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-level operator, which describes the details how new-solutions are generated in the search space.

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
In PSO, the scheme of generating new solutions is mainly based on exploring and learning, 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 $P_1\%$ of individuals and keep far away from the last $P_2\%$ 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 (Capability 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;

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

\[
X_{\text{new}}^d = X_{\text{old}}^d + \xi \cdot n(\mu, \sigma) \tag{1}
\]

where \(X_{\text{new}}\) is the new solution, \(X_{\text{old}}\) 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 \(\mu\) and variance value of \(\sigma\), and \(\xi\) is the coefficient which is defined as

\[
\xi = \text{logsig}(0.5 \cdot T - t) \cdot rand() \tag{2}
\]

where \(\text{logsig()}\) 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 \(\text{rand()}\) 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]

\[
h_i^t = \left| \frac{\min(F_i^{t-1}, F_i^t)}{\max(F_i^{t-1}, F_i^t)} \right| \tag{3}
\]

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

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

\[
s = \frac{\min(F_{\text{best}}, F_i^t)}{\max(F_{\text{best}}, F_i^t)} \tag{4}
\]

where \(F_{\text{best}}\) is the best fitness value of all individuals at the \(t\)-th iteration, and \(F_i^t\) 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

\[
w_i^t = f(h_i^t, s) = \alpha \cdot (1 - h_i^t) + \beta \cdot s \tag{5}
\]

where \(\alpha\) and \(\beta\) are two parameters within the range [0, 1], and \(\alpha + \beta = 1\). It can be obtained that initially \(w_i^0 = 0\) because \(h_i^0 = 1\) and \(s = 0\). With the increase of the number of iterations, \(w_i^t\) also increases. Finally, when the algorithm converges, both the evolutionary speed factor of each individual \(h_i^t\) and the aggregation degree of the swarm \(s\) will approach 1.

For emulation, the \(\text{emulating()}\) 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 \(\text{learning()}\) operator is defined in Eq. (6).

\[
X_{\text{new}}^d = X_{\text{old}}^d + r_1 \cdot (g_{\text{best}}^d - X_{\text{old}}^d) + r_2 \cdot (\text{center}^d - X_{\text{old}}^d) \tag{6}
\]

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

\[
X_{\text{old}}^d = q \cdot X_a^d + (1 - q) \cdot X_b^d \tag{7}
\]

where \(q\) is a random value within [0, 1].
Algorithm 1 proposed BSO-AL algorithm

1: Require: $N$, number of population; $M$, number of clusters; $T$, max iterations
2: for $i := 1$ to $N$ > init
3: randomly generate solution $X_i$
4: evaluate the fitness of $X_i$
5: while $t < T$
6: cluster $N$ solutions into $M$ clusters > Clustering
7: for $i := 1$ to $M$
8: set solution with best fitness as cluster center
9: for $i := 1$ to $N$ > Generating
10: if $rand() < p_1$ then
11: randomly select a cluster $C_1$
12: if $rand() < p_2$ then
13: $X_i \leftarrow$ cluster center
14: else
15: $X_i \leftarrow$ random solution in $C_1$
16: compute $w_i^t$ according to Eqs. (3)-(5)
17: if $w_i^t > t/T$ then
18: $X_{new} \leftarrow$ emulating($X_i$)
19: else
20: $X_{new} \leftarrow$ learning($X_i$)
21: else
22: randomly select two clusters $C1, C2$
23: if $rand() < p_3$ then
24: $X_{i1}, X_{i2} \leftarrow$ two cluster centers
25: else
26: $X_{i1}, X_{i2} \leftarrow$ random solutions in $C1, C2$
27: compute $w_i^t$ according to Eqs. (3)-(5)
28: if $w_i^t > t/T$ then
29: $X_{new} \leftarrow$ emulating($X_{i1}, X_{i2}$)
30: else
31: $X_{new} \leftarrow$ learning($X_{i1}, X_{i2}$)
32: select $X_i$ or $X_{new}$ based on their fitness values
33: $t \leftarrow t + 1$

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:

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

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

Rastrigin function:

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

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

Rosenbrock function:

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

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

Griewank function:

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

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 $M$, i.e., the new rotated variable $y = M \times x$, where $x$ is the original variable [21].

In this paper, the orthogonal matrix is generated by using Gram-Schmidt orthonormalization method, the new rotated variable $y$ is used to evaluate the fitness values. The rotated benchmark functions are described as follows.

Rotated Rastrigin function:

$$f(x) = 10n + \sum_{i=1}^{n} (y_i^2 - 10 \cos(2\pi y_i)), y = M \times x$$
which is evaluated on \( x \in [-5.12, 5.12] \), the global minima is \( f(x^*) = 0 \) at \( x^* = (0, \ldots, 0) \).

Rotated Rosenbrock function:

\[
f(x) = \sum_{i=1}^{n} \left[ 100(y_{i+1} - y_i^2)^2 + (1 - y_i)^2 \right], \quad y = M \ast x \tag{13}
\]

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

Rotated Griewank function:

\[
f(x) = 1 + \sum_{i=1}^{n} \frac{y_i^2}{4000} - \prod_{i=1}^{n} \cos\left(\frac{y_i}{\sqrt{i}}\right), \quad y = M \ast x \tag{14}
\]

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

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

<table>
<thead>
<tr>
<th>( n )</th>
<th>( m )</th>
<th>( p_1 )</th>
<th>( p_2 )</th>
<th>( p_3 )</th>
<th>( p_4 )</th>
<th>( k )</th>
<th>max_iteration</th>
<th>( \mu )</th>
<th>( \sigma )</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>5</td>
<td>0.2</td>
<td>0.8</td>
<td>0.4</td>
<td>0.5</td>
<td>20</td>
<td>500</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

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 \( \alpha \) and \( \beta \) 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 Rastrigin 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 Schwefel function, as well as the rotated Rosenbrock and Schwefel 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 \( P_\% \) individuals to avoid weaknesses, and learn from the top \( P_\% \) individuals to improve themselves. It is worth to mention that this learning strategy was added at the end of each iteration from...
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 \( P \% \) falls into local optima, or the last \( P \% \) 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.


