An Improved Brain Storm Optimization with Dynamic Clustering Strategy

Zijian Cao , Xiaofeng Rong, Zhiqiang Du

School of Computer Science and Engineering, Xi'an Technological University, Xi'an 710021, China

Abstract: Intelligence algorithms play an increasingly important role in the field of intelligent control. Brain storm optimization (BSO) is a new kind of swarm intelligence algorithm inspired by emulating the collective behavior of human beings in the problem solving process. To improve the performance of the original BSO, many variants of BSO are proposed. In this paper, an improved BSO algorithm with dynamic clustering strategy (BSO-DCS) is proposed as a variant of BSO for global optimization problems. The basic framework of BSO is firstly introduced. Then to reduce the time complexity of the original BSO, a new grouping method named dynamic clustering strategy (DCS) is proposed to improve the clustering method in the original BSO. To verify the effectiveness of the proposed BSO-DCS, it is tested on 12 benchmark functions of CEC 2005 with 30 dimensions. Experimental results show that DCS is an effective strategy to reduce the time complexity, and the improved BSO-DCS performs greatly better than the original BSO algorithm.

1 Introduction

In the past few decades, a lot of swarm intelligence optimization algorithms, such as particle swarm optimization (PSO) [1], ant colony optimization (ACO) [2], bee colony optimization (BCO) [3], firefly optimization algorithms (FFO) [4], bacterial forging optimization (BFO) [5], artificial raindrop algorithm (ARA) [6], have been proposed to tackle increasingly complex real-world optimization problems in the field of intelligent control [7-9].

Brain storm optimization (BSO) is a new kind of swarm intelligence algorithm inspired by emulating the collective behavior of human beings in the problem solving process [10, 11]. Like other swarm intelligence algorithms, BSO has achieved successful applications in areas such as optimal satellite formation reconfiguration [12], the design of DC Brushless Motor [13], economic dispatch considering wind power [14], and multi-objective optimization problems (MOPs) [15-16].

However, in swarm intelligence research, there have always been attempts to further improve the performance of any given findings. In this paper, an improved BSO algorithm with dynamic clustering strategy (BSO-DCS) is proposed as a variant of BSO for global optimization problems. After we firstly introduce the basic framework of BSO, in order to reduce the time complexity of the original BSO, a new grouping method named dynamic clustering strategy (DCS) is proposed to improve the clustering method in the original BSO. To verify the effectiveness of the proposed DCS, it is tested on 12 benchmark functions of CEC 2005 with 30 dimensions.

The rest of this paper is organized as follows. Section 2 introduces briefly the BSO algorithm. Section 3 describes the BSO with a dynamic clustering strategy (BSO-DCS). Section 4 presents the test benchmark functions, the experimental setting for each algorithm and the experimental results. Section 5 discusses the differences between the BSO-DCS and the original BSO algorithm, and conclusions are also given in this Section.

2 Brain storm optimization

The BSO algorithm is motivated by the philosophy of brainstorming. Brainstorming is a widely used tool for increasing creativity in organizations which has achieved wide acceptance as a means of facilitating creative thinking [17]. Similar to other swarm intelligence algorithms, BSO is also a population-based stochastic optimization technique. A potential solution in the fitness landscape is named an idea in BSO. BSO sticks to the rules of interchange of ideas by a team and uses clustering, replacing and creating operators to produce global optimum generation by generation. As presented in [10-11], the process of BSO can be described as follows.

Firstly, N ideas are randomly initialized within the searching space. Then each idea is evaluated and its fitness value is calculated. Next m points of cluster center are also randomly initialized like N ideas, where m is lesser than N.

Next, like other evolutionary algorithms and swarm intelligence algorithms, the main operations will be

divided into converging operation and diverging operation.

2.1 Converging operation

The converging operation primarily is the clustering of ideas. Clustering is a process of grouping similar objects together, and during each generation, all the ideas are clustered into *m* clusters according to idea individual features such as the distance. The best idea in each center is chosen as the cluster center, and the clustering operation can refine a search area. In BSO, the basic method of clustering is *k*-means.

2.2 Diverging operation

The diverging operation mainly includes disrupting cluster center and creating individuals.

2.2.1 Cluster center disrupting

Cluster center disrupting operation randomly chooses an idea in a cluster center and replaces it with a newly-generated idea with a probability of *p_replace*, which is also named as the replacing operation. The value of *p_replace* is utilized to control the probability to replace a cluster center by a randomly generated solution. This can avoid the premature convergence, and help idea individuals "jump out" of the local optima.

2.2.2 Creating individuals operation

To maintain the diversity of population, a new idea individual can be generated based on one idea or two in one cluster or two, respectively. In the creating operation, BSO first randomly chooses one cluster or two according to a probability of *p_one*. Then in the basis of choosing one cluster or two, an idea of cluster center or a random idea is selected with a probability of *p_one_center* and *p_two_center*. The selecting operation is defined below as

$$x_{select} = \begin{cases} x_{ij}, & \text{one cluster} \\ rand * x_{i1,j} + (1-rand) * x_{i2,j}, & \text{two clusters} \end{cases} \quad (1)$$

where *rand* is a random value between 0 and 1. After choosing one idea or two, the selected idea(s) is updated according to as

$$X_{new} = X_{select} + \xi * normrnd(0,1) \quad (2)$$

where *normrnd* is the Gaussian random with mean 0 and variance 1, ξ is an adjusting factor slowing the convergence speed down as the evolution goes, which is expressed as

$$\xi = rand * \log sig(\frac{0.5 * \max\_iteration - current\_iteration}{k}) \quad (3)$$

where *rand* is a random value between 0 and 1. The *max_iteration* and *current_iteration* denote the maximum number of iteration and current number of iteration respectively. The *logsig* is a logarithmic sigmoid transfer function, and such form is beneficial to global search ability at the beginning of the evolution and enhances local search ability when the process is approaching to the end. *k* is a predefined parameter for changing slopes of the *logsig* function. The new created idea is evaluated, and if the fitness value is better than the current idea, the old idea will be replaced by the new one.

3 BSO with dynamic clustering strategy

3.1 Dynamic clustering strategy

In the original BSO, a *k*-means clustering method was adopted to group similar ideas into serval groups in the converging operation. As we all known, the *k*-means clustering method is time consuming and needs heavier time computational burden. During the evolutionary process, BSO executes the *k*-means clustering in every generation to group ideas. However, it is not necessary to use *k*-means clustering method to group the ideas into different groups in every generation.

In our proposed algorithm, the dynamic clustering strategy is used to improve the *k*-means clustering method. The main thinking of the dynamic clustering strategy is that we periodically execute the *k*-means clustering after a certain number of generations (re-clustering period), so that the exchange of information covers all ideas in the clusters to achieve proper exploration ability.

Hence, in our dynamic clustering strategy, the key point is the size of re-clustering period. If the re-clustering period is higher, the algorithm will get a high degree of exploitation and a high convergence rate, and the time complexity of the algorithm will be correspondingly higher. On the contrary, if the re-clustering period is lower, the algorithm will achieve more exploration and diversity, and the run time of the algorithm will be reduced in proportion. So, an appropriate re-clustering period helps to balance the exploitation and exploration of the algorithm, and will achieve better performance than the original BSO algorithm.

In this paper, we use a probabilistic parameter *p_dynamic* to denote the re-clustering period. *p_dynamic* is a value between 0 to 1. According to the above analysis, the pseudo-code for the implementation of dynamic clustering strategy is summarized in Algorithm 1.

Algorithm1: pseudo-code of Dynamic Clustering

```
1  Begin
2     if rand()<p_dynamic
3        execute k-means method
4     end
5  end
```

The setting of the parameter *p_dynamic* will be introduced in detail in Section 4.2.

3.2 Dynamic step size parameter control

For an optimization algorithm, the good balance between exploration and exploitation is comparatively necessary. Exploration refers to the ability of the algorithm to explore

or search different areas of the feasible search space whereas exploitation means the ability of convergence of all the individuals to the near optimal solutions as fast as possible. Too much stress on exploration would result in pure random search, whereas too much stress on exploitation would result in pure local search.

To adjust the convergence speed as the evolution goes in idea generation, the original BSO algorithm defines an adjusting factor ξ described in Formula (3). Through numerical experiments, we find that at first the adjusting factor keeps around 1, while when half the number of generations has been reached, it rapidly turns to near 0. This method to control the size of step can also balance exploration and exploitation at different searching generations. However, it just takes effect only for very short interval. Hence, we introduce a simple dynamic step size strategy. The dynamic function is described as the following

$$\xi = rand * \exp(1 - \frac{\max\_iteration}{\max\_iteration - current\_iteration + 1}) \quad (4)$$

where *rand* is a random value between 0 and 1. The *max_iteration* and *current_iteration* also denote the maximum number of iterations and current number of iteration, respectively.

3.3 Pseudo-code of the BSO-DCS

As described above, the pseudo-code of the improved BSO-DCS is summarized in Algorithm 2.

Algorithm2: BSO-DCS( )

```
Begin
  Randomly initialize N ideas and evaluate their fitness
  Initialize m cluster centers (m<N)
  while(stopping condition not met)
    Execute Algorithm 1: Dynamic Clustering( )
    for (i=1 to N)
      if rand()<p_one
        Randomly select a cluster
        if rand()<p_one_center
          Give the cluster center idea to X_selected
        else
          Randomly select an idea in a cluster to X_selected
        endif
      else
        Randomly select two clusters
        if rand()<p_two_center
          Combine the two clusters' center ideas to X_selected
        else
          Combine two random ideas to X_selected
        endif
      endif
      Create X_new using X_selected with formula (2) and (4)
      Accept X_new if f(X_new) is better than f(Xi )
    endfor
  endwhile
end
```

In Algorithm 2, the main process of BSO-DCS is similar with the original BSO. In converging operation, we improve the *k*-means clustering method with the dynamic clustering strategy, and in creating individuals operation, we adopt a new dynamic step size strategy described in Section 3.2. As a whole, we do not change the main framework of the original BSO for a fair comparison with BSO.

4 Benchmark tests and experimental results

4.1 Benchmark functions

In this paper, we choose 12 generally known shifted and rotated benchmark functions in CEC2005 given in Table 3[18]. All functions are tested on 30 dimensions. The searching range and theory optima for all functions are also given in the Table 1. Among 12 benchmark functions, F1 to F5 are shifted unimodal functions, F6 to F12 are shifted or rotated multimodal functions.

Table 1. Numerical benchmark function definition.

Fun.	Function Name	Range	Optima
F1	Shifted sphere	[-100 100]	0
F2	Shifted Schwefel	[-100 100]	0
F3	Shifted rotated high conditioned elliptic	[-100 100]	0
F4	Shifted Schwefels problem 1.2 with noise	[-100 100]	0
F5	Schwefels problem 2.6	[-100 100]	0
F6	Shifted Rosenbrock	[-100 100]	0
F7	Shifted rotated Griewank	[0 600]	0
F8	Shifted rotated Ackley	[-32 32]	0
F9	Shifted Rastrigin	[-5 5]	0
F10	Shifted rotated Rastrigin	[-5 5]	0
F11	Shifted rotated Weierstrass	[-0.5 0.5]	0
F12	Schwefels problem 2.13	[-100 100]	0

4.2 Parameter settings of comparative algorithms

For a fair comparison, all the experiments are conducted on the same machine with an Intel 3.4 GHz CPU, 4GB memory. The operating system is Windows 7 with MATLAB 8.0 (R2012b).

Compared with the original BSO, Our BSO-DCS algorithm requires a new parameter (p_dynamic) called re-clustering period. Fig.1 shows the variation of convergence performance under different p_dynamic which of values are 0.1, 0.3, 0.5, 0.7 and 0.9 for function F1.

From the Fig.1, we can observe that for function F1, higher re-clustering period leads to higher accuracy. However, the higher the re-clustering period, the more time complexity is required. We set *p_dynamic*=0.5 to get a trade-off between solution accuracy and time complexity.

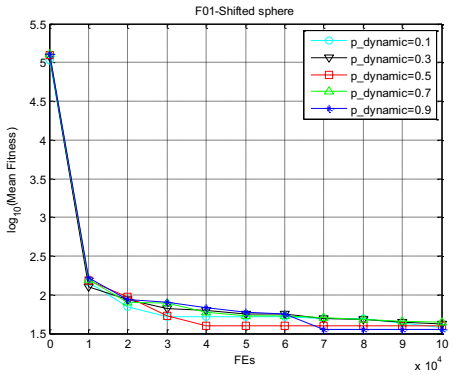


Figure 1. Convergence performance under different *p_dynamic*

Because the BSO algorithm is analogy with the PSO algorithm, we specially compare the performance of BSO-DCS with the PSO. To eliminate influences of statistical errors, each problem function is independently simulated 25 times which is a prescribed evaluation criterion in CEC2005 [18]. For all algorithms, the population size is set to 30. The same stopping criterion is used in all algorithms, i.e. reaching certain number of iterations or function evaluations (FEs).

In our experiments, we have run all algorithms on the benchmark functions using the same FEs of 10000*D for a fair comparison, where D is the size of the problem dimension. The parameter settings of all the algorithms are given in Table 2.

Table 2. The parameter settings.

NO.	Algorithm	Parameter Setting
1	BSO-DCS	M=5,p_one=0.8,p_one_center=0.4, p_two_center=0.5,p_dynamic=0.5
2	BSO	M=5,p_replace=0.2, p_one=0.8, p_one_center=0.4, p_two_center=0.5,K=20
3	PSO	ω: 0.9~0.4, c1= c2=1.49445, global version

4.3 Comparison results

4.3.1 Comparisons on solution accuracy

The results of solution accuracy are given in Table 3 in terms of the mean optimum solution and the standard deviation of the solutions obtained in the 25 independent runs by each algorithm over 300,000 FEs on 12 benchmark functions. In all experiments, the dimensions of all problems are 30. In each row of the table, the mean values are listed in the first part, and the standard deviations are listed in the last part, and the two parts are divided with a symbol "±". The best results among the algorithms are shown in boldface.

Table 3. The results of solution accuracy.

Fun.	BSO-DCS	BSO	PSO
F1	**8.79E-14±9.36E-14**	2.65E-03±3.03E-03	4.67E+03±2.86E+03
F2	**3.50E-04±2.58E-04**	2.76E+00±8.16E-01	6.51E+03±8.15E+03
F3	**8.57E+05±3.69E+05**	1.33E+06±3.58E+05	2.29E+07±4.93E+07
F4	**1.48E-01±1.00E-01**	1.29E+04±9.06E+03	4.56E+03±6.54E+03
F5	**3.03E+04±4.55E+02**	3.12E+04±6.02E+02	3.43E+04±2.28E+03
F6	**4.20E+02±4.92E+02**	1.52E+03±2.71E+03	8.15E+08±9.30E+08
F7	**5.12E+03±1.30E+02**	5.35E+03±2.32E+02	5.36E+03±5.34E+02
F8	**2.02E+01±6.10E-02**	2.05E+01±1.17E-01	2.09E+01±6.56E-02
F9	**3.11E+01±7.30E+00**	4.97E+01±1.02E+01	6.90E+01±2.28E+01
F10	**2.89E+01±6.82E+00**	4.59E+01±1.19E+01	1.26E+02±3.82E+01
F11	**9.99E+00±2.36E+00**	2.43E+01±2.58E+00	2.02E+01±2.59E+00
F12	4.95E+03±4.32E+03	3.16E+04±1.70E+04	**4.83E+03±5.14E+03**
w/t/l	-	12/0/0	11/0/1

The comparison results among BSO-DCS and other algorithms are summarized as "*w/t/l*" in the last row of the Table, which means that BSO-DCS wins in *w* functions, ties in *t* function and loses in *l* functions. From the Table 5 it can be observed that the mean value and the standard deviation value of the BSO-DCS performs better for all 12 functions than the original BSO and PSO. We can conclude that the BSO-DCS algorithm remains a better performance of the solution accuracy for complex shifted and rotated benchmark functions than the original BSO and PSO.

4.3.2 The Comparison results of convergence speed

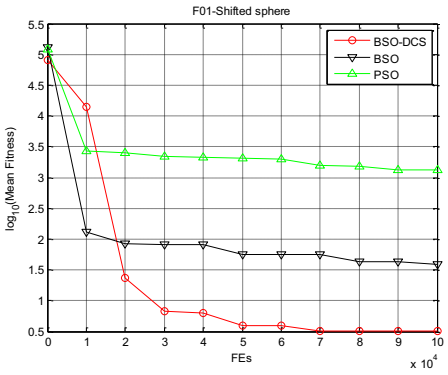
To compare the convergence speed of different algorithms, the mean CPU times of all algorithms in one run are shown in Table 4.

Table 4. The mean CPU times of 3 algorithms.

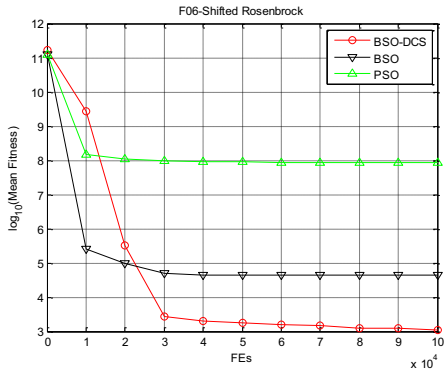
Fun.	BSO-DCS	BSO	PSO
F1	3.31E+02	3.93E+02	3.48E+02
F2	3.46E+02	4.16E+02	3.52E+02
F3	4.76E+02	5.72E+02	4.97E+02
F4	3.59E+02	4.10E+02	3.67E+02
F5	3.40E+02	3.90E+02	3.55E+02
F6	3.34E+02	3.82E+02	3.41E+02
F7	4.75E+02	5.39E+02	5.14E+02
F8	4.76E+02	5.90E+02	5.05E+02
F9	3.33E+02	4.70E+02	3.38E+02
F10	4.75E+02	6.56E+02	4.93E+02
F11	6.19E+02	8.15E+02	6.42E+02
F12	3.83E+02	5.05E+02	3.91E+02
Total	**4.95E+03**	6.14E+03	5.14E+03
Rank	1	3	2

Table 6 shows that the mean CPU times of BSO-DCS are shorter than BSO and PSO for functions F1 to F12.

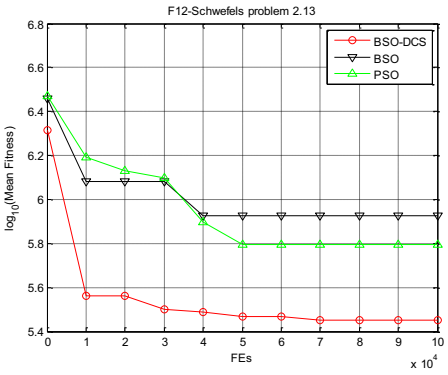
Fig.2 presents the convergence graphs in terms of the mean fitness values achieved by each of 3 algorithms for 25 runs. From the Fig.2 we can observe that BSO-DCS has fast or similar convergence speed than the other two algorithms.



(1) F01-Shifted sphere



(2) F06-Shifted Rosenbrock



(3) F12-Schwefels problem 2.13

Figure 2. Convergence performance on part of functions

5 Discussions and conclusions

There are mainly two differences between the improved BSO-DCS and the original BSO.

Firstly, to reduce the time complexity of the algorithm, the proposed BSO-DCS adopted the dynamic clustering strategy to adjust the *k*-means clustering method. The experimental results show that the mean CPU times of BSO-DCS is lower significantly than the original BSO algorithm.

Besides, we modified the creating individuals' operator with a dynamic step size parameter control strategy. The step size can dynamically adjust the convergence speed as the evolution goes in idea generation, and may overcome the shortage of the original adjusting factor which updates in very short interval. On the other hand, the dynamic step size parameter control strategy also reduces the load of parameter settings, since it is no longer required the parameter *k*.

In this paper, an improved BSO with dynamic clustering strategy named BSO-DCS is proposed for solving complex global optimization problems. Experiments on the 12 chosen test problems were carried out. From the results of experiments, it is observed that the dynamic clustering strategy can greatly reduce the run time of the BSO-DCS algorithm, and with the aid of dynamic step size parameter control, the solution accuracy of the BSO-DCS also is better than the original BSO. From the result analysis, we can conclude that BSO-DCS significantly improves the performance of the original BSO.

We expect that BSO-DCS will be further applied to constrained, uncertain and dynamic, large scale optimization domains. Moreover, the algorithm may be used to solve some real-world intelligent control problems, such as the fuzzy model predictive control of non-linear processes, chaotic time series prediction, etc.

References

1 J. Kennedy, R. C. Eberhart, Particle swarm optimization, Proceedings of IEEE International Conference on Neural Network, pp. 1942-1948, 1995.
2 M. Dorigo, V. Maniezzo, A. Colorni, Ant system: Optimization by a colony of cooperating agents. IEEE Transactions on Systems, Man. Cybernetics B, vol. 26, no. 2, pp. 9-41, Feb. 1996.
3 D. Karaboga, B. Basturk, A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm, Journal of Global Optimization, vol. 39, no. 3, pp. 459-471, Nov. 2007.
4 X. Yang, Nature-inspired metaheuristic algorithms, Beckington, UK: Luniver Press, 2008.
5 K. M. Passino, Bacterial foraging optimization, International Journal of Swarm Intelligence Research, Vol.1, no.1, pp.1-16, 2010.
6 Q. Y. Jiang, L. Wang, X. H. Hei, et al, Optimal appro ximation of stable linear systems with a novel and eff icient optimizaion algorithm, IEEE Congress on Evol utionary Computation, pp. 840-844, Jul. 2014.
7 H. Sarimveis, G. Bafas. Fuzzy model predictive control of non-linear processes using genetic algorithms. Fuzzy Sets and Systems. 2003, 139: 59-80.
8 J. Wang, D. Chi, J. Wu, H. Y. Lu . Chaotic time series method combined with particle swarm optimization and trend adjustment for electricity demand forecasting. Expert Systems with Applications, 38(7): 8419-8429, 2011
9 V. A. Gromov, A. N. Shulga. Chaotic time series prediction with employment of ant colony optimization. Expert Systems with Applications, 39(9): 8474-8478, 2012
10 Y. H. Shi, Brain storm optimization algorithm, The second International Conference on Swarm Intelligence, pp. 303–309, 2011.

11 Y. H. Shi, An optimization algorithm based on brainstorming process, International Journal of Swarm Intelligence Research, vol. 2, no. 4, pp. 35–62, Oct. 2011.
12 C. H. Sun, H. B. Duan, Y. H. Shi, Optimal satellite formation reconfiguration based on closed-loop brain storm optimization, IEEE Computational Intelligence Magazine, vol. 8, no. 4, pp. 39-51, 2013.
13 H. B. Duan, S. T. Li, Y. H. Shi, Predator-prey based brain storm optimization for DC brushless motor. IEEE Transactions on Magnetics, vol. 49, no. 10, pp. 5336-5340, 2013.
14 H. T. Jadhav, U. Sharma, J. Patel, et al, Brain storm optimization algorithm based economic dispatch considering wind power. IEEE International Conference on Power and Energy, pp. 588-593, 2012.
15 Y. H. Shi, Multi-objective optimization based on brain storm optimization algorithm, International Journal of Swarm Intelligence Research, vol. 4, no. 3, pp. 1-21, Jul. 2013.
16 J. Q. Xue, Y. L. Wu, Y. H. Shi, S. Cheng, Brain storm optimization algorithm for multiobjective optimization problems, The third International Conference on Swarm Intelligence, pp. 513-519, 2012.
17 A. F. Osborn, Applied imagination: Principles and procedures of creative problem solving. Charles Scribner's Son, New York, 1963.
18 P. N. Suganthan, N. Hansen, J. J. Liang, et al, Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization, Nanyang Technological University, Singapore, Technical Report, 2005.