Construction of Fuzzy Models for Dynamic Systems Using Multi-population Cooperative Particle Swarm Optimizer

Ben Niu[1, 2], Yunlong Zhu[1], and Xiaoxian He[1, 2]

[1] Shenyang Institute of Automation, Chinese Academy of Sciences,
110016, Shenyang, China,
[2] School of Graduate, Chinese Academy of Sciences,
100039, Beijing, China
{niuben, ylzhu}sia.cn

Abstract. A new fuzzy modeling method using Multi-population Cooperative Particle Swarm Optimizer (MCPSO) for identification and control of nonlinear dynamic systems is presented in this paper. In MCPSO, the population consists of one master swarm and several slave swarms. The slave swarms execute Particle Swarm Optimization (PSO) or its variants independently to maintain the diversity of particles, while the particles in the master swarm enhance themselves based on their own knowledge and also the knowledge of the particles in the slave swarms. The MCPSO is used to automatic design of fuzzy identifier and fuzzy controller for nonlinear dynamic systems. The proposed algorithm (MCPSO) is shown to outperform PSO and some other methods in identifying and controlling dynamic systems.

1 Introduction

The identification and control of nonlinear dynamical systems has been a challenging problem in the control area for a long time. Since for a dynamic system, the output is a nonlinear function of past output or past input or both, and the exact order of the dynamical systems is often unavailable, the identification and control of this system is much more difficult than that has been done in a static system. Therefore, the soft computing methods such as neural networks [1-3], fuzzy neural networks [4-6] and fuzzy inference systems [7] have been developed to cope with this problem.

Recently, interest in using recurrent networks has become a popular approach for the identification and control of temporal problems. Many types of recurrent networks have been proposed, among which two widely used categories are recurrent neural networks (RNN) [3, 8, 9] and recurrent fuzzy networks (RFNN) [4, 10].

On the other hand, fuzzy inference systems have been developed to provide successful results in identifying and controlling nonlinear dynamical systems [7, 11]. Among the different fuzzy modeling techniques, the Takagi and Sugeno's (T-S) type fuzzy controllers have gained much attention due to its simplicity and generality [7, 12]. T-S fuzzy model describes a system by a set of local linear input-output relations and it is seen that this fuzzy model can approach highly nonlinear dynamical systems.

L. Wang and Y. Jin (Eds.): FSKD 2005, LNAI 3613, pp. 987 – 1000, 2005.
© Springer-Verlag Berlin Heidelberg 2005

The bottleneck of the construction of a T-S model is the identification of the antecedent membership functions, which is a nonlinear optimization problem. Typically, both the premise parameters and the consequent parameters of T-S fuzzy model are adjusted by using gradient descent optimization techniques [12-13]. Those methods are sensitive to the choice of the initial parameters, easily got stuck in local minima, and have poor generalization properties. This hampers the aposteriori interpretation of the optimized T-S model.

The advent of evolutionary algorithm (EA) has attracted considerable interest in the construction of fuzzy systems [14-16]. In [15], [16], EAs have been applied to learn both the antecedent and consequent part of fuzzy rules, and models with both fixed and varying number of rules have been considered. As compared to traditional gradient-based computation system, evolutionary algorithm provides a more robust and efficient approach for the construction of fuzzy systems.

Recently, a new evolutionary computation technique, the particle swarm optimization (PSO) algorithm, is introduced by Kennedy and Eberhart [17, 18], and has already come to be widely used in many areas [19-21]. As already has been mentioned by Angeline [22], the original PSO, while successful in the optimization of several difficult benchmark problems, presented problems in controlling the balance between exploration and exploitation, namely when fine tuning around the optimum is attempted.

In this paper we try to deal with this issue by introducing a multi-population scheme, which consists of one master swarm and several slave swarms. The slave swarms evolve independently to supply new promising particles (the position giving the best fitness value) to the master swarm as evolution goes on. The master swarm updates the particle states based on the best position discovered so far by all the particles both in the slave swarms and its own. The interactions between the master swarm and the slave swarms control the balance between exploration and exploitation and maintain the population diversity, even when it is approaching convergence, thus reducing the risk of convergence to local sub-optima.

The paper is devoted to a novel fuzzy modeling strategy to the fuzzy inference system for identification and control of nonlinear dynamical systems. In this paper, we will use the MCPSO algorithm to design the T-S type fuzzy identifier and fuzzy controller for nonlinear dynamic systems, and the performance is also compared to other methods to demonstrate its effectiveness.

The paper is organized as follows. Section 2 gives a review of PSO and a description of the proposed algorithm MCPSO. Section 3 describes the T-S model and a detailed design algorithm of fuzzy model by MCPSO. In Section 4, simulation results of one nonlinear plant identification problem and one nonlinear dynamical system control problems using fuzzy inference systems based on MCPSO are presented. Finally, conclusions are drawn in Section 5.

2 PSO and MCPSO

2.1 Particle Swarm Optimization

Particle Swarm Optimization (PSO) is inspired by natural concepts such as fish schooling, bird flocking and human social relations. The basic PSO is a population

based optimization tool, where the system is initialized with a population of random solutions and searches for optima by updating generations. In PSO, the potential solutions, called particles, fly in a *D*-dimension search space with a velocity which is dynamically adjusted according to its own experience and that of its neighbors.

The location and velocity for the *i*th particle is represented as $x_i = (x_{i1}, x_{i2}, ...x_{iD})$ and $v_i = (v_{i1}, v_{i2}, ...v_{iD})$, respectively. The best previous position of the *i*th particle is recorded and represented as $P_i = (P_{i1}, P_{i2}, ..., P_{iD})$, which is also called *pbest*. The index of the best particle among all the particles in the population is represented by the symbol g, and p_g is called *gbest*. At each time step t, the particles are manipulated according to the following equations:

$$v_i(t+1) = v_i(t) + R_1 c_1 (P_i - x_i(t)) + R_2 c_2 (P_g - x_i(t)) \tag{1}$$

$$x_i(t+1) = x_i(t) + v_i(t) \tag{2}$$

where R_1 and R_2 are random values within the interval [0, 1], c_1 and c_2 are acceleration constants. For Eqn. (1), the portion of the adjustment to the velocity influenced by the individual's own *pbest* position is considered as the cognition component, and the portion influenced by *gbest* is the social component.

A drawback of the aforementioned version of PSO is associated with the lack of a mechanism responsible for the control of the magnitude of the velocities, which fosters the danger of swarm explosion and divergence. To solve this problem, Shi and Eberhart [23] later introduced an inertia term w by modifying (1) to become:

$$v_i(t+1) = w \times v_i(t) + R_1 c_1 (P_i - x_i(t)) + R_2 c_2 (P_g - x_i(t)) \tag{3}$$

They proposed that suitable selection of w will provides a balance between global and local explorations, thus requiring less iteration on average to find a sufficiently optimal solution. As originally developed, w often decreases linearly from about 0.9 to 0.4 during a run. In general, the inertia weight w is set according to the following equation:

$$w = w_{\max} - \frac{w_{\max} - w_{\min}}{iter_{\max}} \times iter \tag{4}$$

where $iter_{\max}$ is the maximum number of iterations, and $iter$ is the current number of iterations.

2.2 Multi-population Cooperative Particle Swarm Optimization

The foundation of PSO is based on the hypothesis that social sharing of information among conspecifics. It reflects the cooperative relationship among the individuals (fish, bird, insect) within a group (school, flock, swarm). Obviously it is not the case

of the nature. In natural ecosystem, many species have developed cooperative interactions with other species to improve their survival. Such cooperative—also called symbiosis—co-evolution can be found in organisms going from cells (e.g., eukaryotic organisms resulted probably from the mutualistic interaction between prokaryotes and some cells they infected) to superior animals (e.g., African tick birds obtain a steady food supply by cleaning parasites from the skin of giraffes, zebras, and other animals) [24, 25].

Inspired by the phenomenon of symbiosis in the natural ecosystem, a master-slave mode is incorporated into the PSO, and a Multi-population (species) Cooperative Optimization (MCPSO) is thus presented. In our approach, the population consists of one master swarm and several slave swarms. The symbiotic relationship between the master swarm and slave swarms can keep a right balance of exploration and exploitation, which is essential for the success of a given optimization task.

The master-slave communication model is shown in Fig.1, which is used to assign fitness evaluations and maintain algorithm synchronization. Independent populations (species) are associated with nodes, called slave swarms. Each node executes a single PSO or its variants, including the update of location and velocity, and the creation of a new local population. When all nodes are ready with the new generations, each node then sends the best local individual to the master node. The master node selects the best of all received individuals and evolves according to the following equations:

$$v_i^M(t+1) = wv_i^M(t) + R_1c_1(p_i^M - x_i^M(t)) + R_2c_2(p_g^M - x_i^M(t)) + R_3c_3(p_g^S - x_i^M(t)) \quad (5)$$

$$x_i^M(t+1) = x_i^M(t) + v_i^M(t) \quad (6)$$

where M represents the master swarm, c_3 is the migration coefficient, and R_3 is a uniform random sequence in the range [0, 1]. Note that the particle's velocity update in the master swarm is associated with three factors:

i. p_i^M : Previous best position of the master swarm.

ii. P_g^M : Best global position of the master swarm.

iii. p_g^S : Previous best position of the slave swarms.

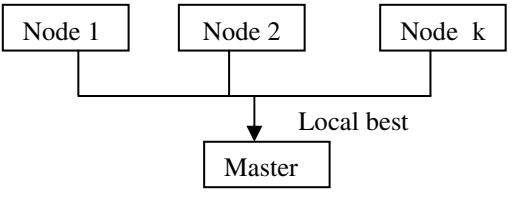

Fig. 1. The master-slave model

As Shown in Eqn. (5), the first term of the summation represents the inertia (the particle keeps moving in the direction it had previously moved), the second term represents memory (the particle is attracted to the best point in its trajectory), the third term represents cooperation (the particle is attracted to the best point found by all particles of master swarm) and the last represents information exchange (the particle is attracted to the best point found by the slave swarms). The pseudocode for the MCPSO algorithm is listed in Fig 2.

```
Algorithm MCPSO
Begin
    Initialize all the populations
    Evaluate the fitness value of each particle
    Repeat
        Do in parallel
            Node i, 1 ≤ i ≤ K  //K is the number of slaver swarms
        End Do in parallel
        Barrier synchronization  //wait for all processes to finish
        Select the fittest local individual p_g^S from the slave swarms
        Evolve the mast swarm
        // Update the velocity and position using (5) and (6), respectively
        Evaluate the fitness value of each particle
    Until a terminate-condition is met
End
```

Fig. 2. Pseudocode for the MCPSO algorithm

3 Fuzzy Model Based on MCPSO

3.1 T-S Fuzzy Model Systems

In this paper, the fuzzy model suggested by Takagi and Sugeno is employed to represent a nonlinear system. A T-S fuzzy system is described by a set of fuzzy IF-THEN rules that represent local linear input-output relations of nonlinear systems. The overall system is then an aggregation of all such local linear models. More precisely, a T-S fuzzy system is formulated in the following form:

$$R^l : if\ x_1 \text{ is } A_1^l \text{ and } \ldots x_n \text{ is } A_n^i, \text{ then } \hat{y}^l = \alpha_0^l + \alpha_1^l x_1 + \ldots + \alpha_n^l x_n, \tag{7}$$

where $\hat{y}^l (1 \le l \le r)$ is the output due to rule R^l and $\alpha_i^l (1 \le i \le n)$, called the consequent parameters, are the coefficients of the linear relation in the *l*th rule and will be identified. $A_i^l(x_i)$ are the fuzzy variables defined as the following Gaussian membership function:

$$A_i^l(x_i) = \exp[-\frac{1}{2} * (\frac{x_i - m_i^l}{\sigma_i^l})^2], \tag{8}$$

where $1 \le l \le r, ..., 1 \le i \le n, x_i \in R$, m_i^l and σ_i^l represent the center (or mean) and the width (or standard deviation) of the Gaussian membership function, respectively. m_i^l and σ_i^l are adjustable parameters called the premise parameters, which will be identified.

Given an input $(x_1^0(k), \cdots, x_n^0(k))$, the final output $\hat{y}(k)$ of the fuzzy system is inferred as follows:

$$\hat{y}(k) = \frac{\sum_{l=1}^{r} \hat{y}^l(k)(\prod_{i=1}^{n} A_i^l(x_i^0(k)))}{\sum_{l=1}^{r}(\prod_{i=1}^{n} A_i^l(x_i^0(k)))} = \frac{\sum_{l=1}^{r} \hat{y}^l(k) w^l(k)}{\sum_{l=1}^{r} w^l(k)}, \tag{9}$$

where the weight strength $w^l(k)$ of the lth rule, is calculated by:

$$w^l(k) = \prod_{i=1}^{n} A_i^l(x_i^0(k)). \tag{10}$$

3.2 Fuzzy Model Strategy Based on MCPSO

The detailed design algorithm of fuzzy model by MCPSO is introduced in this section. The overall learning process can be described as follows:

(1) *Parameter representation*

In our work, the parameter matrix, which consists of the premise parameters and the consequent parameters described in section 3.1, is defined as a two dimensional matrix, i.e.,

$$\begin{bmatrix} m_1^1 & \sigma_1^1 & \cdots & m_n^1 & \sigma_n^1 & \alpha_0^1 & \alpha_1^1 & \cdots & \alpha_n^1 \\ m_1^2 & \sigma_1^2 & \cdots & m_n^2 & \sigma_n^2 & \alpha_0^2 & \alpha_1^2 & \cdots & \alpha_n^2 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ m_1^r & \sigma_1^r & & m_n^r & \sigma_n^r & \alpha_0^r & \alpha_1^r & \cdots & \alpha_n^r \end{bmatrix}$$

The size of the matrix can be represented by $D = r \times (3n+1)$.

(2) *Parameter learning*

a) In MCPSO, the master swarm and the slave swarm both work with the same parameter settings except for the velocity update equation. Initially, $N \times n (N \ge 2,\ n \ge 2)$ individuals forming the population should be randomly generated and the individuals can be divided into N swarms (one master swarm and $N-1$ slave swarms). Each swarm contains n individuals with random

positions and velocities on D dimensions. These individuals may be regarded as particles in terms of PSO. In T-S fuzzy model system, the number of rules, r, should be assigned in advance. In addition, the maximum iterations $w_{\max}$, minimum inertia weight $w_{\min}$ and the learning parameters c_1, c_2, the migration coefficient c_3 should be assigned in advance. After initialization, new individuals on the next generation are created by the following step.

b) For each particle, evaluate the desired optimization fitness function in D variables. The fitness function is defined as the reciprocal of RMSE (root mean quadratic error), which is used to evaluate various individuals within a population of potential solutions. Considering the single output case for clarity, our goal is to minimize the error function:

$$RMSE = \sqrt{\frac{1}{K}\sum_{k=1}^{K}(y_p(k+1) - y_r(k+1))^2} \tag{11}$$

where K is the total time steps, $y_p(k+1)$ is the inferred output and $y_r(k+1)$ is the desired reference output..

c) Evaluate the fitness for each particle.

d) Compare the evaluated fitness value of each particle with it's $pbest$. If current value is better than $pbest$, then set the current location as the $pbest$ location in D-dimension space. Furthermore, if current value is better than $gbest$, then reset $gbest$ to the current index in particle array. This step will be executed in parallel for both the master swarm and the slave swarms.

e) In each generation, after step d) is executed, the best-performing particle p_g^S among the slave swarms should be marked.

f) Update the velocity and position of all the particles in $N-1$ slave swarms according to Eqn. (3) and Eqn. (2), respectively (Suppose that $N-1$ populations of SPSO with the same parameter setting are involved in MCPSO as the slave swarms).

g) Update the velocity and position of all the particles in the master swarm according to Eqn. (5) and Eqn. (6), respectively.

(3) *Termination condition*

The computations are repeated until the premise parameters and consequent parameters are converged. It should be noted that after the operation in master swarm and slaver swarm the values of the individual may exceed its reasonable range. Assume that the domain of the ith input variable has been found to be $[\min(x_i),\ \max(x_i)]$ from training data, then the domains of m_i^l and σ_i^l are defined as $[\min(x_i) - \delta_i,\ \max(x_i) + \delta_i]$ and $[d_i - \delta_i,\ d_i + \delta_i]$, respectively, where δ_i is a small positive value defined as $\delta_i = (\max(x_i) - \min(x_i))/10$, and d_i is the predefined width of the Gaussian membership function, the value is set as $(\max(x_i) - \min(x_i))/r$, the variable r is the number of fuzzy rules.

4 Illustrative Examples

In this section, two nonlinear dynamic applications, including an example of identification of a dynamic system and an example of control of a dynamic system are conducted to validate the capability of the fuzzy inference systems based on MCPSO to handle the temporal relationship. The main reason for using these dynamic systems is that they are known to be stable in the bounded input bounded output (BIBO) sense.

A. Dynamic System Identification

The systems to be identified are dynamic systems whose outputs are functions of past inputs and past outputs as well. For this dynamic system identification, a serial-parallel model is adopted as identification configuration shown in Fig.3.

Example 1: The plant to be identified in this example is guided by the difference equation [2, 4]:

$$y_p(k+1) = f[y_p(k), y_p(k-1), y_p(k-2), u(k), u(k-1)], \tag{12}$$

where

$$f[x_1, x_2, x_3, x_4, x_5] = \frac{x_1 x_2 x_3 x_5 (x_3 - 1) + x_4}{1 + x_3^2 + x_2^2}. \tag{13}$$

Here, the current output of the plant depends on three previous outputs and two previous inputs. Unlike the authors in [1] who applied a feedforward neural network with five input nodes for feeding appropriate past values of $y_p(k)$ and $u(k)$, we only use the current input $u(k)$ and the output $y_p(k)$ as the inputs to identify the output of the plant $y_p(k+1)$. In training the fuzzy model using MCPSO for the nonlinear plant, we use only ten epochs and there are 900 time steps in each epoch. Similar to the inputs used in [2, 3]. The input is an independent and identically distributed (*iid)* uniform sequence over [-2, 2] for about half of the 900 time steps and a single sinusoid given by $1.05 * \sin(\pi k / 45)$ for the remaining time steps. In applying MCPSO to this plant, the number of swarms $N = 4$, the population size of each swarm $n = 20$, are chosen, i.e., 80 individuals are initially randomly generated in a population. The number r of the fuzzy rules is set to be 4. For master swarm, inertial weights $w_{\max}$, $w_{\min}$, the acceleration constants c_1, c_2, and the migration coefficient c_3, are set to 0.35, 0.1, 1.5, 1.5 and 0.8, respectively. In slave swarms the inertial weights and the acceleration constants are the same as those used in master swarm. To show the superiority of MCPSO, the fuzzy identifiers designed by PSO are also applied to the same identification problem. In PSO, the population size is set as 80 and initial individuals are the same as those used in MCPSO. For fair comparison, the

other parameters, $w_{\max}$, $w_{\min}$, c_1, c_2 are the same as those defined in MCPSO. To see the identified result, the following input as used in [3, 4] is adopted for test:

$$\begin{aligned} u(k) &= \sin(\pi k/25), \quad k < 250 \\ &= 1.0, \quad 250 \le k < 500 \\ &= -1.0, \quad 500 \le k < 750 \\ &= 0.3\sin(\pi k/25) + 0.1\sin(\pi k/32) + 0.6\sin(\pi k/10), \quad 750 \le k < 1000. \end{aligned} \tag{14}$$

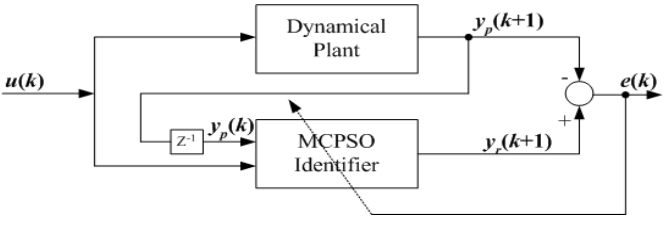

Fig. 3. Identification of nonlinear plant using MCPSO

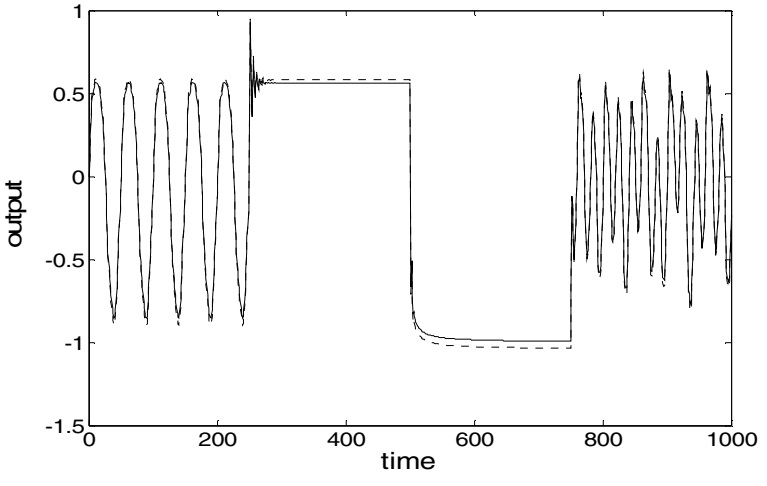

Fig. 4. Identification results using MCPSO in Example 1, where the solid curve denotes desired output and the dotted curve denotes the actual output

Table 1. Performance comparisons with different methods for Example 1

Method	RSONFIN	RFNN	TRFN-S	PSO	MCPSO
RMSE(train)	0.0248	0.0114	0.0084	0.0386	0.0146
RMSE (test)	0.0780	0.0575	0.0346	0.0372	0.0070

Fig.4 shows the desired output (denoted as a solid curve) and the inferred output obtained by using MCPSO (denoted as a dotted curve) for the testing input signal. Table 1 gives the detailed identification results using different methods, where the results of the methods RSONFIN, RFNN and TRFN-S come from literature [5]. From the comparisons, we see that the fuzzy controller designed by MCPSO is superior to the method using RSONFIN, and is slight inferior to the methods using RFNN and TRFN-S. However, among the three types of methods, it achieves the highest identification accuracy in the test part, which demonstrates its better generalized ability. The results of MCPSO identifier also demonstrate the improved performance compared to the results of the identifiers obtained by PSO. The abnormal phenomenon that the test RMSE is smaller than the train RMSE using fuzzy identifier based on PSO and MCPSO may attributes to the well-regulated input data in test part (during time steps [250, 500] and [500, 750], the input data is equal to a constant).

B. Dynamic System Control

As compare to linear systems, for which there now exists considerable theory regarding adaptive control, very litter is known concerning adaptive control of plants governed by nonlinear equations. It is in the control of such systems that we are primarily interested in this section. Based on MCPSO, the fuzzy controller is designed for the control of dynamical systems. The control configuration and input-output variables of MCPSO fuzzy controller are shown in Fig.5, and are applied to one MISO (multi-input-single-output) plant control problem in the following example. The comparisons with other control methods are also presented.

Example 2: The controlled plant is the same as that used in [2] and [5] and is given by

$$y_p(k+1) = \frac{y_p(k)\,y_p(k-1)(y_p(k)+2.5)}{1+y_p^2(k)+y_p^2(k-1)} + u(k). \tag{15}$$

In designing the fuzzy controller using MCPSO, the desired output y_r is specified by the following 250 pieces of data:

$$y_r(k+1) = 0.6\,y_r(k) + 0.2\,y_r(k-1) + r(k), 1 \le k \le 250,$$
$$r(k) = 0.5\sin(2\pi k/45) + 0.2\sin(2\pi k/15) + 0.2\sin(2\pi k/90).$$

The inputs to MCPSO fuzzy controller are $y_p(k)$ and $y_r(k)$ and the output is $u(k)$. There are five fuzzy rules in MCPSO fuzzy controller, i.e., $r = 5$, resulting in total of 35 free parameters. Other parameters in applying MCPSO are the same as those used in Example 1. The fuzzy controller designed by PSO is also applied to the MISO control problem and the parameters in using PSO are also the same as those defined in Example 1.The evolution is processed for 100 generations and is repeated for 50 runs. The averaged best-so-far RMSE value over 50 runs for each generation is shown in Fig.6.

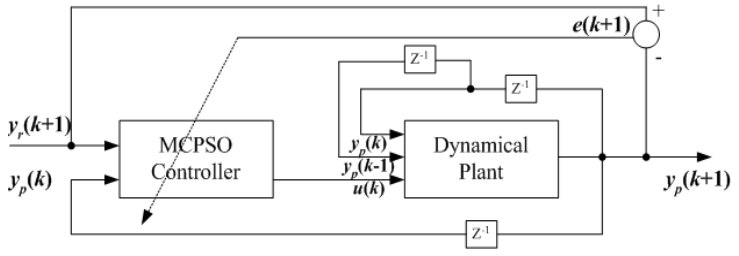

Fig. 5. Dynamical system control configuration with MCPSO fuzzy controller

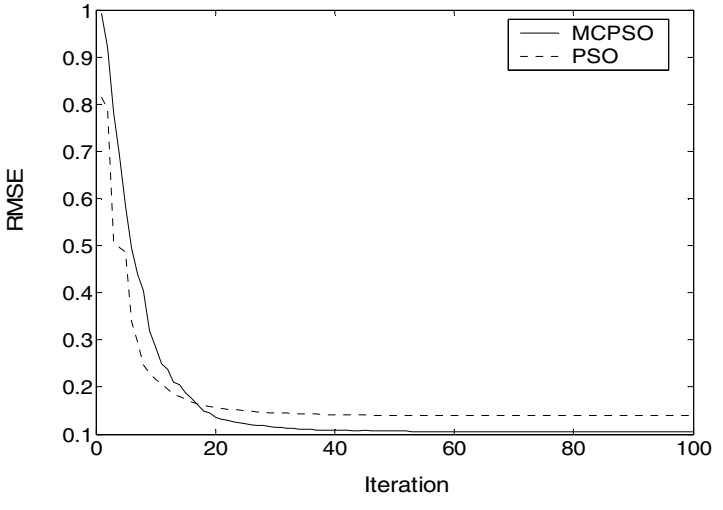

Fig. 6. Average best-so-far RMSE in each generation for PSO and MCPSO in Example 2

From the figure, we can see that MCPSO converges with a higher speed compared to PSO and obtains a better result. In fact, since the competition relationships of the slave swarms, the master swarm will not be influenced much when a certain slave swarms gets stuck at a local optima. Avoiding premature convergence allows MCPSO continue search for global optima in optimization problems

The best and averaged RMSE error for the 50 runs after 100 generations of training for each run are listed in Table 2, where the results of the methods GA and HGAPSO are from [6]. It should be noted that the TRFN controller designed by HGPSO (or GA) is evolved for 100 generations and repeated for 100 runs in literature [6]. To test the performance of the designed fuzzy controller, another reference input $r(k)$ is given by:

$$r(k) = 0.3\sin(2\pi k / 50) + 0.2\sin(2\pi k / 25) + 0.4\sin(2\pi k / 60), 251 \le k \le 500.$$

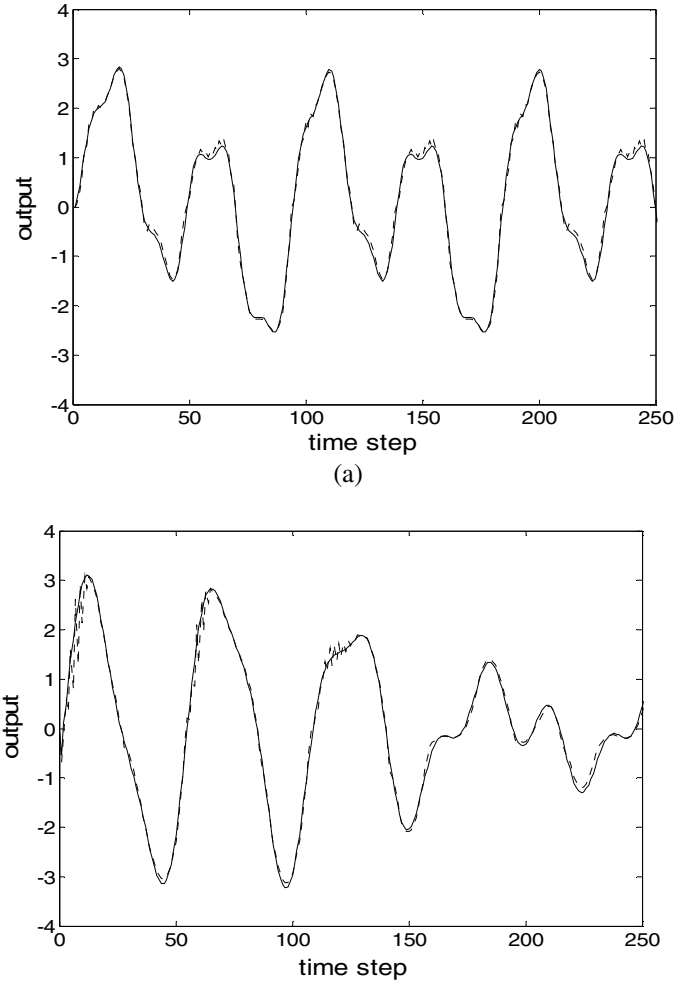

Fig. 7. The tracking performance by MCPSO controller in Example 2 for (a) training and (b) test reference output, where the desired output is dotted as solid curve and the actual output by a dotted curve

The best and averaged control performance for the test signal over 50 runs is also listed in Table 2. From the comparison results, we can see that the fuzzy controller based on MCPSO outperforms those based on GA and PSO greatly especially in the test results, and reaches the same control level with the TRFN controller based on HGPSO.

To demonstrate control performance using the MCPSO fuzzy controller for the MISO control problem, one control performance of MCPSO is shown in Fig.7 for both training and test control reference output.

Table 2. Performance comparisons with different methods for Example 2

Method	GA	PSO	HGAPSO	MCPSO
RMSE (train mean)	0.2150	0.1364	0.0890	0.1024
RMSE (train best)	0.1040	0.0526	0.0415	0.0518
RMSE (test mean)	—	0.1526	—	0.1304
RMSE (test best)	—	0.1024	—	0.0704

5 Conclusions

The paper proposed a multi-population cooperative particle swarm optimizer to identify the T-S fuzzy model for processing nonlinear dynamic systems. In the simulation part, we apply the suggested method to respectively design a fuzzy identifier for a nonlinear dynamic plant identification problem and a fuzzy controller for a nonlinear dynamic plant control problem. To demonstrate the effectiveness of the proposed algorithm MCPSO, its performance is compared to several typical methods in dynamical systems.

Acknowledgements

This work is supported by the National Natural Science Foundation of China (No.70431003) and the National Basic Research Program of China (No. 2002CB312200). The first author would like to thank Prof. Q.H Wu of Liverpool University for many valuable comments. Helpful discussions with Dr. B. Ye, Dr. L.Y. Yuan and Dr. S. Liu are also gratefully acknowledged.

References

1. Narenda, K. S., Parthasarathy, K: Adaptive identification and control of dynamical systems using neural networks. In: Proc. of the 28th IEEE Conf. on Decision and Control, Vol. 2. Tampa, Florida, USA (1989) 1737-1738
2. Narenda, K. S., Parthasarathy, K.: Identification and control of dynamical systems using neural networks. IEEE Trans. Neural Networks 1 (1990) 4-27
3. Sastry, P. S., Santharam, G., Unnikrishnan, K. P.: Memory neural networks for identification and control of dynamical systems. IEEE Trans. Neural Networks 5 (1994) 306-319
4. Lee, C. H., Teng, C. C.: Identification and control of dynamic systems using recurrent fuzzy neural networks. IEEE Trans. Fuzzy Syst. 8 (2000) 349-366
5. Juang, C. F.: A TSK-type recurrent fuzzy network for dynamic systems processing by neural network and genetic algorithms. IEEE Trans. Fuzzy Syst. 10 (2002) 155-170
6. Juang, C. F.: A hybrid of genetic algorithm and particle swarm optimization for recurrent network design. IEEE Trans. Syst. Man Cyber. B 34 (2004) 997-1006
7. Tseng, C. S., Chen, B. S., Uang, H. J.: Fuzzy tracking control design for nonlinear dynamical systems via T-S fuzzy model. IEEE Trans. Fuzzy Syst. 9 (2001) 381-392

8. Chow, T. W. S., Yang, F.: A recurrent neural-network-based real-time learning control strategy applying to nonlinear systems with unknown dynamics, IEEE Trans. Industrial Electronics 45 (1998) 151-161
9. Gan, C., Danai, K.: Model-based recurrent neural network for modeling nonlinear dynamic systems. IEEE Trans. Syst., Man, Cyber. 30 (2000) 344-351
10. Juang, C. F., Lin, C. T.: A Recurrent self-constructing neural fuzzy inference network. IEEE Trans. neural networks. 10 (1999) 828-845
11. Wang, L. X., Mendel, J.M.: Back-propagation fuzzy systems as nonlinear dynamic systems identifiers. In: Proc. IEEE Int. Conf. Fuzzy Syst., San Diego, USA (1992) 1409-1418
12. Takagi, T., Sugeno, M.: Fuzzy identification of systems and its application. IEEE Trans. Syst., Man, Cyber. 15 (1985) 116-132
13. Tanaka K., Ikeda, T., Wang, H. O.: A unified approach to controlling chaos via an LMI-based fuzzy control system design. IEEE Trans. Circuits and Systems 45 (1998) 1021-1040
14. Karr, C. L.: Design of an adaptive fuzzy logic controller using a genetic algorithm. In: Proc. of 4th Int. Conf. Genetic Algorithms, San Diego, USA (1991) 450–457
15. Wang, C. H., Hong, T. P., Tseng, S.S.: Integrating fuzzy knowledge by genetic algorithms. IEEE Trans. Evol. Comput. 2 (1998) 138-149
16. Ishibuchi, H., Nakashima, T. and Murata, T.: Performance evaluation of fuzzy classifier systems for multi dimensional pattern classification problems. IEEE Trans. Syst., Man, Cyber. B 29 (1999) 601-618
17. Eberhart, R. C., Kennedy, J.: A new optimizer using particle swarm theory. In: Proc. of Int. Sym. Micro Mach. Hum. Sci., Nagoya, Japan (1995) 39-43
18. Kennedy, J., Eberhart, .R. C.: Particle swarm optimization. In: Proc. of IEEE Int. Conf. on Neural Networks, Piscataway, NJ (1995) 1942-1948
19. Zhang, C., Shao, H., Li, Y.: Particle swarm optimization for evolving artificial network. In: Proc. of IEEE Int. Conf. Syst., Man, Cyber., Vol.4. Nashville, Tennessee, USA (2000) 2487–2490
20. Engelbrecht, A. P., Ismail, A. : Training product unit neural networks. Stability Control: Theory Appl. 2 (1999) 59–74
21. Mendes, R., Cortez, P. Rocha, M. and Neves, J.: Particle swarms for feedforward neural network training. In: Proc. of Int. Joint Conf. on Neural Networks, Honolulu, USA (2002) 1895–1899
22. Angeline, P. J.: Evolutionary optimization versus particle swarm optimization: philosophy and performance difference. In: Proc. of the 7th Annual Conf. on Evolutionary Programming, San Diego, USA (1998) 601-610
23. Shi, Y., Eberhart, R. C.: A modified particle swarm optimizer. Proc. of IEEE Int. Conf. on Evolutionary Computation, Anchorage, USA (1998) 69-73
24. Moriartv, D., Miikkulainen: Reinforcement learning through symbiotic evolution Machine learning. 22 (1996) 11-32
25. Wiegand, R. P.: An analysis of cooperative coevolutionary Algorithms. PhD thesis, George Mason University, Fairfax, Virginia, USA (2004)