, and so the Smith predictor structure has significantly facilitated controller design in the conventional feedback systems. However, the exact pure time is hard to obtain. Another important predictive control strategy, Model Predictive Control (MPC), developed rapidly in the past two decades, has an increasing acceptance in engineering field recently [3]. A major problem engineers have to face that the design of a MPC system relies heavily upon an explicit mathematical model of the system under control, which has been studied extensively. However, the traditional approach is usually based on approximate linearization theory, which is often difficult to identify an accurate mathematical system model and imposes serious restrictions on the structure of nonlinear system.

This work is supported in part by supported by the project (2007AA04Z158) of the National High Technology Research and Development Program of China (863 Program), the project (60674073) of the National Nature Science Foundation of China, the project (2006BAB14B05) of the National Key Technology R&D Program of China and the project (2006CB403405) of the National Basic Research Program of China (973 Program).

Artificial neural networks (ANNs), which have been widely studied and applied to various research fields, have superiority as compared with other conventional modeling methods. The advantage of ANNs is that it does not need any knowledge about the process, which is a black box model. For nonlinear plants, the ability of the MPC to make accurate predictions can be enhanced if an NN is used to learn the plant dynamic instead of standard nonlinear modeling techniques. The NN based predictive control strategies have been found to be effective in controlling a wide class of nonlinear processes in the past [4]. In the NN predictive control (NNPC), the NN is used as the prediction model of the nonlinear plant and the system performance is greatly dependent on the online optimization. Some network structures have been successfully applied to the actual application, such as feedforward neural networks, radial basis networks, and dynamic neural networks. Dynamic neural networks, due to the more complex time sequence structure, could fully express the nonlinear relationship between the input and output [5]. Furthermore, several algorithms are commonly implemented in the training process of the NN, such as gradient–descent and Newton methods. However, these gradient-based optimization methods usually fall into local optima, and also this is a complex procedure for calculating the Jacobian and Hessian matrix. Therefore, the convergence speed of neural networks is slow and sensitive to the initial parameter values [6]. A number of different evolutionary algorithms have been proposed to improve the capability of the neural networks.

In order to enhance the approximation capability of the single neuron model, particle swarm optimization is introduced to train the model. PSO developed by Kennedy and Eberhart [7] in 1995, is an evolutionary computation technique for optimizing hard numerical functions on metaphor of social behavior of flocks of birds and schools of fish. It is an evolutionary computation technique based on swarm intelligence, and is also the latest evolutionary optimization technology used for solving a wide range of real world optimization problem as a substitute for genetic algorithm (GA) [8]. A swarm consists of individuals, called “particles”, which change their position over time. Each particle represents a potential solution to the problem. Due to the simple concept, easy implementation, and quick convergence, nowadays, PSO has gained much attention and wide applications in different

978-1-4244-2794-9/09/$25.00 ©2009 IEEE

fields. However, the conventional PSO may easily get trapped in a local optimum when tackling complex problems as well as other evolutionary methods [9]. Great efforts could be found in the literature. At present, Niu [10] introduces a bio-inspired mechanism to make the basic PSO more intelligent by separating the population in PSO into two parts, a master swarm and several slave swarms. A hybrid PSO model has been reported by Lovbjerg [11], in which the velocity and position update relations employ the concepts of breeding and subpopulation. However, due to the limitation of dynamic response capability for common NN structure and the drawback of PSO algorithm lacking of enough ability to sustainable development, a simple hybrid of these two optimization algorithms cannot obtain approving generalization effect [12,13].

Therefore, this paper proposes an adaptive dynamic feedforward neural network based on modified particle swarm optimization algorithm to the long time delay nonlinear system predictive control. Between input layer and first hidden layer, and also the last hidden layer and output layer, the dynamic time delay operators are adopted, which could build the relationship between the current output and the previous time inputs. Those are dynamic adjusted with the iterative system identification process using the PSO algorithm. The delay operators in the output part can accurately identify the pure time delay, and corresponding operators in the input part can enhance the NNs dynamic characters. After these operators in output part removed, the actual system model without any delay is built exactly, so some classic predict control strategies, such as model predictive control, could achieve satisfactory control effect. Consequently, the predictor and compensator in MPC are solved exactly by our modified NN structure. Otherwise, use the weight digression method to improve the PSO algorithm's global search capability. Then, the parameters in the NNPC are trained by modified PSO method, avoiding the complexity of gradient calculation and trapping in the local optimal points. That makes the neural networks follow the system change as fast as possible and perform real-time response.

The remainder of this paper is organized as follows. Section II presents the modified PSO algorithm (MPSO). Section III describes the dynamic delay evolutional feedforward neural networks. Furthermore, in Section IV, we discuss the combination of the above two modified parts for the nonlinear delay system control. Section V highlights the potential of the proposed approach through experimental examples. Concluding remarks are presented in Section VI.

II. MODIFIED PSO ALGORITHM

A. Standard PSO

PSO is an evolutionary algorithm which maintains a swarm of candidate solutions, referred to as 'particles'. The best solution each particle has achieved so far is P_{best}, which is denoted as $P_i = (p_{i1}, p_{i2}, \cdots, p_{iD})$. The best position P_{gbest} among all the particles in the population is represented as $P_g = (p_{g1}, p_{g2}, \cdots, p_{gD})$. PSO relies on the exchange of information between particles, in other words this is a hypothesis that social sharing of information among individuals offers an evolutionary advantage. PSO is similar to other evolutionary algorithms in that the system is initialized with a population of random solutions. The velocity for each particle in D dimensional problem space is dynamically adjusted according to the flying experiences of its own and its fellows. There are four main parameters in the whole PSO algorithm. The location of particle for i is represented as $x_i = \{x_{i1}, x_{i2,} \cdots, x_{iD}\}$, where $x_{id} \in [x_{\min,d}, x_{\max,d}]$, $d = 1, \cdots, D$. $x_{\min,d}$ and $x_{\max,d}$ are the lower and upper bounds for the dimension d, respectively. Let $V_i = (v_{i1}, v_{i2}, \cdots, v_{iD})$ be the current velocity for particle i, which is limited to a maximum velocity $V_{\max} = (v_{\max,1}, \cdots, v_{\max,D})$ .At each step, the velocity and position of each particle are updated toward its P_{best} and P_{gbest} locations according to (1)

$$\begin{cases} v_{id}(t+1) = wv_{id}(t) + c_1 rand_1()[p_{id}(t) - x_{id}(t)] \\ \qquad\qquad + c_2 rand_2()[p_{gd}(t) - p_{id}(t)] \\ x(t+1) = x(t) + v(t+1) \end{cases} \tag{1}$$

where $w \in [0,1]$ is the inertia weight, determining how much of the previous velocity of the particle is preserved. And $rand_1()$, $rand_2() \in [0,1]$ denote two uniform random numbers samples, c_1, c_2 are acceleration constants. From the iterative (1), it is shown that the second part represents the private thinking by itself, and the third part is the social part, which represents the cooperation among the individuals. In order to use the control theory to analyze the particle swarm optimization algorithm, change the formula (1) to matrix format

$$\begin{bmatrix} v(t+1) \\ x(t+1) \end{bmatrix} = A(t) \begin{bmatrix} v(t) \\ x(t) \end{bmatrix} + \begin{bmatrix} \varphi_1(t) & \varphi_2(t) \\ \varphi_1(t) & \varphi_2(t) \end{bmatrix} \begin{bmatrix} p_i(t) \\ p_g(t) \end{bmatrix} \tag{2}$$

where the variable $A(t) = \begin{bmatrix} \omega & -\varphi(t) \\ \omega & 1-\varphi(t) \end{bmatrix}$, $\varphi_1(t) = c_1 r_1(t)$, $\varphi_2(t) = c_2 r_2(t)$, $\varphi(t) = \varphi_1(t) + \varphi_2(t)$.

B. Modified Particle Swarm Optimization (MPSO)

The inertial weight w of the PSO algorithm coordinates the particles global and local search capabilities. When the parameter w is larger, the change rate for each particle is larger to keep the global search capability, whereas owning a better local approximate capacity. There has been a lot of research in determining the inertia weight. The linear decreasing weight was presented by Shi [14]and described as

$$w = w_{\max} - \frac{(w_{\max} - w_{\min})}{Maxstep} \times iter \tag{3}$$

where $w_{\min} = 0.4$, $w_{\max} = 0.9$, which are commonly accepted minimum and maximum change range for inertial weight [0,1]. $iter$ and $Maxstep$ denote the current iterative time and the maximum iterative time, respectively. Eberhart [15] proposed the PSO with a random inertia weight factor, where the inertia weight w changes according to

$$w = 0.5 - rand()/2 \tag{4}$$

where $rand()$ is a uniformly distributed random number within the range [0,1]. However, it is found that the inertia weight in these modified methods could not combine with the neural networks. In Equation (3), the parameter w is in the process of the transition at the most iterative times. So the global and local search capability of PSO algorithm is weakened. And also the Equation (4) is random method, which could not keep the NN continual approximate ability and achieve the stable predictive control. Therefore, this article adopts nonlinear ways to gradually reduce the weight of inertia, in accordance with cosine curve. Furthermore, the system error shrinks with the weight w, which ensures that the neural networks continuously approach the controlled system in the latter period. The update formula is shown as (5)

$$w = w_{\min} + (w_{\max} - w_{\min}) \times \frac{1 + \cos[(iter - 1)\pi / (Maxstep - 1)]}{2} \tag{5}$$

With the increase of variable *iter* and the decrease for system identification error, inertial weight w diminishes gradually. Therefore, the feedforward neural network with modified PSO algorithm could hold local approximate capacity in the later period of system control process. The pseudocode of the whole modified particle swarm optimization algorithm is shown as follow:

```
Start
  initialize the population
    evaluate the fitness: = f(x)
    update P_i and P_g
    while ( termination condition = false )
      do
        update inertia weight w using the formula (5)
        for ( i = 1 to number of particles )
            for ( d = 1 to number of dimensions )
                calculate new velocity v_id and update the
                    →position x_id using the formula (2)
            increase d
            evaluate the fitness: = f(x)
            update P_i and P_g
        increase i
    end do
end
```

III. DYNAMIC FEEDFORWARD NEURAL NETWORKS

Choose a multi-input, single-output feedforward neural network as an example to illustrate proposed dynamic feedforward Neural Networks (DNN), the structure is shown in Fig. 1. There is only one connection between each of the two neurons. The delayed of dynamic operators $z^{-\tau^{(1)}}$ and $z^{-\tau^{(2)}}$ are accessioned in the first hidden layer and input layer, the last hidden layer and output one, respectively. These two dynamic parameters are together adaptive adjusted with the connection weights w and the threshold b.

Implied in the hidden layers between the input and output layer relationships can be expressed, as (6) and (7), respectively

$$net_j^{(l)}(t) = \sum_{i=1}^{N^{(l-1)}} w_{ji}^{(l-1)} O_i^{(l-1)}(t) + b_{lj} \tag{6}$$

$$O_j^{(l)}(t) = f[net_j^{(l)}(t)] \tag{7}$$

where $net_j^{(l)}(t)$ and $O_j^{(l)}(t)$ denote the input and output of the jth neural in the lth layer at time t, respectively. The weight connecting the jth neuron in the lth layer to the ith neuron in the (l-1)th layer are represented by $w_{ji}^{(l)}$. Note that j varies from 1 to N^l, i varies from 1 to N^l. b_{lj} denotes the threshold parameter of the jth neural in the lth layer. $f[\cdot]$ is a nonlinear activation function, written as

$$f(x) = K\frac{1 - e^{-cx}}{1 + e^{-cx}} \tag{8}$$

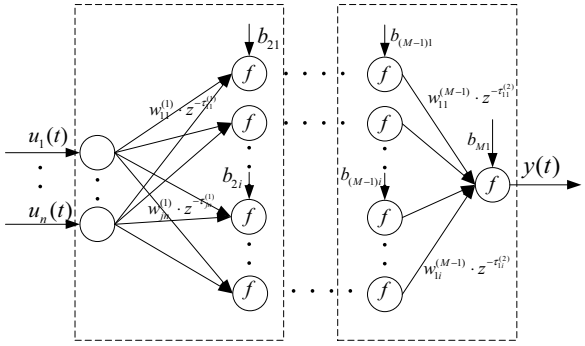

Fig. 1. The structure of proposed dynamic neural networks.

Joined the dynamic delay operators of the neural network connection, the description of the first hidden nodes to input and output neurons can be expressed respectively as

$$net_j^{(2)}(t) = \sum_{i=1}^{N^{(1)}} w_{ji}^{(1)} O_i^{(1)}(t - \tau_{ji}^{(1)}) + b_{2j} \tag{9}$$

$$net_j^{(M)}(t) = \sum_{i=1}^{N^{(M-1)}} w_{ji}^{(M-1)} O_i^{(M-1)}(t - \tau_{ji}^{(2)}) + b_{Mj} \tag{10}$$

here $\tau_{ji}^{(1)}$ and $\tau_{ji}^{(2)}$ denote the associated delay connecting the hidden layers with input and output layers, respectively.

In the proposed dynamic feedforward neural networks structure as shown in Fig. 1, consider the effect of input $u_1(t-1)$ to output $y(t)$, which is shown in (11)

$$y(t) = f[\sum_{1}^{N^{(M-1)}} w_{pq}^{(M-1)} \cdot (f \cdots (f(w_{ji}^{(1)} u_1(t - 1 - \tau_{ji}^{(1)} - \tau_{pq}^{(2)}) + b_{2j}) \cdots) + b_{M1}] \tag{11}$$

The parameters $\tau_{ji}^{(1)}$ and $\tau_{pq}^{(2)}$ are adaptive adjusted by MPSO

algorithm until reaching the minimum value of fitness function, and vary form 0 to $\tau_{\max}$. The weight connecting the *j*th neuron in the *l*th layer to the *i*th neuron in the (*l*-1)th layer are represented by $w_{ji}^{(l)}$. Furthermore, b_{lj} denotes the threshold parameter of the *j*th neural in the *l*th layer. As a result, each connection between the neurons can be seen on a random time delay so that the network has the time sequence. The current moment output may be related to a few moments of inputs, which enhances the dynamic nature of DNN. Through the identification of the controlled long time-delay system, the pure time delays are obtained by the delay operators between hidden and output layers. Therefore, the proposed DNN could successfully represent the nonlinear and delay characters of the system only with fewer parameters. After training, if the delay operators $\tau_{ji}^{(2)}$ were removed, the exact controlled plant without any delay is obtained, so predictive controller in MPC is obtained exactly. Due to the precise system outputs are predicted, the satisfactory control effect could be achieved.

IV. MPC BASED ON MPSO+DNN

The MPSO method is adopted to adaptively adjust the value of the connection weights w, threshold b and delay operators $z^{-\tau}$ in the DNN. This way avoids the complex calculation on repeated partial derivation in error feedback gradient descent methods, and accelerates the pace of convergence to the global optimum. The variables w, b, τ constitute the location of the particle k, as shown in Fig. 2

$x_k(t)$	$w_{11}^{(1)}(t),\cdots,w_{1i}^{(M-1)}(t)$	$b_{21}(t),\cdots,b_{M1}(t)$	$\tau_{11}^{(1)}(t),\cdots,\tau_{1i}^{(2)}(t)$

Fig. 2. The structure for the particles.

Then calculate the objective function of the particle fitness, and update each particle velocity, location and the optimal location of the groups. The objective function $f(x)$ is defined based on users' desired specification. Typical output specifications in the time domain are peak over-shooting, rise time, settling time, and steady-state error. The integral absolute error (IAE) performance index described in Equation (12) is usually considered as the objective function,

$$IAE = \int_0^{\infty} \sum_{i=1}^{n} \left| e_i(t) \right| dt \tag{12}$$

The purpose of MPC is to select signal $u(t)$, such that the output of system $y(t)$ is made as close as possible to be a expectation set-point $r(t)$. A schematic of the MPC based on MPSO+DNN is illustrated in Fig. 3.

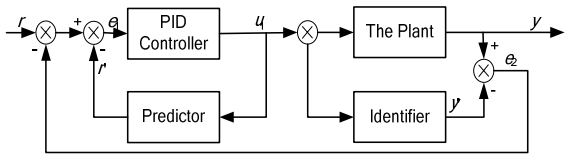

Fig. 3. The system block diagram of model predictive control on MPSO+DNN.

The main controller adopts the classic PID methods, and the identifier is our proposed MPSO+DNN after the system identification. On the other hand, the predictor is the same MPSO+DNN structure except the delay operators omitted between the last hidden layers and output layers, which could predict the plant output without pure delay exactly. Hence, through the first close-loop, the predictive single $r'(t)$ is transferred to the PID Controller in time. And then, the main controller reposes to change the single $u(t)$ according to r'. Otherwise, in the second close-loop, the identification error $e_2(t)$ is the difference between the actual plant output $y(t)$ and the identifier output $y'(t)$. Then the error $e_2(t)$ is transferred to PID controller to revise the deflection during the identification process. In addition, the feedback part of error $e_2(t)$ could overcome the disturbance in the place of controlled plant and the effects on the change of system model successfully.

V. THE SIMULATION ON PH NEUTRALIZATION PROCESS

PH neutralization process widely exists in the chemical industry, sewage treatment, biotechnology, power generation, and other industrial processes. Acid-base neutralization process owns high nonlinear features, and other factors, such as the valve and sensors, results in this process existing long time delay. The literature [16] provides Hammerstein's mathematical model to describe actual pH process, written as (13) and (14)

$$v(k) = u(k) - 1.207u^2(k) + 1.15u^3(k) \tag{13}$$

$$y(k) - 1.588y(k-1) + 0.597y(k-2) = 0.0185v(k-20) + 0.017v(k-21) + 0.597v(k-22) \tag{14}$$

Use the PID controller to control the above-mentioned system, and choose 1800 group input and output data as a teacher single to MPSO+DNN. Among these data, the first 900 group signals are adopted to train the neural networks, the rest part to test the generalization capability. Fig. 4 depicts the results of the system identification. And also the simulation condition is shown in Table I.

TABLE I
SIMULATION CONDITION

Name	Value
NN structure	2-20-1
Function	$f(x)=D(\frac{1-e^{-x}}{1+e^{-x}})$
Amplitude	$D=4$
Iterative Times	30000
Particle Number	15
Accelerate Coefficient	$c_1=2, c_2=2$
Dimension	141
Research Range	$(-25,25)^n$
Maximum velocity	$v_{\max}=0.5$
Maximum inertia weight	$w_{\max}=0.9$
Minimum inertia weight	$w_{\min}=0.4$

.Fig. 4 shows the improved algorithm can also achieve very good results to identify actual industrial process with the

nonlinear and long time delay. The training error is 0.000200, the generalization error is 0.000144, and the consuming time is 17.74 minute. The modified NN could achieve a satisfactory accuracy and better generalization capability. What's more, the consuming time is also greatly reduced, and the whole convergence process is speed up significantly. As shown in the Equation (14), the delay time is 20 sample period. After the NN system identification process, the delay operators between the last hidden and output layer are the integer in the range of [18, 21]. Therefore, our proposed MPSO+DNN structure could obtain the pure time parameters.

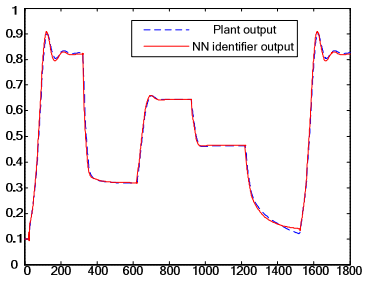

Fig. 4. The system identification results.

In order to verify the effectiveness of our proposed predictive control strategy, three control methods are compared with the example of PH neutralization process. The first one is classic PID single-closed-loop control, the second is PID control with the Smith predictor, and the last one is our proposed model predictive control with MPSO+DNN. The input single is the square wave signal, whose sampling period is 0.1s, and three thousand points in all. Fig. 5 presents the control results obtained by these control strategies for square signal and it shows a good tracking result.

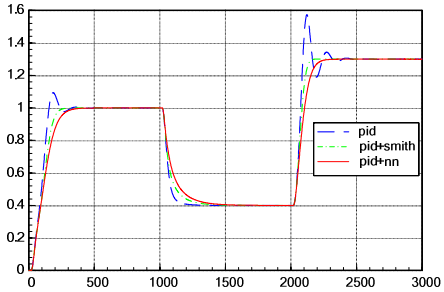

Fig. 5. The comparison of control performance effect with three methods.

From the figure, it is shown that our proposed control system could comes to a steady state rapidly without the any overshoot, and the performance is very close to the Smith predictive control. However, the main advantage of our predictive controller is designed in the case of system model unknown.

Furthermore, in order to analyze the disturbance situation, or the operation process and environment causing the change of system model, the experiment on robust characters is designed. At the 600th step, both the model coefficient and pure time delay parameter have varied as the following Equation (15)

$$v(k)=u(k)-1.207u^2(k)+1.15u^3(k)$$
$$y(k)-1.588y(k-1)+0.597y(k-2)=$$
$$0.0585v(k-15)+0.017v(k-16)+0.597v(k-17) \quad (15)$$

The performance is shown in Fig. 6. It is found that the novel model predictive control with MPSO+DNN could get back to the stable state with small transition time compared with the Smith method in the mismatch situation. Thus our proposed predictive control is less dependent on the system model and the robust character is better.

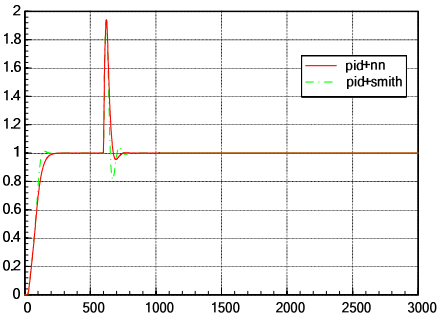

Fig. 6. The robust character analysis in the mismatch situation

VI. CONCLUSION

In this paper, a modified PSO algorithm is introduced to train the dynamic neural networks. Through the inertia weights decreasing, the PSO method could be fit for the neural network training process. What's more, the dynamic NN structure could represent the system nonlinear and the pure time delay exactly, and overcomes the drawback of gradient descent way easily falling into local optima points. The introduced dynamic delay operators make that the pure delay time could be obtain precisely for the model predict control. The simulation results show that the proposed model predict control with MPSO+DNN is superior to other predictive control methods. Moreover, the method is of strong robusticity on background disturbance and mismatch situation.

REFERENCES

[1] D. W. Clarke, C. Mohtadi, and P. S. Tuffs, "Generalized predictive control—Part I and Part II," Automatica, vol. 23, no. 2, pp. 137-160, 1987.

[2] Z. Song and A. Kusiak, Optimization of Temporal Processes: A Model Predictive Control Approach, IEEE Transactions on Evolutionary Computation, vol. 13, no. 1, pp. 169-179, 2009.

[3] B. Potocnik, G. Music, I. Skrjanc, and B. Zupancic, Model-based predictive control of hybrid systems: A probabilistic neural-network approach to real-time control, Journal of Intelligent & Robotic Systems, vol. 51,no. 7, pp. 45-63, 2008.

[4] X. J. Wu, X. J. Zhu, G. Y. Cao, and H. Y. Tu, Predictive control of SOFC based on a GA-RBF neural network model, Journal of Power Sources, vol. 179, no. 1, pp. 232-239, 2008.

[5] M. Jalili, S. Atashbari, S. Momenbellah, and F. H. Roudsari, Neural networks as a tool for nonlinear predictive control: Application to some benchmark systems, International Journal of Wavelets Multiresolution and Information Processing, vol. 5,no. 1, pp. 69-99, 2007.

[6] K. S. Narendra and K. Parthasarathy, “Identification and control of dynamical systems using neural networks,” IEEE Transaction on Neural Networks, vol. 1, no. 1, pp. 4-27, 1990.
[7] J. Kennedy, R. C. Eberhart. Particle Swarm Optimization. Proc IEEE Int Conf on Neural Networks. Piscataway, NJ: IEEE Press, 1995: 1942-1948.
[8] X. F. Xie, W. J. Zhang, Z. L. Yang. A Dissipative Particle Swarm Optimization. Congress on Evolutionary Computation, Honolulu, HI: USA, 2002: 1456-1461.
[9] L. Zhao, Y. Yang. “PSO-based single multiplicative neuron model for time series prediction” Expert system with application, in press.
[10] B. Niu, Y.L. Zhu, X.X. He, X.P. Zeng, MCPSO: amulti-swarm cooperative particle swarm optimizer. Application mathematic computation, vol. 185, no. 2, pp: 1050-1062, 2007.
[11] M. Lovbjerg, T.K. Rasmussen, T. Krink. Hybrid Particle Swarm Optimizer with Breeding and Subpopulations.Proc. Third Genetic and Evolutionary Computation Congress, San Diego, USA, p.469-476, 2001
[12] U. Yuzgec, Y. Becerikli, and M. Turker, Dynamic neural-network-based model-predictive control of an industrial baker's yeast drying process, IEEE Transactions on Neural Networks, vol. 19, no. 1, pp. 1231-1242, 2008.
[13] T. J. Zhang and J. H. Lu, A pso-based multivariable Fuzzy Decision-Making predictive controller for a once-through 300-mw power plant, Cybernetics and Systems, vol. 37, no. 5, pp. 417-441, 2006.
[14] Y. Shi, R.C. Eberhart, A Modified Particle Swarm Optimizer. Proc. IEEE Int. Conf. on Evolutionary Computation,Anchorage, Alaska, p.69-73, 1998.
[15] R.C. Eberhart, Y. Shi. Tracking and Optimizing Dynamic Systems with Particle Swarms. Proc. IEEE Congress on Evolutionary Computation, Seoul, Korea, pp.94-97. 2001a
[16] R. Paravi Torghabeh and H. Khaloozadeh. Neural Networks Hammerstein Model Identification Based On Particle Swarm Optimization, in Networking, Sensing and Control, 2008. ICNSC 2008. IEEE International Conference on, 2008, pp. 363-367.