ISSN 2347-4289

Improved Simulated Annealing With Cobweb Spreading In Continuous Space

Mahdi Imani, Seyede Fatemeh Ghoreishi, Masoud Shariat-Panahi

Department of Electrical and Computer Engineering, University of Tehran. Teran, Iran.
Department of Mechanical Engineering, University of Tehran. Teran, Iran.
Associate Professor of Department of Mechanical Engineering, University of Tehran. Teran, Iran.
Email: f.ghoreishi@ut.ac.ir, Email: m.imani88@ut.ac.ir

ABSTRACT: An innovative approach based on Simulated Annealing method to optimize the problems in continuous space is introduced in this paper. The Simulated Annealing is a popular method which finds the optimum value of functions based on temperature changes during its search. Specifying the trend of temperature change and step length is the most critical issue in the original SA method. The SA algorithm with large or small initial step length cannot reach the optimum value of the function efficiently. The initial temperature and its trend of decrease affect the results of the search in the space. In addition, swarm algorithms like PSO and GA are the powerful methods in optimization. In this paper, a new method called Cobweb Simulated Annealing (CSA) with swarm search in the continuous space is presented. The number of population, temperature and step length are adaptive during the search in this algorithm. The searching points spread to explore the entire search space especially in the first stages. This method is applied to several benchmark functions and the results have shown its reliability and efficiency to find the optimum value of functions in comparison with some powerful modifications of SA. CSA is able to search more extensive area of the whole space with less computational cost. The other significant capability of CSA algorithm is finding more local minimums than other modified algorithms.

Keywords : Simulated Annealing, continuous space, Cobweb Simulated Annealing (CSA), swarm search, adaptive step length and temperature

1 INTRODUCTION

Simulated Annealing (SA) which its name is borrowed from the annealing process of metal is a powerful technique used in optimization of functions versus former techniques in optimization. The SA method has resolved the deficiencies such as not converging to the optimum value in practical number of steps in former methods. Also SA easily deals with ridges and plateaus which were not included in latest methods until SA has appeared due to the fact that it does not have some constraints. The basics of SA methods are first published by Kirkpatrick et al. [1] in 1983. It is based on generalizing the Monte Carlo methods of testing the equation of state and frozen states of a system with n dimensional space. It explores the sample space by moving downhill and uphill to find the optimum value of a test function. The simulated annealing method is frequently used to solve the problems like flow shop sequencing [2], quadratic assignment [3], bandwidth minimization [4], continuous optimization [5], image processing [6], communication systems [7-10], travelling salesman [11,12] and so on. However the SA method has some deficiencies in optimization procedure like: It may be trapped in local minima and does not converge to the best solution in some cases, and it may not find considerable number of local minimums during the search due to its blindness in entire space. In some papers, adaptive parameters were proposed to obviate these deficiencies [13,7]. Most of the modifications of SA algorithm focus on discrete spaces. Latest modification on SA algorithm in a continuous space to reach the best solution in minimum time and cost is called nonu-SA which is released by Zhao Xinchao in 2011 [8]. This algorithm with adaptive neighbourhood borrows dynamic non-uniform mutation operation from evolutionary algorithm (EA) [9]. Non- uniform mutation is one of the operators responsible for both improving fine tuning capabilities of the system and reducing the disadvantage of random mutation. This dynamic operator has the features of searching the space uniformly at early stages and locally at final stages [8]. The optimization methods ha lots of application such as [14]. In this paper, an innovative method (CSA) based on the SA and swarm optimization is introduced. CSA method has the ability to spread through the space adaptively. Instead of choosing one single point in each stage, CSA may choose up to n-point, which n is the dimension of the search space. The other advantage of CSA is that it finds a large percent of local minimum points of function during the searching process. Since knowing the local minimums in addition to the global minimum is practical in some engineering problems, this algorithm with this ability is useful in these problems. In the second section, the paper focuses on the introduction of the proposed CSA method. The definition of parameters and the equations used in CSA are discussed in the third section. Then simulation results of applying CSA and some other modifications of SA on different common benchmark functions are presented and compared. Finally the conclusion is included.

2 CSA ALGORITHM

First of all, SA algorithm and its basic characteristics are presented and then CSA is discussed.

SA algorithm

This algorithm searches the space of the function to find the optimum value through a step by step searching. The initial temperature T_0 and step length are specified in the beginning of the optimization process of function $F(x)$. The algorithm starts with the initial point and the next point in each stage is calculated as follow:

$$X' = X + \rho.R \qquad (1)$$

Where X' and X are the new point and the current point in the n-dimensional space respectively. ρ is a uniformly distributed random number in [-1, 1] and R is the step length. To minimize $F(x)$, if the value of the function in the new point (X') is less than the current value of function, X' will re-

place the X and the algorithm with this new point continues in the same way. If the value of the function in the new point is more than the current value, the X' replaces the X if the random number (P_r) generated in [0,1] is less than p_m which is the probability taken from Metropolis criteria and is defined as:

$$p_m = e^{\frac{f(x')-f(x)}{T_n}} \qquad (2)$$

Where T_n is the temperature of the n^{th} step. If the random number is more than p_m, the algorithm continues its search with the previous point (X). T_n decreases during the searching process to locate the final solution near the optimal value. The decrease of temperature reduces the probability of moving toward the point with higher function value and also causes the method to converge. The initial temperature T_0 must be large enough to ensure exploration of the space to find the optimum value.

CSA algorithm

For papers accepted for publication, it is essential that the electronic version of the manuscript and artwork match This algorithm is based on SA with some modifications explained in the following. In this algorithm, n initial points are chosen stochastically. In each stage n random points are selected in the neighborhood of each of the initial points in the space. Initial points are considered as the current points (ni) and nij is the jth selected point around ith current point. Similar to basic SA method, Xi is replaced with Xij* (the least value function) if its value is less than the function value of Xi, if not, it is replaced with Xij* by the probability in Eq. 2. The best obtained point (X*) is kept and updated during the search. In each stage, in addition to Xij*, the function value of other points ({Xi1, Xi2, …, Xin}-{Xij*}) are compared with the function value of X* and all the points with function values less than X*, are added to the current points of the next step. So the current points of the next stage are both the points obtained by the comparison of the previous points with the generated points in their neighborhood and the points with value functions less than the best value saved in previous stages. In the beginning of each stage, X* is compared with all of the current points and in the case of presence of other point better than X*, it is updated. One advantage of CSA is that it considers the dimension of space to choose the number of random points and also it is capable to spread in the space like a cobweb. These spreading points go forward until there is no point with value function less than the current point in each branch. This algorithm continues till the entire spreading branches stop. Then n points with less function value among all the stopped points are selected and the algorithm starts again. This algorithm continues until the number of function evaluations reaches to the number (N_c) decided by the user. It is worthwhile to mention that in some situations such as simple functions with one minimum, the increasing number of spreading points causes meaningless proliferation of function evaluations, so maximum number of $2\times n$ points in each stage is considered which has been obtained with trial and error.

3. MOVING FROM SA TO CSA

The parameters and formulae used in the proposed algorithm are discussed in this section. - The initial temperature T_0 is calculated by the following equation[8]:

$$T_0 = -\frac{\Delta S_0}{\ln(\chi_0)} \qquad (3)$$

Where χ_0 is the initial acceptance rate for the worse solution. ΔS_0 is the difference between the worse and the optimal solution. - Cooling method used in this paper is as follows:

$$T_{n_c+1} = \frac{T_0}{Ln(1+n_c)} \qquad (4)$$

Where n_c is the number of current function evaluations. The probability of accepting a large function value decreases exponentially toward zero, so the final solution is near optimal when the temperature approaches zero. - To obtain the next points around the current points, one of the dimensions of the current points is selected randomly. The selected dimension x_k' is produced by the following equation and the new generated point is $X_{i,j} = \{x_1, ..., x_k', ..., x_n\}$.

$$x'_k = \begin{cases} x_k + (U_k - x_k)\left(1-\rho^{(1-\frac{n_c}{N_c})^b}\right) & \eta > 0.5 \\ x_k - (x_k - L_k)\left(1-\rho^{(1-\frac{n_c}{N_c})^b}\right) & \eta \le 0.5 \end{cases} \qquad (5)$$

Where η is a random number in [0,1], and U_k, L_k are the upper and lower bounds of the variable x_k respectively, n_c is the number of current function evaluations, N_c is the number decided by the user, ρ is a uniform random number in [0,1] and b is a system parameter determining the degree of dependency on iteration number (non-uniformity) [9]. This adaptive step length allows the algorithm to explore the whole space in initial steps and in the later stages, the step length decreases to prevent the point to distance from the optimum value. The flowchart of the CSA algorithm is demonstrated in Figure 1.

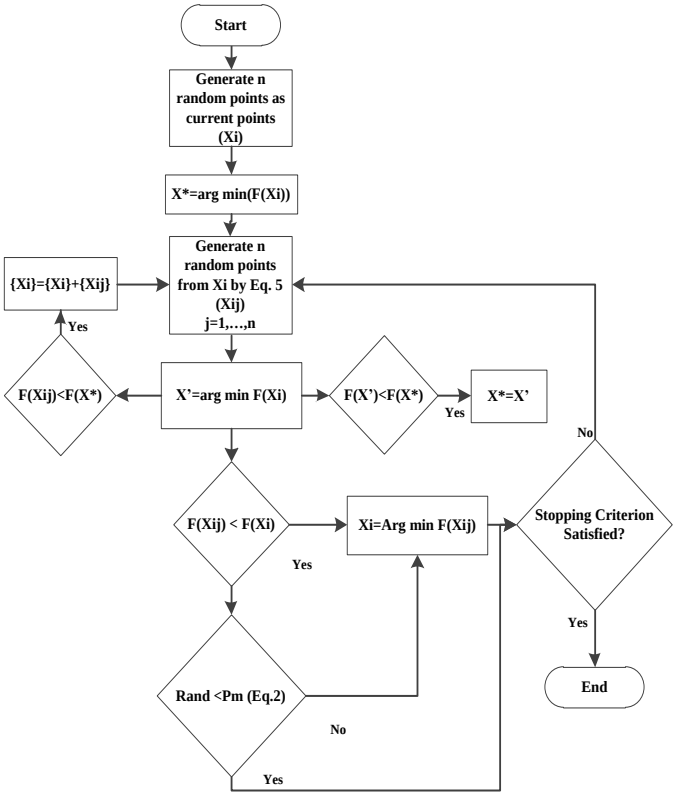

Figure 1 Flowchart of CSA algorithm

4. COMPUTATIONAL RESULTS

The CAS method applied to several benchmark functions ensures the consistency of the method to find the optimum value of functions. This method is also compared with GSA method and nonu-SA method which are powerful modifications of SA. Figure 2 shows a sample benchmark function which is tested by CSA method. The function is as follows:

$$f(x_1,x_2)=\left\{\sum_{i=1}^{5} i.\cos\left[(i+1)x_1+i\right]\right\}$$
$$\left\{\sum_{i=1}^{5} i.\cos\left[(i+1)x_2+i\right]\right\} \qquad (6)$$
$$+\frac{1}{2}\left[(x_1+0.80032)^2+(x_2+1.42513)^2\right]$$
$$-5<x_1,x_2<5$$

This benchmark function has two global minimum values which are located nearby two local maximums. Figure 3 shows the track of selected points in different branches of CSA method to reach the optimum value of this function from different directions.

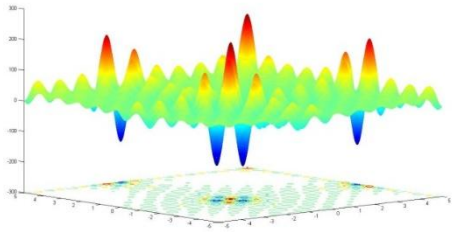

Figure 1. Benchmark Function (Eq. 6)

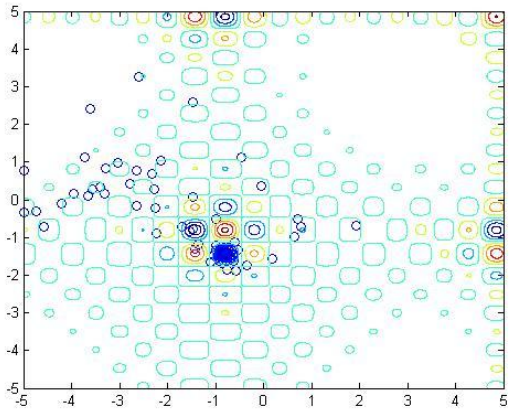

Figure 2. Searching process of CSA method for Benchmark Function (Eq. 6) with Nc = 160

Table 1 shows the results of GSA, nonu-SA and CSA methods applied to benchmark functions in Table 2 in Appendix. The coefficients of f1, f2 and f3 are shown in Tables 3, 4 and 5 respectively. The results are the statistic results over 30 independent runs and they are obtained for the same number of function Evaluations for all three methods. As it is shown in the table, CSA is able to reach the lowest value function in the same number of evaluations. Average of all 30 runs in each algorithm shows that CSA can find the value nearby global minimum more than other methods. For example in the benchmark function Eq. 6 the results show that the CSA method reaches around the global minimum point in more than 86% of runs while the SA reaches this point just in 39% of runs. As is clear from the results, the CSA method with the abilities to spread in the space and also by considering the dimension of space can explore a large portion of the function space. So, in this method the dimension of problem is considered as an effective parameter. This allows the algorithm to reach the optimum value in less number of evaluations. In addition, using adaptive step length during exploration and exploitation process increases the efficiency and accuracy of the results. Another advantage of CSA algorithm is its ability to find a large number of local minimums during the search and also by keeping the best point in each stage, it adapts its spreading to reach the global optimum.

Table 1. The results of applying GSA, nonu-SA and CSA to 10 benchmark functions over 30 independent runs

GSA			nonu-SA		CSA		No. Function Evaluations
F	Best	Mean	Best	Mean	Best	Mean	
1	-3.21	-3.02	-3.31	-3.24	-3.32	-3.29	2000
2	9.53e-3	2.31e-2	7.13e-4	4.27e-3	5.89e-4	3.03e-3	2000
3	0.998(22)	10.34	0.998(18)	0.998(2)	0.998(01)	0.998(12)	2000
4	64.34	78.23	3.24	9.83	9.72e-2	7.94e-1	4000
5	15.86	23.49	2.43e-2	5.18e-2	9.48e-5	7.98e-4	4000
6	1.78e3	4.15e3	17.54	51.60	0.224	4.73	4000
7	2.84	6.08	6.8e-2	3.42e-1	5.44e-3	1.37e-2	4000
8	2.19e-2	2.948e-2	13.55	39.85	0.23	2.37	5000
9	0.36	0.63	1.84e-6	1.14e-2	4.15e-4	9.55e-3	5000
10	18.84	23.65	2.94e-6	0.68	3.74e-7	6.9e-4	5000
11	1.32	2.24	0.78e-9	3.98e-3	3.4e-10	5.36e-5	5000

5. CONCLUSION

In this paper an innovative method based on SA algorithm and swarm optimization is presented. The noticeable feature of this algorithm is its ability to spread adaptively based on the dimension of the space to cover the entire space during the search. Another advantage of this method is that it uses adaptive Step length which helps the algorithm to escape the local minimums in the first steps of search process and also helps the algorithm to converge to the final solution. Obtaining a large number of local minimums in this method is another feature which is practical in lots of engineering problems. The result of applying this method to some benchmark functions shows great efficiency of CSA in comparison with other modifications.

REFERENCE

[1] Kirkpatrick, S., Vecchi, M.: Optimization by simmulated annealing. science 220(4598), 671-680 (1983).

[2] Ishibuchi, H., Misaki, S., Tanaka, H.: Modified simulated annealing algorithms for the flow shop sequencing problem. European Journal of Operational Research 81(2), 388-398 (1995).

[3] Misevicius, A.: A modified simulated annealing algorithm for the quadratic assignment problem. Informatica 14(4), 497-514 (2003).

[4] Rodriguez-Tello, E., Hao, J.K., Torres-Jimenez, J.: An improved simulated annealing algorithm for bandwidth minimization. European Journal of Operational Research 185(3), 1319-1335 (2008).

[5] Locatelli, M.: Convergence of a simulated annealing algorithm for continuous global optimization. Journal of Global Optimization 18(3), 219-233 (2000).

[6] Carnevali, P., Coletti, L., Patarnello, S.: Image processing by simulated annealing. IBM Journal of Research and Development 29(6), 569-579 (1985).

[7] Salcedo-Sanz, S., Santiago-Mozos, R., Bousoño-Calzón, C.: A hybrid Hopfield network-simulated annealing approach for frequency assignment in satellite communications systems. Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on 34(2), 1108-1116 (2004).

[8] Xinchao, Z.: Simulated annealing algorithm with adaptive neighborhood. Applied Soft Computing 11(2), 1827-1836 (2011).

[9] Zhao, X., Gao, X.S., Hu, Z.C.: Evolutionary programming based on non-uniform mutation. Applied Mathematics and Computation 192(1), 1-11 (2007).

[10] Paik, C., Soni, S.: A simulated annealing based solution approach for the two-layered location registration and paging areas partitioning problem in cellular mobile networks. European journal of operational research 178(2), 579-594 (2007).

[11] Černý, V.: Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. Journal of optimization theory and applications 45(1), 41-51 (1985).

[12] Lin, S., Kernighan, B.W.: An effective heuristic algorithm for the traveling-salesman problem. Operations research 21(2), 498-516 (1973).

[13] Low, C.: Simulated annealing heuristic for flow shop scheduling problems with unrelated parallel machines. Computers & Operations Research 32(8), 2013-2025 (2005).

[14] Tajeddini, M. A., Kebriaei. H, Imani, M.: Bidding Strategy in pay as bid markets by Multi-Agent Reinforcement Learning. The 28th International Power System Conference (PSC2013)

APPENDIX

TABLE 2. MULTIMODAL AND UNIMODAL BENCHMARK FUNCTIONS WITH HIGH AND LOW DIMENSIONS

Benchmark Function	m	D	Minimum Value of Function
1	6	$[0,1]^m$	-3.32
2	4	$[-5,5]^m$	0.0003075
3	2	$[-65.536,65.536]^m$	0.998
4	30	$[-100,100]^m$	0
5	30	$[-1.28,1.28]^m$	0
6	30	$[-30,30]^m$	0
7	30	$[-100,100]^m$	0
8	30	$[-5.12,5.12]^m$	0
9	30	$[-600,600]^m$	0
10	30	$[-32,32]^m$	0
11	30	$[-50,50]^m$	0

TABLE 3. COEFFICIENT OF R.HARTMAN'S FUNCTION (F1)

i	a_{ij} , j=1,…,6						ci	P_{ij} , j=1,…,6					
1	10	3	17	3.5	1.7	8	1	0.1312	0.1696	0.5569	0.0124	0.8283	0.5886
2	0.05	10	17	0.1	8	14	1.2	0.2329	0.4135	0.8307	0.3736	0.1004	0.9991
3	3	3.5	1.7	10	17	8	3	0.2348	0.1415	3552	0.2883	0.3047	0.6650
4	17	8	0.05	10	0.1	14	3.2	0.4047	0.8828	0.8732	0.5743	0.1091	0.381

TABLE 4. COEFFICIENT OF N. KOWALIK'S FUNCTION (F2)

I	1	2	3	4	5	6	7	8	9	10	11
Ai	0.1957	0.1947	0.1735	0.16	0.0844	0.0627	0.0456	0.0342	0.0323	0.0233	0.0246
bi-1	0.025	0.5	1	2	4	6	8	10	12	14	16

TABLE 5. COEFFICIENT OF R.HARTMAN'S FUNCTION (F3)

i	a_{ij} , j=1,…,25										
1	-32	-16	0	16	32	-32	-16	…..	0	16	32
2	-32	-32	-32	-32	-32	-16	-16	…..	32	32	32