International Journal of Innovative Computing, Information and Control
Volume **9**, Number **7**, July **2013**

ICIC International ©2013 ISSN 1349-4198
pp. **2963–2977**

MEMBRANE OPTIMIZATION ALGORITHM BASED ON MUTATED PSO AND ITS APPLICATION IN NONLINEAR CONTROL SYSTEMS

Jun Wang[1], Tao Wang[2], Peng Shi[3,4], Min Tu[1] and Fan Yang[1]

[1]School of Electrical and Information Engineering
Xihua University
No. 999, Jinzhou Road, Chengdu 610039, P. R. China
junwang66@tom.com

[2]School of Electrical Engineering
Southwest Jiaotong University
No. 111, North Sec. 1, 2nd Ring Road, Chengdu 610031, P. R. China
wangatao2005@163.com

[3]School of Engineering and Science
Victoria University Melbourne
Vic 8001, Australia

[4]School of Electrical and Electronic Engineering
The University of Adelaide
Adelaide, SA 5005, Australia

Received April 2012; revised August 2012

Abstract. *The PID neural network model was proposed for complex control systems in order to achieve desirable control performance. However, the conventional backward-propagation (BP) algorithm restrains the model's wide applications in control field due to the known reasons. In this paper, a novel variant of particle swarm optimization (PSO), named membrane optimization algorithm based on mutated particle swarm optimization (MO-MPSO) is proposed. The MO-MPSO algorithm is an appropriate combination of membrane computing, evolution rules of PSO algorithms and a mutation operator. Comparison experiments on benchmark functions and the case study show the effectiveness and advantages of the presented algorithm.*
Keywords: Membrane computing, PSO, PIDNN, Mutation

1. **Introduction.** Membrane computing, firstly introduced by G. Păun in 2000, is a new attractive research field of computer science aiming to abstract computing models from the functioning and structures of living cells as well as from the way the cells are organized in tissues or higher order structures. The obtained models, called membrane systems or P systems, are distributed and parallel computing models [1]. Under the same running conditions, the rational utilization of optimization technique can not only improve productivity effect, but also decrease the energy wastage and allocate resources reasonably. Therefore, the investigations about optimization technique attract many scholars coming from both home and abroad. As a novel class of bio-inspired computing models, membrane computing has attracted so much attention and has been gradually applied into different areas [2], such as optimization [3], fuzzy knowledge representation [4], image processing [5], fault diagnosis [6]. The computing units that abstract from biological cells can accomplish specific computing independently with maximum parallel manner, so the computing efficiency of this kind of systems will surpass the current electronic computer. Therefore, it possesses tremendous potentiality of application to optimization field. In recent years, some researchers have introduced P systems into optimization field. The

basic idea of these works is to combine the membrane structures and their characteristics in membrane computing with evolution operations of the existing evolutionary computations, such as tabu search [6], DNA computing [7], particle swarm optimization (PSO) [8], quantum calculation [9].

The traditional PID controller has been widely employed in industrial control system, but it was inadequate for complicated nonlinear systems, which were usually restricted by real-world environment. In order to effectively control strong coupled nonlinear multi-input and multi-output (MIMO) systems, Shu and Shu [10] proposed a novel dynamic neural network model named PID neural network (PIDNN), which was a multi-layer forward network. This kind of models, which were constructed by proportion, integral and differential neurons that were mutually linked with each other, were adequate for many different kinds of systems without measuring and identifying internal structure and parameters of the controlled plant, so they were better than other conventional controllers. Besides, this kind of models combined advantages of both PID controller and neural network, such as shot training time, good dynamic property, legible hierarchical structure. However, the BP algorithm (i.e., gradient descent algorithm) restrains the wide applications of these models due to its known shortcomings, such as weakly global searching ability, easily trapping into local optimum and the sensitivity of initial weight and learning rate. The training process of PIDNN controller can actually be regarded as an optimization problem that searches for the optimum in the solution space of weight causing minimum output error. So far, some bio-inspired algorithms and their variants have been employed to train PIDNN, such as GA [11], PSO [12]. In [12], cooperated particle swarm optimization (CPSO) was proposed to optimize the design of PIDNN controllers in order to obtain a good control performance. However, this method usually needs big population size which will increase the amount of calculation and slow down running speed.

A new algorithm combined the idea of membrane computing and PSO, called particle swarm optimization based on P systems (PSOPS) algorithm, was first presented in [13], and experiments based on seven bench function optimization problems and time-frequency atom decomposition addressed its effectiveness and optimization ability. The global searching ability of PSOPS is improved due to the introduction of membrane structure, but particles of the algorithm may show great homoplasy in the searching process and then lead the algorithm into prematurity due to the small size of populations in each elementary membrane. In order to fully make use of the parallelism feature to overcome this problem, a novel evolutionary algorithm called membrane optimization algorithm based on mutated PSO (in short, MO-MPSO) is proposed in this paper and a number of trial function experiments are done. Furthermore, we develop a MIMO PID neural network (MPIDNN) controller based on PIDNN and employ the proposed MO-MPSO instead of BP algorithm to control the strong coupled nonlinear MIMO system. Comparison results of the MPIDNN controllers based on BP, PSO, CPSO and MO-MPSO indicate that MO-MPSO-based MPIDNN controller is more effective than the other three in controlling MIMO systems.

This paper is organized as follows. Structure of PIDNN and its control system are presented in Section 2. In Section 3, membrane optimization algorithm based on mutated PSO (MO-MPSO) is described. In Section 4, performance measurements of the proposed algorithm based on benchmark functions are carried out. One case study is given in Section 5. Finally, conclusions are discussed in Section 6.

2. **Structure of PIDNN and Its Algorithm.**

The PID neural network (PIDNN) is a new dynamic neural network model proposed by Shu and Shu [10]. This kind of model

defines neurons possessed function of proportion (P), integral (I) and differential (D). So, it combines the ability of approximating arbitrary functions of neural network and dynamic characteristic of quick input-output response which especially suit for controlling the complex and nonlinear objects. PIDNN generally can be categorized into two kinds: single argument PIDNN (SPIDNN) and multivariable PIDNN (MPIDNN). The MPIDNN can be constructed by SPIDNN because their activation function of neurons in each layer, performance index and learning algorithm are about the same. So, we only introduce the structure and algorithm of SPIDNN.

2.1. **Structure of PIDNN and its control system.** The basic form of PIDNN is SPIDNN, which is a three layer structure of $2 \times 3 \times 1$ form, called input layer, hidden layer and output layer. Its network structure and single argument control system are shown in Figure 1. The input layer is constructed by two proportion neurons used to input the setting value and feedback value. The output layer is a proportion neuron used to export the control quantity of control system. The hidden layer is the hard-core part which is constructed by three neurons that are proportion neuron, integral neuron and differential neuron, used to complete proportion, integral and differential operation, respectively (i.e., deal with and convert the input data).

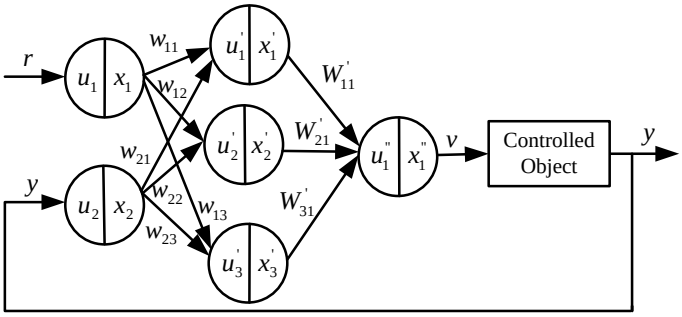

FIGURE 1. Structure drawing of PIDNN control system

2.2. **Forward-propagation algorithm of PIDNN.** The function of forward-propagation algorithm is to accomplish the network's output on the basis of input data, current weights and state function in each layer. This algorithm is composed by three parts: input layer, hidden layer and output layer.

Input layer: the input and output definitions of the two neurons in this layer are described as:

$$x_i(k) = \begin{cases} q', & u_i(k) \geq q, \\ u_i(k), & -q < u_i(k) < q, \\ -q', & u_i(k) \leq -q. \end{cases} \tag{1}$$

where $i = 1, 2$ is number mark of proportion neurons in input layer; k is sample time; q is the upper limit; q' is the real output value when q surpasses its maximum limit; $u_i(k)$ and $x_i(k)$ are input and output values of the ith neuron at time k , respectively.

Hidden layer: the input definition of the neurons is written as:

$$u_j' = \sum_{i=1}^{2} \omega_{ij} \cdot x_j(k) \tag{2}$$

where $j = 1, 2, 3$, is number mark of neurons in hidden layer; ω_{ij} is weight between input layer and hidden layer. The input and output definition of proportion, integral and

differential neurons are shown in Equations (3)-(5), respectively:

$$x'_1(k) = \begin{cases} q', & u'_1(k) \geq q, \\ u'_1(k), & -q < u'_1(k) < q, \\ -q', & u'_1(k) \leq -q; \end{cases} \tag{3}$$

$$x'_2(k) = \begin{cases} q', & u'_2(k) \geq q, \\ x'_2(k-1) + u'_2(k), & -q < u'_2(k) < q, \\ -q', & u'_2(k) \leq -q; \end{cases} \tag{4}$$

$$x'_3(k) = \begin{cases} q', & u'_3(k) \geq q, \\ u'_3(k) + u'_3(k-1), & -q < u'_3(k) < q, \\ -q', & u'_3(k) \leq -q. \end{cases} \tag{5}$$

Output layer: this layer contains only one proportion neuron which exports the sum value of the network, and its input definition is written as:

$$u''_h = \sum_{j=1}^{3} \omega_{jh} \cdot x'_j(k) \tag{6}$$

where $h = 1$, is number mark of the output neuron; ω_{jh} is weights between hidden layer and output layer. The input and output definition of proportion neuron in this layer is shown as:

$$x''_h(k) = \begin{cases} q', & u''_h(k) \geq q, \\ u''_h(k), & -q < u''_h(k) < q, \\ -q', & u''_h(k) \leq -q. \end{cases} \tag{7}$$

2.3. **Backward-propagation algorithm of PIDNN.** The function of backward-propagation (BP) algorithm is to complete process of modification, learning and memory of the weights. Its learning process is to obtain the minimum J, shown in Equation (8), which is square mean value of the time sequence error between real output and ideal output of the nets.

$$J = \frac{1}{l} \sum_{k=1}^{l} [r(k) - y(k)]^2 \tag{8}$$

where $r(k)$, $y(k)$ are input and output of the control system, respectively; l is sampling number.

After nth step, the weights modification formula of BP algorithm is shown in Equation (9):

$$\Delta\omega_{ij} = \eta \frac{dJ}{d\omega_{ij}} = \begin{cases} -\dfrac{2}{l} \displaystyle\sum_{k=1}^{l} [r(k) - y(k)] \cdot \frac{dy}{dx''} \cdot \omega_{jh}(k) \cdot 1 \cdot x_i, & P_{neuron}, \\ -\dfrac{2}{l} \displaystyle\sum_{k=1}^{l} [r(k) - y(k)] \cdot \frac{dy}{dx''} \cdot \omega_{jh}(k) \cdot u'_j \cdot x_i, & I_{neuron}, \\ -\dfrac{2}{l} \displaystyle\sum_{k=1}^{l} [r(k) - y(k)] \cdot \frac{dy}{dx''} \cdot \omega_{jh}(k) \cdot \frac{dx'_j}{du'_j} \cdot x_i, & D_{neuron}, \end{cases}$$

$$\Delta\omega_{ij} = \eta \frac{dJ}{d\omega_{ij}} = -\frac{2}{l} \sum_{k=1}^{l} [r(k) - y(k)] \cdot \frac{dy}{dx''} \cdot x'_j(k),$$

$$\omega_{ij}(n+1) = \omega_{ij}(n) - \Delta\omega_{ij}, \quad \omega_{jh}(n+1) = \omega_{jh}(n) - \Delta\omega_{jh}. \tag{9}$$

where η is learning rate; $\frac{dy}{dx''}$ and $\frac{dx'_j}{du'_j}$ are common factors in this formula and their positive and negative characters determine changing direction of the weights. Besides, because the values of $\frac{dy}{dx''}$ and $\frac{dx'_j}{du'_j}$ only have effect in changing pace of weights which can be adjusted

by learning rate, they can be replaced by $\operatorname{sgn}\left(\frac{y(k+1)-y(k)}{x''(k)-x''(k-1)}\right)$ and $\operatorname{sgn}\left(\frac{x'_j(k)-x'_j(k-1)}{u'_j(k)-u'_j(k-1)}\right)$, respectively.

3. **Membrane Computing Optimization Algorithm Based on Mutated PSO.** The traditional PSO algorithm has multifarious population in beginning period of iteration, but the particles begin to show great homoplasy and "*gathering*" phenomena along with the iterations, which cause falling down in multiformity and premature of the algorithm. The solutions of this problem are usually two ways: (1) enlarging the population size; (2) introducing mutation operator. The shortcoming of former method is not only increasing amount of calculation, but also has not solve the problem of premature fundamentally. The latter one can maintain multiformity based on small population size only through proper mutation operator which is neither influencing calculation, nor the programming and realization. Combining the idea of membrane computing and mutation operator can strengthen local searching ability of the algorithm through segmentation and maintain multiformity in each elementary membrane to conquer premature.

3.1. **Membrane computing.** Membrane computing (known as P systems) is one of the youngest branches of natural computing, firstly introduced by G. Păun. They consist of three key ingredients: membrane hierarchical structures (composed by several membranes), multisets of symbol-objects and evolution rules. The membrane structure is a hierarchical arrangement of membranes; the outermost layer is called skin membrane which separates P systems from environment; if one membrane contains no more other one, and it is called elementary membrane. Each membrane defines a region which contains a multiset of objects and a set of evolving rules, where the objects are expressed by particular symbols. So far, P systems can be categorized into three main types: i) cell-like P systems, ii) tissue-like P systems, iii) neural-like P systems. The membrane structure of a cell-like P system can be formalized as follows [2]:

$$\Pi = (O, H, \mu, \omega_1, \ldots, \omega_m, R_1, \ldots, R_m, i_0)$$

where O and H are alphabets of objects and labels of membranes, respectively; μ is a membrane structure with m membranes; $\omega_1, \ldots, \omega_m \in O^*$ are strings which represent multisets; R_i, $1 \leq i \leq m$, represents the set of evolving rules; $i_0 \in H \cup \{e\}$ represents the output membrane, where e is not a reserved symbol of H.

The main kinds of hierarchical structure of membrane optimization algorithm generally contain two ones: changeless membrane structure and dynamic changing membrane structure, where the changeless structures can be categorized into three kinds: (1) The nested type membrane structure (OLMC), shown in Figure 2(a), which contains m membranes [3]. In this kind of structure, the innermost membrane, called output region, is contained by only one elementary membrane. In each region, some revolution rules, communication rules and solution sets such as multisets are contained. In process of algorithm performance, revolution rules in each membrane are carrying out independently and the solutions are transported to adjacent region by the communication rules (the communication barely takes place between membranes that next door to each other). Usually, the best and worst solutions, with respect to the optimization criterion, are sent to the adjacent inner and outer regions simultaneously. (2) The one level membrane structure (OLMS), shown in Figure 2(b), which contains m elementary membranes and one skin membrane [13]. Each membrane contains its own evolution rules which can be the same or not. The rules in each region are working simultaneously and the best solutions of each elementary membrane are sent to the skin membrane. Then, the global search strategy is used to get the global optimum solution of this algorithm which will be returned to

each elementary region to influence the next revolution. (3) The hybrid type membrane structure, shown in Figure 2(c), which contains m elementary membranes and one skin membrane [14]. In this kind of structure, both the nested type membrane structure and the one level membrane structure are contained. When the algorithm is running, evolution and communication rules in those two types of structures execute according to the way shown as above, respectively.

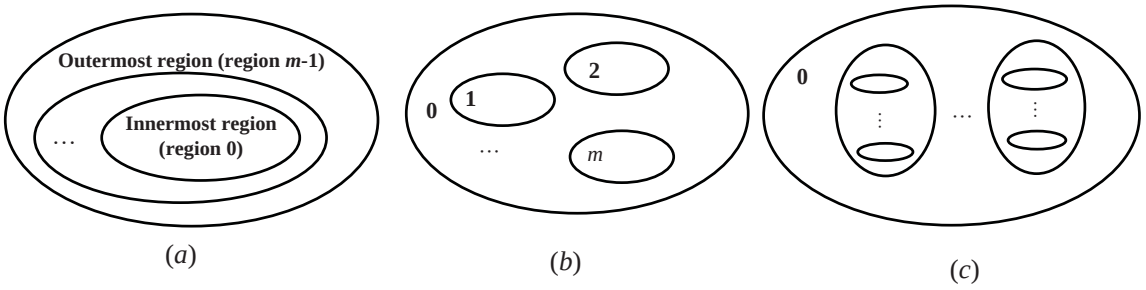

FIGURE 2. Three kinds of membrane structure

3.2. **Mutation operator.** In order to make each particle with preferable local searching ability, we introduced horizontal hybrid mutation operator into PSOPS algorithm [15]. In the evolutionary process of population, mutation operators are executed in each elementary membrane so as to improve population diversity and the particles performed mutation will search in the expanded areas where new best solutions would be discovered. From this, the algorithm will spring out local optimum and succeeded in finding the globally optimal solution.

Then main idea of this mutation operator is shown as follows: in the beginning, we decide whether a particle will mutate or not by mutant power (mc_i) which is settled according to empirical Formula (10):

$$mc_i = 0.05 + 0.45\frac{\exp\left(\frac{5(i-1)}{Popsize-1}\right) - 1}{\exp(5) - 1} \tag{10}$$

where $Popsize$ expresses the population size, i is the current particle. The pseudocode algorithm of horizontal hybrid mutation operator is shown in Figure 3(a).

The main mutagenic factors are usually divided into three kinds shown as follows [15]. (1) The changeless constant mutation probability which is a value usually chose in $[0, 1]$, such as 0, 0.1, 0.2, ..., 1. If the value of mutation probability is too large, population will get muddled while the population diversity is increased, which is sure to slow down rapidity of convergence of the algorithm. On the contrary, the algorithm will not be able to spring out local optimum rapidly and efficiently. (2) The self-adapting mutagenic factor which can produce an even distribution value chose in $[0.4, 0.7]$. (3) The decreasing mutagenic factor which general contains three kinds of function shown as follows:

$$\text{Linear function: } p_m = 1 - t/Iter \tag{11}$$

$$\text{Exponential function: } p_m = 1 - (\exp(t\log 2/Iter) - 1) \tag{12}$$

$$\text{Sigmoid function: } \begin{cases} f(r,t) = 1/(1+\exp(-rt)), \\ p_m(t) = 1 - f(t - Iter/2, r) \end{cases} \tag{13}$$

where t expresses current iteration time, $Iter$ is the total number of iterations of the algorithm.

3.3. **MO-MPSO algorithm.** We introduce the idea of membrane computing and mutation operator into PSO algorithm and propose a novel evolutionary algorithm, called membrane optimization algorithm based on mutated PSO (in short, MO-MPSO). In this algorithm, the horizontal hybrid mutation operation, OLMS and communication rules of membrane computing and evolution rules in PSO are employed. Particles expressed by x_i in the MO-MPSO algorithm are updated as the following:

$$\begin{cases} v_{id}^{(t+1)} = \omega v_{id}^{(t)} + c_1 r_1 \left(p_{id}^{(t)} - x_{id}^{(t)} \right) + c_2 r_2 \left(G_{gd}^{(t)} - x_{id}^{(t)} \right) \\ x_{id}^{(t+1)} = x_{id}^{(t)} + v_{id}^{(t+1)} \\ \omega = \omega_{\max} - (\omega_{\max} - \omega_{\min}) * (t+1)/I \end{cases} \tag{14}$$

where $i = 1, 2, \ldots, n$ express the ith particle and the population size is n; $p_{id}^{(t)}$ express personal optimum value of the ith particle in tth iteration; $G_{gd}^{(t)}$ indicate best solution of the algorithm in tth iteration; r_1, r_2 are random numbers in $[0, 1]$; the value of v_d is clamped in the range $[-v_{d\max}, v_{d\max}]$; ω is the weight coefficient decrease from $\omega_{\max}$ to $\omega_{\min}$ which make the particles maintain free fall; $[c_1, c_2]$ are acceleration factors, usually chose in $[0, 2]$; I is iteration of the algorithm.

The flowchart of MO-MPSO is shown in Figure 3(b), and the execution steps of the algorithm are shown as follows:

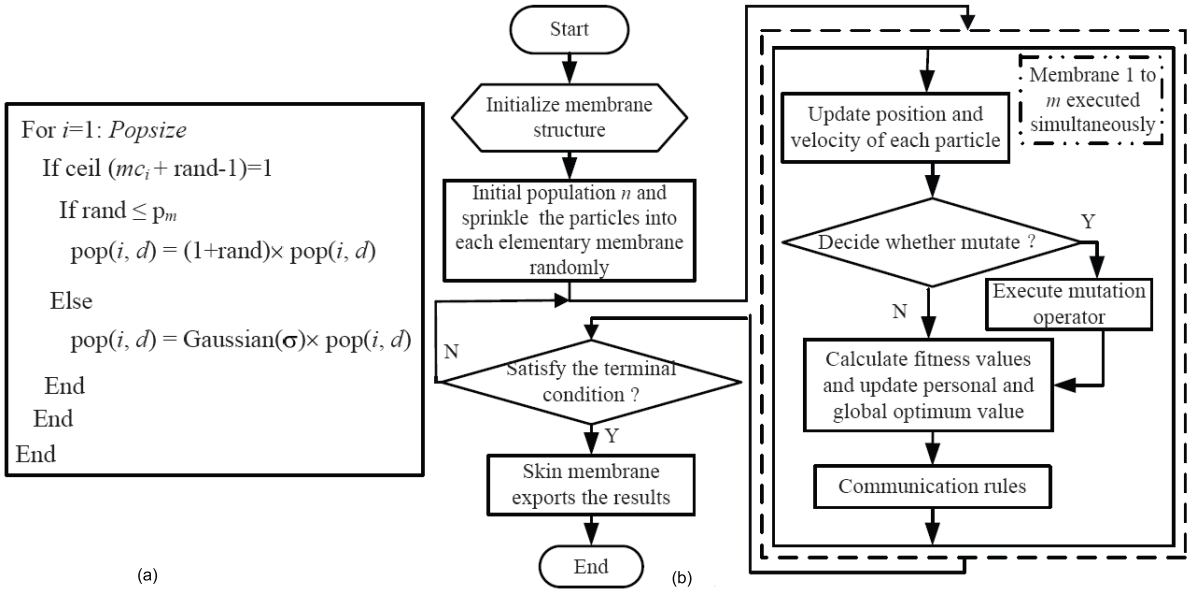

FIGURE 3. Pseudocode algorithm of a horizontal hybrid mutation operation and flow chart of MO-MPOS algorithm

Step 1: Initialize parameters, such as parameters of PSO as above, searching range of the optimization problem, terminal condition, and membrane structure $[[_0[_1]_1, [_2]_2, [_3]_3, \ldots, [_m]_m]_0]$ with elementary membranes and one skin membrane is denoted by 0;

Step 2: A particle swarm population with n particles is initialized, and each particle is assigned randomly into elementary membranes without duplication. Ensure that each elementary membrane includes one particle at least. Multisets are initialized as follows:

$\omega_0 = \lambda$,

$\omega_1 = q_1 q_2 q_3 \ldots q_{n_1}$, $n_1 < n$,

$\omega_2 = q_{n_1+1} q_{n_1+2} \ldots q_{n_2}$, $n_1 + n_2 < n$,

.....

$\omega_m = q_{n_{(m-1)}+1} q_{n_{(m-1)}+2} \ldots q_{n_m}$, $n_1 + n_2 + \ldots + n_m < n$,
where q_i $(i = 1, 2, \ldots, n)$ is an individual;

Step 3: Each elementary membrane updates position and velocity of the particles according to Formula (14) simultaneously;

Step 4: Calculate mutant power (mc_i) and decide whether a particle mutate or not, the implementation process of mutation operator is shown in Figure 3(a) shown as above;

Step 5: Calculate fitness values of the subalgorithm in each elementary membrane and update personal optimum values of each particle and best solution of the algorithm;

Step 6: Communication rules between each elementary membrane and the skin membrane send the best solution generated by the best particle of each elementary membrane into skin membrane and select the particle with best solution from the particles, and then send it back to each elementary membrane to affect the next generation update of all individuals in each region;

Step 7: If the algorithm reaches terminal condition, loop ends and the optimized result is exported by skin membrane; if not, then go to Step 3.

4. **Performance Measurement Based on Benchmark Functions.** In order to evaluate the algorithm in terms of the search capability, the performance of MO-MPSO is compared with PSO and PSOPS using test functions. The information of the chosen functions including the dimensions (expressed by D), function expression, admissible ranges of variables and optima are summarized described in Table 1. The types of these functions include unimodal (containing only one optimum) and multimodal (containing many local optima, but only one global optimum), continuous and discontinuous, or nimization and maximization.

The experiments are accomplished by programming m file in Matlab7.5. To compare the proposed MO-MPSO with other algorithms, for each benchmark function, the population

TABLE 1. Properties of the benchmark functions

Function	Search space $[x_{\min}, x_{\max}]$	Optimum point
$f_1 = 418.9829D - \sum_{i=1}^{D} x_i \sin(\sqrt{\lvert x_i \rvert})$	$[-500, 500]$	0 (min)
$f_2 = 6D + \sum_{i}^{D} \lfloor x_i \rfloor$	$[-5.12, 5.12]$	0 (min)
$f_3 = \sum_{i=1}^{D} \left\lvert \frac{\sin(10\pi x_i)}{10\pi x_i} \right\rvert$	$[-0.5, 0.5]$	0 (min)
$f_4 = \sum_{i=1}^{D} x_i^2$	$[-5.12, 5.12]$	0 (min)
$f_5 = -c_1 \times \exp\left(-c_2 \sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D} \cos(c_3 x_i)\right) + c_1 + e; \quad c_1 = 20, c_2 = 0.2, c_3 = 2\pi$	$[-32.786, 32.786]$	0 (min)
$f_6 = \sum_{i=1}^{D-1} \left(100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right)$	$[-5.12, 5.12]$	0 (min)
$f_7 = \sum_{i=1}^{D} (x_i^2 - 10\cos(2\pi x_i) + 10)$	$[-5.12, 5.12]$	0 (min)
$f_8 = \frac{1}{4000}\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D} \left(\cos\left(\frac{x_i}{\sqrt{i}}\right)\right) + 1$	$[-600, 600]$	0 (min)
$f_9 = \sum_{i=1}^{D} \lfloor x_i + 0.5 \rfloor^2$	$[-100, 100]$	0 (min)
$f_{10} = -\sum_{i=1}^{D} \left[\sin(x_i) + \sin(\frac{2x_i}{3})\right]$	$[3, 13]$	1.21598D (max)

size are set to the same 30 and three dimensions tested: 10, 50, and 100. For each dimension, best, worst and mean values are seek to evaluate the performance of these algorithms by running 30 times independently. The other parameters are shown in Table 2, where I_1, I_2 express iterations of the skin membrane and elementary membrane in PSOPS and MO-MPSO, respectively; p_m is mutation probability; m is number of the elementary membrane.

The performance measurement for the ten functions is listed in Tables 3-5. From the tables, it can be seen that searching ability of the three algorithm for functions

TABLE 2. Parameters setting of each algorithm

Algorithm			PSO	PSOPS	MO-MPSO
Weight	ω		0.6	0.6	–
	$\omega_{\max}$		–	–	0.9
	$\omega_{\min}$		–	–	0.4
Acceleration factor	c_1		2	2	2
	c_2		2	2	2
Dimension and iteration	$D = 10$	I	5000	–	–
		I_1	–	1000	1000
		I_2	–	5	5
	$D = 50$	I	5000	–	–
		I_1	–	1000	1000
		I_2	–	5	5
	$D = 100$	I	10000	–	–
		I_1	–	2000	2000
		I_2	–	5	5
other			–	$m = 16$	$p_m = 0.1$, $m = 16$

TABLE 3. Simulation results for $D = 10$

Function		f_1	f_2	f_3	f_4	f_5
PSO	Best	3825.6	10	1.0093e-04	3.2914e-06	0.0027
	Worst	3973.5	10	0.0060	0.0105	1.6467
	Mean	3884.7	10	7.6467e-04	1.4337e-03	0.4228
PSOPS	Best	3641.8	10	1.8024e-04	1.8314e-11	1.2095e-05
	Worst	3831.1	10	0.0014	1.5726e-08	1.1551e-03
	Mean	3743.9	10	5.1512e-04	1.4650e-09	1.5041e-04
MO-MPSO	Best	3641.8	10	1.2376e-04	2.0155e-11	3.2797e-06
	Worst	3831.1	10	7.6107e-04	2.3214e-09	7.9547e-05
	Mean	3732.2	10	4.6661e-04	3.3512e-10	2.0713e-05
Function		f_6	f_7	f_8	f_9	f_{10}
PSO	Best	4.000	4.9809	0.0025	0	12.1598
	Worst	9.6703	24.8741	0.3890	0	9.8934
	Mean	8.0550	11.5543	0.0179	0	12.1598
PSOPS	Best	9.0070e-06	1.9950	0.0025	0	12.1598
	Worst	0.0106	17.9092	0.0459	0	12.1598
	Mean	4.9411e-04	6.7008	0.0067	0	12.1598
MO-MPSO	Best	3.7471e-07	0.9950	0.0025	0	12.1598
	Worst	2.1448e-04	10.9546	0.0049	0	12.1598
	Mean	2.6888e-05	5.9138	0.0030	0	12.1598

TABLE 4. Simulation results for $D = 50$

Function		f_1	f_2	f_3	f_4	f_5
PSO	Best	18747	60	0.0042	3.7984	2.8569
	Worst	19675	183	0.2300	11.7916	4.7638
	Mean	19121	122.0667	0.1235	8.2883	3.6890
PSOPS	Best	18646	50	0.0023	4.5240e-04	0.0099
	Worst	19371	50	0.1670	0.0490	0.5586
	Mean	18915	50	0.0317	0.0074	0.0509
MO-MPSO	Best	7996.5	50	0.0083	1.4914e-04	6.1728e-04
	Worst	17030	50	0.0103	4.8303e-04	0.0140
	Mean	10574	50	0.0287	3.1147e-04	7.4122e-03
Function		f_6	f_7	f_8	f_9	f_{10}
PSO	Best	85.2974	199.8787	0.2006	0	59.5073
	Worst	306.3575	301.1568	0.4393	2	52.3043
	Mean	172.7819	241.7006	0.2687	0.8667	56.2628
PSOPS	Best	45.6843	35.8854	0.0026	0	60.7988
	Worst	102.9447	113.5624	0.0101	0	51.7334
	Mean	53.8111	65.7832	0.0036	0	58.5163
MO-MPSO	Best	47.0533	13.9560	0.0025	0	60.7990
	Worst	49.0155	70.8069	0.0026	0	51.7333
	Mean	48.0190	43.1200	0.0025	0	58.6031

TABLE 5. Simulation results for $D = 100$

Function		f_1	f_2	f_3	f_4	f_5
PSO	Best	37274	236	0.0184	15.9770	4.0138
	Worst	41104	424	0.6443	52.6535	5.5322
	Mean	38490	354	0.2975	32.6921	4.4212
PSOPS	Best	15998	100	0.0343	0.0025	0.0153
	Worst	36066	100	0.2971	0.0088	0.0292
	Mean	22133	100	0.1063	0.0048	0.0224
MO-MPSO	Best	16654	100	0.0210	0.0012	0.0013
	Worst	21973	100	0.1623	0.0014	0.0018
	Mean	19887	100	0.0777	0.0014	0.0016
Function		f_6	f_7	f_8	f_9	f_{10}
PSO	Best	308.0915	587.9143	0.2884	0	115.9488
	Worst	769.0072	734.5523	0.7627	6	100.1787
	Mean	482.7527	640.0129	0.4808	2.0667	107.0654
PSOPS	Best	95.5706	90.0850	0.0065	0	119.263
	Worst	98.4515	180.9966	0.3927	0	101.1851
	Mean	96.6156	123.1359	0.0218	0	111.0966
MO-MPSO	Best	94.6141	34.9797	0.0026	0	121.5957
	Worst	96.6574	129.5744	0.0053	0	103.4665
	Mean	95.3805	89.19254	0.0028	0	115.2947

$f_1, f_2, f_3, f_7, f_8, f_9$ and f_{10} is shown about the same when $D = 10$, but for f_4, f_5 and f_6 the results of PSO are worse than that of both PSOPS and MO-MPSO. However, the superior performance of membrane computing optimization algorithm is gradually showing while the dimension increases that the results of both PSOPS and MO-MPSO are much better

than that of PSO, especially when $D = 100$. The degree of accuracy of optimum solutions get through MO-MPSO is usually higher than that of PSOPS due to the fact that the number n ($1 \leq n \leq 30 - m + 1$, $m = 16$ in this experiment) of particles in each elementary membrane is too small which will cause prematurity. Thanks to the mutation mechanism and membrane structure, the proposed MO-MPSO will maintain multiformity in each elementary membrane as well as balance the global and local optimization ability in search procedure which makes the searching ability, degree of optimal accuracy and stability of MO-MPSO better than PSOPS.

5. **Case Study.** In order to control the nonlinear MIMO system efficiently and to extend the application scope of PIDNN, we develop the MIMO PIDNN controller based on PIDNN, and employ the proposed MO-MPSO to take the place of traditional BP algorithm, which is marked as MPIDNN-MO-MPSO controller. Simulation results of the MPIDNN controller based on BP, PSO, CPSO and MO-MPSO algorithms are taken to analysis and comparison.

5.1. **MPIDNN control system.** In this experiment, a typical three input three output complex nonlinear and close coupling control system is employed and its transfer function is shown in Formula (15) [16]. To realize the decoupling control of this MIMO system, MPIDNN control system based on PIDNN is constructed shown in Figure 4. Compared with SPIDNN control system, the output layer of MPIDNN control system is codetermined by outputs of the elementary SPIDNN.

$$\left.\begin{aligned} y_1(k) &= 0.4 * y_1(k-1) + u_1(k-1)/[1 + u_1(k-1)^2] + 0.2 * u_1(k-1)^3 \\ &\quad + 0.5 * u_2(k-1) + 0.3 * y_2(k-1) \\ y_2(k) &= 0.2 * y_2(k-1) + u_2(k-1)/[1 + u_2(k-1)^2] + 0.4 * u_2(k-1)^3 \\ &\quad + 0.2 * u_1(k-1) + 0.3 * y_3(k-1) \\ y_3(k) &= 0.3 * y_3(k-1) + u_3(k-1)/[1 + u_3(k-1)^2] + 0.4 * u_3(k-1)^3 \\ &\quad + 0.4 * u_2(k-1) + 0.3 * y_1(k-1) \end{aligned}\right\} \tag{15}$$

5.2. **Simulation result and analysis.** In Figure 4, r_1, r_2, r_3 express control targets of the control quantity, v_1, v_2, v_3 express control laws of the controller, y_1, y_2, y_3 express current value of the control quantity. In this experiment, the weights initialize randomly, the initial values of control quantity take [0 0 0], the control targets are set as [0.7 0.4 0.6], space of control time is set as 0.001s, the number of SPIDNN is 2, the number of output neurons is 3, the learning rate of weights is set as 0.05, the upper limit of input value is $p = 1$, the real output value which is the value q' when q surpasses its maximum limit is set as -1.

The parameters of the employed algorithms are set as follows (except BP algorithm): population size is set as 30, the number of elementary membranes is set as 8 (i.e., $m = 8$), the dimension is set as 45, the number of sub-swarm of CPSO is set as 3 and each sub-swarm contains 10 particles, the maximum number of generations of PSO is set as 100, the maximum number of skin membrane and each elementary membrane are all set as 20 and 5, respectively; the other parameters are set the same as shown in Table 2. Simulation curves of the control strategy based on BP, PSO, CPSO and MO-MPSO algorithms in terms of the given object model are shown in Figure 5 to Figure 8, and the error curves are shown in Figure 9.

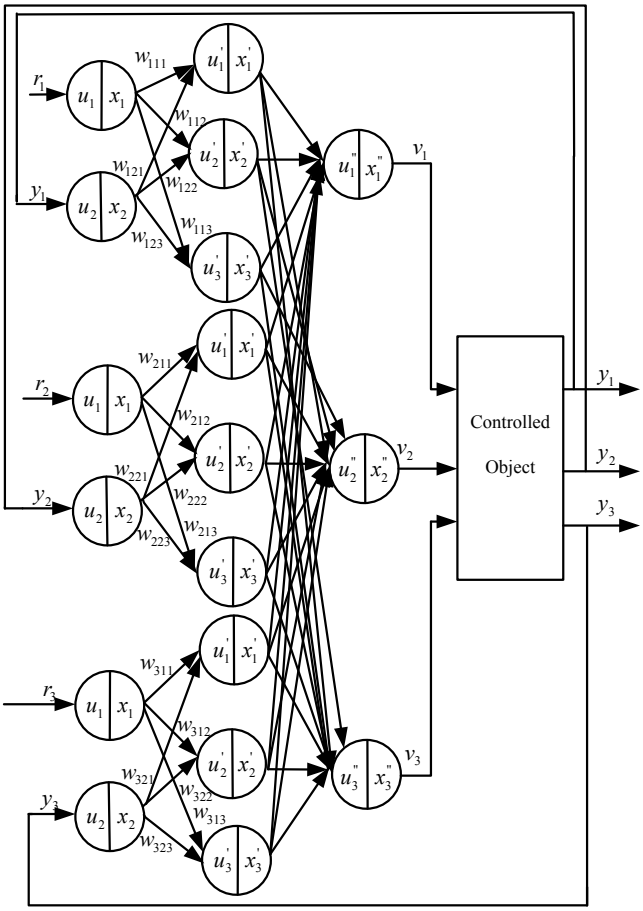

FIGURE 4. Structure drawing of MPIDNN control system based on the controlled object

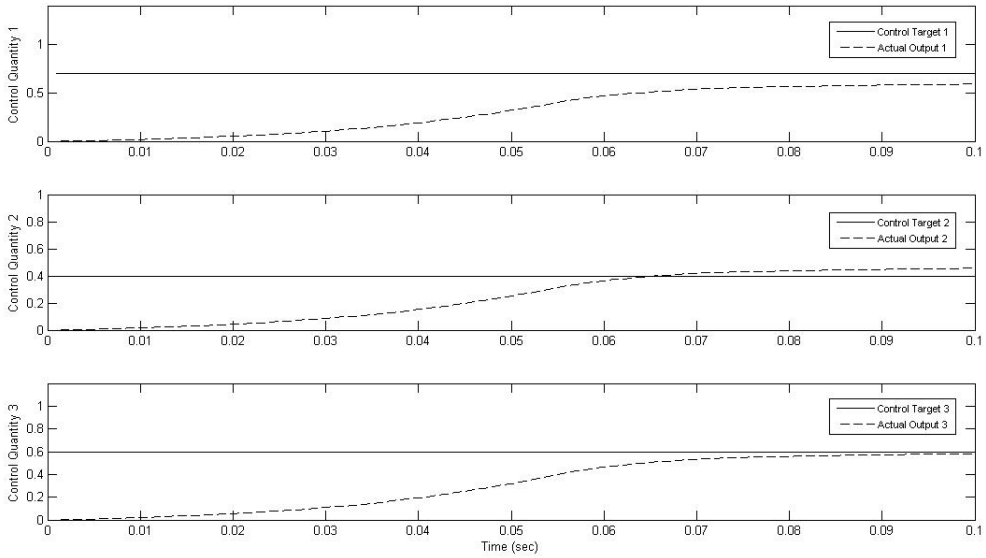

FIGURE 5. Simulation curves of the control strategy based on BP algorithm

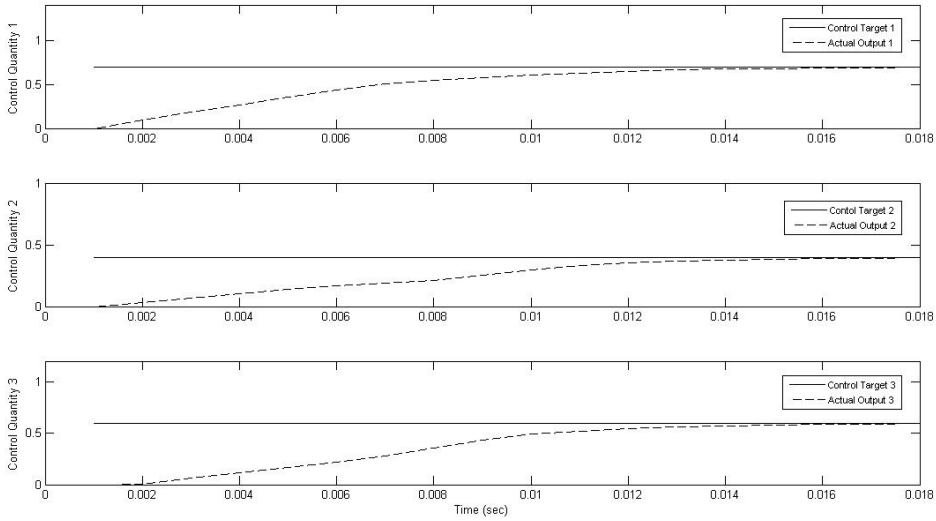

FIGURE 6. Simulation curves of the control strategy based on PSO algorithm

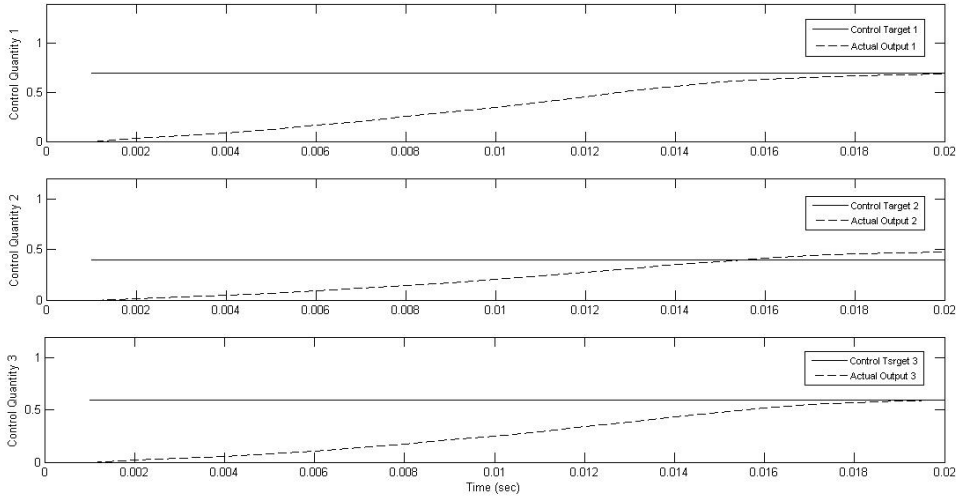

FIGURE 7. Simulation curves of the control strategy based on CPSO algorithm

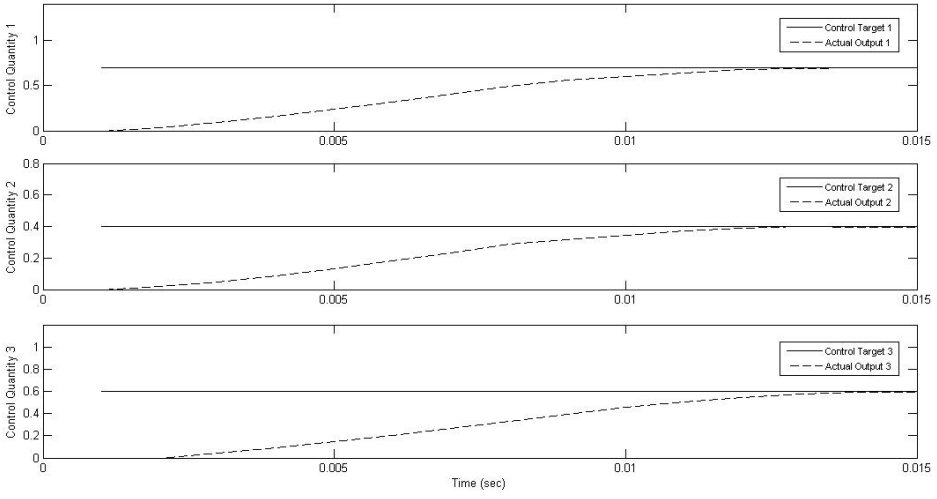

FIGURE 8. Simulation curves of the control strategy based on MO-MPSO algorithm

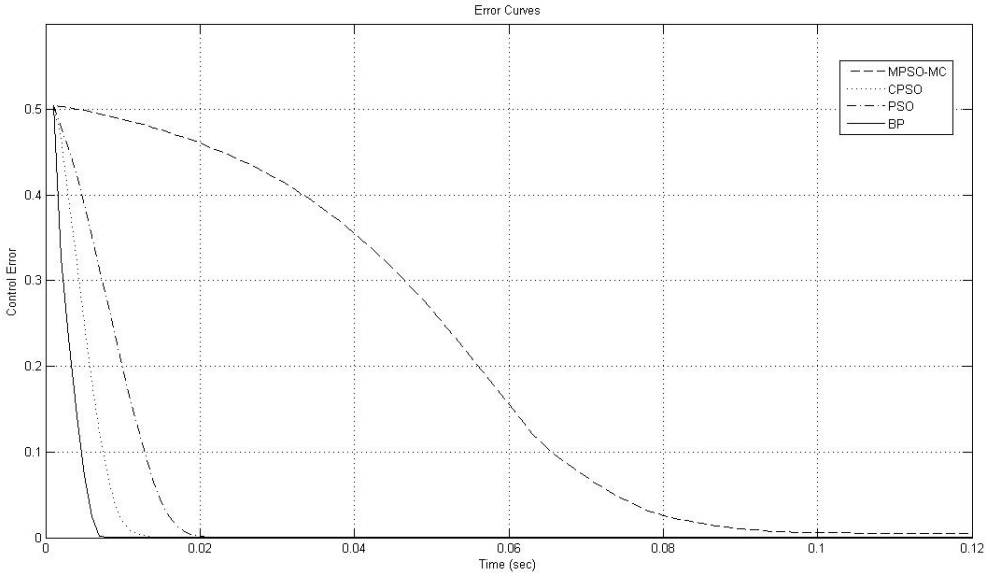

FIGURE 9. Error curves of the four methods

From Figures 5-9, it can be seen that the MPIDNN control system can success execute decoupling control based on the four adoptive methods, but the control effects are different, and the results based on BP and CPSO algorithms are worse than that got from the PSO and MO-MPSO systems. The causes lie in the fact that the global searching ability of BP algorithm is weak which leads it easily to fall into local optimum; in spite of the fact that the CPSO algorithm performs well in [17] (where the population size was set as 90), it is unsuccessful in this experiment due to the fact that the population size is set as 30 which shows that CPSO algorithm cannot maintain its favorable global searching ability in small population size, even worse than PSO; the searching ability, degree of accuracy of optima and stability of MO-MPSO are strengthened powerfully due to the introduction of membrane structure and the mutation operator. Therefore, in the case of the same small population size 30, the proposed MO-MPSO algorithm converges to the minimum error amount with the fastest velocity of convergence which improves the control accuracy and tracking velocity of the MPIDNN control system to reach the optimal control effect.

6. **Conclusions.** This paper proposes a new kind of algorithm called MO-MPSO to take the place of traditional BP algorithm in MPIDNN. Simulation results of the MPIDNN controller based on BP, PSO, CPSO and MO-MPSO algorithms indicate that MO-MPSO-based MPIDNN controller is more effective than the other three in controlling the MIMO nonlinear systems. The MO-MPSO algorithm makes MPIDNN controller better in performances than BP algorithm in optimization accuracy, stability and robustness. Furthermore, our future work may focus on more experiments and a systematic analysis of the introduced algorithm and its applications in electric power system.

Acknowledgment. This work was partially supported by the National Natural Science Foundation of China (Grant No. 61170030), the Open Research Fund of Key Laboratory of High Performance Scientific Computing, Xihua University (No. SZJJ2012-002), and Research Fund of Sichuan Provincial Key Discipline of Power Electronics and Electric Drive, Xihua University (No. SZD0503-09-0).

REFERENCES

[1] G. Păun, Computing with membranes, *Journal of Computer System Sciences*, vol.61, no.1, pp.108-143, 2000.

[2] G. Păun, G. Rozenberg and A. Salomaa, *Handbook of Membrane Computing*, Oxford University Press, Oxford, 2009.

[3] T. Y. Nishida, An approximate algorithm for NP-complete optimization problems exploiting P systems, *Proc. of Brainstorming Workshop on Uncertainty in Membrane Computing*, Palma de Majorca, pp.185-192, 2004.

[4] J. Wang, P. Shi, H. Peng, M. J. Pérez-Jiménez and T. Wang, Weighted fuzzy spiking neural P systems, *IEEE Transactions on Fuzzy Systems*, http://dx.doi.org/10.1109/TFUZZ.2012.2208974, 2012.

[5] H. Peng, J. Wang, M. J. Pérez-Jiménez and P. Shi, A novel image thresholding method based on membrane computing and fuzzy entropy, *Journal of Intelligent & Fuzzy Systems*, http://dx.doi.org/10.3233/IFS-2012-0549, 2012.

[6] H. Peng, J. Wang, H. Wang, J. Shao and T. Wang, Fuzzy reasoning spiking neural P system for fault diagnosis, *Information Sciences*, http://dx.doi.org/10.1016/j.ins.2012.07.015, 2012.

[7] L. Huang and I. H. Suh, Controller design for a marine diesel engine using membrane computing, *International Journal of Innovative Computing, Information and Control*, vol.5, no.4, pp.899-912, 2009.

[8] T. Wang, J. Wang, H. Peng and M. Tu, Optimization of PID controller parameters based on PSOPS algorithm, *ICIC Express Letters*, vol.6, no.1, pp.273-280, 2012.

[9] G. X. Zhang, M. Gheorghe and C. Z. Wu, A quantum-inspired evolutionary algorithm based in P systems for knapsack problem, *Fundamenta Informaticae*, vol.87, no.1, pp.93-116, 2008.

[10] H. L. Shu and H. Shu, Simulation study of PID neural network temperature control system in plastic injecting-moulding machine, *Proc. of the 6th International Conference on Machine Learning and Cybernetics*, Hong Kong, China, pp.492-497, 2007.

[11] J.-S. Kim, J.-H. Kim, J.-M. Park, S.-M. Park et al., Auto tuning PID controller based on improved genetic algorithm for reverse osmosis plant, *World Academy of Science, Engineering and Technology*, vol.47, pp.384-389, 2008.

[12] H. G. Piao, Z. X. Wang and H. Q. Zhang, Nonlinear control system of PID neural network based on cooperated particle swarm optimization (PSO), *Control Theory & Applications*, vol.26, no.12, pp.1317-1324, 2009.

[13] G. X. Zhang, J. X. Cheng and M. Cheorghe, An membrane-inspired approximate algorithm for traveling salesman problem, *Romanian Journal of Information Science and Technology*, vol.14, no.1, pp.3-19, 2011.

[14] L. Huang, N. Wang and J. H. Zhao, Multiobjective optimization for controller design, *Acta Automatica Sinica*, vol.34, no.4, pp.472-477, 2008.

[15] Y. M. Liu, Q. Z. Zhao, C. L. Sui and Z. Z. Shao, Particle swarm optimizer based on dynamic neighbourhood topology and mutation operator, *Control and Decision*, vol.25, no.7, pp.968-974, 2010.

[16] F. Shi, X. C. Wang, L. Yu and Y. Li, *Analysis of 30th Neural Network Cases Based on MATLAB*, Beijing University of Aeronautics and Astronautics Press, Beijing, 2010.

[17] S.-Y. Ho, H.-S. Lin, W.-H. Liauh and S.-J. Ho, OPSO: Orthogonal particle swarm optimization and its application to task assignment problems, *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol.38, no.2, pp.288-298, 2008.