BSO-AL: Brain Storm Optimization Algorithm with Adaptive Learning Strategy

Yang Shen*, Jian Yang†, Shi Cheng‡ and Yuhui Shi§

*†§*Department of Computer Science and Engineering*

Southern University of Science and Technology, Shenzhen, China 518055

‡*Shaanxi Normal University, Xi'an, China 710119*

Email: *sheny3@mail.sustech.edu.cn, †yangj38@mail.sustech.edu.cn, ‡cheng@snnu.edu.cn, §shiyh@sustech.edu.cn*

***Abstract*—Swarm intelligence algorithms have been widely and successfully used to optimize many science and engineering problems, the collective behavior of the agents lead to the emergence of intelligence. These interactions among agents can be classified into three categories: exploring, emulating and learning. Brain Storm Optimization (BSO) is a novel swarm intelligence algorithm which is inspired by the human brainstorming process, and generates new ideas by emulating existing ideas. In this paper, a new BSO algorithm with an adaptive learning strategy (BSO-AL) is proposed. By considering the evolutionary speed factor of each individual and the aggregation degree of the swarm, the proposed BSO-AL generates new individuals by exploring, emulating or learning adaptively. Comparative experiments were conducted on a set of benchmark functions with different dimensions. The experimental results show that the proposed BSO-AL algorithm outperforms the classic BSO algorithm and the other two state-of-the-art algorithms, which demonstrates the effectiveness of the learning strategy.**

***Index Terms*—brain storm optimization, adaptive learning, evolutionary speed, aggregation degree**

1. Introduction

In the past a few decades, global optimization has been playing an important role in many different fields of science and engineering. Swarm Intelligence (SI), which is inspired by the collective behavior of biological systems, has attracted extensive attention from researchers. Many nature-inspired swarm intelligence algorithms have been proposed, such as ant colony optimization (ACO) [1], particle swarm optimization (PSO) [2], artificial bee colony (ABC) algorithm [3], bacterial foraging optimization (BFO) [4], firefly optimization (FFO) algorithm [5], etc. The procedure of a unified swarm intelligence algorithm is shown in Fig. 1 [6]. In Fig. 1, the capacity developing is a top-level operator, which describes an algorithm's learning ability to adaptively change its parameters and structures at different search stages. The capability learning is the bottom-level operator, which describes the details how new solutions are generated in the search space.

Corresponding author: Yuhui Shi (e-mail: shiyh@sustech.edu.cn).

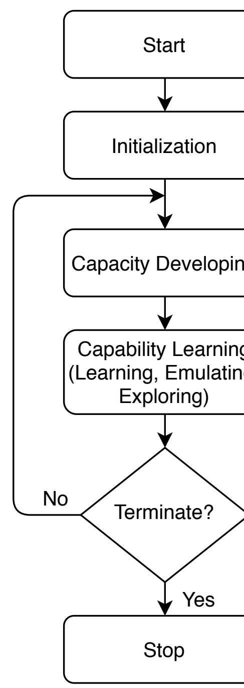

Figure 1. Procedure of a unified swarm intelligence algorithm

Swarm intelligence system is decentralized and self-organized, and each agent follows very simple rules. However, the collective behavior of the agents lead to the emergence of intelligence, which allows the swarm to complete complex missions, i.e., the interactions among agents make significant contributions to swarm intelligence. The interactions among agents can be classified into three categories: exploring, emulating and learning [6]. In other words, when talking about generating new solutions, a good swarm intelligence algorithm should possess all these three strategies, which is shown in Fig 2. Exploring ensures the searching in

978-1-7281-6929-3/20/$31.00 ©2020 IEEE

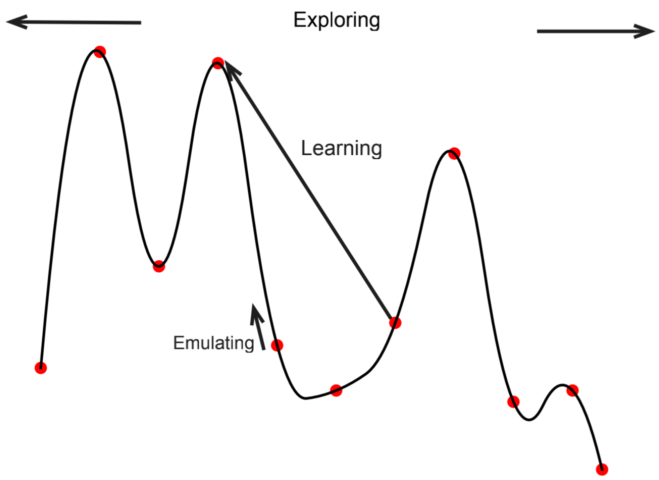

Figure 2. Exploring, emulating and learning in swarm intelligence algorithms

decision space and the diversity of solutions, especially when there is no prior knowledge. For example, random search is the most widely used strategy for exploring. In addition, emulating others can also generate new solutions that could match or surpass the old solutions. Last but not least, learning aims to reduce the distance between the current solution and the object solution, the object solution is usually local or global best solution. Thus, learning ensures that agents would move towards better solutions, and can also ensure the convergence of the SI algorithm. Taking the PSO algorithm as an example, PSO has two main interactions, which are exploring and learning. In general, exploring is obtained by re-initializing, and learning is obtained by updating the particle's velocity and position, which include both inertia, self-cognition and social cognition.

Recently, a novel SI algorithm, i.e., brain storm optimization (BSO) [7], was proposed. BSO is inspired by human brainstorming process, and acts by emulating the collective behavior of human beings to solve problems. Many different variants of BSO have also been proposed to improve the classic BSO algorithm. To reduce the computational cost of the k-means clustering method in the classic BSO algorithm, many other clustering methods were proposed, such as simple grouping method [8], affinity propagation clustering method [9], k-medians clustering algorithm [10], random grouping strategy [11], etc. In addition, in the version of BSO in objective space (BSO-OS) [12], clustering strategy was even removed to solve large scale problems. In the aspect of generating new solutions, many strategies and operations have been proposed, such as adaptive step-size strategy [13], partial re-initialization strategy [14], chaotic operations [15], etc. However, most of the newly proposed variants of BSO algorithms mentioned above did not consider the interaction of learning among individuals like the learning in PSO, the scheme of generating new solutions mainly focus on emulating, i.e., new individuals were generated by emulating one or two selected individuals, which imitates generating new ideas based on someone else's existing ideas in a brainstorming process. Therefore, in original BSOs, there exist two operators, i.e., emulating and exploring, but not learning operator. In fact, during a realistic brainstorming process, new ideas are generated based on not only someone else's ideas, but also his/her own existing ideas with inspiration from other's ideas. As a consequence, a good BSO should consist of all three operators, i.e., learning, emulating, and exploring operators. In [16], individuals were ranked according to their fitness values, a learning strategy was added by letting the individuals learn from the top $P_e\%$ of individuals and keep far away from the last $P_l\%$ of individuals. In [17], an active learning strategy with intra-cluster learning and inter-cluster learning was proposed to improve the original BSO algorithm.

With three operators included in a BSO, it is critical to know how to execute the three operators, for example, in a sequential or parallel order, and with what percentage, which should be considered in the top layer (Capacity Developing) of a meta-heuristic algorithm while the three operators should be considered in the bottom layer (Capability Learning) of the algorithm [6], [18]. In this paper, we propose a new variant of BSO algorithm named BSO with adaptive learning strategy (BSO-AL). The proposed BSO-AL algorithm takes the evolutionary speed of each individual into account, as well as the aggregation degree of the swarm. The decision of whether to emulate or learn is adaptively changed according to the status of the swarm. By doing so, each individual will have both self-cognition and social-cognition, and new individuals could be generated by learning from good individuals in the swarm. Under this situation, individuals would seek more benefits and avoid weaknesses, which is more similar to human behavior.

The rest of this paper is organized as follows. Section 3 briefly introduces the classic BSO algorithm. Section 3 first discusses the evolutionary speed factor and the aggregation degree, and then describes the proposed BSO-AL algorithm. Benchmark functions, parameter settings, and experimental results are given in Section 4. Section 5 concludes the paper and proposes the future work.

2. Brain Storm Optimization Algorithm

Inspired by the human brainstorming process, Shi [7] first proposed the brain storm optimization (BSO) algorithm in 2011. The detailed procedure of the BSO algorithm is described as follows.

- Step 1: Generate n initial solutions randomly;
- Step 2: Cluster n solutions into m clusters by using k-means clustering algorithm;
- Step 3: Evaluate the fitness values of n solutions;

- Step 4: Sort the solutions in each cluster according to their fitness values, and set the best ones as cluster centers;
- Step 5: Generate a random value within $(0, 1)$, if $rand(0, 1) < p_1$, then randomly select a cluster center, and replace it with a randomly generated new solution;
- Step 6: Generate a random value, if $rand(0, 1) < p_2$, then generate a new solution based on a solution, if $rand(0, 1) < p_3$, generate the new solution from a randomly selected cluster center, else generate the new solution from a randomly selected solution; otherwise, if $rand(0, 1) < p_4$, generate the new solution according to two cluster centers, else generate the new solution from two random solutions.
- Step 7: If n new solutions have been generated, go to Step 8; otherwise go to Step 6;
- Step 8: Terminate if the maximum number of iterations has been reached; otherwise, go to Step 2 to run the next iteration.

In the classic BSO algorithm, the new solution is generated by

$$X^d_{new} = X^d_{old} + \xi * n(\mu, \sigma) \tag{1}$$

where X_{new} is the new solution, X_{old} is the selected solution, d means the d-th dimension of the solution, $n(\mu, \sigma)$ is the Gaussian random function with mean value of μ and variance value of σ, and ξ is the coefficient which is defined as

$$\xi = logsig(\frac{0.5 * T - t}{k}) * rand() \tag{2}$$

where $logsig()$ is the logarithmic sigmoid transfer function, T is the pre-defined maximum number of iterations, t is the current number of iterations, k is the slope; and $rand()$ is a random value between 0 and 1.

3. Proposed BSO with Adaptive Learning Strategy

As discussed above, a good swarm intelligence algorithm should possess three components: exploring, emulating, and learning. Classic BSO algorithm only has two components, which are exploring and emulating. As we all know, the PSO algorithm has exploration and learning, the learning in PSO is based on the self-cognition and social cognition of each particle. To address learning, both the status of each individual and the swarm should be considered. In this paper, we consider two factors, which are the evolutionary speed factor of each individual and the aggregation degree of the swarm [19]. These two factors have been widely used in many applications.

The evolutionary speed factor of the individual i at the t-th iteration is defined as [19]

$$h^t_i = \left| \frac{min(F^{t-1}_i, F^t_i)}{max(F^{t-1}_i, F^t_i)} \right| \tag{3}$$

where F^t_i is the fitness value of the i-th individual at the t-th iteration. It can be obtained that $0 \leq h \leq 1$, and h^t_i reflects the evolutionary speed factor of the individual i, the smaller the value of h, the faster the evolutionary speed of the individual i.

The aggregation degree of the swarm is defined as [19]

$$s = \left| \frac{min(F_{tbest}, \overline{F_t})}{max(F_{tbest}, \overline{F_t})} \right| \tag{4}$$

where F_{tbest} is the best fitness value of all individuals at the t-th iteration, and $\overline{F_t}$ is the mean fitness value of all individuals in the swarm at the t-th iteration. It can be obtained that $0 \leq s \leq 1$, the larger the value of s, the more aggregated the swarm is.

For each individual in the swarm, to emulate or to learn should be determined by the dynamic status. The proposed method considers the combination of the two dynamic factors mentioned above, which is defined as

$$w^t_i = f(h^t_i, s) = \alpha * (1 - h^t_i) + \beta * s \tag{5}$$

where α and β are two parameters within the range $[0, 1]$, and $\alpha + \beta = 1$. It can be obtained that initially $w^0_i = 0$ because $h^0_i = 1$ and $s = 0$. With the increase of the number of iterations, w^t_i also increases. Finally, when the algorithm converges, both the evolutionary speed factor of each individual h^T_i and the aggregation degree of the swarm s will approach 1.

For emulation, the $emulating()$ operator is inherited from the classic BSO algorithm in Eq. (1), which adds random noise to existing solutions to generate new solutions. For learning, an individual learns from others with itself as the starting point, i.e., moves towards "good" individuals (global best individual and its cluster center). The $learning()$ operator is defined in Eq. (6).

$$\begin{aligned} X^d_{new} = X^d_{old} &+ r_1 * (gbest^d - X^d_{old}) \\ &+ r_2 * (center^d - X^d_{old}) \end{aligned} \tag{6}$$

where d is the dimension, r_1 and r_2 are random values within $[0, 1]$, $gbest$ is the global best individual, i.e., the one with the best fitness value, and $center$ is the center of the cluster which contains X_{old}. If an individual is learning from one individual, then X_{old} is selected within the cluster, which is either the cluster center or a random individual in the cluster. If an individual is learning from two individuals X_a and X_b, then X_{old} is defined as the linear weighting of X_a and X_b in Eq. (7).

$$X^d_{old} = q * X^d_a + (1 - q) * X^d_b \tag{7}$$

where q is a random value within $[0, 1]$.

Algorithm 1 proposed BSO-AL algorithm

Require: N, number of population; M, number of clusters; T, max iterations
for $i := 1$ to N **do** ▷ init
    randomly generate solution X_i
    evaluate the fitness of X_i
while $t < T$ **do**
    cluster N solutions into M clusters ▷ Clustering
    for $i := 1$ to M **do**
        set solution with best fitness as cluster center
    for $i := 1$ to N **do** ▷ Generating
        if $rand() < p_1$ **then**
            randomly select a cluster C_1
            if $rand() < p_2$ **then**
                $X_i \leftarrow$ cluster center
            else
                $X_i \leftarrow$ random solution in C_1
            compute w_i^t according to Eqs. (3)-(5)
            if $w_i^t > t/T$ **then**
                $X_{new} \leftarrow emulating(X_i)$
            else
                $X_{new} \leftarrow learning(X_i)$
        else
            randomly select two clusters $C1, C2$
            if $rand() < p_3$ **then**
                $X_{i1}, X_{i2} \leftarrow$ two cluster centers
            else
                $X_{i1}, X_{i2} \leftarrow$ random solutions in $C1, C2$
            compute w_i^t according to Eqs. (3)-(5)
            if $w_i^t > t/T$ **then**
                $X_{new} \leftarrow emulating(X_{i1}, X_{i2})$
            else
                $X_{new} \leftarrow learning(X_{i1}, X_{i2})$
        select X_i or X_{new} based on their fitness values
    $t \leftarrow t + 1$

A BSO algorithm with learning has the unified framework shown in Fig. 1, in which the three operators, i.e., learning, emulating, and exploring operator, are in the Capability Learning layer, and Capacity Developing layer determines how the three operators in the Capability Learning layer are organized and in what percentage. There can be different ways to implement the Capacity Developing layer as there are different ways to implement the three operators. In general, initially, all the individuals of the swarm should explore the decision space as much as possible, and finally converge. Thus, the emulation operation should be performed more in the early stage, and the learning operation should be performed more in the late stage in the whole procedure. Since w_i^t is used to represent the status of swarm and changes over iterations, in this paper, we compare it with t/T to determine whether the emulating or the learning operator should be taken, which is one way to implement the Capacity Developing layer.

According to the parameter investigation in BSO [20], the replacing operator in classic BSO has very limited or even no contributions. Thus, in the proposed BSO-AL algorithm, the replacing operator is removed to simplify the algorithm [20]. The pseudocode of the proposed BSO-AL algorithm is shown in Alg. 1, which is one way to implement the unified BSO algorithm.

4. Experiments and Discussions

The effectiveness of the proposed BSO-AL algorithm is tested on a set of benchmark functions, which have been widely used to evaluate metaheuristic algorithms. In this paper, a set of benchmark functions are chosen for evaluation, which are shown as follows. More details and features of the benchmark functions are listed in Table 1.

Sphere function:

$$f(x) = \sum_{i=1}^{n} x_i^2 \tag{8}$$

which is evaluated on $x \in [-100, 100]$, the global minima is $f(x^*) = 0$ at $x^* = (0, \ldots, 0)$.

Rastrigin function:

$$f(x) = 10n + \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i)) \tag{9}$$

which is evaluated on $x \in [-5.12, 5.12]$, the global minima is $f(x^*) = 0$ at $x^* = (0, \ldots, 0)$.

Rosenbrock function:

$$f(x) = \sum_{i=1}^{n} |100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2| \tag{10}$$

which is evaluated on $x \in [-5, 10]$, the global minima is $f(x^*) = 0$ at $x^* = (1, \ldots, 1)$.

Griewank function:

$$f(x) = 1 + \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} \cos(\frac{x_i}{\sqrt{i}}) \tag{11}$$

which is evaluated on $x \in [-600, 600]$, the global minima is $f(x^*) = 0$ at $x^* = (0, \ldots, 0)$.

Apart from the classic benchmark functions, the rotated and more complicated benchmark functions are also tested. The rotated functions are generated by multiplying an orthogonal matrix $\mathbf{M}$, i.e., the new rotated variable $\mathbf{y} = \mathbf{M} * \mathbf{x}$, where $\mathbf{x}$ is the original variable [21]. In this paper, the orthogonal matrix is generated by using Gram-Schmidt orthonormalization method, the new rotated variable $\mathbf{y}$ is used to evaluate the fitness values. The rotated benchmark functions are described as follows.

Rotated Rastrigin function:

$$f(x) = 10n + \sum_{i=1}^{n} (y_i^2 - 10\cos(2\pi y_i)), \mathbf{y} = \mathbf{M} * \mathbf{x} \tag{12}$$

TABLE 1. BENCHMARK FUNCTIONS

Functions	Expressions	Features	Search Range	Global Optima
Sphere	$f(x)=\sum_{i=1}^{n} x_i^2$	unimodal, convex	$[-100, 100]$	$f(x^*)=0$ at $x^*=(0,\dots,0)$
Rastrigin	$f(x)=10n+\sum_{i=1}^{n}(x_i^2-10\cos(2\pi x_i))$	multimodal, non-convex	$[-5.12, 5.12]$	$f(x^*)=0$ at $x^*=(0,\dots,0)$
Rosenbrock	$f(x)=\sum_{i=1}^{n}\lvert 100(x_{i+1}-x_i^2)^2+(1-x_i)^2\rvert$	multimodal, non-convex	$[-5, 10]$	$f(x^*)=0$ at $x^*=(1,\dots,1)$
Griewank	$f(x)=1+\sum_{i=1}^{n}\frac{x_i^2}{4000}-\prod_{i=1}^{n}\cos(\frac{x_i}{\sqrt{i}})$	unimodal, non-convex	$[-600, 600]$	$f(x^*)=0$ at $x^*=(0,\dots,0)$
Rotated Rastrigin	$f(x)=10n+\sum_{i=1}^{n}(y_i^2-10\cos(2\pi y_i))$	multimodal, non-convex	$[-5.12, 5.12]$	$f(x^*)=0$ at $x^*=(0,\dots,0)$
Rotated Rosenbrock	$f(x)=\sum_{i=1}^{n}\lvert 100(y_{i+1}-y_i^2)^2+(1-y_i)^2\rvert$	multimodal, non-convex	$[-2.048, 2.048]$	$f(x^*)=0$ at $x^*=(1,\dots,1)$
Rotated Griewank	$f(x)=1+\sum_{i=1}^{n}\frac{y_i^2}{4000}-\prod_{i=1}^{n}\cos(\frac{y_i}{\sqrt{i}})$	unimodal, non-convex	$[-600, 600]$	$f(x^*)=0$ at $x^*=(0,\dots,0)$

which is evaluated on $x \in [-5.12, 5.12]$, the global minima is $f(x^*)=0$ at $x^*=(0,\dots,0)$.

Rotated Rosenbrock function:

$$f(x)=\sum_{i=1}^{n}\lvert 100(y_{i+1}-y_i^2)^2+(1-y_i)^2\rvert, \mathbf{y}=\mathbf{M}*\mathbf{x} \quad (13)$$

which is evaluated on $x \in [-2.048, 2.048]$, the global minima is $f(x^*)=0$ at $x^*=(1,\dots,1)$.

Rotated Griewank function:

$$f(x)=1+\sum_{i=1}^{n}\frac{y_i^2}{4000}-\prod_{i=1}^{n}\cos(\frac{y_i}{\sqrt{i}}), \mathbf{y}=\mathbf{M}*\mathbf{x} \quad (14)$$

which is evaluated on $x \in [-600, 600]$, the global minima is $f(x^*)=0$ at $x^*=(0,\dots,0)$.

For each benchmark function, three sets of experiments with dimensions $10, 20$ and 40 were conducted over 50 runs each. In this paper, two other state-of-the-art variations of BSO with learning methods are also evaluated, which are BSOLS [16] and ALBSO [17]. Since all the experiments are based on BSO, the shared parameters for different algorithms were taken the same value. For the classic BSO algorithm, the parameter settings are shown in Table 2.

TABLE 2. PARAMETER SETTINGS

n	m	p_1	p_2	p_3	p_4	k	$max\_iteration$	μ	σ
100	5	0.2	0.8	0.4	0.5	20	500	0	1

In Table 2, n is the size of population, m stands for number of clusters, p_1, p_2, p_3 and p_4 are the predefined probabilities used for generating new solutions. For other newly introduced parameters in BSOLS and ALBSO, the same values of parameters in their original papers were taken. For the proposed BSO-AL algorithm, the α and β in Eq. (5) are both set as 0.5.

The proposed algorithm was programmed in C++, and all the experiments were conducted on an Intel i5-6500 CPU@3.60GHz with 16GB RAM. The mean, minimum, maximum fitness values, and the variances are shown in Table 3, the best solutions found were marked with bold fonts.

As it can be observed from Table 3, the proposed BSO-AL algorithm outperforms the classic BSO, the BSOLS and the ALBSO algorithm on all benchmark functions for all different dimensions except the rotated Rosenbrock function on the dimension of 10, which demonstrate the effectiveness of the proposed BSO algorithm with adaptive learning strategy.

Taking the experiments with 40 dimensions as examples, for benchmark functions Rastrigin and Griewank, as well as the rotated Rastrigin and Griewank functions, the proposed BSO-AL algorithm can always find global optima under all dimensions, which means that the search ability and stability of the proposed algorithm perform significantly better than the classic BSO, the BSOLS and the ALBSO algorithms. Additionally, the Rastrigin function is multimodal, while the Griewank function is unimodal, therefore, the proposed BSO-AL algorithm works on optimizing both multimodal and unimodal functions. For the Sphere, Rosenbrock and Schewefel function, as well as the rotated Rosenbrock and Schewefel functions, the mean fitness values of the proposed BSO-AL are much smaller than the classic BSO, the BSOLS, and the ALBSO algorithms. In addition, the minimum values found by the proposed BSO-AL algorithm are very close to the global minima. Last but not least, the proposed BSO-AL algorithm also achieves smaller variances in all experiments, which illustrates the stability of the proposed algorithm.

Intuitively, adding learning to BSO will outperform the classic BSO. However, the results show that the BSOLS and the ALBSO algorithms work worse than the classic BSO algorithm on finding global optima for some experiments. In BSOLS, individuals keep far away from the last $P_l\%$ individuals to avoid weaknesses, and learn from the top $P_e\%$ individuals to improve themselves. It is worth to mention that this learning strategy was added at the end of each iteration from

TABLE 3. Experimental results over 50 runs

Function	Dimension	BSO				BSOLS				ALBSO				BSO-AL			
		Mean	Min	Max	Variance	Mean	Min	Max	Variance	Mean	Min	Max	Variance	Mean	Min	Max	Variance
Sphere	10	1.54E-07	3.19E-12	6.73E-06	8.96E-13	1.17E-05	1.20E-12	5.83E-04	6.65E-09	1.08E+02	2.16E+01	3.18E+02	3.88E+03	**5.31E-100**	**2.39E-143**	**2.66E-98**	**1.38E-197**
	20	2.74E-02	3.76E-10	1.32E+00	3.43E-02	6.86E-02	5.13E-07	1.75E+00	7.13E-02	5.08E+02	2.26E+02	9.10E+02	2.75E+04	**4.85E-82**	**4.21E-124**	**2.37E-80**	**1.10E-161**
	40	2.11E+01	4.26E-02	4.54E+02	5.97E+03	1.10E+02	2.94E-01	6.60E+02	2.92E+04	1.86E+03	7.99E+02	3.38E+03	3.52E+05	**2.94E-73**	**3.32E-104**	**9.98E-72**	**2.24E-144**
Rastrigin	10	6.46E+00	1.93E+00	1.66E+01	8.08E+00	9.63E+00	3.59E+00	1.63E+01	7.38E+00	3.33E+01	1.42E+01	5.39E+01	7.63E+01	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	20	2.96E+01	1.56E+01	5.34E+01	8.72E+01	5.74E+01	3.84E+01	7.56E+01	8.47E+01	1.11E+02	6.25E+01	1.45E+02	2.95E+02	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	40	1.36E+02	5.00E+01	1.95E+02	1.09E+03	1.90E+02	1.18E+02	2.23E+02	3.29E+02	2.82E+02	2.25E+02	3.38E+02	7.26E+02	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
Rosenbrock	10	4.72E-01	3.12E-04	6.46E+00	9.73E-01	1.07E+00	1.01E-03	9.32E+00	3.01E+00	1.36E+03	1.56E+02	3.79E+03	6.97E+05	**3.31E+00**	**8.47E-02**	**8.92E+00**	**7.59E+00**
	20	9.13E+01	3.08E+00	4.05E+02	8.15E+03	1.02E+02	7.74E+00	3.79E+02	5.88E+03	6.44E+03	2.77E+03	1.14E+04	4.32E+06	**1.43E+01**	**3.47E+00**	**1.90E+01**	**1.24E+01**
	40	7.21E+02	1.03E+02	2.10E+03	1.62E+05	6.80E+02	7.60E+01	2.40E+03	1.67E+05	2.77E+04	1.56E+04	5.22E+04	6.00E+07	**3.45E+01**	**2.40E+01**	**3.90E+01**	**7.98E+00**
Griewank	10	1.98E+00	2.14E-01	4.46E+00	7.39E-01	4.29E+00	1.76E+00	1.25E+01	5.09E+00	1.86E+00	1.10E+00	3.97E+00	3.72E-01	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	20	5.53E+00	1.51E+00	1.09E+01	5.94E+00	1.20E+01	3.53E+00	4.10E+01	5.16E+01	5.83E+00	1.93E+00	1.12E+01	4.38E+00	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	40	1.97E+01	1.10E+01	3.18E+01	2.37E+01	3.97E+01	2.04E+01	7.88E+01	1.72E+02	1.83E+01	1.06E+01	3.00E+01	1.71E+01	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
Rotated Rastrigin	10	5.70E+00	1.06E+00	1.12E+01	4.35E+00	1.51E+01	1.95E-01	2.70E+01	3.92E+01	2.91E+01	1.31E+01	4.26E+01	6.22E+01	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	20	1.97E+01	8.03E+00	3.50E+01	3.22E+01	7.31E+01	3.47E+01	9.49E+01	1.86E+02	1.07E+02	6.04E+01	1.40E+02	2.47E+02	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	40	6.31E+01	3.84E+01	1.20E+02	5.01E+02	2.24E+02	1.93E+02	2.66E+02	3.42E+02	2.86E+02	2.33E+02	3.29E+02	5.73E+02	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
Rotated Rosenbrock	10	7.36E+00	2.52E+00	1.41E+01	7.15E+00	**6.16E+00**	**3.03E+00**	**1.07E+01**	**2.91E+00**	1.52E+01	8.12E+00	4.16E+01	3.57E+01	8.18E+00	4.40E+00	9.00E+00	1.25E+00
	20	2.01E+01	1.29E+01	3.15E+01	1.88E+01	1.99E+01	1.24E+01	6.63E+01	5.68E+01	5.64E+01	3.36E+01	1.05E+02	2.09E+02	**1.88E+01**	**1.77E+01**	**1.90E+01**	**1.15E-01**
	40	6.55E+01	4.99E+01	9.46E+01	2.07E+02	1.08E+02	6.27E+01	1.63E+02	1.07E+03	1.54E+02	9.54E+01	2.42E+02	1.45E+03	**3.90E+01**	**3.88E+01**	**3.90E+01**	**2.35E-03**
Rotated Griewank	10	2.03E+00	2.56E-01	4.05E+00	8.82E-01	1.07E+00	4.55E-01	4.25E+00	4.38E-01	1.84E+00	7.62E-01	4.07E+00	4.12E-01	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	20	5.02E+00	1.52E+00	1.09E+01	4.03E+00	6.57E-01	3.21E-02	2.30E+00	1.89E-01	5.34E+00	2.67E+00	1.40E+01	4.45E+00	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	40	2.35E+01	1.40E+01	3.54E+01	3.70E+01	8.84E+00	1.60E+00	1.97E+01	2.75E+01	1.82E+01	1.08E+01	2.68E+01	1.40E+01	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**

the first iteration, after the updating operator in the classic BSO algorithm. At the beginning, the population was initialized randomly in the decision space. If the top $P_e\%$ falls into local optima, or the last $P_l\%$ are located near the global optima but with worse fitness values, this learning strategy may lose its effectiveness. For ALBSO, the results obtained were the worst among all algorithms over 500 iterations. In their original paper, the experiments were conducted over $300,000$ function evaluations with the population size of 30, so each experiment was run over $10,000$ iterations. In ALBSO, all the new individuals were generated by the proposed differential learning strategy, i.e, emulating was removed, which might be the reason why the ALBSO algorithm achieved the worst results over the same iterations comparing to other algorithms.

Performing learning strategy at the beginning may decrease the exploration of the population, make the algorithm converge faster, and more easily fall into local optima. In addition, individuals should do more exploration at the initial stage to ensure the searching in the decision space, as well as the diversity of the population. Therefore, to add learning, the status of the swarm should be considered.

5. Conclusions and Future Work

In this paper, we have addressed the importance of the interactions in swarm intelligence and discussed the three categories of interactions, which are exploring, emulating and learning. Since there are no interactions of learning in the classic BSO algorithm, we proposed a new BSO algorithm with an adaptive learning strategy (BSO-AL). Experiments on a set of benchmark functions were conducted to demonstrate the effectiveness of the proposed algorithm. The obtained experimental results showed that the proposed BSO-AL algorithm outperforms the classic BSO, the BSOLS and the ALBSO algorithms significantly in terms of searching ability and stability, no matter the dimensions and features of the benchmark functions.

Apparently, there are different ways to add a learning strategy to BSO, which should be considered in the Capacity Developing layer of the unified BSO. For our future work, the proposed BSO-AL will be evaluated on more complex benchmark functions, and will be compared with more swarm intelligence algorithms, as well as evolutionary algorithms. Moreover, more learning strategies will be explored and compared.

Acknowledgment

This work is partially supported by National Key R&D Program of China under the Grant No. 2017YFC0804003, National Science Foundation of China under grant number 61761136008 and 61806119, Shenzhen Peacock Plan under Grant No. KQTD2016112514355531, Program for Guangdong Introducing Innovative and Entrepreneurial Teams under grant number 2017ZT07X386, and the Science and Technology Innovation Committee Foundation of Shenzhen under the Grant No. ZDSYS201703031748284.

References

[1] M. Drigo, "The ant system: optimization by a colony of cooperating agents," *IEEE Transactions on Systems, Man, and Cybernetics-Part B*, vol. 26, no. 1, pp. 1–13, 1996.

[2] Y. Shi and R. Eberhart, "A modified particle swarm optimizer," in *1998 IEEE international conference on evolutionary computation proceedings. IEEE world congress on computational intelligence (Cat. No. 98TH8360)*. IEEE, 1998, pp. 69–73.

[3] D. Karaboga and 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, 2007.

[4] K. M. Passino, "Bacterial foraging optimization," *International Journal of Swarm Intelligence Research (IJSIR)*, vol. 1, no. 1, pp. 1–16, 2010.

[5] X.-S. Yang, "Firefly algorithms for multimodal optimization," in *International symposium on stochastic algorithms*. Springer, 2009, pp. 169–178.

[6] Y. Shi, "Unified swarm intelligence algorithms," in *Critical Developments and Applications of Swarm Intelligence*. IGI Global, 2018, pp. 1–26.

[7] ——, “Brain storm optimization algorithm,” in *International conference in swarm intelligence*. Springer, 2011, pp. 303–309.

[8] Z.-h. Zhan, J. Zhang, Y.-h. Shi, and H.-l. Liu, “A modified brain storm optimization,” in *2012 IEEE Congress on Evolutionary Computation*. IEEE, 2012, pp. 1–8.

[9] J. Chen, S. Cheng, Y. Chen, Y. Xie, and Y. Shi, “Enhanced brain storm optimization algorithm for wireless sensor networks deployment,” in *International Conference in Swarm Intelligence*. Springer, 2015, pp. 373–381.

[10] H. Zhu and Y. Shi, “Brain storm optimization algorithms with k-medians clustering algorithms,” in *2015 Seventh International Conference on Advanced Computational Intelligence (ICACI)*. IEEE, 2015, pp. 107–110.

[11] C. Li, D. Hu, Z. Song, F. Yang, Z. Luo, J. Fan, and P. Liu, “A vector grouping learning brain storm optimization algorithm for global optimization problems,” *IEEE Access*, vol. 6, pp. 78 193–78 213, 2018.

[12] Y. Shi, “Brain storm optimization algorithm in objective space,” in *2015 IEEE Congress on evolutionary computation (CEC)*. IEEE, 2015, pp. 1227–1234.

[13] D. Zhou, Y. Shi, and S. Cheng, “Brain storm optimization algorithm with modified step-size and individual generation,” in *International Conference in Swarm Intelligence*. Springer, 2012, pp. 243–252.

[14] S. Cheng, Y. Shi, Q. Qin, T. O. Ting, and R. Bai, “Maintaining population diversity in brain storm optimization algorithm,” in *2014 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2014, pp. 3230–3237.

[15] Z. Yang and Y. Shi, “Brain storm optimization with chaotic operation,” in *2015 Seventh International Conference on Advanced Computational Intelligence (ICACI)*. IEEE, 2015, pp. 111–115.

[16] H. Wang, J. Liu, W. Yi, B. Niu, and J. Baek, “An improved brain storm optimization with learning strategy,” in *International Conference on Swarm Intelligence*. Springer, 2017, pp. 511–518.

[17] Z. Cao and L. Wang, “An active learning brain storm optimization algorithm with a dynamically changing cluster cycle for global optimization,” *Cluster Computing*, pp. 1–17, 2019.

[18] Y. Shi, “Developmental swarm intelligence: Developmental learning perspective of swarm intelligence algorithms,” *International Journal of Swarm Intelligence Research (IJSIR)*, vol. 5, no. 1, pp. 36–54, 2014.

[19] X. Yang, J. Yuan, J. Yuan, and H. Mao, “A modified particle swarm optimizer with dynamic adaptation,” *Applied Mathematics and Computation*, vol. 189, no. 2, pp. 1205–1213, 2007.

[20] Z.-h. Zhan, W.-n. Chen, Y. Lin, Y.-j. Gong, Y.-l. Li, and J. Zhang, “Parameter investigation in brain storm optimization,” in *2013 IEEE symposium on swarm intelligence (SIS)*. IEEE, 2013, pp. 103–110.

[21] J. J. Liang, A. K. Qin, P. N. Suganthan, and S. Baskar, “Comprehensive learning particle swarm optimizer for global optimization of multimodal functions,” *IEEE transactions on evolutionary computation*, vol. 10, no. 3, pp. 281–295, 2006.