Available online at www.sciencedirect.com
SCIENCE DIRECT®
Chaos, Solitons and Fractals 30 (2006) 94–109
CHAOS SOLITONS & FRACTALS
www.elsevier.com/locate/chaos

A chaos search immune algorithm with its application to neuro-fuzzy controller design

X.Q. Zuo *, Y.S. Fan

Department of Automation, Tsinghua University, Beijing 100084, China

Accepted 24 August 2005

Abstract

In this paper, a chaos search immune algorithm (CSIA) is proposed by integrating the chaos optimization algorithm and the clonal selection algorithm. First, optimization variables are expressed by chaotic variables through solution space transformation. Then, taking advantages of the ergodic and stochastic properties of chaotic variables, a chaos search is performed in the neighbourhoods of high affinity antibodies to exploit local solution space, and the motion of the chaotic variables in their ergodic space is used to explore the whole solution space. Furthermore, a generalized radial basis function neuro-fuzzy controller (GRBFNFC) is constructed and designed automatically by the proposed CSIA. Application of the CSIA-designed GRBFNFC to real-time control of an inverted pendulum system is discussed. Experimental results demonstrate that the designed GRBFNFC has very satisfactory performance.
© 2005 Elsevier Ltd. All rights reserved.

1. Introduction

Over the last few years, artificial immune systems (AISs) have become an active research area [1]. AISs use ideas gleaned from immunology to develop intelligent systems capable of learning and adapting, and have been widely applied to the areas of computer security [2], failure diagnosis [3], abnormal detection [4], pattern recognition [5] and machine learning [6]. Moreover, optimization computation is also a popular research area of AISs. Scholars have proposed several immune algorithms for solving engineering optimization problems. Mori et al. [7] proposed an immune algorithm that improved the global search performance of a genetic algorithm (GA) by using the immune network theory. Chun et al. [8] further developed Mori's algorithm and applied it to the shape optimization of electromagnetic devices. Jiao and Wang [9] proposed an immune algorithm that used an immune operator to accelerate the convergence speed of a GA. De Castro and Von Zuben [10] developed a clonal selection algorithm (CLONALG) for optimization computation and pattern recognition. Zuo and Li [11] proposed an immune algorithm for solving function optimization problems, and further applied it to the optimal design of an fuzzy controller [12].

Chaos is a universal phenomenon of nonlinear dynamic systems. Since Lorenz [13] found the canonical chaotic attractor in 1963, chaos has been extensively studied within the engineering and mathematics communities [14]. Chaos is apparently an irregular motion, seemingly unpredictable random behavior exhibited by a deterministic nonlinear

* Corresponding author. Tel.: +86 10 6278 9636x1059.
E-mail address: zuoxq@tsinghua.edu.cn (X.Q. Zuo).

0960-0779/$ - see front matter © 2005 Elsevier Ltd. All rights reserved.
doi:10.1016/j.chaos.2005.08.126

system under deterministic conditions. Chaotic variables can go through every state in a certain area according to their own regularity without repetition. Due to the ergodic and dynamic properties of chaos variables, chaos search is more capable of hill-climbing and escaping from local optima than random search [15], and thus has been applied to the area of optimization computation. Li and Jiang [16] proposed a chaos optimization algorithm (COA) to solve complex optimization problems. Lu et al. [17] used the COA to seek the simplex control vectors for nonlinear sliding mode control systems. Hasegawa et al. [18] proposed a chaos search method including both effects of chaotic dynamics and tabu search. Ji and Tang [19] developed a chaos-based simulated annealing algorithm by integrating a chaos system and simulated annealing.

In this paper, a novel chaos search immune algorithm (CSIA) is proposed by integrating chaos optimization algorithm and clonal selection algorithm, The basic ideas of the algorithm are as follows. First, the solution space of the optimization problem is mapped to the ergodic space of chaotic variables; and thus optimization variables are represented by chaotic variables which are coded into an antibody. Then, taking full advantages of the ergodic and stochastic properties of chaotic variables, a chaos search is performed in the neighbourhoods of high affinity antibodies to exploit the local solution space, and the motion of chaotic variables in their ergodic space is used to explore the whole solution space. The numerical simulation results demonstrate the effectiveness of the proposed algorithm. Furthermore, the CSIA is used to design a generalized RBF neuro-fuzzy controller. The structure of the controller is based on an RBF neural network with generalized Gaussian basis functions. The designed GRBFNFC is used to control an actual inverted pendulum system, and experimental results demonstrate that the designed GRBFNFC has very good performances.

2. Chaos search immune algorithm

The immune system protects our body against bacteria and viruses etc, and can adapt or ''learn'' to recognize various antigens, even those never encountered before. The clonal selection theory reveals this learning principle of immune system. The immune system consists of a large number of lymphocytes, and there are mainly two kinds of lymphocytes: B cells and T cells. Each B cell has one special kind of receptors on its surface. Both the antigens and the receptors of B cells have sophisticated three-dimensional structures. When the structures of an antigen and a B cell receptor are complementary, the antigen is recognized by the B cell, and the more complementary the structures are, the higher the affinity between the antigen and the B cell is. When an antigen invades a body, some B cells recognize the antigen. Under the second signal generated by T_H cells, these B cells are activated to proliferate. The receptors of the proliferated B cells mutate with high frequency, which is known as somatic hypermutation. Somatic hypermutation leads to an increase in the affinity of B cells and generates B cells with a higher affinity for the antigen. The generated higher affinity B cells are also subject to the process of proliferation and somatic hypermutation. After several generations, B cells with the highest affinity for the antigen are generated, which is called affinity maturation. The highest affinity B cells differentiate into plasma cells that produce a large number of high affinity antibodies to eliminate the antigen.

The clonal selection principle described above reveals that high affinity antibodies are generated by cumulative selection and blind variation of B cells, which can be interpreted as a microcosm of Darwin's theory of evolution. Based on the evolutionary principle in the immune system, De Castro and Von Zuben [10] proposed a clonal selection algorithm. The steps of the algorithm can be described as follows: (1) randomly choose an antibody repertoire, and calculate the affinity of each antibody in the repertoire; (2) select a few highest affinity antibodies from the repertoire; (3) the selected antibodies are cloned and the repertoire of clones is submitted to an mutation process; (4) replace some low affinity antibodies in the mutated clones by random antibodies. In the algorithm, the antibody clone and mutation are employed to perform greed search and the global search is carried out by replacing low affinity antibodies. The algorithm can solve combination optimization and pattern recognition problems effectively; however, when the algorithm is applied to continuous parameter optimization problems, there is hamming cliff phenomenon in the greed search process and the global search efficiency is low, so that the performances of the algorithm is not nice.

The reason of existing hamming cliff phenomenon is that local search is realized by the greed search mechanism but the antibody is encoded as a binary code. Hence, in the process of greed search, a high affinity antibody mutating slightly may generate a low affinity antibody that is far away form the high affinity antibody in the solution space, so the search efficiency is reduced. Aiming at this problem, we construct a neighbourhood for each high affinity antibody and the antibody is expressed by real code that is more fitness for neighbourhood search, then neighbourhood search is carried out around the high affinity antibodies to realize the greed search mechanism, so that the hamming cliff phenomenon is avoided and the search efficiency is improved.

Chaos is a general phenomenon that exists in nature. Recent study revealed that chaos also exists in the process of immune response [20]. In this paper, taking advantages of the ergodic and dynamic properties of chaos system, we

introduce the chaotic search mechanism of chaos optimization algorithm into the CLONALG to improve its search efficiency. Firstly, the neighbourhood search is realized by the second carrier wave of chaos optimization algorithm, i.e., the movement of chaotic variables in antibody neighbourhoods is used to exploit local solution space. Secondly, aiming at the low global search efficiency of CLONALG, the first carrier wave of chaos optimization algorithm is introduced to explore global solution space to avoid sticking at local optima.

The chaos system used in this paper is the well-known Logistic mapping defined by

$$z^{j+1} = \mu z^j(1 - z^j), \quad z^j \in [0,1], \ \ j = 1,2,\ldots, \tag{1}$$

where z^j is the value of the variable z at the jth iteration, and μ is a chaotic attractor. If $\mu \in (3.56, 4.0)$, then the above system enters into a chaos state and the chaotic variable z is produced. The chaos system has some special characteristics such as ergodicity, randomicity and extreme sensitivity to the initial conditions.

The proposed CSIA is used to solve the following continuous parameter optimization problem:

$$(\mathrm{P}) \quad \begin{cases} \max & f(X_1,\ldots,X_r), \\ \text{s.t.} & X_i \in [a_i, b_i], \quad i = 1,2\ldots,r, \end{cases} \tag{2}$$

where $[a_i, b_i] \subset R$; f is a real-valued continuous function; r is the number of optimization variables.

In the CSIA, a feasible solution of optimization problem is expressed by an antibody and the evaluation value of the solution corresponds to the affinity of the antibody. Detailed operations of the CSIA are developed as follows.

2.1. Initialization

The r chaotic variables are generated by the following Logistic mappings:

$$z_i^{j+1} = \mu_i z_i^j(1 - z_i^j), \quad i = 1,2,\ldots,r, \ \ j = 1,2,\ldots, \tag{3}$$

where i is the serial number of chaotic variables, and $\mu_i = 4$. Let $j = 0$, and given the r chaotic variables different initial values z_i^0 $(i = 1,2,\ldots,r)$, then the values of the r chaotic variables z_i^1 $(i = 1,2,\ldots,r)$ are produced by the Logistic equation and encoded into a real-coded antibody. Let $j = 1,2,\ldots,N-1$, and then other $N-1$ antibodies are produced by the same method.

2.2. Solution space transformation

Each antibody consists of r chaotic variables, and each chaotic variable corresponds to an optimization variable. The range of chaotic variables is $[0,1]$, and maybe different from that of optimization variables, so the mapping relationship between chaotic variables and optimization variables must be determined. The ergodic space of the chaos system (3) is mapped to the solution space of the optimization problem (P) by Eq. (4), and thus the r optimization variables can be expressed by the r chaotic variables.

$$X_i = a_i + (b_i - a_i)x_i, \quad i = 1,2,\ldots,r, \tag{4}$$

where X_i is the ith optimization variable of the problem (P), and x_i is the ith chaotic variable of an antibody. Each antibody is decoded by using Eq. (4) to obtain the r optimization variables. Then the continuous function f of the problem (P) is viewed as the objective function, and the antibody affinity can be calculated.

2.3. Structure of antibody neighbourhoods

The n highest affinity antibodies are selected $(n < N)$ and sorted in descending order according to their affinities, i.e., $Ab_1,\ldots,Ab_n$. In order to perform neighbourhood search around these antibodies, neighbourhoods must be constructed. We construct a neighbourhood for each selected antibody, and the neighbourhood structure of the kth antibody Ab_k $(1 \leqslant k \leqslant n)$ is as follows:

$$N(Ab_k) = \left\{C_y \middle| \|C_y - Ab_k\| \leqslant \alpha_k, \ \ C_y \in [0,1]^r\right\}, \tag{5}$$

where $\|\bullet\|$ is Euclidean norm and α_k is a real-valued constant. $N(Ab_k)$ is composed of all the antibodies whose Euclidean distance from Ab_k is no greater than the constant α_k, and is a sphere with center Ab_k and radius α_k. The size of the neighbourhood is determined by α_k which is determined by

$$\alpha_k = \exp\left(-\frac{\rho(n-k)}{n}\right), \tag{6}$$

where n is the number of the selected antibodies, and $\rho > 0$ is the control factor determining the neighbourhood size. Eq. (6) assures that an antibody with high affinity has a neighbourhood with small size. That means for a high (low) affinity antibody, a small (large) step length is used to search subtly (roughly) around the antibody.

2.4. Chaotic search in antibody neighbourhoods

Each antibody selected has a neighbourhood determined by Eqs. (5) and (6). The chaos sequences produced by the chaos system are used to carry out chaotic search in the antibody neighbourhood to generate new antibodies. The higher the affinity of an antibody is, the more chaotic iterations are produced in the antibody neighbourhood, i.e., the more new antibodies are generated. The number of chaotic iterations in the neighbourhood of the antibody Ab_k $(1 \leqslant k \leqslant n)$ can be calculated by

$$M_k = \text{round}\left(\frac{\beta \cdot N}{k}\right), \tag{7}$$

where round($\cdot$) denotes the operator that rounds its argument toward the closest integer, and β is the multiplying factor determining the number of chaotic iterations.

If the number of optimization variables is small, then the antibody neighbourhoods can be determined by Eqs. (5) and (6). But for high dimensional functions with many optimization variables, it is difficult to calculate antibody neighbourhoods by these two equations. Hence, we use a hypercube to approximate the sphere neighbourhood of an antibody in this paper to determine antibody neighbourhoods conveniently. The chaotic search in the hypercube neighbourhood of an antibody can be described as follows:

Suppose the kth antibody Ab_k is expressed by a vector $\mathbf{x} = (x_1, x_2, \ldots, x_r)^{\mathrm{T}}$, then the new antibody $\mathbf{x}' = (x'_1, x'_2, \ldots, x'_r)^{\mathrm{T}}$ generated by chaotic search in the neighbourhood of $\mathbf{x}$ is obtained by

$$\mathbf{x}' = \mathbf{x} + \alpha_k(2\mathbf{z}^{j+1} - 1), \tag{8}$$

where $\mathbf{z}^{j+1} = (z_1^{j+1}, z_2^{j+1}, \ldots, z_r^{j+1})^{\mathrm{T}}$ is a chaotic vector, and z_i^{j+1} $(i = 1, 2, \ldots, r)$ is the value of the ith chaotic variable at the jth iteration. Since the ergodic range of z_i^{j+1} is $[0, 1]$, the ergodic space of the vector $\mathbf{z}^{j+1}$ is $[0, 1]^r$. From Eq. (8), it is clear that $\mathbf{x}'$ is also a chaotic vector whose ergodic space is a r-dimensional hypercube, i.e., $\prod_{i=1}^{r}[x_i - \alpha_k, x_i + \alpha_k]$. The vector $\boldsymbol{x}'$ is restricted within $[0, 1]^r$ by the following equation:

$$x'_i = \begin{cases} 0, & x'_i < 0, \\ x'_i, & 0 \leqslant x'_i \leqslant 1, \\ 1, & x'_i > 1. \end{cases} \tag{9}$$

When the chaotic vector $\mathbf{z}^{j+1}$ moves in its ergodic space through chaotic iterations of chaos system (3), $\mathbf{x}'$ moves in the neighbourhood of $\mathbf{x}$ to perform chaotic search.

2.5. Chaotic search in whole solution space

The affinities of the new antibodies generated by chaotic search are calculated and the n highest affinity antibodies are selected. Chaos sequences are produced by the chaos system (3) to compose new antibodies, and d new antibodies are generated to replace the lowest affinity antibodies in the selected antibodies. This process corresponds to the first carrier wave of chaos optimization algorithm, and is used to carry out chaotic search in the whole solution space to avoid sticking at local optima.

2.6. Calculation strategy of the CSIA

The steps of the CSIA can be described as follows:

Step 1: N antibodies are generated by the chaos system (3).
Step 2: The affinity of each antibody is calculated through solution space transformation.
Step 3: The n highest affinity antibodies are selected.
Step 4: Construct a neighbourhood for each selected antibody.
Step 5: Chaotic search is performed in the antibody neighbourhoods to generate new antibodies.

Step 6: Chaotic search is performed in the whole solution space by replacing some low affinity antibodies.
Step 7: Elitist strategy: If the affinity of the best antibody in the current generation is lower than the previous generation, then the lowest affinity antibody in the current generation is replaced by the highest affinity antibody in the previous generation.
Step 8: Return to step 3, until the stopping criterion is fulfilled.

3. Characteristics of the CSIA

In the CSIA, each antibody denotes a point in $[0,1]^r$. Parallel chaotic search is performed in the neighbourhoods of multi-antibodies. The higher the affinity of an antibody is, the smaller the size of the antibody neighbourhood is and the more chaotic iterations are performed in the neighbourhood (see Fig. 1). In Fig. 1, the high affinity antibody Ab_1 has a small neighbourhood and the low affinity antibody Ab_2 has a large neighbourhood. The lowest affinity antibody Ab_3 is eliminated and replaced by a new antibody generated by chaos system to carry out chaotic search in the whole solution space. The small neighbourhood search is used to perform local search in the high affinity areas of the solution space while the large neighbourhood search is employed to search in the whole solution space to find high affinity areas.

The CLONALG has nice learning capability but its global and local search capabilities are not nice. The COA uses the chaotic variables with abundant dynamic to search, and thus has nice local search capability and its global search capability is also nice in the condition that the solution space is not very large. However, the serial search mechanism of the COA makes it have low search efficiency, and the COA cannot utilize the experience information obtained before in the search process. The CSIA employs the evolutionary model of the CLONALG and the chaotic optimization mechanism of COA to integrate the advantages of the two algorithms, and has following merits:

(1) Taking advantages of the parallel search mechanism of the CLONALG, the single-point serial chaotic search of the COA is extended to multi-point parallel chaotic search to improve the optimization efficiency of chaotic search.
(2) By using the learning capability of the CLONALG, the experience information obtained before can be utilized by the chaotic search to improve its search efficiency and avoid the weakness that COA is not effective in the condition of large solution space.
(3) By using the characteristics of ergodicity and dynamic of chaos variables, the chaotic optimization mechanism of COA is introduced into CLONALG to improve its local and global search capabilities.

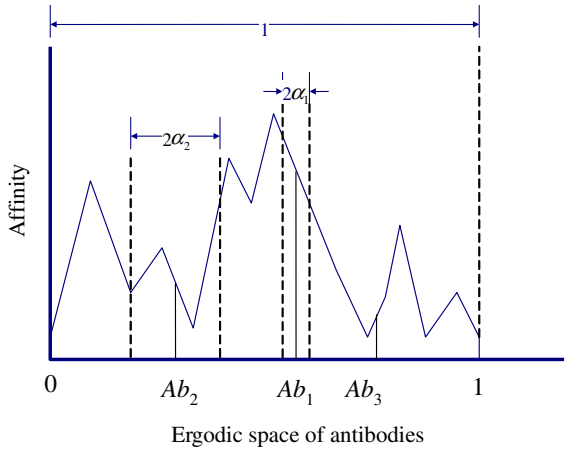

Fig. 1. Multi-neighbourhood search of CSIA.

4. Performances of the CSIA

4.1. Experimental results

In order to verify the performance of the CSIA, the algorithm was applied to the following test functions, and compared with CLONALG [10], COA [16] and simple genetic algorithm (SGA) [21].

(1) Sphere function

$$F_1 = 10 - \sum_{i=1}^{3}(x_i - 5)^2, \tag{10}$$

where $1 \leqslant x_i \leqslant 10$. The global optimum is 10 when $x_i = 5$. The criterion value for success is 9.99.

(2) Step function

$$F_2 = 30 + \sum_{i=1}^{5}[x_i], \tag{11}$$

where $1 \leqslant x_i \leqslant 10.1$. The global optimum is 80 when each variable reaches 10. The criterion value for success is 78.

(3) Rosenbrock function

$$F_3 = 10 - \sum_{i=1}^{5}[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2], \tag{12}$$

where $1 \leqslant x_i \leqslant 10$. The global optimum is 10 when each variable reaches 1. Criterion of success is 9.8.

(4) Shekel's Foxholes function

$$F_4 = \frac{1}{500} + \sum_{j=1}^{25}\frac{1}{j + \sum_{i=1}^{2}(x_i - a_{ij})^6}, \tag{13}$$

where $|x_i| \leqslant 65.56$. The global optimum is about 1.002 when (x_1, x_2) approaches $(-32, 32)$. The measure for success is 1.

(5) Sinc function

$$F_5 = \frac{\sin\left(\sum_{i=1}^{4}|x_i - 5|\right)}{\sum_{i=1}^{4}|x_i - 5|}, \tag{14}$$

where $1 \leqslant x_i \leqslant 10$. The global optimum is 1 when $x_i = 5$. The criterion value for success is 0.999.

Sphere and step functions are used to test general performances of these algorithms. Rosebrock function is employed to test the searching directivity of these algorithms. Foxholes and Sinc functions are complicated multi-modal functions and used to verify the global search capability. The parameters of the CSIA were chosen as $N = 50$, $n = 20$, $\beta = 0.4$ and $d = 5$, and the parameter ρ was given in Table 1 according to experience. For CLONALG, the code length for each variable was 20, the parameters were $N = 50$, $n = 20$, and other parameters were given to make the algorithm have the optimal performance. For SGA, the population size was 100; the code length for each variable was 20; one-point crossover and the elitist strategy were employed; for each function, the crossover probability and mutation probability were given in the intervals $[0.6, 0.8]$ and $[0.001, 0.02]$, respectively according to experience. The COA has two parameters: the first carrier wave constant k and the second carrier wave constant α. In the experiments, $k = 3000$, and α was given in Table 2 to make the algorithm have optimal performance.

The 4 algorithms were all written in Matlab language. Their codes were run by a Matlab computation system on a personal computer at 1.5 GHz CPU. Each algorithm optimized each function 30 times. In each of the 30 computations,

Table 1
The values of parameter ρ of CSIA

Function	F_1	F_2	F_3	F_4	F_5
ρ	4.5	2	6.5	3.5	4.8

Table 2
The values of parameter α of COA

Function	F_1	F_2	F_3	F_4	F_5
α	0.05	1.5	0.05	0.05	0.02

SGA, CLONALG, and CSIA employed different initial populations, and the COA chosen a stochastic number in the range $[0, 1]$ as initial value. The maximum number of generations executed for SGA, CLONALG and CSIA was 200, and the COA stopped if the objective function value was not improved for 3000 iterations in the second carrier wave. If the algorithms attained a solution with a function value equal to the criterion value, they were regarded as converging to the global optimum. Otherwise, the algorithms were regarded as getting stuck at the local optimum. The experimental results are given in Tables 3–7. In these tables, "NF" and "NS" are the numbers of failures and successes for converging to the global optimum at 30 runs, respectively, and they reflect the global search performance of these algorithms; "BV", "WV" and "AV" are the best, worst and average optimum value gained by these algorithms in 30 runs, respectively, and they express the accuracy of the solution gained; "AE" is the average number of objective function evaluations required by these algorithms to attain the criterion values of the functions, and reflects the convergence speed of these algorithms; "AT" is the average CPU implementation time (seconds) required for attaining the criterion values of the functions, through which the simplicity of these algorithms can be inferred.

Table 3
The optimization results of Sphere function

	NF	NS	BV	WV	AV	AE	AT
SGA	0	30	10.000	9.992	9.998	2820	5.981
CLONALG	0	30	10.000	9.992	9.997	1510	2.428
COA	0	30	10.000	10.000	10.000	5655	0.736
CSIA	0	30	10.000	10.000	10.000	660	0.219

Table 4
The optimization results of Step function

	NF	NS	BV	WV	AV	AE	AT
SGA	0	30	80.000	80.000	80.000	3080	7.610
CLONALG	0	30	80.000	80.000	80.000	1259	3.212
COA	0	30	80.000	80.000	80.000	3476	0.553
CSIA	0	30	80.000	80.000	80.000	266	0.112

Table 5
The optimization results of Rosenbrock function

	NF	NS	BV	WV	AV	AE	AT
SGA	26	4	9.920	4.927	8.373	12800	37.849
CLONALG	27	3	9.912	−4.643	6.047	12620	34.159
COA	17	13	9.971	9.659	9.786	10478	2.275
CSIA	3	27	9.992	9.776	9.856	9230	3.971

Table 6
The optimization results of Shekel's Foxholes function

	NF	NS	BV	WV	AV	AE	AT
SGA	7	23	1.002	0.818	0.965	2712	5.617
CLONALG	9	21	1.002	0.691	0.943	3310	4.714
COA	13	17	1.002	0.169	0.746	6983	3.233
CSIA	0	30	1.002	1.002	1.002	4279	2.723

Table 7
The optimization results of Sinc function

	NF	NS	BV	WV	AV	AE	AT
SGA	18	12	1.000	0.128	0.864	7275	17.879
CLONALG	17	13	1.000	0.910	0.988	6613	13.365
COA	0	30	1.000	1.000	1.000	6483	0.980
CSIA	0	30	1.000	1.000	1.000	5366	1.982

In order to further demonstrate the performance of the CSIA, the evolutionary curves of optimizing each function by the 4 algorithms are given. The population sizes of SGA, CLONALG and CSIA are different, which makes the numbers of objective function evaluations required by each algorithm to evolve one generation are also deferent, and thus the number of generations cannot reflect the convergence speed properly. So the number of objective function evaluations is used to evaluate the convergence speed. Since the 4 algorithms are all stochastic algorithms, it is not reasonable to compare their performances by optimizing these functions only one time. To reduce the stochastic influences, each algorithm optimized each function for 20 times, and the average optimal function value was calculated by

$$O(T) = \frac{\sum_{i=1}^{20} O_i(T)}{20} \tag{15}$$

where T is the number of objective function evaluations; $O_i(T)$ is the optimal function value obtained in ith optimization computation through T objective function evaluations; $O(T)$ is the average optimal function value obtained through T objective function evaluations. The evolutionary curves of average optimal function values for each function are illustrated in Figs. 2–6.

4.2. Analysis of experimental results

It can be seen from Table 3–7 that the local search capability of CLONALG is not nice and the high precision solution cannot be obtained by CLONALG. When the functions have many local optimal solutions, the algorithm is easy to stick at local optima. These tables also show that the global search capability of COA is better than CLONALG but the convergence speed of COA is slow. It can be observed that the CSIA has nice global search performance to avoid local optima and the optimal solution gained by the CSIA is the most accurate. Moreover, the convergence speed of the CSIA is faster in most cases. As optimizing function F_4, the convergence speed of the CSIA is slower slightly than that of SGA and CLONALG, but the CSIA found the global optimization solution in all of the 30 computations while other two algorithms stuck at local optima in most of cases, so the slower convergence speed is the necessary cost for the CSIA to find the global optimization. The simulation results also demonstrate that the average CPU time required for the CSIA to converge is much less than SGA and CLONALG, and similar to COA, which shows the operations of the CSIA is simple.

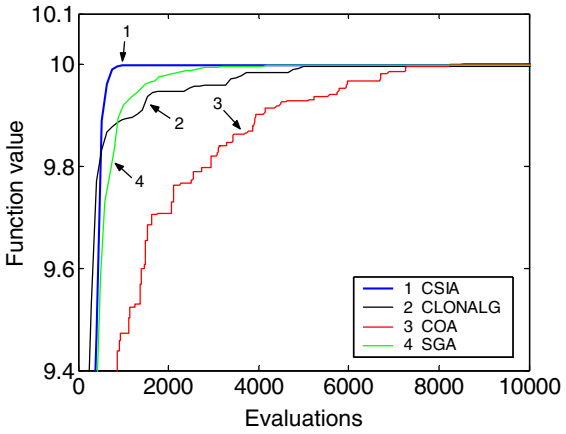

Fig. 2. The evolution curves of average optimal function values for Sphere function.

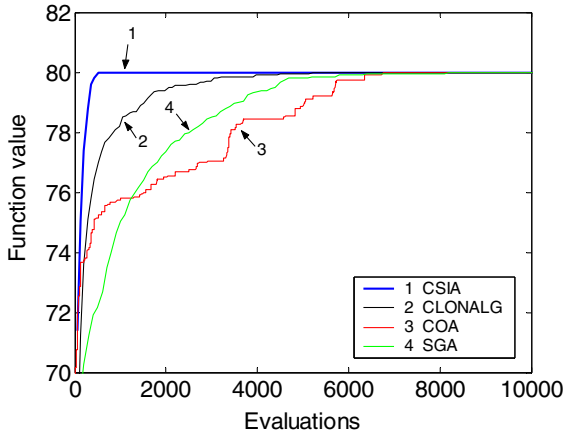

Fig. 3. The evolution curves of average optimal function values for Step function.

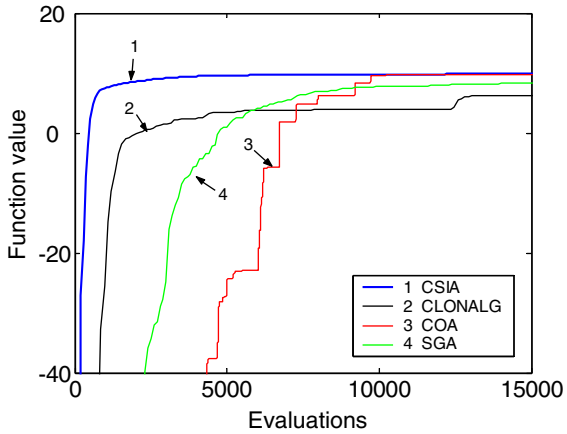

Fig. 4. The evolution curves of average optimal function values for Rosebrock function.

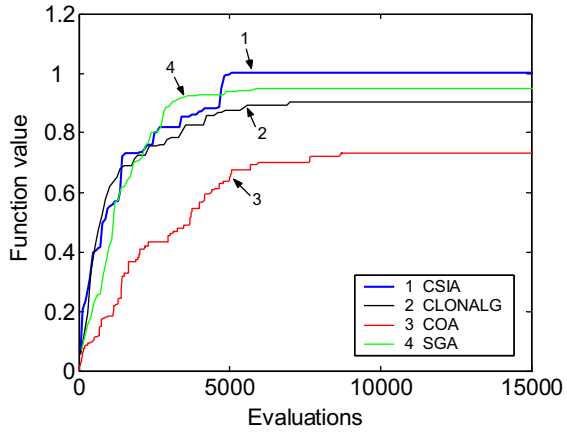

Fig. 5. The evolution curves of average optimal function values for Foxholes function.

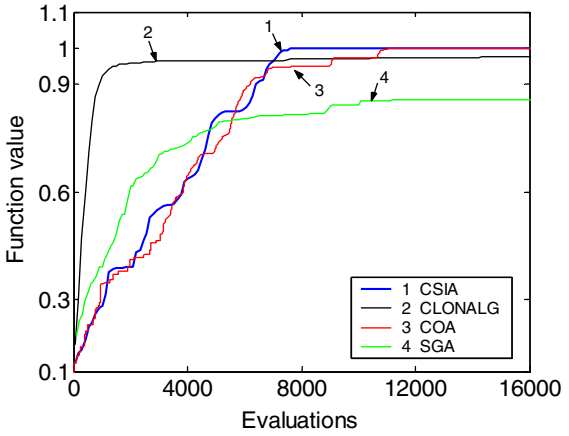

Fig. 6. The evolution curves of average optimal function values for Sinc function.

From Figs. 2–6, it can be seen that the convergence speed of the COA is slow, and the global search capability of COA is nice when the solution space is small, but when the solution space is large (for example, function F_4), the optimization results of COA become worse. The CLONALG can converge quickly, but its global search capability is weak and cannot avoid local optima. It can be observed that the CSIA cannot only converge to the global optimal solution quickly but also has nice global search capability to avoid sticking at local optima. For the Rosebrock and Foxholes functions, the shapes of evolutionary curves of CSIA and CLONALG are similar and the two algorithms all converged quickly, but the CSIA has more powerful global search capability than CLONALG to avoid local optima due to the fact that chaotic search is introduced into CSIA. For the Sinc function, the shapes of evolutionary curves of the CSIA and COA are similar and the two algorithms can all avoid local optima, but the convergence speed of the CSIA is faster than that of COA because the parallel search mechanism of CLONALG is employed in CSIA. The shapes of the evolution curves of CSIA are similar to that of CLONALG or COA, which demonstrates that the CSIA has the characteristics of these two algorithms.

5. Design generalized RBF neuro-fuzzy controller by CSIA

5.1. Generalized RBF neuro-fuzzy controller

The construction of a generalized RBF neural network is shown in Fig. 7 (where the thick lines represent vectors). The output of the network is expressed by

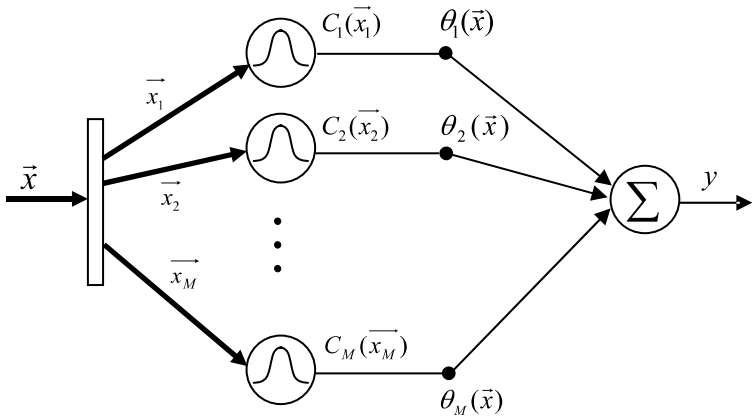

Fig. 7. Generalized RBF neural network.

$$y = f(\vec{x}) = \sum_{i=1}^{M} C_i(\vec{x}_i)\theta_i(\vec{x}) = \vec{C}'\vec{\theta}, \tag{16}$$

where $\rightarrow$ denotes a vector and $'$ is the transpose operator; $y \in R$ is the network output; $\vec{x} \in R^N$ is the input vector. The network has M hidden units, and each hidden unit has a radial basis function. The radial basis function for the ith hidden unit is represented by $C_i(\vec{x}_i)$, where $\vec{x}_i \in R^{N_i}$, $N_i < N$. Each input vector $\vec{x}_i$ of the hidden unit is a subset of the network input vector $\vec{x}$, i.e., $\vec{x}_i \subseteq \vec{x}$. The output of each hidden unit is multiplied by the weighting function $\theta_i(\vec{x})$ and these values are summed to form the network output [22]. If the network is a normalized form, then its output is described by

$$y = f(\vec{x}) = \frac{\sum_{i=1}^{M} C_i(\vec{x}_i)\theta_i(\vec{x})}{\sum_{i=1}^{M} C_i(\vec{x}_i)}. \tag{17}$$

The basis function of the hidden unit is a generalized form of a Gaussian function, which is expressed by

$$C_i(\vec{x}_i) = \exp[-(\vec{x}_i - \vec{c}_i)'\Lambda_i(\vec{x}_i - \vec{c}_i)], \tag{18}$$

where $\vec{c}_i \in R^{N_i}$ is the center vector, and $\Lambda_i \in R^{N_i \times N_i}$ is the width matrix described by

$$\Lambda_i = \mathrm{diag}\left[\left(\frac{1}{\sigma_1^i}\right)^2, \left(\frac{1}{\sigma_2^i}\right)^2, \ldots, \left(\frac{1}{\sigma_{N_i}^i}\right)^2\right]. \tag{19}$$

The basis function described above is generalized in the sense that the standard Gaussian basis functions have the same width parameters in each dimension, while the generalized Gaussian function has different width parameters in each direction, and is defined on a hyperellipsoid in the input space as opposed to a hypersphere as in the standard Gaussian RBF.

The generalized RBF neural network is functionally equivalent to a fuzzy inference system. Each hidden unit of the network represents a fuzzy if-then rule. The centers and widths of the generalized Gaussian basis function for the ith hidden unit correspond to the centers and widths of the Gaussian membership functions of the antecedent part for the ith fuzzy rule, and the output weight of the hidden unit corresponds to the center of the membership function of the consequent part for the fuzzy rule.

Based on the functional equivalence between the generalized RBF neural network and fuzzy system, a generalized RBF neuro-fuzzy controller can be constructed, which is shown in Fig. 8. The controller has N inputs $(x_1, x_2, \ldots, x_N)$ and 1 output y. There are M hidden units in the hidden layer, and the output weight for the ith hidden unit is $\overline{y^i}$. The basis function of each hidden unit is an N-dimensional generalized Gaussian function, which is also called the hyperellipsoid Gaussian function.

5.2. Control inverted pendulum system by the GRBFNFC

The GRBFNFC is employed to control an actual inverted pendulum system manufactured by Googol Technology (HK) Co. Ltd. The nonlinear dynamic equation of the inverted pendulum system is given by

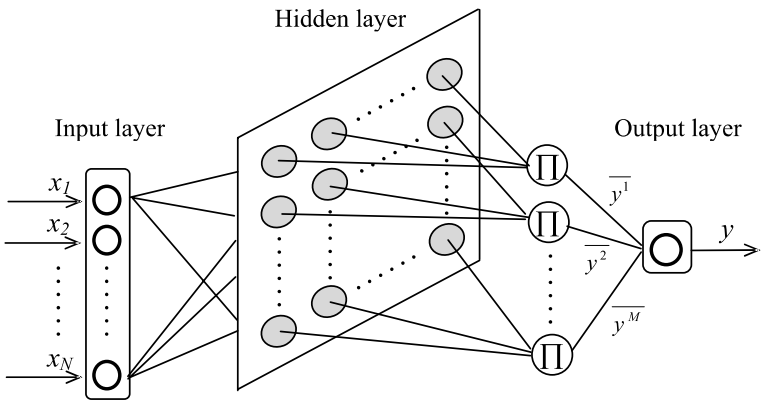

Fig. 8. A N-input and one-output generalized RBF neuro-fuzzy controller.

$$(M+m)\ddot{x} + ml\cos(\theta)\ddot{\theta} = -b\dot{x} + ml\dot{\theta}^2\sin(\theta) + u, \tag{20}$$

$$ml\ddot{x}\cos(\theta) + (I+ml^2)\ddot{\theta} = -f\dot{\theta} + mlg\sin(\theta), \tag{21}$$

where u is the applied force; x is the displacement of the cart; θ is the angle of the pole with respect to the vertical axis; $g = 9.8\ \mathrm{m/s^2}$ is gravity acceleration. Other parameters are: the mass of the cart $M = 1.096$ kg; the mass of the pole $m = 0.109$ kg; the friction coefficient of the cart $b = 0.1$ N/m/s; the length between the axle center and the barycenter of the pole $l = 0.25$ m; the inertia of the pole $I = 0.0034\ \mathrm{kg\ m^2}$; the frictional resistance moment coefficient of the pole $f = 0.00218\ \mathrm{kg\ m^2/s}$.

The inputs of the GRBFNFC are the 4 state variables of the inverted pendulum, namely, x, θ, $\dot{x}$, $\dot{\theta}$, and the output of the controller is the applied force (u). The multi-variable inputs would result in combinatorial explosion of the number of fuzzy control rules, and the size of the GRBFNFC would become very large. In this paper, a state variable synthesis scheme is employed to greatly reduce the size of the GRBFNFC [23]. The state variables x and θ are synthesized as synthesis error e, and $\dot{x}$ and $\dot{\theta}$ are synthesized as synthesis change rate of error ec by the following equations:

$$e = \begin{cases} 1 & k_1x + k_2\theta > 1, \\ k_1x + k_2\theta & |k_1x + k_2\theta| \leqslant 1, \\ -1 & k_1x + k_2\theta < -1, \end{cases} \tag{22}$$

$$ec = \begin{cases} 1 & k_3\dot{x} + k_4\dot{\theta} > 1, \\ k_3\dot{x} + k_4\dot{\theta} & |k_3\dot{x} + k_4\dot{\theta}| \leqslant 1, \\ -1 & k_3\dot{x} + k_4\dot{\theta} < -1, \end{cases} \tag{23}$$

where k_1, k_2, k_3 and k_4 are synthesis coefficients. So the GRBFNFC has only 2 inputs. Each input linguistic variable is covered by 3 linguistic values, which are labeled as positive (P), zero (Z) and negative (N); thus the GRBFNFC has only $3 \times 3 = 9$ hidden units (fuzzy control rules). The inverted pendulum control system with the state variable synthesis scheme is illustrated in Fig. 9. The controller has 4 synthesis coefficients and 1 output scaling factor. Each of the hidden units of the GRBFNFC has 2 centers (c_1^i, c_2^i), 2 widths (σ_1^i, σ_2^i) and 1 output weight $\overline{y^i}$. Thus, a total of 50 parameters $(4 + 1 + 5 \times 9)$ need to be determined. This paper uses CSIA to design all of the 50 parameters automatically.

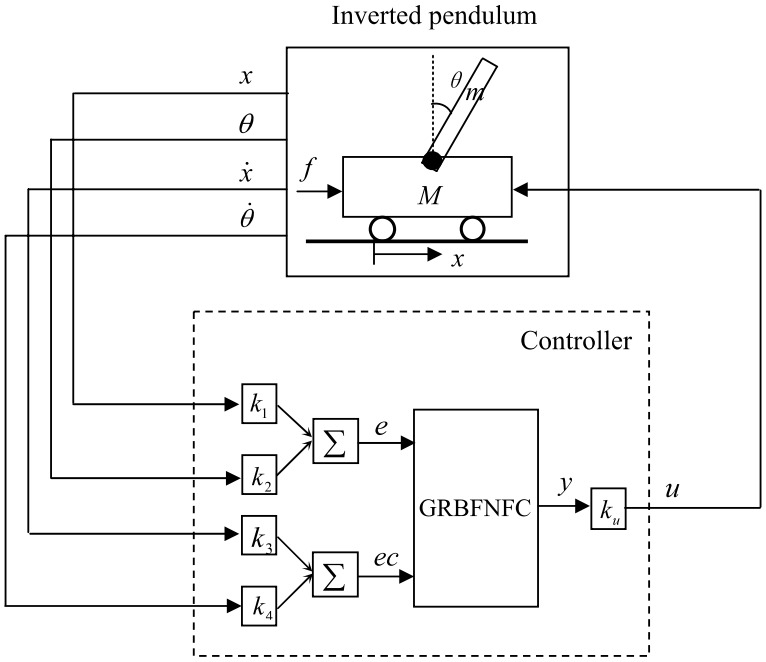

Fig. 9. The inverted pendulum control system with state variable synthesis scheme.

5.3. Design generalized RBF neuro-fuzzy controller by CSIA

The proposed CSIA is employed to optimize the 50 parameters of the controller shown in Fig. 9. First, the initial population is generated. Secondly, each antibody in a population is decoded into a set of controller parameters and represents a controller. Thirdly, the controller and the inverted pendulum compose a control system, and multi-simulations are performed in the control system to evaluate its performance. Thus the affinity of the antibody can be calculated. Fourthly, the operations of the CSIA are used to evolve the populations until the given number of generations is reached. Finally, the highest affinity antibody in the last generation is selected and decoded to obtain the designed GRBFNFC.

Since the CSIA deals with coded parameters, all of the 50 controller parameters that need to be tuned must be encoded into a finite length of string called an antibody. The coding scheme of the antibody is illustrated in Fig. 10, in which each gene of the antibody represents a controller parameter and the antibody indicates a controller. Each gene of the antibody is a value of the chaotic variable, so the following linear mapping is needed to decode the antibody.

$$p_i = g_{i,\min} + (g_{i,\max} - g_{i,\min}) \times g_i, \quad g_i \in [0,1], \ \ i = 1,2,\ldots,50, \tag{24}$$

where g_i is the ith gene of the antibody; p_i is the ith controller parameter; $g_{i,\max}$ and $g_{i,\min}$ are the upper and lower bounds of the ith controller parameter range, respectively. In order to express the linguistic values of the GRBFNFC properly, the ranges of the centers of Gaussian membership functions for positive, zero and negative linguistic values are given by $[0.3,1]$, $[-0.3,0.3]$ and $[-1,-0.3]$, respectively, and the range of the widths of these membership functions is given by $[0.1,0.7]$.

Each antibody in a population represents a controller. The controller and the mathematical model of the actual inverted pendulum system compose an inverted pendulum control system. Multi-simulations are performed in the control system, and the quadratic performance index is selected to evaluate the performance of the control system in each simulation. The performance indexes gained in these simulations are summed to obtain the controller performance index

$$J = \sum_{i=1}^{L} \sum_{k=1}^{S} [X'(k)QX(k) + u'(k)Ru(k)], \tag{25}$$

where $X'(k) = [x(k), \theta(k), \dot{x}(k), \dot{\theta}(k)]$ is the state vector of the control system; $u(k)$ is the output of the controller; $Q = \mathrm{diag}[10,1,0,0]$ and $R = 0.001$ are weight matrixes; $S = 200$ is the simulation length; $L = 8$ is the number of simulations. Each simulation is performed with a different initial state. Multi-simulations are used to evaluate the controller to enable the evaluation to cover a wider operating range [24]. The performance index of the controller refers to the affinity of the antibody. A smaller performance index equals higher affinity, so the antibody affinity can be defined by

$$\text{Affinity} = \frac{d}{J + c}, \tag{26}$$

where $c = 10^{-2}$ ensures the denominator is not zero and $d = 10^2$ ensures that the affinity is not too small.

5.4. Experimental results

The CSIA was used to tune the controller parameters to maximize the antibody affinity in Eq. (26). The parameters of the CSIA were chosen as $N = 50$, $n = 20$, $\beta = 0.6$, $d = 10$, and $\rho = 4$. The software was written with Matlab language, and was run in a Matlab version 6.0 environment. In order to allow the CSIA to find the optimal or near-optimal con-

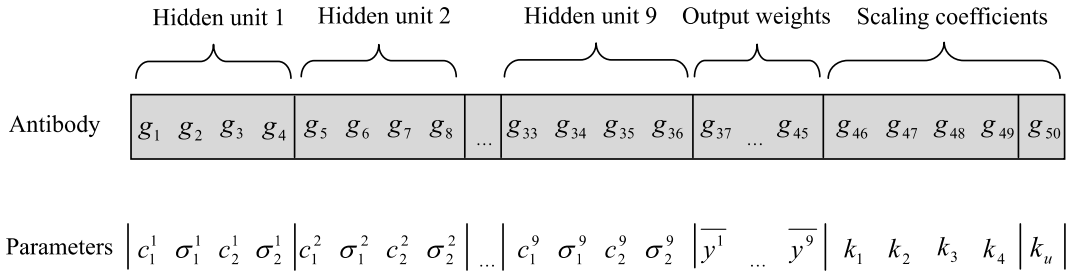

Fig. 10. Structure of an antibody.

Table 8
The basis functions and weights of CSIA-designed GRBFNFC

Hidden units	Linguistic values of E	Membership functions		Linguistic values of EC	Membership functions		$\overline{y^i}$
		c_1^i	σ_1^i		c_2^i	σ_2^i	
1	P	0.6212	0.5347	P	0.6407	0.3438	−0.6595
2	P	0.7324	0.3974	Z	−0.0392	0.3747	−0.6122
3	P	0.6243	0.4414	N	−0.6640	0.3581	−0.0419
4	Z	0.0050	0.3914	P	0.6106	0.4267	−0.5711
5	Z	0.0308	0.3674	Z	−0.0313	0.4901	−0.0642
6	Z	0.0239	0.3387	N	−0.5918	0.3289	0.7058
7	N	−0.6205	0.3699	P	0.6327	0.3687	0.0749
8	N	−0.6053	0.4788	Z	−0.0073	0.3606	0.6389
9	N	−0.5781	0.3569	N	−0.6439	0.4280	0.7294

troller parameters quickly, initial controller parameters were given according to experiences. The CSIA was executed on a 1.5 GHz personal computer for 100 generations. At the last generation, the highest affinity antibody was selected from the population and taken as the CSIA-designed GRBFNFC. The synthesis coefficients k_1, k_2, k_3, k_4 and the output scaling factor k_u of the designed GRBFNFC are 1.3854, 2.2696, 0.7791, 0.4209 and 110.9326, respectively, and the basis functions and weights of the controller are presented in Table 8.

The CSIA-designed GRBFNFC was employed to control the actual inverted pendulum system. The inverted pendulum control system included a personal computer, a kinetic control card, an electric cabinet and an inverted pendulum system. The arrangement of the whole experimental setup is illustrated in Fig. 11. The CSIA-designed GRBFNFC was written in C language. The control signals generated by the execution code compiled from the C program were set to the DSP of the kinetic control card through an ISA bus. The DSP processed the received signals to produce the control actions, and these control actions were delivered to a servo motor through a servo driver to apply suitable force to the cart. The states of the inverted pendulum system were sensed by a photoelectric encoder and fed back to the execution code of the controller through the kinetic control card.

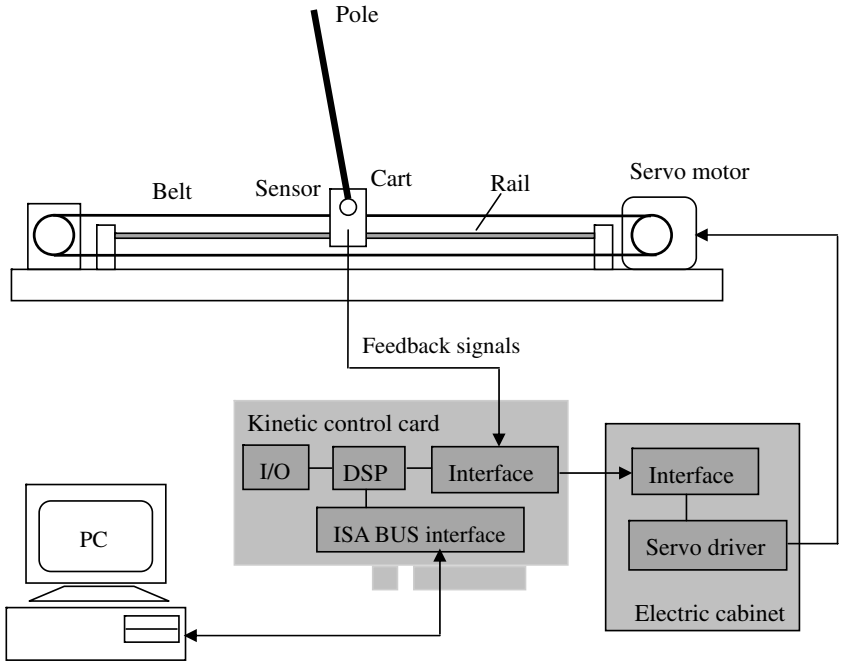

Fig. 11. Experimental setup of the inverted pendulum control system.

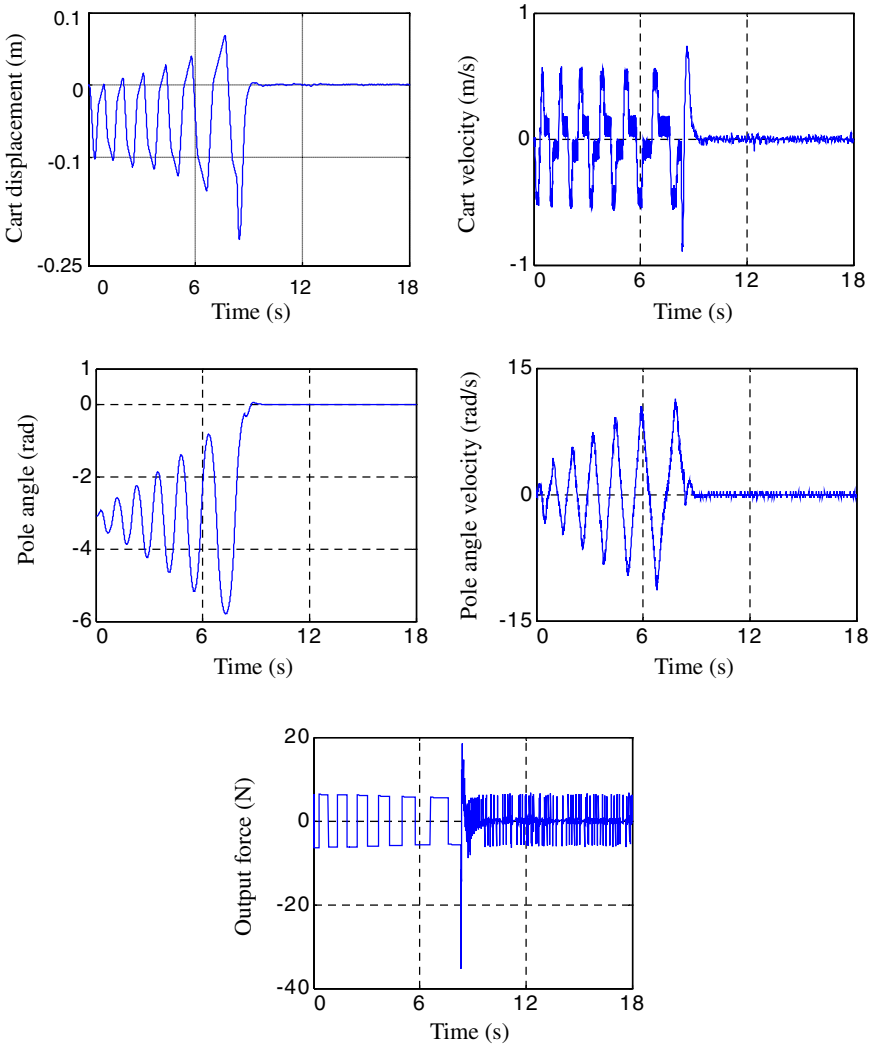

Fig. 12. The responses of the actual inverted pendulum control system when the pole is swung up.

The CSIA-designed GRBFNFC was used to drive the cart to the zero position and stabilize the inverted pendulum simultaneously. The sampling interval was 0.006 s. The inverted pendulum was swung up from a pendulum position by the program provided by the manufacturer. When the pole angle was less than 0.3 rad, the CSIA-designed GRBFNFC was switched to control the inverted pendulum. The dynamic responses of the inverted pendulum control system are illustrated in Fig. 12. When the system was in a steady state, the displacement of the cart was bounded within 0.002 m and the angle of the pendulum was within 0.3°.

6. Conclusion

In this paper, we have proposed a chaos search immune algorithm that integrates chaos optimization algorithm and the clonal selection algorithm. The chaotic search mechanism of chaos optimization algorithm is introduced into clonal selection algorithm to improve its local and global search capabilities, and the learning process and parallel search mechanism of clonal selection algorithm is employed to improve the search efficiency of the chaos optimization algorithm. Moreover, a generalized RBF neuro-fuzzy controller was designed automatically by the CSIA, and applied to real-time control of an actual inverted pendulum system. Experimental results demonstrate that the designed GRBFNFC has very nice performances.

Acknowledgement

This work was granted financial support from China Postdoctoral Science Foundation (023209022).

References

[1] De Castro LN, Timmis J. Artificial immune systems: a new computational intelligence approach. Springer-Verlag; 2002.
[2] Forrest S, Perelson AS, Allen L, Cherukuri R. Self-nonself discrimination in a computer. In: Proc IEEE symp on research in security and privacy, Oakland, CA; 1994. p. 202–12.
[3] Dasgupta D, Forrest S. Tool breakage detection in milling operations using a negative-selection algorithm. Department of Computer Science, University of New Mexico. Technical Report CS95-5; 1995.
[4] Dasgupta D, Forrest S. Novelty detection in time series data using ideas from immunology. In: Proc 5th int conf on intelligent systems, Reno, Nevada, USA; 1996. p. 19–21.
[5] Tang Z, Yamaguchi T, Tashima K, Ishizuka O, Tanno K. A multiple valued immune network and its applications. IEICE Trans Fundam 1999;E82-A(6):1102–8.
[6] Hunt JE, Cooke DE. Learning using an artificial immune system. J Network Comput Appl 1996;19:189–212.
[7] Mori M, Tsukiyama M, Fukuda T. Immune algorithm with searching diversity and its application to resource allocation problem. Trans Inst Electr Eng Jpn 1993;113-C(10):872–8.
[8] Chun JS, Kim MK, Jung HK. Shape optimization of electromagnetic devices using immune algorithm. IEEE Trans Magn 1997;33(2):1876–9.
[9] Jiao LC, Wang L. A novel genetic algorithm based on immunity. IEEE Trans Syst Man Cybernet A 2000;30(5):552–61.
[10] De Castro LN, Von Zuben FJ. Learning and optimization using the clonal selection principle. IEEE Trans Evol Comput 2002;6(3):239–51.
[11] Zuo XQ, Li SY. Solving function optimization problem with immune principle. J Syst Eng Electron 2004;15(4):702–9.
[12] Zuo XQ, Li SY. Design of a fuzzy logical controller by immune algorithm with application to an inverted pendulum system. In: Proc IEEE congress on evolutionary computation, Canberra, Australia; 2003. p. 100–6.
[13] Lorenz EN. Deterministic non-periodic flows. J Atmos Sci 1963;20:130–41.
[14] Chen G, Dong X. From chaos to order: methodologies, perspectives and applications. Singapore: World Scientific; 1998.
[15] Zhou C, Chen T. Chaotic annealing for optimisation. Phys Rev E 1997;55(3):2580–7.
[16] Li B, Jiang W. Optimizing complex functions by chaos search. Int J Cybernet Syst 1998;29(4):409–19.
[17] Lu Z, Shieh LS, Chen G, Coleman NP. Simplex sliding mode control for nonlinear uncertain systems via chaos optimization. Chaos, Solitons & Fractals 2005;23:747–55.
[18] Hasegawa M, Ikeguchi T, Aihara K, Itoh K. A novel chaotic search for quadratic assignment problems. Eur J Oper Res 2002;139:543–56.
[19] Ji MJ, Tang HW. Application of chaos in simulated annealing. Chaos, Solitons & Fractals 2004;21:933–41.
[20] Canabarro AA, Gl'eria IM, Lyra ML. Periodic solutions and chaos in a non-linear model for the delayed cellular immune response. Physica A 2004;342:234–41.
[21] Goldberg DE. Genetic algorithm in search, optimization and machine learning. MA: Addison-Wesley; 1989.
[22] Hunt KJ, Hass R, Murray-Smith R. Extending the functional equivalence of radial basis function networks and fuzzy inference systems. IEEE Trans Neural Networks 1996;7(3):776–81.
[23] Cheng FY, Zhong GM, Li YS. A parameter fuzzy controller for a double inverted pendulum. Inform Control 1995;24(3):189–92 (in Chinese).
[24] Seng TL, Khalid MB. Tuning of a neuro-fuzzy controller by genetic algorithm. IEEE Trans Syst Man Cybernet B 1999;29(2):226–36.