Chaotic Brain Storm Optimization Algorithm

Eva Tuba[1], Edin Dolicanin[2], and Milan Tuba[3(✉)]

[1] Faculty of Mathematics, University of Belgrade, Belgrade, Serbia
[2] Department of Technical Sciences, State University of Novi Pazar, Novi Pazar, Serbia
[3] Graduate School of Computer Science, John Naisbitt University, Belgrade, Serbia
tuba@ieee.org

Abstract. Swarm intelligence algorithms are stochastic optimization algorithms that are very successfully used for hard optimization problems. Brain storm optimization is a recent swarm intelligence algorithm that has been proven successful in many applications but is still not researched enough. Many swarm intelligence algorithm have been recently improved by introduction of chaotic maps that better than random sequences contributed to search quality. In this paper we propose an improvement of the brain storm optimization algorithm by introducing chaotic maps. Two one-dimensional chaotic maps were incorporated into the original brain storm optimization algorithm. The proposed algorithms were tested on 15 standard benchmark functions from CEC 2013 and compared to the original brain storm optimization algorithm and particle swarm optimization. Our proposed chaos based methods obtained better results where for this set of benchmark functions circle maps were superior.

Keywords: Brain storm optimization algorithm · BSO · Metaheuristic algorithms · Chaos · Global optimization problems · Swarm intelligence

1 Introduction

Numerous real life problems can be represented as optimization problems where global optimum (minimum or maximum) of the objective function needs to be found. In most cases these optimization problems are hard optimization problems and often highly nonlinear. Standard deterministic algorithms are incapable to find the solution for such problems due to computational complexity and numerous local optima, hence different approaches are needed. In the past decades various stochastic optimization algorithms were proposed. One group of such algorithms are nature inspired algorithms where the idea is to mimic some processes from the nature.

Nature inspired algorithms can be divided into two main categories: evolutionary and swarm intelligence algorithms. Evolutionary algorithms are inspired by the biological evolution processes such as reproduction, recombination, mutation and selection. One of the oldest and the most famous member of this group

H. Yin et al. (Eds.): IDEAL 2017, LNCS 10585, pp. 551–559, 2017.
https://doi.org/10.1007/978-3-319-68935-7_60

is genetic algorithm. Swarm intelligence algorithms are inspired by collective behavior of different agents in the nature. Each individual follows simple rules and interacts with other members of the swarm. These simple agents collectively exhibit remarkable intelligence that is used for solving optimization problems.

Swarm intelligence algorithms are inspired by processes from nature such as ant colonies, animal herding, food harvesting, nesting and others. By mimicking these processes two main parts of the algorithms have to be implemented: exploration and exploitation. Exploitation represents a local search around promising solutions that were found while exploration is global search where better solutions are looked for in different areas of the search space in order to prevent the algorithm to get stacked in some local optimum.

Among the oldest swarm intelligence algorithms are particle swarm optimization (PSO) and ant colony optimization. Consequently numerous different swarm intelligent algorithms have been proposed such as firefly algorithm [10,18,20], fireworks algorithm [13], krill herd algorithm [6,11], and others. Swarm intelligence algorithms have been applied for solving different problems such as traveling salesman problem [19], multilevel thresholding [14], support vector machine optimization [15,16], image registration [17], etc.

Optimization algorithms are constantly being improved by different modifications and hybridization. In recent years advances in theory and applications of chaos have been widely used in numerous fields and one of them is optimization algorithms. Chaotic maps such as circle, Gauss/mouse, logistic, piecewise, sine, sinusoidal and others were introduced in swarm intelligence algorithms instead of some random parameters [4]. For example, parameters of the bat algorithm were replaced by different chaotic maps and the results were compared to the original bat algorithm [5]. Results have shown that some chaotic bat algorithms can outperform the version with random numbers. Chebyshev map was introduced into fruit fly optimization algorithm and modified chaotic version of the algorithm has shown superior and more reliable behavior compared to the original one in [8]. In [7] ten different one-dimensional chaotic maps were used for improving fireworks algorithm. The best performance was when circle maps were used.

In this paper we introduce chaos into recent brain storm optimization algorithm [9] and propose chaos based BSO (CBSO). Since different chaotic maps can lead to different behavior of the optimization algorithm, we proposed two chaotic maps. The proposed algorithm was tested on standard benchmark functions proposed in CEC 2013 and it was favorably compared to the original BSO and the standard PSO.

The rest of the paper is organized as follows. Our proposed brain storm optimization algorithm with chaotic maps is presented in Sect. 2. Simulation results along with comparison with other algorithms are given in Sect. 3. At the end the conclusion and proposition for further work is presented in Sect. 4.

2 Chaos Based Brain Storm Optimization Algorithm

Brain storm optimization algorithm (BSO) was proposed by Yuhui Shi in 2011 [9]. This algorithm has been applied to numerous problems [2,12]. During

the last few years different improved and modified BSO versions were proposed [1,3]. Inspiration for the algorithm was human idea generation process or brainstorming process. Brainstorming was summarized in several steps which were transformed into the brain storm optimization algorithm.

Brainstorming process contains step of generating initial ideas followed by choosing more promising ideas and generating new idea based on better solutions as well as new ones regardless of the previous ideas. It is expected that after a several iteration some good solution will be obtained. Brain storm optimization algorithm is presented in Algorithm 1.

In the Algorithm 1 several parameters of the BSO are mentioned. The first one is n, the number of solutions or individuals in each generation of the ideas and the second one is parameter m which represents the number of clusters. Beside these two parameters, four different parameters need to be set, parameters p_{5a}, p_{6b}, p_{6bi} and p_{6c} that determine how a new solution will be created according to the Algorithm 1.

Algorithm 1. Pseudo-code of the BSO algorithm

Initialization
Randomly generate n potential solutions.
repeat
  Cluster n solutions into m clusters.
  Rank solutions in each cluster and set the best one as cluster center.
  Randomly generate a value r between 0 and 1.
  if $r < p_{5a}$ **then**
    Randomly select a cluster center.
    Randomly generate an individual to replace the selected cluster.
  end if
  repeat
    Generate new solutions.
    Randomly generate a value r between 0 and 1.
    if $r < p_{6b}$ **then**
      Randomly select a cluster with probability p_{6bi}.
      Randomly generate a value r_1 between 0 and 1.
      if $r_1 < p_{6bii}$ **then**
        Select the cluster center and add random values to it to generate new individual.
      else
        Randomly select a solution from the chosen cluster and add random value to the solution to generate new one.
      end if
    else
      Randomly select two clusters.
      Generate random value r_2 between 0 and 1
      if $r_2 < p_{6c}$ **then**
        Two cluster centers are combined to generate new solution.
      else
        Two solutions from each selected cluster are randomly chosen to be combined to generate new individual.
      end if
    end if
    The newly generated solution is compared with the same solution index and the better one is kept.
  until n new solution is generated.
until Maximal iteration number is reached.

New solutions are generated by the following equation:

$$x_{new} = x_{selected} + \zeta * n(\mu, \sigma) \tag{1}$$

where x_{new} is a new solution in the d-dimensional space, $x_{selected}$ represents solution selected to be potentially changed, $n(\mu, \sigma)$ is a random number generated from Gaussian distribution with mean μ and variance σ, while ζ is the coefficient that controls the influence of the Gaussian random value. Parameter ζ is calculated in each generation by the following expression:

$$\zeta = \text{logsig}((0.5 * maxIteration - currentIteration)/k) * rand() \tag{2}$$

where *maxIteration* and *currentIteration* represent maximal number of iterations and the number of the current iteration, respectively. Parameter k changes the logsig() function's slope where logsig is a logarithmic sigmoid transfer function. Finally, *rand*() represents random value from uniform distribution within [0,1]. In this paper we propose using chaotic maps instead of random values.

2.1 Chaotic Maps

Chaotic optimization algorithms are optimization algorithms that use chaotic variables rather then random values. The characteristics of chaotic maps such as non-repetition and ergodicity may improve search in the optimization algorithms [21]. In this paper two different one-dimensional maps were considered: circle map and sinusoidal map. Circle map is defined by the following equation:

$$x_{k+1} = \left[x_k + b - \frac{a}{2\pi}\sin(2\pi x_k)\right] \mod 1 \tag{3}$$

where for $a = 0.5$ and $b = 0.2$ the generated chaotic sequence is within (0, 1).

Sinusoidal map is defined as:

$$x_{k+1} = a x_k^2 \sin(\pi x_k) \tag{4}$$

where for $a = 2.3$ and $x_0 = 0.7$ the following simplified form can be used:

$$x_{k+1} = \sin(\pi x_k) \tag{5}$$

The proposed chaotic maps were used to generate chaos sequence of numbers that were used in Eq. 2.

3 Simulation Results

Our proposed method was implemented in Matlab R2016a and simulations were performed on the platform with Intel ® Core™ i7-3770K CPU at 4 GHz, 8 GB RAM, Windows 10 Professional OS.

The proposed algorithms with chaotic maps were tested on 15 well-known benchmark functions proposed in CEC 2013. We tested on 5 unimodal and 10

multimodal 10-dimensional functions and the details about these functions are presented in Table 1. Parameters for brain storm algorithm were set empirically. Number of individuals was set to 100 and number of the clusters was 5. Probabilities p_{5a}, p_{6b}, p_{6bi} and p_{6c} were set to 0.2, 0.8, 0.4 and 0.5, respectively. Maximal number of iterations was 5000. All tests were run 30 times.

Table 1. Benchmark function details

No	Function	Optimal
	Unimodal functions	
1	Sphere function	−1400
2	Rotated high conditioned elliptic function	−1300
3	Rotated bent cigar function	−1200
4	Rotated discus function	−1100
5	Different powers function	−1000
	Basic multimodal functions	
6	Rotated Rosenbrock's function	−900
7	Rotated Schaffers F7 function	−800
8	Rotated Ackley's function	−700
9	Rotated Weierstrass function	−600
10	Rotated Griewank's function	−500
11	Rastrigin's function	−400
12	Rotated Rastrigin's function	−300
13	Non-Continuous rotated Rastrigin's function	−200
14	Schwefel's Function	−100
15	Rotated Schwefel's Function	100

We compared the original brain storm optimization algorithm with two versions when chaotic maps were introduced. In the first version we used circle map and we named this chaotic brain storm optimization algorithm CBSO-C. The second version had implemented sinusoidal map and it was named CBSO-S. We also compared the results with results of standard PSO algorithm [22]. In [22] maximal number of objective function evaluation was set to 100,000 while in our proposed algorithm it was 50,000 since larger number of evaluations did not improve results. For each function algorithms were executed 30 times and median, standard deviation, the best and the worst solutions were calculated as in [22]. The obtained results are presented in Table 2.

For benchmark function f_1, the sphere, all algorithms successfully found global minimum in all cases since standard deviation was 0. Similarly, all algorithms found the global optimum for function f_5 in almost all cases. The smallest standard deviation was achieved by sinusoidal chaotic BSO. From the results presented in Table 2 it can be seen that PSO achieved the same best solutions in

Table 2. Comparison of PSO, BSO, CBSO-C and CBSO-S

Fun.		PSO	BSO	CBSO-C	CBSO-S
f_1	median	−1.400E+03	−1.400E+03	−1.400E+03	−1.400E+03
	std	0.000E+00	0.000E+00	0.000E+00	0.000E+00
	best	−1.400E+03	−1.400E+03	−1.400E+03	−1.400E+03
	worst	−1.400E+03	−1.400E+03	−1.400E+03	−1.400E+03
f_2	median	3.504E+04	**8.614E+03**	3.769E+05	1.470E+04
	std	7.356E+04	**1.240E+04**	3.054E+05	2.572E+04
	best	7.597E+02	**4.859E+02**	1.808E+05	2.963E+03
	worst	4.755E+05	**2.790E+04**	9.384E+05	6.751E+04
f_3	median	2.670E+05	**5.629E+04**	1.067E+05	7.762E+04
	std	1.656E+07	**3.062E+05**	1.514E+07	6.967E+05
	best	**−1.200E+03**	**−1.200E+03**	3.844E+04	5.471E+03
	worst	8.251E+07	**7.235E+05**	3.505E+07	1.611E+06
f_4	median	7.769E+03	−7.443E+02	−9.728E+02	**−9.953E+02**
	std	4.556E+03	2.078E+02	1.406E+04	**1.818E+02**
	best	2.454E+02	−1.097E+03	2.353E+03	**−1.099E+03**
	worst	1.856E+04	−6.070E+02	3.874E+04	**−6.570E+02**
f_5	median	−1.000E+03	−1.000E+03	−1.000E+03	−1.000E+03
	std	3.142E−05	1.574E−04	2.378E+01	**4.957E−05**
	best	−1.000E+03	−1.000E+03	−1.000E+03	−1.000E+03
	worst	−1.000E+03	−1.000E+03	−9.433E+02	−1.000E+03
f_6	median	−8.902E+02	**−8.997E+02**	−8.953E+02	−8.995E+02
	std	4.974E+00	4.164E+00	7.499E−01	**7.893E−02**
	best	−9.000E+02	**−9.000E+02**	−8.954E+02	−8.996E+02
	worst	−8.898E+02	−8.904E+02	−8.935E+02	**−8.994E+02**
f_7	median	−7.789E+02	−7.349E+02	−7.640E+02	**−7.885E+02**
	std	1.327E+01	2.639E+01	1.976E+01	**4.589E+00**
	best	−7.974E+02	−7.646E+02	−7.723E+02	**−7.987E+02**
	worst	−7.434E+02	−7.036E+02	−7.232E+02	**−7.831E+02**
f_8	median	−6.789E+02	−6.797E+02	**−6.800E+02**	−6.798E+02
	std	6.722E−02	9.599E−02	**1.873E−03**	5.318E−02
	best	−6.789E+02	−6.799E+02	**−6.800E+02**	−6.799E+02
	worst	−6.796E+02	−6.797E+02	**−6.800E+02**	−6.797E+02
f_9	median	−5.952E+02	−5.911E+02	**−5.959E+02**	−5.923E+02
	std	1.499E+00	1.114E+00	1.112E+00	**7.699E−01**
	best	−5.987E+02	−5.936E+02	**−5.989E+02**	−5.926E+02
	worst	−5.929E+02	−5.911E+02	**−5.931E+02**	−5.910E+02

(*continued*)

Table 2. (*continued*)

Fun.		PSO	BSO	CBSO-C	CBSO-S
f_{10}	median	−4.997E+02	**−4.999E+02**	**−4.999E+02**	**−4.999E+02**
	std	2.713E−01	8.620E−02	**1.631E−02**	1.795E−02
	best	−4.999E+02	**−5.000E+02**	**−5.000E+02**	−4.999E+02
	worst	−4.989E+02	−4.998E+02	**−4.999E+02**	−4.998E+02
f_{11}	median	−3.891E+02	−3.582E+02	**−3.940E+02**	−3.473E+02
	std	5.658E+00	8.425E+00	**8.899E−01**	1.777E+01
	best	**−3.970E+02**	−3.682E+02	**−3.970E+02**	−3.781E+02
	worst	−3.731E+02	−3.473E+02	**−3.940E+02**	−3.353E+02
f_{12}	median	−2.861E+02	−2.552E+02	**−2.920E+02**	−2.682E+02
	std	6.560E+00	2.464E+01	**2.725E+00**	1.054E+01
	best	**−2.970E+02**	−2.881E+02	**−2.970E+02**	−2.731E+02
	worst	−2.682E+02	−2.323E+02	**−2.891E+02**	−2.483E+02
f_{13}	median	−1.792E+02	−1.371E+02	**−1.834E+02**	−1.398E+02
	std	9.822E+00	2.110E+01	**5.168E+00**	9.588E+00
	best	−1.946E+02	−1.637E+02	**−1.952E+02**	−1.434E+02
	worst	−1.523E+022	−1.134E+02	**−1.802E+02**	−1.201E+02
f_{14}	median	7.338E+02	1.180E+03	**3.858E+02**	1.093E+03
	std	1.282E+02	1.687E+02	**9.365E+01**	1.444E+02
	best	2.228E+02	8.796E+02	**1.677E+02**	9.702E+02
	worst	1.109E+03	1.340E+03	**5.526E+02**	1.303E+03
f_{15}	median	8.743E+02	9.160E+02	**4.329E+02**	9.160E+02
	std	2.507E+02	**1.409E+02**	2.329E+02	2.323E+02
	best	4.372E+02	7.272E+02	**3.585E+02**	1.136E+03
	worst	1.705E+03	1.124E+03	**8.540E+02**	1.346E+03

the case of benchmark functions f_3, f_{11} and f_{12}. In all other cases standard PSO was outperformed by BSO or our proposed chaos based BSO.

For function f_2 original BSO achieved the best results, however with a large error. It can be said that all algorithms failed to find global optimum. Similarly to f_2, for f_3 again not nearly good solutions were found by any algorithm but BSO managed to find the closest solutions. For functions f_4 and f_7 the best results were achieved when BSO with sinusoidal map was used. For f_4 where optimal solution is -1100, the best found one was -995.3 which is significantly different, but the improvement over the original BSO is noticeable. Original BSO had -744.3 as median in 30 runs which is worse then the results obtained by both chaotic based BSO. Chaotic based BSO improved performance of the original BSO for function f_7, from -734.9 to 764.0 by CBSO-C and -788.5 by CBSO-S.

For functions from f_8 to f_{15} the best solutions were obtained by CBSO-C, chaotic BSO with circle map. Improvements are in some cases smaller, such as for functions f_8 and f_{10}. In the case of f_8 median with the original BSO was -679.7 while this solution is improved by CBSO-S to -679.8 and finally, CBSO-C obtained median -680.0. Function f_{10} was successfully solved by all three BSO versions, but the most robust one was CBCO-C since it had smallest standard deviation. In all other cases chaos based brain storm optimization with circle map improved results of original BSO significantly and also outperformed PSO. For functions f_9–f_{15}, except for the function f_{10}, BSO with circle map obtained results better than the original BSO and also better than BSO with sinusoidal map. This shows that different maps are more suitable for some functions.

4 Conclusion

In this paper chaos based brain storm optimization algorithm was proposed. Two different one-dimensional chaotic maps were implemented into the original BSO: circle and sinusoidal maps. Proposed algorithms were tested on 15 benchmark functions from CEC 2013 and the results were compared to the original BSO and standard PSO. The best results in the most cases were obtained by BSO algorithm with circle map. In other cases the best performance was achieved by the original BSO or BSO with sinusoidal map. BSO as well as two modifications outperformed standard PSO in all cases. In further work different chaotic maps can be used and compared to other chaotic optimization algorithms.

Acknowledgment. This research is supported by the Ministry of Education, Science and Technological Development of Republic of Serbia, Grant No. III-44006.

References

1. Cao, Z., Shi, Y., Rong, X., Liu, B., Du, Z., Yang, B.: Random grouping brain storm optimization algorithm with a new dynamically changing step size. In: Tan, Y., Shi, Y., Buarque, F., Gelbukh, A., Das, S., Engelbrecht, A. (eds.) ICSI 2015. LNCS, vol. 9140, pp. 357–364. Springer, Cham (2015). doi:10.1007/978-3-319-20466-6_38
2. Chen, J., Cheng, S., Chen, Y., Xie, Y., Shi, Y.: Enhanced brain storm optimization algorithm for wireless sensor networks deployment. In: Tan, Y., Shi, Y., Buarque, F., Gelbukh, A., Das, S., Engelbrecht, A. (eds.) ICSI 2015. LNCS, vol. 9140, pp. 373–381. Springer, Cham (2015). doi:10.1007/978-3-319-20466-6_40
3. Chen, J., Wang, J., Cheng, S., Shi, Y.: Brain storm optimization with agglomerative hierarchical clustering analysis. In: Tan, Y., Shi, Y., Li, L. (eds.) ICSI 2016. LNCS, vol. 9713, pp. 115–122. Springer, Cham (2016). doi:10.1007/978-3-319-41009-8_12
4. Gandomi, A., Yang, X.S., Talatahari, S., Alavi, A.: Firefly algorithm with chaos. Commun. Nonlinear Sci. Numer. Simul. **18**(1), 89–98 (2013)
5. Gandomi, A.H., Yang, X.S.: Chaotic bat algorithm. J. Comput. Sci. **5**(2), 224–232 (2014)
6. Gandomi, A.H., Alavi, A.H.: Krill herd: a new bio-inspired optimization algorithm. Commun. Nonlinear Sci. Numer. Simul. **17**(12), 4831–4845 (2012)

7. Gong, C.: Chaotic adaptive fireworks algorithm. In: Tan, Y., Shi, Y., Niu, B. (eds.) ICSI 2016. LNCS, vol. 9712, pp. 515–525. Springer, Cham (2016). doi:10.1007/978-3-319-41000-5_51
8. Mitic, M., Vukovic, N., Petrovic, M., Miljkovic, Z.: Chaotic fruit fly optimization algorithm. Knowl.-Based Syst. **89**, 446–458 (2015)
9. Shi, Y.: Brain storm optimization algorithm. In: Tan, Y., Shi, Y., Chai, Y., Wang, G. (eds.) ICSI 2011. LNCS, vol. 6728, pp. 303–309. Springer, Heidelberg (2011). doi:10.1007/978-3-642-21515-5_36
10. Strumberger, I., Bacanin, N., Tuba, M.: Enhanced firefly algorithm for constrained numerical optimization. In: IEEE Congress on Evolutionary Computation (CEC), pp. 2120–2127. IEEE (2017)
11. Strumberger, I., Bacanin, N., Tuba, M.: Hybridized krill herd algorithm for large-scale optimization problems. In: 15th International Symposium on Applied Machine Intelligence and Informatics (SAMI), pp. 473–478. IEEE (2017)
12. Sun, C., Duan, H., Shi, Y.: Optimal satellite formation reconfiguration based on closed-loop brain storm optimization. IEEE Comput. Intell. Mag. **8**(4), 39–51 (2013)
13. Tan, Y., Zhu, Y.: Fireworks algorithm for optimization. In: Tan, Y., Shi, Y., Tan, K.C. (eds.) ICSI 2010. LNCS, vol. 6145, pp. 355–364. Springer, Heidelberg (2010). doi:10.1007/978-3-642-13495-1_44
14. Tuba, E., Alihodzic, A., Tuba, M.: Multilevel image thresholding using elephant herding optimization algorithm. In: 14th International Conference on Engineering of Modern Electric Systems (EMES), pp. 240–243. IEEE (2017)
15. Tuba, E., Mrkela, L., Tuba, M.: Support vector machine parameter tuning using firefly algorithm. In: 26th International Conference Radioelektronika, pp. 413–418. IEEE (2016)
16. Tuba, E., Tuba, M., Beko, M.: Support vector machine parameters optimization by enhanced fireworks algorithm. In: Tan, Y., Shi, Y., Niu, B. (eds.) ICSI 2016. LNCS, vol. 9712, pp. 526–534. Springer, Cham (2016). doi:10.1007/978-3-319-41000-5_52
17. Tuba, E., Tuba, M., Dolicanin, E.: Adjusted fireworks algorithm applied to retinal image registration. Stud. Inform. Control **26**(1), 33–42 (2017)
18. Tuba, M., Bacanin, N.: Improved seeker optimization algorithm hybridized with firefly algorithm for constrained optimization problems. Neurocomputing **143**, 197–207 (2014)
19. Tuba, M., Jovanovic, R.: Improved ACO algorithm with pheromone correction strategy for the traveling salesman problem. Int. J. Comput. Commun. Control **8**(3), 477–485 (2013)
20. Yang, X.-S.: Firefly algorithms for multimodal optimization. In: Watanabe, O., Zeugmann, T. (eds.) SAGA 2009. LNCS, vol. 5792, pp. 169–178. Springer, Heidelberg (2009). doi:10.1007/978-3-642-04944-6_14
21. Yuan, X., Zhao, J., Yang, Y., Wang, Y.: Hybrid parallel chaos optimization algorithm with harmony search algorithm. Appl. Soft Comput. **17**, 12–22 (2014)
22. Zambrano-Bigiarini, M., Clerc, M., Rojas, R.: Standard particle swarm optimisation 2011 at CEC-2013: a baseline for future PSO improvements. In: IEEE Congress on Evolutionary Computation (CEC), pp. 2337–2344. IEEE (2013)