An Improved Brain Storm Optimization with Learning Strategy

Hong Wang[1,2], Jia Liu[1], Wenjie Yi[1], Ben Niu[1,3(✉)], and Jaejong Baek[3(✉)]

[1] College of Management, Shenzhen University, Shenzhen, China
Drniuben@gmail.com

[2] Department of Mechanical Engineering, Hong Kong Polytechnic University, Hung Hom, Hong Kong

[3] School of Computing, Informatics and Decision Systems Engineering, Arizona State University, Tempe, AZ 85281, USA
jbaek7@asu.edu

Abstract. Brain Storm Optimization (BSO) algorithm is a brand-new and promising swarm intelligence algorithm by mimicking human being's behavior of brainstorming. This paper presents an improved BSO, i.e., BSO with learning strategy (BSOLS). It utilizes a novel learning strategy whereby the first half individuals with better fitness values maintain their superiority by keeping away from the worst ones while other individuals with worse fitness values improve their performances by learning from the excellent ones. The improved algorithm is tested on 10 classical benchmark functions. Comparative experimental results illustrate that the proposed algorithm performs significantly better than the original BSO and standard particle swarm optimization algorithm.

Keywords: Brain Storm Optimization · Improved BSO · Learning strategy · Benchmark functions

1 Introduction

Brain Storm Optimization (BSO) is a swarm intelligence algorithm that simulates the problem-solving process of human brainstorming. The basic framework of BSO was proposed by Shi [1, 2], who designed the clustering and creating operators by mimicking brainstorming process based on Osborn's four rules in 2011.

As a young and promising algorithm, BSO can be further improved by developing various searching strategies. In [3–6], a variety of clustering methods were utilized into BSO instead of k-means clustering to reduce the computational burden of the algorithm. Zhou et al. [7] employed an adaptive step size and generated new individuals in a batch-mode. Krishnanand et al. [8] presented a hybrid algorithm combining BSO and Teaching-Learning-Based Optimization algorithm. Sun et al. [9] designed a closed-loop strategy based BSO (CLBSO) by taking advantage of feedback information and developed three versions of CLBSO. Cao et al. [10] incorporated differential evolution strategy into the creating operator of individuals and introduced a new step size control method. Cheng et al. [11] removed the clustering strategy and divided the solutions into elitist and normal classes. Yadav et al. [12] modified BSO with the inclusion of a mathematical theory called fractional calculus.

In addition, the basic BSO and its variants have been applied successfully to several kinds of real-world problems. Most applications of BSOs focused on electric power systems [8], design problems in aeronautics field [9], wireless sensor networks [4] and optimization problems in finance [13]. However, in evolutionary computation research, there have always been attempts to further improve any given findings. In this paper, we present an improved variant of the BSO algorithm named BSO with learning strategy (BSOLS). In original BSO, new individuals were generated by only one or two individuals. It may trap into local optima easily. The BSOLS implement a novel learning strategy imitating human behavior of seeking benefits and avoiding weakness. The proposed algorithm is tested on 10 benchmark functions, and the results show that the BSOLS algorithm significantly improves the performance of BSO and the diversity of population.

The remaining paper is organized as follows. Section 2 will give a brief introduction of the original BSO algorithm. The proposed learning strategy and the improved algorithm are described in detail in Sect. 3. Simulations on benchmark functions and experimental results are given in Sect. 4. Finally, the conclusions and future works are made in Sect. 5.

2 Original Brain Storm Optimization Algorithm

Derived from the human brainstorming process, Shi [1] first proposed BSO algorithm and gained success. The detailed procedure of BSO can be described as follow:

Step 1: Generate n potential solutions randomly and calculate their fitness values. Step 2: Cluster n solutions into m classifications using k-means clustering method. Then select the best solution as a cluster center in each classification. Step 3: Utilize a newly generated idea in place of a randomly selected cluster center with a small probability P_1 to explore more potential solutions. Step 4: If rand (0, 1) is smaller than P_2, randomly choose one cluster. Otherwise, two randomly chosen clusters are utilized to obtain X_{old}. Based on one cluster, a cluster center or a random solution is selected according to P_3. On the basis of choosing two clusters, combine two cluster centers or two random solutions from selected clusters according to P_4. The combination is defined as

$$X_{old} = rand() \cdot X_1 + (1 - rand()) \cdot X_2 \tag{1}$$

Step 5: Update X_{old} into X_{new} according to

$$X_{new} = X_{old} + \xi \cdot N(\mu, \sigma). \tag{2}$$

Where $N(\mu, \sigma)$ is the Gaussian random value with mean μ and variance σ. ξ is an alterable factor which can be expressed as:

$$\xi = \log sig(\frac{0.5T - t}{k}) \cdot rand(0, 1). \tag{3}$$

Where *logsig* () is a logarithmic sigmoid transfer function, T and t are respectively the maximum and current iteration number, and k is for changing the slope of *logsig* () function. Step 6: Compare the newly generated idea with the previous one and keep the better one as the next iteration of the new information.

The assigned values to the set of parameters of BSO are presented in Table 1.

Table 1. Parameter settings for original BSO

m	P_1	P_2	P_3	P_4	k	μ	σ
5	0.2	0.8	0.4	0.5	20	0	1

3 Brain Storm Optimization with Learning Strategy

In this paper, we propose a novel learning strategy inspired by the learning capacity of humans. In general, normal individuals have a willingness to make progress. However, different individuals have different learning capacity and learning strategy. Some individuals are so outstanding in special fields that what they need to pay attention to is avoiding mistakes. Others perform poorly in one area so that it is required for them to make great efforts for improvement by learning from those individuals who are excellent. Based on the above strategies, we introduce an improved BSO with learning strategy called BSOLS.

The original BSO selects only one idea or two combined idea as X_{old} to generate X_{new}, which may obtain local optima easily. In this paper, we add a learning strategy after updating operator to enhance the population diversity and to jump out of local optima.

All the updated individuals X_{new} are ranked from the best to the worst according to their fitness values to differentiate which ones are better. The top P_e % of individuals will be categorized as "elitists" while the last P_l % will be categorized as "laggards". The first half individuals have already been better so their ideas only need to keep far away from laggards' ideas. While, the second half individuals have great space to improve. So they should learn towards elitists. The learning rule is as follow:

$$X'_{new} = \begin{cases} X_{ranked} - (X_{last} - X_{ranked}) \cdot q_1 \cdot rand() \\ X_{ranked} + (X_{top} - X_{ranked}) \cdot q_2 \cdot rand() \end{cases} \text{if } i < n/2 + 1 \tag{4}$$

Where X'_{new} s is a new idea after learning operator, X_{last} is randomly selected from last P_l percentage ideas, X_{last} is randomly selected from top P_e percentage ideas, q_1 and q_2 are similar to c_2 in PSO which expresses the influence of other individuals.

According to the parameters investigation in [14], the current replacing operator makes less or even no contributions to the BSO. To simplify the algorithm, we remove the replacing operator from BSO. The procedure of the proposed BSO with learning strategy can be described as follows:

(1) The initialization, evaluating, cluster, selecting and updating operator are the same as the basic BSO. While the replacing operator in Step 3 of original BSO is omitted.
(2) Sorting. Sort fitness values for each idea X_{new} in ascending or descending order (It depends on the expected fitness value, maximum or minimum) to obtain X_{ranked}.
(3) Learning. If the index of X_{ranked} is no more than half of total individuals, execute the learning strategy of avoiding weakness. Otherwise, implement the learning strategy of seeking benefits.

The assigned values to the set of parameters of BSOLS are presented in Table 2. In BSOLS, the percentage of elitists and laggards are both set to be 0.1. q_1 and q_2 are equal to 0.13 and 0.15, respectively.

Table 2. Parameter settings for BSOLS

m	P_2	P_3	P_4	k	μ	σ	P_e	P_l	q_1	q_2
5	0.8	0.4	0.5	20	0	1	0.1	0.1	0.13	0.15

4 Benchmark Tests and Experimental Results

4.1 Test Problems

To validate the BSOLS, we use 10 benchmark functions, which have often been used to test population-based algorithms in the literature. All the 10 benchmark functions and their dynamic ranges are from [15], among which the first five functions are unimodal functions and the remaining five functions are multimodal functions. All functions are minimization problems with minimum being zero. Each benchmark function will be tested with three different dimension setting, i.e., 10, 20 and 30, respectively. The proposed BSOLS will be compared with the original BSO and the standard particle swarm optimization (SPSO). To obtain reasonable statistical results, the tested BSO algorithms for each benchmark function will be run 30 times. All the experiments are run under the MATLAB R2014a environment on the same machine with an Intel 2.2 GHz CPU, 4 GB memory. The operating system is Windows 10.

4.2 Parameter Settings

The common parameters for both BSO algorithms are set the same for the purpose of comparison, that is, the population size n is set to be 100, the maximum number of iterations is 2000. The parameters for BSO and BSOLS are given in Tables 1 and 2, respectively. In SPSO, we set c1 = c2 = 2, and inertia weight decrease from 0.9 to 0.5.

4.3 Results and Analysis

The mean fitness values, the minimum and the maximum obtained by the three algorithms are listed in Table 3. The best of all the numerical values obtained on each function are emphasized by using a bold type. Additionally, the average results obtained by the three algorithms with 30 dimensions are visually shown for the example in Fig. 1. The red dotted line, the blue line and the green line are the means of each iteration for running 30 times of BSOLS, BSO and SPSO, respectively.

Table 3. Experiment results on benchmark functions

Function	D	BSOLS			BSO			SPSO		
		Mean	Best	Worst	Mean	Best	Worst	Mean	Best	Worst
f_1 *Sphere*	10	2.88E−54	**7.47E−77**	8.64E−53	3.55E−44	4.88E−45	6.61E−44	**2.42E−59**	4.01E−66	**6.39E−58**
	20	**3.57E−53**	**2.54E−75**	**7.47E−52**	3.11E−43	1.87E−43	4.06E−43	4.75E−26	1.95E−29	4.99E−25
	30	**4.18E−53**	**3.89E−76**	**1.25E−51**	9.58E−43	6.29E−43	1.48E−42	3.10E−14	1.92E−16	3.65E−13
f_2 *Schwefel's P221*	10	**1.81E−31**	**1.06E−38**	**5.29E−30**	1.19E−22	7.91E−23	1.59E−22	3.06E−18	6.80E−21	4.48E−17
	20	**1.43E−31**	**6.77E−40**	**4.15E−30**	2.75E−04	2.66E−12	3.29E−03	5.77E−03	7.55E−04	1.53E−02
	30	**2.38E−33**	**4.39E−40**	**4.00E−32**	6.93E−02	9.77E−03	1.94E−01	2.58E+00	8.51E−01	6.25E+00
f_3 *Step*	10	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
	20	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
	30	**0**	**0**	**0**	3.33E−02	**0**	2.00E+00	**0**	**0**	**0**
f_4 *Schwefel's P222*	10	4.60E−26	**1.75E−37**	5.44E−25	4.68E−22	3.48E−22	7.29E−22	**1.03E−34**	7.81E−37	**7.81E−34**
	20	**3.88E−29**	**3.44E−38**	**8.97E−28**	4.02E−10	1.30E−21	1.19E−08	3.26E−17	1.13E−18	1.84E−16
	30	**8.63E−26**	**1.06E−37**	**1.00E−24**	1.63E−02	4.34E−17	2.79E−02	2.50E−10	2.63E−11	7.94E−10
f_5 *Quartic Noise*	10	**4.42E−05**	**4.42E−06**	**1.63E−04**	4.25E−04	5.92E−05	1.73E−03	1.57E−03	1.99E−04	3.78E−03
	20	**3.76E−05**	**6.26E−07**	**1.26E−04**	2.21E−03	3.38E−04	5.08E−03	7.59E−03	2.39E−03	1.32E−02
	30	**4.58E−05**	**2.82E−06**	**2.50E−04**	1.13E−02	3.44E−03	2.33E−02	2.00E−02	7.08E−03	4.78E−02
f_6 *Ackely*	10	**8.88E−16**	**8.88E−16**	**8.88E−16**	4.44E−15	4.44E−15	4.44E−15	5.39E−15	4.44E−15	7.99E−15
	20	**8.88E−16**	**8.88E−16**	**8.88E−16**	7.16E−15	4.44E−15	1.15E−14	3.89E−14	1.15E−14	1.82E−13
	30	**8.88E−16**	**8.88E−16**	**8.88E−16**	1.52E−14	4.44E−15	2.93E−14	5.36E−08	2.86E−09	1.56E−07
f_7 *Rastrigin*	10	**0**	**0**	**0**	4.44E+00	1.99E+00	7.96E+00	1.03E+00	**0**	3.98E+00
	20	**0**	**0**	**0**	1.96E+01	1.19E+01	3.18E+01	1.07E+01	4.03E+00	1.89E+01
	30	**0**	**0**	**0**	3.59E+01	2.19E+01	5.67E+01	3.05E+01	1.49E+01	4.97E+01
f_8 *Rosenbrock*	10	**0**	**0**	**0**	6.12E+00	4.16E+00	1.23E+01	2.71E+00	8.34E−03	1.16E+01
	20	**0**	**0**	**0**	2.26E+01	1.59E+01	9.18E+01	3.34E+01	1.35E+00	1.11E+02
	30	**0**	**0**	**0**	6.15E+01	2.60E+01	1.24E+02	4.38E+01	4.94E+00	1.20E+02
f_9 *Schwefel's P226*	10	**1.27E−04**	**1.27E−04**	**1.27E−04**	1.35E+03	4.74E+02	2.47E+03	2.21E+02	**1.27E−04**	4.74E+02
	20	**7.90E+01**	**2.55E−04**	2.37E+03	3.32E+03	2.11E+03	4.72E+03	6.75E+02	3.55E+02	**1.30E+03**
	30	**3.82E−04**	**3.82E−04**	**3.82E−04**	5.17E+03	3.57E+03	8.27E+03	1.10E+03	4.74E+02	2.01E+03
f_{10} *Griewank*	10	**0**	**0**	**0**	2.04E+00	7.06E−01	4.29E+00	5.90E−02	2.46E−02	9.83E−02
	20	**0**	**0**	**0**	1.02E−01	**0**	1.44E+00	3.92E−02	**0**	1.20E−01
	30	**0**	**0**	**0**	1.18E−02	1.97E−13	6.63E−02	1.28E−02	7.77E−16	8.60E−02

According to Table 3, we can draw the following conclusions:

BSOLS performs significantly better than the original BSO on all the tested benchmarks, which means the learning strategy is effective in terms of the search accuracy no matter the dimensions and categories of the functions.

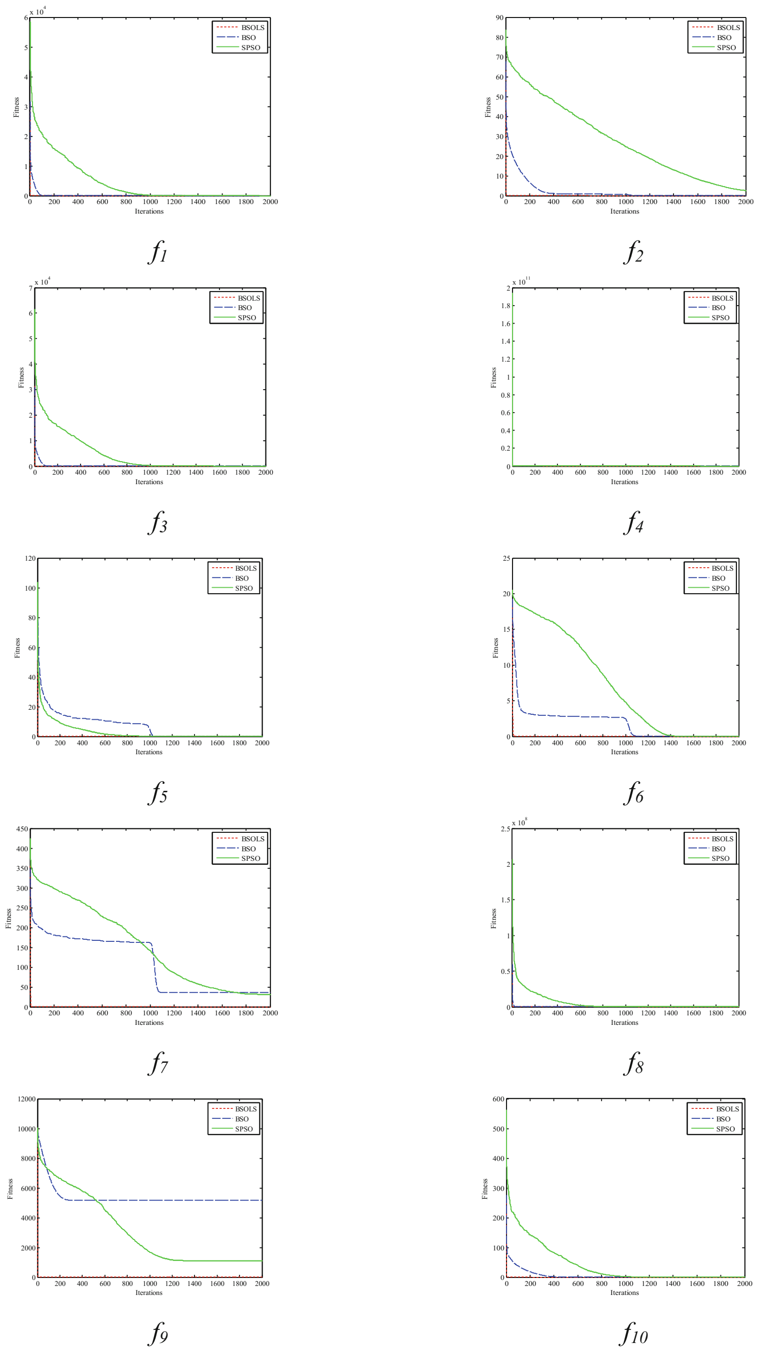

Fig. 1. Convergence Curves of BSOLS, BSO and SPSO (Color figure online)

For f_3, f_7, f_8 and f_{10}, BSOLS can find the optimum zero all the time. It means that the search stability and superiority of BSOLS in specific optimization problem is best as compared to BSO and SPSO.

Although the mean fitness values of BSOLS on f1 and f4 with 10 dimensions are slightly worse than SPSO, BSOLS still performs quite well in terms of the minimum.

We observed visually from Fig. 1 that BSOLS achieves the highest quality of the solution and the most rapid convergence rate. Altogether, whether in unimodal functions or in multimodal functions, it can be observed that BSOLS outperforms the other two algorithms.

5 Conclusions

In this paper, we propose a novel learning strategy and combine it with BSO algorithm. We apply BSOLS, the original BSO and SPSO to a set of benchmark functions to demonstrate the effect of our proposed algorithm. The results on benchmark functions show that the learning strategy significantly improves the performance of the original BSO.

In the future, BSOLS will be compared with more state-of-art algorithms and also be applied to some real-world problems. Moreover, our future work will focus on the development of modified BSO with a lighter computational burden to promote its efficiency.

Acknowledgment. This work is partially supported by The National Natural Science Foundation of China (Grants Nos. 71571120, 71271140, 71471158, 71001072, 61472257), Natural Science Foundation of Guangdong Province (2016A030310074) and Shenzhen Science and Technology Plan (CXZZ20140418182638764).

References

1. 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
2. Shi, Y.: An optimization algorithm based on brainstorming process. Int. J. Swarm Intell. Res. (IJSIR) **2**(4), 35–62 (2011)
3. Zhan, Z.H., Zhang, J., Shi, Y.H., Liu, H.L.: A modified brain storm optimization. In: 2012 IEEE Congress on Evolutionary Computation (CEC), pp. 1–8, June 2012
4. Chen, J., Xie, Y., Ni, J.: Brain storm optimization model based on uncertainty information. In: 2014 Tenth International Conference on Computational Intelligence and Security, pp. 99–103 (2014)
5. Zhu, H., Shi, Y.: Brain storm optimization algorithms with k-medians clustering algorithm. In: Proceedings of the Seventh International Conference on Advanced Computational Intelligence (ICACI 2015), pp. 107–110. IEEE (2015)
6. 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

7. Zhou, D., Shi, Y., Cheng, S.: Brain storm optimization algorithm with modified step-size and individual generation. In: Tan, Y., Shi, Y., Ji, Z. (eds.) ICSI 2012. LNCS, vol. 7331, pp. 243–252. Springer, Heidelberg (2012). doi:10.1007/978-3-642-30976-2_29
8. Krishnanand, K.R., Hasani, S.M.F., Panigrahi, B.K., Panda, S.K.: Optimal power flow solution using self–evolving brain–storming inclusive teaching–learning–based algorithm. In: Tan, Y., Shi, Y., Mo, H. (eds.) ICSI 2013. LNCS, vol. 7928, pp. 338–345. Springer, Heidelberg (2013). doi:10.1007/978-3-642-38703-6_40
9. 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)
10. Cao, Z., Wang, L., Hei, X., Shi, Y., Rong, X.: An improved brain storm optimization with differential evolution strategy for applications of ANNs. Math. Probl. Eng. **2015**, 1–18 (2015)
11. Shi, Y.: Brain storm optimization algorithm in objective space. In: Proceedings of 2015 IEEE Congress on Evolutionary Computation, (CEC 2015), pp. 1227–1234. IEEE, Sendai (2015)
12. Yadav, P.: Case retrieval algorithm using similarity measure and adaptive fractional brain storm optimization for health informaticians. Arab. J. Sci. Eng. **41**, 1–12 (2016)
13. Niu, B., Liu, J., Liu, J., Yang, C.: Brain storm optimization for portfolio optimization. In: Tan, Y., Shi, Y., Li, L. (eds.) ICSI 2016. LNCS, vol. 9713, pp. 416–423. Springer, Heidelberg (2016). doi:10.1007/978-3-319-41009-8_45
14. Zhan, Z.H., Chen, W.N., Lin, Y., Gong, Y.J., Li, Y.L., Zhang, J.: Parameter investigation in brain storm optimization. In: Proceedings of the 2013 IEEE Symposium on Swarm Intelligence (SIS 2013), pp. 103–110 (2013)
15. Yao, X., Liu, Y., Lin, G.: Evolutionary programming made faster. IEEE Trans. Evol. Comput. **3**(2), 82–102 (1999)