22

Social Spider Optimization – Modifications and Applications

Ahmed F. Ali
Department of Computer Science
Suez Canal University, Ismaillia, Egypt

Mohamed A. Tawhid
Department of Mathematics and Statistics, Faculty of Science
Thompson Rivers University, Kamloops, Canada

CONTENTS

22.1 Introduction

The social spider optimization (SSO) is a resent swarm intelligence algorithm proposed by Cuevas et al. [1]. The SSO algorithm simulates the social behav-

ior of the social spiders and their movements on the communal web. The SSO algorithm starts by generating random solutions (spiders) which contain female and male spiders. The number of the females is greater than the number of the males in the colony (population). Each spider transmits its information through the communal web by encoding it as small vibrations. The weight of each spider (fitness function) is presented by calculating its fitness function value. The female spiders update their positions according to the attraction toward or the repulsion from other males. There are two types of male spiders according to their weights. The spider male with the weight greater than the median weight of the other males in the colony is the dominant male, while the other males with weight less than the median weight of other males are non-dominant males. The dominant male is responsible for mating the females in his mating range. The males with the highest weight have a big chance of influencing a new offspring. This chapter is structured as follows. In Section 22.2, we give a brief introduction to the SSO algorithm and how it works. We present some modifications of the SSO algorithm in the female and male cooperative operators in Section 22.3. In Section 22.4, we solve an economic load dispatch problem by a modification of the SSO algorithm. Finally, we summarize the conclusion in Section 22.5.

22.2 Original SSO algorithm in brief

In this section, we highlight the main steps of the SSO algorithm. These steps are shown in Algorithm 24.

22.2.1 Description of the original SSO algorithm

The SSO algorithm starts by initializing the values of the number of solutions N in the population size S, threshold PF and maximum number of iterations max_{itr} as shown in step 1. The number of females N_f and males N_m solutions are assigned in steps 2 as shown in Equations 22.1–22.2.

$$N_f = floor[(0.9 - rand(0,1) \cdot 0.25) \cdot N] \tag{22.1}$$

$$N_m = N - N_f \tag{22.2}$$

The counter of the initial iteration t is initialized in step 3. In steps 4–7, the initial female solutions are randomly generated as shown in Equation 22.3.

$$f_{i,j}^0 = p_j^{low} + rand(0,1) \cdot (p_j^{high} - p_j^{low}) \quad i = 1, 2, \ldots, N_f; j = 1, 2, \ldots, n \tag{22.3}$$

The initial male solutions are randomly generated as shown in Equation 22.4 in steps 9–13.

$$m_{k,j}^{0} = p_j^{low} + rand(0,1) \cdot (p_j^{high} - p_j^{low}) \quad k = 1, 2, \ldots, N_m; j = 1, 2, \ldots, n \tag{22.4}$$

Each solution in the population is evaluated by calculating its weight (fitness function) w_i by assigning the best $best_s$ and worst $worst_s$ solutions as shown in Equation 22.5 in steps 15–17.

$$w_i = \frac{J(s_i) - worst_s}{best_s - worst_s} \tag{22.5}$$

In step 19, the vibrations (transmitted information) between the solution i and the best solution b (s_b) in the population can be defined as follows.

$$Vibb_i = w_b \cdot e^{-d_{i,b}^2} \tag{22.6}$$

while the vibrations (transmitted information) between the solution i and the nearest female solution $f(s_f)$ can be defined as

$$Vibf_i = w_f \cdot e^{-d_{i,f}^2} \tag{22.7}$$

The female spiders (solutions) update their positions in steps 20–25, where α, β, δ and rand are random numbers in [0,1], whereas t is the number of iterations as shown in Equation 22.8.

$$f_i^{t+1} = \begin{cases} f_i^t + \alpha \cdot Vibc_i.(s_c - f_i^t) + \beta \cdot Vibb_i \cdot (s_b - f_i^t) \\ \qquad\qquad + \delta \cdot (rand - 0.5) \quad \text{at } PF \\ f_i^t - \alpha \cdot Vibc_i.(s_c - f_i^t) - \beta \cdot Vibb_i \cdot (s_b - f_i^t) \\ \qquad\qquad + \delta \cdot (rand - 0.5) \quad \text{at } 1 - PF \end{cases} \tag{22.8}$$

where α, β, δ and rand are random numbers in [0,1], whereas t is the number of iterations.

The dominant spider with a weight value above the median value of the other males in the population is calculated in step 26. In steps 27–34, the male spiders update their positions as shown in Equation 22.9

$$m_i^{t+1} = \begin{cases} m_i^t + \alpha \cdot Vibf_i \cdot (s_f - m_i^t) + \delta \cdot (rand - 0.5) & \text{if} \quad w_{N_f+i} > w_{N_f+m} \\ m_i^t + \alpha \cdot \left(\frac{\sum_{h=1}^{N_m} m_h^t \cdot w_{N_f+h}}{\sum_{h=1}^{N_m} w_{N_f+h}} - m_i^t \right) \end{cases} \tag{22.9}$$

where the solution s_f represents the nearest female solution to the male solution i.

The range r (range of mating) is calculated in step 35.

In steps 36–46, the spider with a heavier weight has a big chance to influence the new product. The influence probability Ps_i of each solution

is assigned by the roulette wheel selection method as follows.

$$Ps_i = \frac{w_i}{\sum_{j \in T^t} w_j} \tag{22.10}$$

The iteration counter is increasing in step 47. Finally, the overall best solution is submitted in step 49.

Algorithm 24 Social spider optimization algorithm.

1: Set the initial value of total number of solutions N in the population size S, threshold PF, and maximum number of iterations Max_{itr}
2: Set the number of female spiders N_f and number of males spiders N_m
3: Set $t := 0$ ▷ **Counter initialization**
4: **for** $(i = 1; i < N_f + 1; i++)$ **do**
5: **for** $(j = 1; j < n + 1; j++)$ **do**
6: $f_{i,j}^t = p_j^{low} + rand(0,1) \cdot (p_j^{high} - p_j^{low})$
7: **end for**
8: **end for** ▷ **Initialize randomly the female spider**
9: **for** $(k = 1; k < N_m + 1; k++)$ **do**
10: **for** $(j = 1; j < n + 1; j++)$ **do**
11: $m_{k,j}^t = p_j^{low} + rand(0,1) \cdot (P_j^{high} - p_j^{low})$
12: **end for**
13: **end for** ▷ **Initialize randomly the male spider**
14: **repeat**
15: **for** $(i = 1; i < N + 1; i++)$ **do**
16: $w_i = \frac{J(s_i) - worst_s}{best_s - worst_s}$
17: **end for** ▷ **Evaluate the weight (fitness function) of each spider**
18: **for** $(i = 1; i < N_f + 1; i++)$ **do**
19: Calculate the vibrations of the best local and best global solutions $Vibc_i$ and $Vibb_i$
20: **if** $(r_m < PF)$ **then**
21: $f_i^{t+1} = f_i^t + \alpha \cdot Vibc_i \cdot (s_c - f_i^t) + \beta \cdot Vibb_i \cdot (s_b - f_i^t) + \delta \cdot (rand - 0.5)$
22: **else**
23: $f_i^{t+1} = f_i^t - \alpha \cdot Vibc_i \cdot (s_c - f_i^t) - \beta \cdot Vibb_i \cdot (s_b - f_i^t) + \delta \cdot (rand - 0.5)$
24: **end if**
25: **end for**
26: Find the median male individual (w_{N_f+m}) from M
27: **for** $(i = 1; i < N_m + 1; i++)$ **do**
28: Calculate $Vibf_i$
29: **if** $(w_{N_f i} > w_{N_f+m})$ **then**
30: $m_i^{t+1} = m_i^t + \alpha \cdot Vibf_i \cdot (s_f - m_i^t) + \delta \cdot (rand - 0.5)$
31: **else**
32: $m_i^{t+1} = m_i^t + \alpha \cdot \left(\frac{\sum_{h=1}^{N_m} m_h^t \cdot w_{N_f+h}}{\sum_{h=1}^{N_m} w_{N_f+h}} - m_i^t \right)$
33: **end if**
34: **end for**
35: Calculate the radius of mating r, where $r = \frac{\sum_{j=1}^{n}(p_i^{high} - p_j^{low})}{2 \cdot n}$ ▷ **Perform the mating operation**
36: **for** $(i = 1; i < N_m + 1; i++)$ **do**
37: **if** $(m_i \in D)$ **then**
38: Find E^i
39: **if** E^i is not empty **then**
40: Form s_{new} using the roulette method

```
            if w_new > w_wo then
                Set s_wo = s_new
            end if
        end if
    end if
end for
t = t + 1                                   ▷ Iteration counter is increasing
until (t ≥ Max_itr)                         ▷ Termination criteria are satisfied
Produce the best solution.
```

22.3 Modifications of the SSO algorithm

It is known that the search process in a population-based meta-heuristic is categorized in two essential phases, namely, exploration and exploitation. A random behavior in the exploration phase is to figure out the search space as much as possible. On the other hand, the exploitation toward the promising regions is the essential objective of the latter phase. It is a difficult task to find a suitable balance between these two phases because of the stochastic nature of population-based meta-heuristic algorithms. Researchers show that chaotic maps are able to improve both phases. The literature indicates that combining the chaos theory into population-based algorithms is one of the efficient and effective methods for fostering both exploration and exploitation [8], [9], [10], [11], [12].

This work combined four chaotic maps (namely, Chebyshev, circle, Gauss/-mouse, and iterative) into the population-based meta-heuristic algorithm called Social Spider Optimization algorithm. We present the effect of replacing the random parameters α, β, δ and rand in Equations 22.8–22.9 by the chaotic maps $C1, C2, C3$, and $C4$ (Chebyshev, circle, Gauss/mouse, and iterative), respectively. The proposed algorithm is called Chaotic Social Spider Optimization (CSSO) algorithm as shown in the following subsections.

22.3.1 Chaotic maps

The chaotic maps show a random behavior although they have no random variables. The mathematical forms of these maps are shown in Table 22.1.

TABLE 22.1
Chaotic maps.

No Name	Chaotic map	Range
C1 Chebyshev [5]	$x_{i+1} = \cos(i \cos^{-1}(x_i))$	(-1,1)
C2 Circle [6]	$x_{i+1} = \mod (x_i + b - (\frac{a}{2\pi}) \sin(2\pi x_k), 1), a = 0.5$ and $b = 0.2$	(0,1)
C3 Gauss/mouse [2]	$x_{i+1} = \begin{cases} 1 & x_i = 0 \\ \frac{1}{\mod (x_i, 1)} & \text{otherwise} \end{cases}$	(0,1)
C4 Iterative [7]	$x_{i+1} = \sin(\frac{a\pi}{x_i}), a = 0.7$	(-1,1)

We invoke some of these maps in the female and male spiders position update equations instead of using the standard random parameters to increase the diversity of the search and avoid trapping in local minima. The initial point for the four chaotic maps is random number between [0,1]. We use the initial point $x^0 = 0.7$ as suggested in [4].

22.3.2 Chaotic female cooperative operator

The female spiders update their positions as shown in Equation 22.11 by replacing the random α, β, δ and rand parameters by four chaotic maps.

$$f_i^{t+1} = \begin{cases} f_i^t + C1 \cdot Vibc_i \cdot (s_c - f_i^t) + C2 \cdot Vibb_i \cdot (s_b - f_i^t) \\ \qquad\qquad + C3 \cdot (C4 - 0.5) \quad \text{at } PF \\ f_i^t - C1 \cdot Vibc_i \cdot (s_c - f_i^t) - C2 \cdot Vibb_i \cdot (s_b - f_i^t) \\ \qquad\qquad + C3 \cdot (C4 - 0.5) \quad \text{at } 1 - PF \end{cases} \tag{22.11}$$

where $C1, C2, C3$, and $C4$ are chaotic maps and t is the number of iterations.

22.3.3 Chaotic male cooperative operator

The position of the male spider can be calculated as shown in Equation 22.12.

$$m_i^{t+1} = \begin{cases} m_i^t + C1 \cdot Vibf_i \cdot (s_f - m_i^t) + C3 \cdot (C3 - 0.5) & \text{if } \; w_{N_f+i} > w_{N_f+m} \\ m_i^t + C1 \cdot \left(\frac{\sum_{h=1}^{N_m} m_h^t \cdot w_{N_f+h}}{\sum_{h=1}^{N_m} w_{N_f+h}} - m_i^t \right) \end{cases} \tag{22.12}$$

where the solution s_f represents the nearest female solution to the male solution i, and $C1, C2$ are chaotic maps.

22.4 Application of SSO algorithm for an economic load dispatch problem

In this section, we apply the modified SSO algorithm for solving the economic load dispatch problem (ELD)

22.4.1 Economic load dispatch problem

The economic load dispatch problem (ELD) [3] is a continuous optimization problem. In this problem, we try to find the optimal combination of power generation where the total production cost of the system is minimized. The convex and non-convex cost function of a generator can be defined as follows.

$$F_i(P_i) = a_i P_i^2 + b_i P_i + c_i \tag{22.13}$$

$$F_i(P_i) = a_i P_i^2 + b_i P_i + c_i + |d_i sin[e_i \cdot (P_i^{min} - P_i)]| \tag{22.14}$$

where P_i is the active power output, $F_i(P_i)$ is the generation cost, P_i^{min} is the minimum output limit of the generator and the a_i, b_i, c_i, d_i, e_i are the cost coefficients of the generator. The fuel cost of all generators can be defined in Equation 22.15.

$$MinF(P) = \sum_{i}^{Ng} a_i P_i^2 + b_i P_i + c_i + |d_i sin[e_i \cdot (P_i^{min} - P_i)]| \tag{22.15}$$

where Ng is the number of generating units.

22.4.2 Problem Constraints

The ELD problem is a constrained optimization problem. The total power generated should cover the power demand and the active power losses as shown in Equation 22.16.

$$\sum_{i=1}^{Ng} P_i = P_D + P_{loss} \tag{22.16}$$

where P_D is the total demand load and P_{loss} is the total transmission losses computed and it can be calculated as shown in Equation 22.17.

$$P_{loss} = \sum_{i=1}^{Ng} \sum_{j=1}^{Ng} P_i B_{ij} P_j \tag{22.17}$$

where B_{ij} is the loss coefficient matrix.

The output power of each generator should be within a lower and an upper limit as shown in Equation 22.18.

$$P_i^{min} \leq P_i \leq P_i^{max} \tag{22.18}$$

where P_i^{min} and P_i^{max} are the lower and the upper limit of the *ith* generators.

22.4.3 Penalty function

In order to transform the constrained optimization problem to an unconstrained optimization problem, we use the penalty function technique. We can handle the constraints of the ELD problem as shown in Equation 22.19.

$$\begin{aligned} \text{Min}(Q(P_i)) = &\quad F_c(P_i) + pen \cdot G[h_k(P_i)] \\ \text{Subject to}: &\quad g_j(P_i) \leq 0, j = 1, \ldots, J \end{aligned} \tag{22.19}$$

where *pen* is the penalty factor and $G[h_k(P_i)]$ is the penalty function calculated as shown in Equation 22.20.

$$G[h_k(P_i)] = \varepsilon^2 + \omega^2 \tag{22.20}$$

where ε and ω are equality and inequality constraint violation penalties, respectively and can be calculated as follows.

$$\varepsilon = \left| P_D + P_{loss} - \sum_{j=1}^{Ng} P_j \right| \tag{22.21}$$

and

$$\omega = \begin{cases} |P_i^{min} - P_i| & P_i^{min} > P_i \\ 0 & P_i^{min} < P_i < P_i^{max} \\ |P_i - P_i^{max}| & P_i^{max} < P_i \end{cases} \tag{22.22}$$

22.4.4 How can the SSO algorithm be used for an economic load dispatch problem?

In this subsection, we apply the proposed CSSO algorithm to solve an economic load dispatch problem. The first step is the initialization of all parameters in the algorithm such as the population size S and maximum number of iterations Max_{itr} which are set to be 20 and 500 respectively. The number of female N_f and male N_m spiders are assigned. In step 2, the iteration counter t is initialized. In steps 4–13, the initial population is randomly generated where the problem dimension D is set to be 6 which is the length of the input data in Table 22.3 where the lower and upper bounds of the generated data are set to $L = \{10, 10, 35, 35, 130, 125\}$, $U = \{125, 150, 225, 210, 325, 315\}$. In steps 15-17, the objective function of each solution is evaluated by calculating its fitness function OF(.). The best ($best_s$) and worst ($worst_s$) solutions in the population are assigned; then the weight w_i of each solution is calculated. In steps 18-25, the vibrations of the best local and best global solutions $Vibc_i$ and $Vibb_i$ as shown in Equation 22.12 are calculated. In step 26, the median male individual (w_{N_f+m}) is calculated. In steps 27–34, the male solutions are updated after calculating the radius of mating. The overall process is repeated until the maximum number of iterations is reached.

22.4.5 Description of experiments

We test the proposed CSSO algorithm on six-generator test systems for a total demand of 700 MW and 800 MW. We show the parameters of the proposed CSSO in Table 22.2. To simplify the problem, we set the values of parameters d, e in Equation 22.14 to zero. The limits of the generator active power and the coefficients of the fuel cost are given in Tables 22.3–22.4, respectively. We

TABLE 22.2
Parameter setting.

Parameters	Definitions	Values
n	Population size	20
l	Attractive length scale	1.5
f	Intensity of attraction	0.5
PF	Probability threshold	0.7
Max_{itr}	Maximum number of iterations	500

TABLE 22.3
Limits of the generator active power.

Generator	1	2	3	4	5	6
$P_min(MW)$	10	10	35	35	130	125
$P_max(MW)$	125	150	225	210	325	315

TABLE 22.4
The coefficients of the fuel cost.

No.	a	b	c
1	0.15240	38.53973	756.79886
2	0.10587	46.15916	451.32513
3	0.02803	40.39655	1049.9977
4	0.03546	38.30553	1243.5311
5	0.02111	36.32782	1658.5596
6	0.01799	38.27041	1356.6592

set the number of the population size to 20 and the maximum number of iterations Max_{itr} to 500.

22.4.6 Results obtained

In Figures 22.1–22.2, we show the performance of the proposed CSSO algorithm at total system demand at 700, 800 MW by plotting the number of iterations versus the cost($/h). Also, to test the efficiency of the proposed CSSO algorithm, we compare it with the standard SSO algorithm on six-generator test systems at total demand of 700 MW and 800 MW. We apply the same termination criteria by setting the error tolerance to 0.01 MW or when the maximum number of iterations Max_{itr} reaches 500. We report the best, average (Avg) and standard deviation (Std) results in Tables 22.5–22.6. The results in Tables 22.5–22.6 show that the proposed CSSO algorithm is a promising algorithm and its performance is better than the standard SSO algorithm.

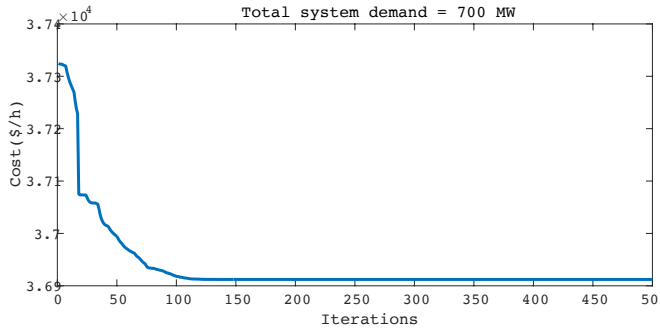

FIGURE 22.1
Total system demand at 700 MW.

TABLE 22.5
Comparison between CSSO and SSO total system demand = 700 MW.

Algorithm		P1(MW)	P2(MW)	P3(MW)	P4(MW)	P5(MW)	P6(MW)	Loss(MW)	Cost($/h)
SSO	Best	28.0745	10.0572	119.987	117.7735	231.1342	212.397	19.4241	36911.82
	Avg	28.3961	10.2698	119.162	119.0377	230.2975	212.255	19.4099	36912.75
	Std	0.6923	0.2655	2.2236	1.7081	2.9535	3.79	0.05938	1.0014
CSSO	Best	28.1475	10.0377	119.715	118.048	231.0216	212.417	19.4304	36911.18
	Avg	28.3622	10.0211	119.0855	118.582	231.1235	212.709	19.4225	36911.75
	Std	0.1477	0.0115	0.4114	0.38235	0.08133	0.2066	0.0132	0.9145

TABLE 22.6
Comparison between CSSO and SSO total system demand = 800 MW.

Algorithm		P1(MW)	P2(MW)	P3(MW)	P4(MW)	P5(MW)	P6(MW)	Loss(MW)	Cost($/h)
SSO	Best	32.46756	14.34413	141.9085	135.7281	257.7262	243.1414	25.3342	41895.74
	Avg	32.58655	14.49139	141.7118	136.2047	257.3585	242.9544	25.3224	41896.15
	Std	0.38255	0.49511	0.97066	0.88625	1.2138	1.3818	0.037027	0.25815
CSSO	Best	32.4951	14.34415	141.9048	135.7519	257.7254	243.1414	25.3341	41895.53
	Avg	32.5745	14.4824	141.5423	135.8414	257.6575	243.0017	25.3384	41895.71
	Std	0.09235	0.09821	0.2135	0.05445	0.9441	0.9462	0.0772	0.09919

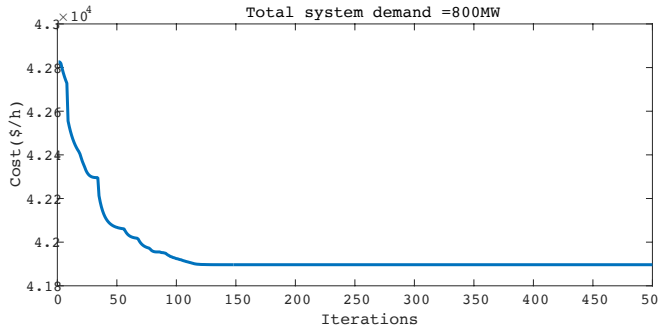

FIGURE 22.2
Total system demand at 800 MW.

22.5 Conclusions

In this chapter, we present a social spider optimization (SSO) algorithm. We apply some modifications to the standard SSO algorithm. The modified SSO algorithm is called the Chaotic Social Spider Optimization (CSSO) algorithm. We solve an economic load dispatch problem by CSSO. The chaotic maps are able to enhance both exploitation and exploration of SSO. Also, we test the modified SSO algorithm on six-generator test systems for a total demand of 700MW and 800MW and compare with the standard SSO algorithm. The results show that the CSSO algorithm is a promising algorithm and can solve an economic load dispatch problem in reasonable time.

References

1. E. Cuevas, M. Cienfuegos, D. Zaldívar and M. Pérez-Cisneros. "A swarm optimization algorithm inspired in the behavior of the social-spider". *Expert Systems with Applications*, vol. 40, no. 16, pp. 6374–6384, 2013.

2. V. Jothiprakash, R. Arunkumar. "Optimization of hydropower reservoir using evolutionary algorithms coupled with chaos". *Water Resour Manag*, vol. 27, no. 7, pp. 1963–1979, 2013.
3. F.E.R.C. Staff. "Economic Dispatch: Concepts, Practices and Issues". Presentation to the Joint Board for the Study of Economic Dispatch, RC Staff, Palm Springs, California, November 13, 2005.
4. S. Saremi,S. Mirjalili, and A. Lewis. "Grasshopper optimisation algorithm: Theory and application". *Advances in Engineering Software*, vol. 105, pp. 30–47, 2017.
5. N. Wang, L. Liu. "Genetic algorithm in chaos". *OR Trans*, vol. 5, pp. 1–10, 2001.
6. LJ. Yang, TL. Chen. "Application of chaos in genetic algorithms". *Commun Theor Phys* vol. 38, pp. 168–172, 2002.
7. G. Zhenyu, C. Bo, Y. Min, C. Binggang. "Self-Adaptive Chaos Differential Evolution. In: Jiao L., Wang L., Gao X., Liu J., Wu F. (eds) Advances in Natural Computation. ICNC 2006. Lecture Notes in Computer Science, vol. 4221, pp. 972–975, Springer, 2006.
8. D. Yang, G. Li, G. Cheng. "On the efficiency of chaos optimization algorithms for global optimization". Chaos, Solitions & Fractals, vol. 34 pp. 136–1375, 2007.
9. G. Gharooni-fard, F. Moein-darbari, H. Deldari, A. Morvaridi. "Scheduling of scientific workflows using a chaos-genetic algorithm". *Procedia Computer Science*, vol. 1, pp. 1445–1454, 2010.
10. A.H. Gandomi, G.J. Yun, X.S. Yang, S. Talatahari. "Chaos-enhanced accelerated particle swarm algorithm". *Communications in Nonlinear Science and Numerical Simulation*, vol. 18, no. 2, pp. 327–340, 2013.
11. B. Alatas. "Chaotic bee colony algorithms for global numerical optimization". *Expert Systems with Applications*, vol. 37, pp. 5682–5687, 2010.
12. J. Mingjun, T. Huanwen. "Application of chaos in simulated annealing". *Chaos, Solitons & Fractals*, vol. 21, pp. 933–941, 2004.