WCCI 2012 IEEE World Congress on Computational Intelligence
June, 10-15, 2012 - Brisbane, Australia IEEE CEC

A Modified Brain Storm Optimization

Zhi-hui Zhan[1], Jun Zhang[1] (Corresponding Author)[*], Yu-hui Shi[2], and Hai-lin Liu[3]
[1]Department of Computer Science, Sun Yat-sen University, Guangzhou, P. R. China, 510275
[1]Key Laboratory of Digital Life, Ministry of Education, P. R. China
[1]Key Laboratory of Software Technology, Education Dept. of Guangdong Province, P. R. China
*junzhang@ieee.org
[2]Xi'an Jiaotong-Liverpool University, Suzhou, P. R. China, 215123
[3]Faculty of Applied Mathematics, Guangdong University of Technology, Guangzhou, P. R. China

Abstract—**Brain storm optimization (BSO) is a new kind of swarm intelligence algorithm inspired by human creative problem solving process. Human being is the most intelligent organism in the world and the brainstorming process popularly used by them has been demonstrated to be a significant and promising way to create great ideas for problem solving. BSO transplants the brainstorming process in human being into optimization algorithm design and gains successes. BSO generally uses the *grouping*, *replacing*, and *creating* operators to produce ideas as many as possible to approach the problem global optimum generation by generation. In this paper, we propose two novel designs to enhance the conventional BSO performance. The first design of the modified BSO (MBSO) is that it uses a simple grouping method (SGM) in the *grouping* operator instead of the clustering method to reduce the algorithm computational burden. The second design is that MBSO uses a novel idea difference strategy (IDS) in the *creating* operator instead of the Gaussian random strategy. The IDS not only contains open minded element to avoid the ideas being trapped by local optima, but also can match the search environment to create better new ideas for problem solving. Experiments have been conducted to illustrate the effectiveness and efficiency of the MBSO algorithm. Moreover, the contributions of SGM and IDS are investigated to show how and why MBSO can perform better than BSO.**

Keywords- Brain storm optimization (BSO), brainstorming process, global optimization

I. INTRODUCTION

Optimization technique has become a significant and promising tool in solving various scientific and engineering problems nowadays. Especially, a kind of optimization technique named evolutionary computation (EC) algorithm has caused great interests all over the world [1]. EC algorithms are based on population and they search for the problem's global optimum through information sharing to cooperate and/or compete among the individuals. From 1960s, EC paradigms such as genetic algorithm, evolution strategy, and evolutionary programming have been invented and developed [2][3]. Late to 1990s, new EC algorithms like ant colony optimization (ACO) [4][5], particle swarm optimization (PSO) [6][7], differential evolution (DE) [8][9], and estimation of distribution algorithm [10] become popular and have been widely studied in recent years. The characteristics of EC algorithms are that they are adaptable optimization methodologies inspired from mechanisms of biological evolution and behaviors of living organisms [11]. Especially, the kind of EC algorithms like ACO and PSO are also named swarm intelligence (SI) algorithms [12], which are regarded as efficient global optimization techniques for their interesting concepts emulate the swarm behaviors in nature and their promising performance in various application problems. Until now, some different SI algorithms such as ACO, PSO, honey bee optimization (HBO), and bacterial forging optimization (BFO) have been proposed. These SI algorithms emulate the collective behaviors of simple insects or animals like ants, birds, bee, and bacteria. Although these social insects and animals can form good intelligence when they cooperate and collaborate, it is still interesting and promising to develop SI algorithm inspired by the social and swarm behaviors of more intelligent organisms, e.g., the human being. In 2011, Shi proposed a novel SI algorithm name brain storm optimization (BSO) by emulating the swarm behaviors of human beings in the problem solving process [13].

This work was supported in part by the National Science Fund for Distinguished Young Scholars No.61125205, National Natural Science Foundation of China (NSFC) No.61070004, NSFC Joint Fund with Guangdong under Key Project U0835002, and the Scholarship Award for Excellent Doctoral Student Granted by Ministry of Education, China.

BSO is a new kind of SI algorithm and it is intuitive to think that BSO should be superior to other SI algorithm because BSO emulates the most intelligent animal in the world (human being) instead of simple objects such as ants in ACO, birds in PSO, bees in HBO, and bacteria in BFO. The BSO algorithm proposed by Shi is motivated by the following intelligent behaviors [13]: when human beings face a difficult problem which every single person may has difficulty to solve, group persons will get together to brainstorm. These persons are usually with different background and they come together for a brainstorming process, which helps them to interactively collaborate to generate great ideas. This way, the problem can usually be solved with high probability. Shi has successfully designed a BSO by emulating this brainstorming process in human being solving problem and conducted simulation results on two typical benchmark functions to validate its usefulness and effectiveness in solving optimization problems [13].

In the BSO framework proposed by Shi [13], there are two features make the algorithm interesting. One feature is the *grouping* operator that groups all the ideas generated in the current generation into some different groups, and the other feature is the *creating* operator that creates new idea by adding random noise to the idea of one selected group or the ideas of two groups, or the ideas from even more groups. In the implementations of Shi's BSO, the *k*-mean clustering method is used for the first feature and a Gaussian random noise is used for the second feature to create new ideas. Although these implementations can make the BSO algorithm able to work, BSO is now just in its infancy and a lot of work and research are needed. In this paper, we focus on the above two features of BSO to propose a *simple grouping method* (SGM) for the first feature and a novel *idea difference strategy* (IDS) for the second feature. We conduct experiments based on benchmark functions to show that our modified BSO (MBSO) can enhance the performance of BSO, and to investigate the reasons why MBSO can work better than BSO.

The rest of this paper is organized as follows. In Section II, the brainstorming process in human beings and the framework of BSO are presented. Then Section III proposes the MBSO algorithm in details. Section IV uses benchmark functions to compares MBSO against BSO in [13] and some other popular EC algorithms like PSO and DE. Finally, conclusions are summarized in Section V.

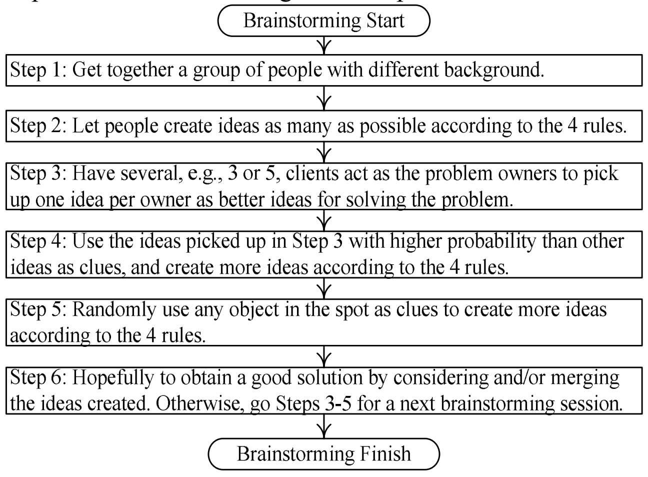
II. Brain Storm Optimization

A. *Brainstorming Process in Human Beings*

Brainstorming is a widely used tool for increasing creativity in organizations which has achieved wide acceptance as a means of facilitating creative thinking. Brainstorming was first developed and coined by Osborn in 1939 in his advertising company. Late in 1957, Osborn systematized this creative problem-solving method in his book *Applied Imagination* [14]. After that, brainstorming has aroused great interest and attention all over the world in both academia and industry [15]. Moreover, this powerful idea generation technique has been further developed in recent years [16].

The brainstorming process is to get together a group of people, especially with different background, to interactively collaborate to generate great ideas for problem solving. A brainstorming process mainly obeys Osborn's four rules to reduce social inhibitions among group members, stimulate idea generation, and increase overall creativity of the group. These four rules are interpreted as "Rule 1: The more ideas, the better", "Rule 2: Withhold criticism for any idea", "Rule 3: Welcome unusual ideas", and "Rule 4: Combine and improve ideas". Based on Osborn's four rules, a typical brainstorming process generally has the steps as shown in Fig. 1 [13].

To execute the steps in Fig. 1, the brainstorming usually requires a free and relaxing brainstorming environment, a facilitator, and something on which to write ideas, such as a white-board. The facilitator's responsibilities include guiding the session, encouraging participation and writing ideas down, but not include generating ideas itself. We can see from Fig. 1 that brainstorming process aims at generating as many ideas as possible, e.g., the Rule 1 and Step 2. This is helpful to maintain the diversity of the ideas, and it is rational to think that the more ideas have been created, the larger chance to come out ideas with good quality. Moreover, all the ideas proposed by anybody won't be criticized until the end of a brainstorming session. This is the Rule 2 which can encourage the participators be open minded as much as possible to speak out whatever they think. Therefore this rule can significantly help to increase the diversity of the ideas. The Rule 3 and Step 5 can also serve for the same purpose to create unusual ideas to increase the diversity to avoid being trapped by previous ideas. At last, the Rule 4 and Step 4 say another important characteristic of the brainstorming process that many later ideas can and should be based on the ideas already created, especially those ideas welcomed by the problem owners. This way, the brainstorming process can progress and gradually approach to the ideal solution. More important, the Step 3 is to cause the brainstorming participators to pay more attention to the better ideas that the problem owners believe to be. This can help to accelerate the idea generation process.


Brainstorming Start

Step 1: Get together a group of people with different background.

Step 2: Let people create ideas as many as possible according to the 4 rules.

Step 3: Have several, e.g., 3 or 5, clients act as the problem owners to pick up one idea per owner as better ideas for solving the problem.

Step 4: Use the ideas picked up in Step 3 with higher probability than other ideas as clues, and create more ideas according to the 4 rules.

Step 5: Randomly use any object in the spot as clues to create more ideas according to the 4 rules.

Step 6: Hopefully to obtain a good solution by considering and/or merging the ideas created. Otherwise, go Steps 3-5 for a next brainstorming session.

Brainstorming Finish

Figure 1. Steps in a typical brainstorming process.

```
Algorithm BSO
Begin
Randomly generate N ideas (Xi, 1<=i<=N) and evaluate their fitness;
While (Not Stop) Do
   Cluster the N ideas into M clusters; //According to the positions
   Record the best idea in each cluster as the cluster center;
   //Probability of replacing a randomly selected cluster center, 0.2
   If (random(0,1)<p_replace)
      Randomly selected a cluster and replace the cluster center
      with a randomly generated idea;
   End of If
   For (i=1 to N)
      //Probability of generating new idea based on one cluster, 0.8
      If (random(0,1)<p_one)
         Randomly select a cluster j with a probability pj;
          //Probability of using the cluster center, p_one_center=0.4
         If  (random(0,1)<p_one_center)
            Add random values to the selected cluster center
            to generate a new idea Yi;
         Else
            Add random values to a random idea of the selected cluster
            to generate a new idea Yi;
         End of If
      Else //Generating new idea based on two clusters
         Randomly select two clusters j1 and j2;
          //Probability of using the cluster center, p_two_center=0.5
         If  (random(0,1)<p_two_center)
            Combine the two selected cluster centers and add with
            random values to generate a new idea Yi;
         Else
            Combine two random ideas from the two selected clusters
            and add with random values to generate a new idea Yi;
         End of If
      End of If
      Evaluate the idea Yi and replace Xi if Yi has better fitness than Xi;
   End of For
End of While
End
```

Figure 2. Pseudocode of the BSO algorithm.

B. Brain Storm Optimization Algorithm

In 2011, Shi invented the BSO algorithm by the inspiration of the brainstorming process in human solving problems [13]. The algorithm is given in Fig. 2 and is described as follows.

In the initialization, a set of N ideas ($\boldsymbol{X_i} = [x_{i1}, x_{i2}, \ldots, x_{iD}]$, where $1 \le i \le N$, N is the population size and D is the problem dimension) are randomly generated as the population. During the evolutionary process, BSO generally uses the *grouping*, *replacing*, and *creating* operators to create new ideas based on the current ideas, so as to improve the ideas generation by generation to approach the problem solution. In the *grouping* operator as lines 4&5 in Fig. 2, BSO uses a k-mean clustering method to group the N ideas into M clusters. However, we will design a simpler grouping method named SGM in this paper instead of the k-mean clustering method to make BSO easier to implement and with lighter computational burden. The *replacing* operator of BSO is given as lines 6-8 in Fig. 2.

In the *creating* operator (line 9 to 26 in Fig. 2), BSO creates N new ideas one by one based on the current ideas. To create a new idea $\boldsymbol{Y_i} = [y_{i1}, y_{i2}, \ldots, y_{iD}]$ ($1 \le i \le N$), BSO first determines whether to create the new idea $\boldsymbol{Y_i}$ based on one selected cluster (lines 10 to 16) or based on two selected clusters (lines 17 to 24). BSO can also use three or more clusters to generate new idea, but would make the algorithm more complex [13]. If use one cluster, the cluster j is selected according to the number of ideas in each cluster by a roulette selection strategy. That is, the more ideas in the cluster, the larger chance it will be selected. The probability of selecting each cluster j is as:

$$p_j = \frac{|M_j|}{N}, \text{ where } |M_j| \text{ is the number of ideas in cluster } j \quad (1)$$

However, if use two clusters, BSO does not select the clusters according to (1), but just randomly select two clusters $j1$ and $j2$ from the M clusters (line 18 in Fig. 2).

After the cluster(s) have been selected, BSO then determines whether create the new idea $\boldsymbol{Y_i}$ base on the cluster center(s) or random idea(s) of the cluster(s). No matter to use the cluster center (lines 13&20) or to use random idea of the cluster (lines 15&22), we can regard the selected based idea as $\boldsymbol{X}$, then the new idea $\boldsymbol{Y_i}$ is created as:

$$y_{id} = x_d + \xi_d \times N(\mu,\sigma)_d \quad (2)$$

where d is the dimension index, $N(\mu,\sigma)$ is the Gaussian random value with mean μ and variance σ, and ξ is a coefficient that weights the contribution of the Gaussian random value, which is calculated as:

$$\xi = \text{logsig}((0.5 \times g - G)/k) \times \text{random}(0,1) \quad (3)$$

where logsig() is a logarithmic sigmoid transfer function whose values are within the range (0,1), g and G are the current generation number and maximum number of generation, k is for changing logsig() function's slope, and random(0, 1) is a random value within (0,1).

There are two issues to be reminded. One is that if y_{id} in (2) is out of the search range, its value should be adjusted. In this paper, we just simple set it as the corresponding boundary value which it exceeds. The other issue is that when create the new idea $\boldsymbol{Y_i}$ based on two clusters, the lines 20 and 22 in Fig. 2 show that the two ideas from the clusters $j1$ and $j2$ first combine themselves and then use Eq. (2) to create $\boldsymbol{Y_i}$. The combination is defined as:

$$\boldsymbol{X} = R \times \boldsymbol{X_1} + (1-\text{R}) \times \boldsymbol{X_2} \quad (4)$$

where R is a random value within (0,1) for all the dimension of $\boldsymbol{X}$, while $\boldsymbol{X_1}$ and $\boldsymbol{X_2}$ are the two ideas from clusters $j1$ and $j2$, respectively.

After the new idea $\boldsymbol{Y_i}$ has been created, BSO evaluates $\boldsymbol{Y_i}$ and replaces $\boldsymbol{X_i}$ if $\boldsymbol{Y_i}$ has a better fitness than $\boldsymbol{X_i}$. This new idea creating process repeats for all the $1 \le i \le N$ to complete a generation. If the termination criteria met, BSO terminates and reports the best idea of the population as the solution. Otherwise, BSO goes to the next generation to repeat the *grouping*, *replacing*, and *creating* processes.

III. Modified Brain Storm Optimization

A. Simple Grouping Method in Grouping Operator

In Shi's BSO [13], a k-mean clustering method was used in the *grouping* operator. However, the k-mean clustering method did make the BSO algorithm more complex for implementation and with heavier computational burden. As the evolutionary process contains stochastic factors during the running time and BSO executes the *grouping* operator in every generation, it is not necessary to use much accurate method like the k-mean clustering method to group the ideas into different groups. In MBSO, the *grouping* operator is implemented by a simple grouping method named SGM as the following steps.

Step 1: Randomly select M different ideas from the current generation as the seeds of the M groups. These M seeds are denoted as $\boldsymbol{S_j}$ ($1 \le j \le M$).

Step 2: For each idea $\boldsymbol{X_i}$ ($1 \le i \le N$) in the current generation, calculate its distance to each group j as:

$$d_{ij} = \|X_i, S_j\| = \sqrt{\left(\sum_{d=1}^{D} (x_{id} - s_{jd})^2\right) / D} \quad (5)$$

Step 3: Compare all the M distance values and assign the idea $\boldsymbol{X_i}$ into the nearest group. Notice that the seed $\boldsymbol{S_j}$ of this group keeps unchanged.

Step 4: Go to Step 2 for the next idea. Otherwise the SGM terminates when all the ideas have been assigned.

B. Idea Difference Strategy in Creating Operator

Refer to Eq. (2), a new idea $\boldsymbol{Y_i}$ is created by adding Gaussian random noise to a based idea $\boldsymbol{X}$. It also can be seen from Eq. (3) that the noise is large in early evolutionary phase and gradually become smaller during the running by the control of the logarithmic sigmoid transfer function. Such a time varying noise strategy is consistent with the commonly intuition that large noises are needed in the early phase for global search while small noises are needed in the late phase for local fine-tuning.

However, Eq. (3) may have two disadvantages. Firstly, the equation uses a fixed logarithmic sigmoid transfer function based on the generation. As the search behaviors of BSO is a stochastic process, the fixed function which takes no feedback information from the search process may not catch the search characteristics well, especially when dealing with different kinds of optimization problems. Secondly, as the logsig() function returns a value within (0,1) and random(0,1) is also a random value within (0,1), their product ξ is still within (0,1). Even though ξ multiplies by a Gaussian random value $N(\mu,\sigma)$ with mean $\mu=0$ and variance $\sigma=1$, this ultimate random noise is with very high probability within the range (-4,4), which may be not efficient enough for global search when the search range is large.

In this paper, we propose to use the IDS to produce the noise value for Eq. (2). The IDS is based on such a consideration. In the human being's brainstorming process, we can image that at the beginning of the process, everyone's idea would be much different. When they create new ideas based on the current ideas, they should take the differences of the current ideas into consideration. For example, when creating a new idea $\boldsymbol{Y_i}$ based on a current idea $\boldsymbol{X_i}$, two distinct random ideas $\boldsymbol{X_a}$ and $\boldsymbol{X_b}$ from all the current ideas are taken to represent the idea difference, and the $\boldsymbol{Y_i}$ is created as:

$$y_{id} = \begin{cases} random(L_d, H_d), & \text{if } random(0,1) < p_r \\ x_{id} + random(0,1)_d \times (x_{ad} - x_{bd}), & \text{otherwise} \end{cases} \tag{6}$$

where d is the dimension index and $1 \le a \ne b \le N$. The p_r in the IF statement is a new parameter to control the open minded element into the idea creation. This emulates the Osborn's Rule 3 (welcome unusual ideas) in the human being's brainstorming process. We will experimentally study this parameter in this paper. Experiments indicate that a value within [0.001, 0.01] would be good. In this paper, we set p_r=0.005.

Using Eq. (6) instead of Eq. (2) to create new ideas, there are two advantages. Firstly, the computational burden of (6) is much lighter than that of (2) for that (2) (including (3)) involves logarithmic sigmoid transfer function, Gaussian distribution function, random function, addition, subtraction, multiplication, and division, while (6) involves random function, multiplication, and subtraction for making up the noise value. Secondly, Eq. (6) can match the search environment of the evolutionary process better than Eq. (2). Be consistent with the brainstorming process for human being in solving problem, the ideas are much different from each other in the beginning, therefore the term $(x_{ad}-x_{bd})$ in (6) is larger and the new created ideas can keep the diversity in the early phase. In the late phase of the brainstorming process, the people may reach a consensus and the idea difference may be smaller. In this condition, the term $(x_{ad}-x_{bd})$ in (6) is also smaller to help refine the ideas. Therefore, Eq. (6) may be good at balance the global search and local search abilities according to the search information during the evolutionary process.

IV. Experimental Studies

A. Functions and Algorithms

Eight benchmark functions listed in Table I are used in the experimental tests [17][18]. In this paper, the functions are divided into two groups, the first group includes 4 unimodal functions and the second group includes 4 complex multimodal functions with a large number of local optima. As EC algorithms are nondeterministic algorithms, we set an 'acceptance' value 1.0×10^{-6} for each test function, except that 0.01 for f_6. If a solution found by an algorithm falls between this value and that of the actual global optimum (0 for all the functions), the solution is judged to be acceptable.

The main purpose of this paper is to improve BSO by using an easy SGM method in the *grouping* operator and designing an efficient IDS strategy in the *creating* operator. Therefore, we compare our MBSO and the BSO algorithm. As BSO is just in its infancy and there exists only one BSO [13] for comparisons, we further adopt two other kinds of popular EC algorithm herein for comparisons. One is the PSO algorithm [19] and another is the DE algorithm [20]. Our purpose to design MBSO is not to propose a powerful algorithm that can beat the elaborately designed algorithms like adaptive PSO [21] or adaptive DE [22], but to show how to improve the BSO algorithm. Therefore, we adopt the conventional PSO and DE algorithms. The parameter settings of all the algorithms are given in Table II.

For a fair comparison among all the algorithms, they use the same maximum number of function evaluations (FEs) 3×10^5 in each run for each test function. For the purpose of reducing statistical errors, each algorithm is tested 30 times independently for every function and the mean results are used in the comparison.

TABLE I. EIGHT TEST FUNCTIONS USED IN THE COMPARISON

	Test function	D	Search Space
Unimodal	$f_1(x)=\sum_{i=1}^{D} x_i^2$	30	$[-100,100]^D$
	$f_2(x)=\sum_{i=1}^{D}\lvert x_i\rvert+\prod_{i=1}^{D}\lvert x_i\rvert$	30	$[-10,10]^D$
	$f_3(x)=\sum_{i=1}^{D}\left(\sum_{j=1}^{i} x_j\right)^2$	30	$[-100,100]^D$
	$f_4(x)=\max(\lvert x_i\rvert), 1\le i\le D$	30	$[-100,100]^D$
Multimodal	$f_5(x)=\sum_{i=1}^{D-1}[100(x_{i+1}-x_i^2)^2+(x_i-1)^2]$	30	$[-10,10]^D$
	$f_6(x)=\sum_{i=1}^{D} -x_i \sin(\sqrt{\lvert x_i\rvert})+418.9829\times D$	30	$[-500,500]^D$
	$f_7(x)=\sum_{i=1}^{D}[x_i^2-10\cos(2\pi x_i)+10]$	30	$[-5.12,5.12]^D$
	$f_8(x)=-20\exp(-0.2\sqrt{1/D\sum_{i=1}^{D} x_i^2})$ $-\exp(1/D\sum_{i=1}^{D}\cos 2\pi x_i)+20+e$	30	$[-32,32]^D$

TABLE II. ALGORITHMS CONFIGURATIONS

Algorithm	Parameters settings	
PSO	ω: 0.9~0.4, c_1= c_2=2.0, $V_{\text{MAX}d}$=0.2×Range, N=40, global version	
DE	F=0.5, CR=0.5, N=50, rand/1 mutation scheme	
BSO [13]	k=20, μ=0, σ=1	*p_replace*=0.2, *p_one*=0.8, N=100, M=5, *p_one_center*=0.4, *p_two_center*=0.5
MBSO	p_r=0.005	

B. Comparisons on Solution Accuracy

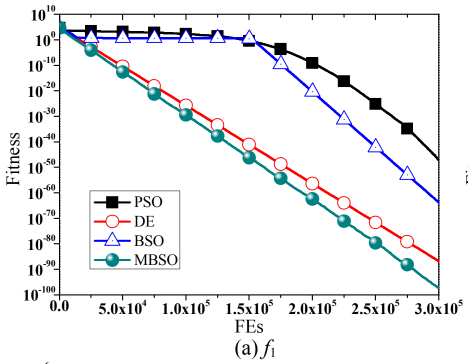
The solutions obtained by each algorithm are compared in Table III. Table III compares the mean values and the standard deviations of the solutions found. The best results are marked with **boldface**. The t-test results between MBSO and other algorithms are also given.

The results show that MBSO does better than BSO on all the functions except f_4. For the four unimodal functions, MBSO obtains the best results among all the four algorithms on f_1, f_2, and f_3. This may be due to that the IDS can match the search environment to provide suitable noise to create better ideas around the global optimal region to refine the solution for high

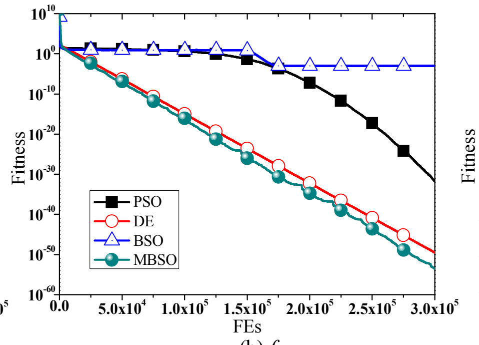
TABLE III. Solutions Accuracy (Mean and Standard Deviation) Comparisons

Functions	PSO	DE	BSO	MBSO
f_1	5.36E-48±1.64E-47†	1.29E-87±1.78E-87†	1.50E-64±3.02E-65†	**3.36E-98±1.37E-97**
f_2	1.77E-32±3.62E-32†	2.83E-50±2.71E-50†	9.93E-04±3.00E-03	**2.60E-54±1.36E-53**
f_3	1.09E-01±9.31E-02†	4.75E+03±1.22E+03†	3.73E-01±1.60E-01†	**3.71E-26±8.70E-26**
f_4	1.87E-01±1.25E-01†	**5.65E-12±1.45E-12†**	7.35E-03±6.89E-03†	6.74E-02±6.73E-02
f_5	3.27E+01±2.60E+01†	1.73E+01±1.08E+00†	2.79E+01±7.68E-01†	**1.46E-01±3.86E-01**
f_6	2.37E+03±3.98E+02†	1.78E+02±1.38E+02†	5.41E+03±7.03E+02†	**3.82E-04±1.57E-12**
f_7	2.56E+01±7.20E+00†	9.28E+01±7.52E+00†	3.08E+01±7.93E+00†	**1.66E-15±1.54E-15**
f_8	1.09E-14±2.35E-15	**4.14E-15±0.00E+00†**	7.58E-15±1.47E-15†	9.59E-15±2.91E-15

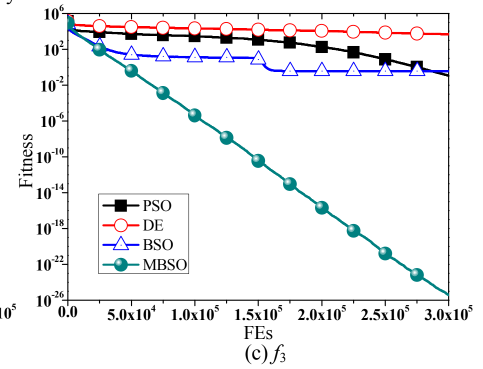
†The value of t with 29 degrees of freedom is significant at α=0.05 by a two-tailed test.



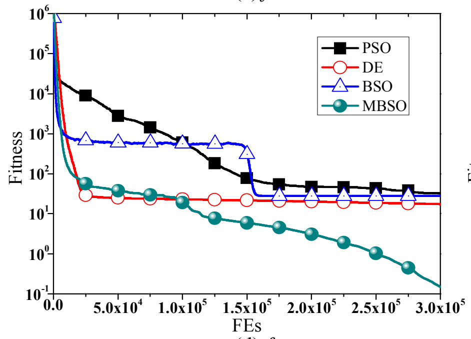
(a) f_1



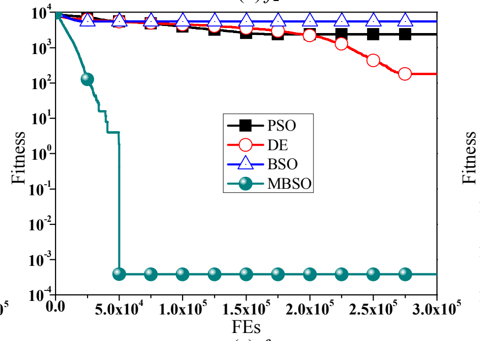
(b) f_2



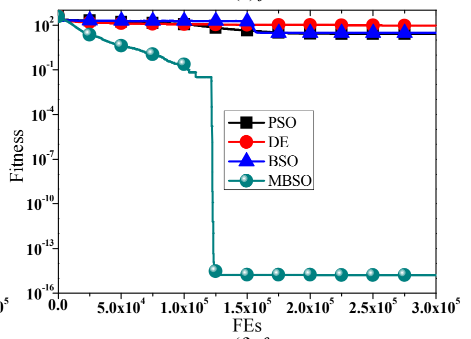
(c) f_3



(d) f_5



(e) f_6



(f) f_7

Figure 3. Convergence progresses of different algorithms.

accuracy. In BSO, refer to Eqs. (2)&(3), the new created idea was disturbed based on the current idea by Gaussian noise. However, this noise may be coarse and is not efficient enough to reach accuracy as high as 1.0×10^{-100}. In the contrast, MBSO uses the difference between two ideas as the disturbed noise as given by Eq. (6). This way, the disturbed noise can be within a comparable order of magnitude with the current ideas. This may cause two advantages. One is that MBSO can use large disturbed noise in the early phase to accelerate the search speed, and the other is that MBSO can use small disturbed noise in the late phase to refine the finial solution. This makes MBSO not only significantly outperforms BSO, but also performs significantly better than PSO and DE on the unimodal functions.

For the four multimodal functions, Table III also shows that MBSO is the best algorithm for f_5-f_7 and the second best algorithm for f_8. Only MBSO can obtain the global optima of all the four functions. The good performance of MBSO on multimodal functions may be due to the open minded element in IDS, as shown in Eq. (6). Similar to the brainstorming process in human being, the open minded element can introduce unusual ideas to avoid the brainstorming group being trapped by the current ideas (avoid the algorithm being trapped by the current local optimum). Moreover, once a new better search region appears, the IDS can make the algorithm fast converge to this new region and to refine the solution again. Therefore, MBSO not only can avoid local optima of multimodal functions, but also can obtain very high accurate solution to these functions. For example, all the other algorithms are trapped by the Schwefel's function (f_6) and the Rastrigin's function (f_7) in very poor local optima, while MBSO not only can find the global optimal region but also can refine the solution to very high accuracy.

We plot the convergence progress of the mean best fitness values during the run for some functions in Fig. 3. It is apparent that MBSO performs better than BSO, PSO, and DE in terms of final results and convergence speed. The fast convergence speed is much evident on the f_3, f_6, and f_7. It also can be observed that MBSO is able to improve solutions steadily for a long period without being trapped in local optima, as that for f_5. Specially, the figures for f_6 and f_7 show that only MBSO can jump out of local optima of these two multimodal functions while others are all trapped. Moreover, MBSO only uses about 5.0×10^4 and 1.2×10^5 FEs to obtain the global optima of these two functions respectively. These are much smaller than the maximal number of FEs 3×10^5.

C. Comparisons on Convergence Speed and Reliability

We have observed that MBSO obtains better results than BSO and other algorithms in the above subsection. It is natural for us to imagine that MBSO has a faster convergence speed than others to approach the global optima of different functions. In order to verify this, more experimental results are given and compared in Table IV. A successful run means that the algorithm can obtain a solution better than the acceptable solution. The FEs values are the average FEs needed to reach the acceptable solutions among the successful runs. In addition, success ratios of the 30 independent runs for each function are also compared. The CPU time is measured by second and is the average running time of the 30 independent runs. All the algorithms are implemented by Visual C++ 6.0 and the experiments are run on same machine with Intel Dual 2.40 GHz CPU, 2 GB memory and the Windows 7 operating system.

TABLE IV. CONVERGENCE SPEED AND RELIABILITY COMPARISONS

Functions		PSO	DE	BSO	MBSO
f_1	FEs	185270	35378	166915	**29958**
	CPU(s)	1.82	**1.22**	5.24	1.95
	Success	**100%**	**100%**	**100%**	**100%**
f_2	FEs	190750	48113	183854	**40687**
	CPU(s)	1.95	**1.35**	5.85	2.07
	Success	**100%**	**100%**	**100%**	**100%**
f_3	FEs	×	×	×	**103371**
	CPU(s)	2.54	**1.96**	7.69	2.61
	Success	×	×	×	**100%**
f_4	FEs	×	**180680**	×	×
	CPU(s)	1.90	**1.36**	6.11	1.95
	Success	×	**100%**	×	×
f_5	FEs	×	×	×	**223348**
	CPU(s)	1.94	**1.39**	7.29	1.95
	Success	×	×	×	**3.33%**
f_6	FEs	×	260480	×	**34644**
	CPU(s)	3.63	**2.12**	7.81	2.89
	Success	×	16.67%	×	**100%**
f_7	FEs	×	×	×	**83935**
	CPU(s)	2.37	**1.89**	7.60	2.59
	Success	×	×	×	**100%**
f_8	FEs	206833	**51735**	180141	73066
	CPU(s)	2.28	**1.69**	6.29	2.64
	Success	**100%**	**100%**	**100%**	**100%**

It can be observed from the table that only MBSO is successful to obtain the acceptable solutions to most of the functions. This is very important for an optimization algorithm because we do not know what kinds of problem will be faced. An algorithm with stronger reliability on different kinds of problems is preferred. Even though MBSO fails to obtain solutions as high accuracy as 1.0×10^{-6} to f_4, the results in Table III do indicate that MBSO can find the global optimal region of this function. Moreover, as we can expect that IDS can bring fast convergence speed to BSO, Table IV shows that MBSO is much faster than BSO on both unimodal and multimodal functions. For example, in solving the Sphere function (f_1), average numbers of FEs, 166915 and 29958, are needed by BSO and MBSO, respectively, to reach the acceptable accuracy 1.0×10^{-6}. Moreover, MBSO is observed to have faster convergence speed than both PSO and DE.

The CPU time shows that the computational burden of MBSO is significantly lighter than that of BSO. This is because MBSO uses a much simpler SGM method in the *grouping* operator instead of the *k*-mean clustering method and that MBSO uses a lighter computational method than BSO does to generate the noise value for creating new ideas. The results also show that PSO, DE, and MBSO have comparable computational burden, with DE being slightly light. Therefore, MBSO is not only efficient for that it significantly reduces the computational burden of BSO, but also is promising for that it has a comparable light computational burden compared with PSO and DE, but with better global search ability to reach higher solution accuracy.

D. Influences of SGM

In the Section IV-C, we can see that MBSO has a much lighter computational burden than BSO. This makes the algorithm easy and suitable for using in wider range of problems. However, whether the advantages brought by MBSO in the computational burden sacrifices the search ability should be investigated. In this subsection, we will compare MBSO using SGM and MBSO using *k*-mean cluster method (denoted as MBSO-Cluster), so as to investigate the influences of SGM.

TABLE V. COMPARISONS OF MBSO AND MBSO-CLUSTER

Functions		Solution (Mean±Std)	FEs	CPU	Success
f_1	MBSO	**3.36E-98±1.37E-97**	**29958**	**1.95**	**100%**
	MBSO-Cluster	2.25E-77±8.56E-77	39396	4.40	**100%**
f_2	MBSO	**2.60E-54±1.36E-53**	**40687**	**2.07**	**100%**
	MBSO-Cluster	1.30E-43±2.31E-43	56311	4.21	**100%**
f_3	MBSO	**3.71E-26±8.70E-26**	**103371**	**2.61**	**100%**
	MBSO-Cluster	6.22E-12±1.78E-11	193781	6.51	**100%**
f_4	MBSO	6.74E-02±6.73E-02	×	**1.95**	×
	MBSO-Cluster	**1.01E-07±1.46E-07**	**255747**	5.69	**100%**
f_5	MBSO	**1.46E-01±3.86E-01**	**223348**	**1.95**	**3.33%**
	MBSO-Cluster	2.06E+00±1.41E+00	×	5.73	×
f_6	MBSO	**3.82E-04±1.57E-12**	**34644**	**2.89**	**100%**
	MBSO-Cluster	**3.82E-04±8.92E-13**	70124	7.02	**100%**
f_7	MBSO	1.66E-15±1.54E-15	**83935**	**2.59**	**100%**
	MBSO-Cluster	**0±0**	135418	6.25	**100%**
f_8	MBSO	9.59E-15±2.91E-15	73066	**2.64**	**100%**
	MBSO-Cluster	**4.97E-15±1.53E-15**	**59408**	6.23	**100%**

The results are compared in Table V. It can be observed that the CPU time used by MBSO-Cluster is usually 2-3 times as the one used by MBSO. Moreover, MBSO generally has a faster convergence speed to reach the acceptable accuracy on most of the functions than MBSO-Cluster does. We also compare the CPU time values of MBSO-Cluster with those of BSO in Table IV. It can be seen that although both BSO and MBSO-Cluster use the *k*-mean clustering method in the *grouping* operator, MBSO-Cluster generally costs less CPU time than BSO. This confirms the advantages of the IDS in reducing the algorithm computational burden.

On the other hand, SGM has some influences on the solution accuracy besides its influences on the computational burden and convergence speed. Table V shows that when solving unimodal functions, MBSO is observed to obtain higher solution accuracy than MBSO-Cluster. This may be due to the faster convergence speed of MBSO, resulting in good refinement in unimodal functions. However, for multimodal functions, MBSO-Cluster seems to be slightly stronger than MBSO. For example, although both MBSO and MBSO-Cluster can find the global optimum of the Rastrigin's function (f_7), MBSO-Cluster even obtains the value 0. This may be due to that the k-mean clustering method can identify different peeks of multimodal functions, and therefore is helpful for the algorithm to find good solution to the problem. Nevertheless, by considerations of the solution accuracy, convergence speed, and computational burden on different kinds of functions, we can conclude that MBSO uses the SGM is promising in global optimization with fast speed.

E. Investigations on IDS

In this subsection, we will make investigations on IDS to study two issues. The first issue is to study how IDS can match the search environment to provide suitable disturbed values for creating new ideas. The other issue is to study the influence of the open minded element in the IDS on the MBSO performance.

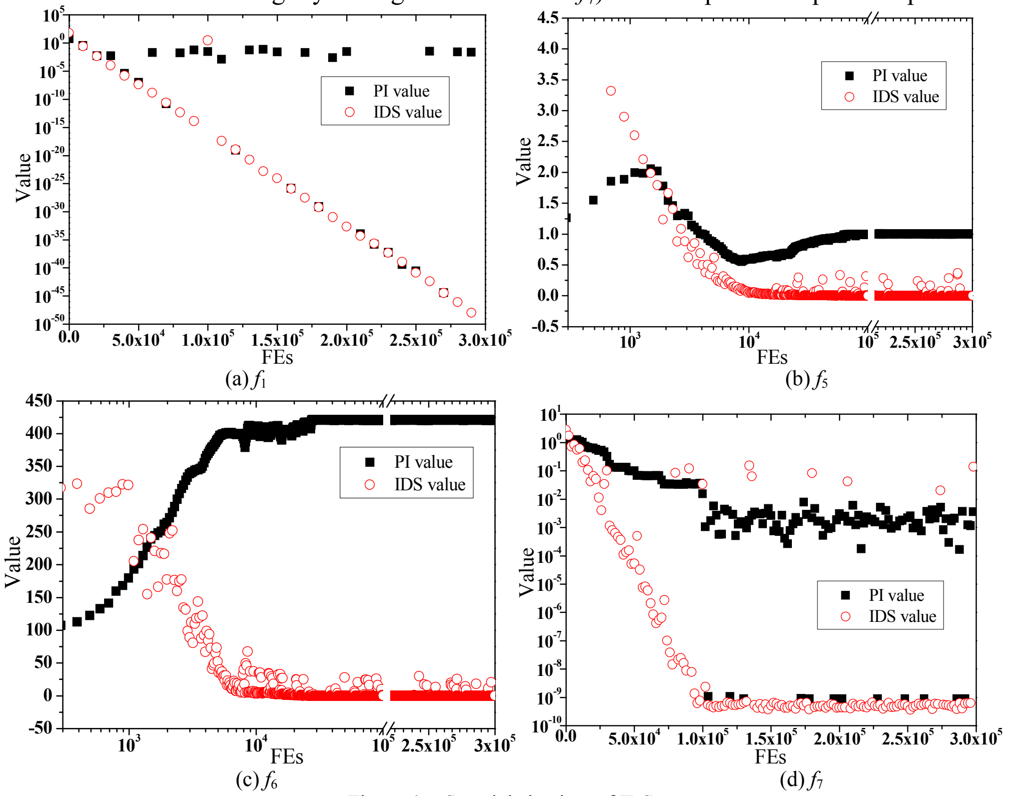


Figure 4. Search behaviors of IDS.

TABLE VI. EXPERIMENTAL RESULTS OF MBSO VARIANTS WITH DIFFERENT P_R VALUES

Functions	MBSO p_r=0	MBSO p_r=0.001	MBSO p_r=0.005	MBSO p_r=0.01	MBSO p_r=0.05	MBSO p_r=0.1
f_1	3.39E-103±1.84E-102	**1.42E-105±4.71E-105**	3.36E-98±1.37E-97	1.04E-86±4.19E-86	6.92E-26±1.60E-25	1.84E-02±1.05E-02
f_2	**3.38E-61±1.27E-60**	6.15E-60±2.11E-59	2.60E-54±1.36E-53	9.53E-49±2.73E-48	1.39E-14±9.19E-15	4.40E-02±1.90E-02
f_3	**9.31E-30±1.76E-29**	5.89E-29±1.97E-28	3.71E-26±8.70E-26	2.56E-22±5.19E-22	2.73E-04±3.73E-04	9.56E+02±3.69E+02
f_4	4.96E-01±4.40E-01	2.70E-01±2.09E-01	6.74E-02±6.73E-02	**2.38E-02±2.57E-02**	1.44E-01±1.18E-01	1.87E+00±5.56E-01
f_5	2.88E+00±2.68E+00	1.82E+00±2.18E+00	1.46E-01±3.86E-01	**2.94E-02±6.15E-02**	8.68E+00±5.99E+00	3.53E+01±2.35E+01
f_6	3.28E+03±7.27E+02	**3.82E-04±1.26E-12**	**3.82E-04±1.57E-12**	**3.82E-04±1.49E-12**	**3.82E-04±1.83E-12**	5.20E+03±9.39E+02
f_7	5.27E+01±1.52E+01	2.65E-01±4.48E-01	1.66E-15±1.54E-15	**8.88E-16±1.85E-15**	5.05E+00±2.13E+01	2.15E+02±1.98E+01
f_8	3.00E-00±6.99E-01	2.87E-07±1.42E-06	**9.59E-15±2.91E-15**	**9.59E-15±2.91E-15**	1.67E-13±2.91E-13	4.39E-02±1.64E-02

For the first issue, we take four typical functions (f_1, f_5, f_6, and f_7) as examples. We plot the position information of the ideas and the IDS information (i.e., the value (x_{ad}–x_{bd}) in (6)) in the same figure. As there are many dimensions (e.g., 30 dimensions are used in this paper) for a function, we just take the first dimension (d=1) to calculate the position information and the IDS information. In every generation, we calculate the position information PI as:

$$PI = \left| \left(\sum_{i=1}^{N} x_{i1} \right) / N \right| \tag{7}$$

and record the IDS information as:

$$IDS = \left| x_{a1} - x_{b1} \right| \tag{8}$$

The mean PI and IDS values of the 30 runs during the evolutionary process are plotted in Fig. 4. These figures confirm our intuitions that IDS can match the search environment to provide suitable disturbed values for creating better ideas. The Fig. 4(a) is for the typical unimodal Sphere function f_1. The figure shows that as the algorithm converging (decrease of the PI value), the IDS value becomes smaller proportional to the degree of the PI value. This is also observed in Fig. 4(d) for the f_7 whose global optimal solution is $\boldsymbol{x}$*=$\{0\}^D$.

When optimizing functions f_5 and f_6 whose global optimal solutions are $\boldsymbol{x}^*=\{1\}^D$ and $\boldsymbol{x}^*=\{420.96\}^D$ respectively, the PI values gradually goes to these optimal positions (as shown in Fig. 4(b)&(c)). This indicates that MBSO indeed has the global search ability on these multimodal functions to find the global optimal regions. More interesting, the IDS values are large in early phase when MBSO hasn't reached the global optimum while become smaller and smaller when MBSO refines the solutions in the global optimal region in late phase. Therefore, IDS is efficient and promising to provide suitable search guidance for MBSO to keep strong global optimization ability.

For the second issue, we investigate the parameter p_r which controls the open minded element. The p_r is set to 0, 0.001, 0.005, 0.01, 0.05, and 0.1 respectively and the experimental results are presented in Table VI. The table shows that small p_r values are good for unimodal functions. As there are no local optima in unimodal functions, it is rational that small p_r values, with less open minded element, can make MBSO converge fast to the global optimum. However, when solving multimodal functions, too small or too large p_r values both deteriorate the MBSO performance. When the p_r value is too small, the algorithm is easy to be trapped in local optima. When the p_r value is too large, the algorithm is like a random search. These two situations are both harmful for the algorithm performance on multimodal functions. The results indicate that a p_r value within the range [0.001, 0.01] would be good. The p_r value is set to 0.005 in MBSO in this paper.

V. Conclusion

In this paper, two novel component designs were proposed to modify the BSO algorithm. The first design is the SGM used in the *grouping* operator so as to reduce the algorithm computational burden. The second design is the IDS used in the *creating* operator. The IDS utilizes not only the open minded element for creating unusual ideas, but also the search information feedback to produce the disturbed values for creating better new ideas. The modified BSO featured with these two novel designs has been tested on a set of global benchmark optimization functions and compared not only with conventional BSO, but also with other EC algorithms as PSO and DE. The experimental results showed that MBSO has generally better performance than all the competitors, in terms of solution accuracy and convergence speed. Moreover, the contributions of SGM in reducing the computational burden and the characteristics of IDS are investigated in the paper.

Although MBSO has significantly enhanced the BSO's performance, we do not compare it with the current most powerful algorithms like APSO [21] and JADE [22] in this paper because our main purpose is to study how to enhance BSO and why MBSO can work better than BSO. As BSO is still in its infancy, many work and research are needed in the future. For example, the parameters of BSO are worth for experimental studies, the new progresses in the brainstorming research field can be used into BSO, and more experimental tests for comparing with other optimization algorithm are needed to be conducted, *etc*.

References

[1] F. Peng, K. Tang, G. L. Chen, and X. Yao, "Population-based algorithm portfolios for numerical optimization," *IEEE Trans. Evol. Comput.*, vol. 14, no. 5, pp. 782-800, Oct. 2010.

[2] J. Zhang, H. Chung, and W. L. Lo, "Clustering-based adaptive crossover and mutation probabilities for genetic algorithms," *IEEE Trans. Evol. Comput.*, vol. 11, no. 3, pp. 326-335, Jun. 2007.

[3] X. M. Hu, J. Zhang, Y. Yu, Y. L. Li, X. N. Luo, and Y. H. Shi, "Hybrid genetic algorithm using a forward encoding scheme for lifetime maximization of wireless sensor networks," *IEEE Trans. Evol. Comput.*, vol. 14, no. 5, pp. 766-781, Oct. 2010.

[4] W. N. Chen and J. Zhang, "Ant colony optimization approach to grid workflow scheduling problem with various QoS requirements," *IEEE Trans. Syst., Man, and Cybern. C.*, vol. 31, no. 1, pp. 29-43, Jan. 2009.

[5] Z. H. Zhan, J. Zhang, Y. Li, O. Liu, S. K. Kwok, W. H. Ip, and O. Kaynak, "An efficient ant colony system based on receding horizon control for the aircraft arrival sequencing and scheduling problem," *IEEE Trans. Intell. Transp. Syst.*, vol. 11, no. 2, pp. 399-412, Jun. 2010.

[6] W. N. Chen, J. Zhang, H. Chung, W. L. Zhong, W. G. Wu, and Y. H. Shi, "A novel set-based particle swarm optimization method for discrete optimization problems," *IEEE Trans. Evol. Comput.*, vol. 14, no. 2, pp. 278-300, April 2010.

[7] W. N. Chen, J. Zhang, Y. Lin, N. Chen, Z. H. Zhan, H. Chang, Y. Li, and Y. H. Shi, "Particle swarm optimization with an aging leader and challengers," *IEEE Trans. Evol. Comput.*,, in press.

[8] Z. H. Zhan and J. Zhang, "Self-adaptive differential evolution based on PSO learning strategy," in *Proc. Genetic Evol. Comput. Conf.*, Jul. 2010, pp. 39-46.

[9] Z. H. Zhan and J. Zhang, "Co-evolutionary differential evolution with dynamic population size and adaptive migration strategy," in *Proc. Genetic Evol. Comput. Conf.*, Jul., 2011, pp. 211-212.

[10] J. H. Zhong, J. Zhang, and Z. Fan, "MP-EDA: A robust estimation of distribution algorithm with multiple probabilistic models for global continuous optimization," *SEAL 2010, LNCS 6457*, pp. 85-94, 2010.

[11] J. Zhang, Z. H. Zhan, Y. Lin, N. Chen, Y. J. Gong, J. H. Zhong, H. Chung, Y. Li, and Y. H. Shi, "Evolutionary computation meets machine learning: A survey," *IEEE Comput. Intell. Mag.*, vol. 6, no. 4, pp. 68-75, Nov. 2011.

[12] J. Kennedy, R. C. Eberhart and Y. H. Shi, *Swarm Intelligence*, San Mateo, CA: Morgan Kaufmann, 2001.

[13] Y. Shi, "Brain storm optimization algorithm," in *Proc. 2nd Int. Conf. on Swarm Intelligence*, 2011, pp. 303-309.

[14] A. F. Osborn, *Applied Imagination*, New York: Scribner, 1957.

[15] R. I. Sutton and A. Hargadon, "Brainstorming groups in context: Effectiveness in a product design firm," *Administrative Science Quarterly*, vol. 41, no. 4, pp. 685-718, Dec. 1996.

[16] R. C. Litchfield, J. Fan, and V. R. Brown, "Directing idea generation using brainstorming with specific novelty goals," *Motivation and Emotion*, vol. 35, pp. 135-143, 2011.

[17] X. Yao, Y. Liu and G. M. Lin, "Evolutionary programming made faster," *IEEE Trans. Evol. Comput.*, vol. 3, no. 2, pp. 82-102, Jul. 1999.

[18] Z. H. Zhan, J. Zhang, Y. Li, and Y. H. Shi, "Orthogonal learning particle swarm optimization," *IEEE Trans. Evol. Comput.*, vol. 15, no. 6, pp. 832-847, Dec. 2011.

[19] R. C. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proc. 6th Int. Symp. Micromachine Human Sci.*, Nagaya, Japan, 1995, pp. 39-43.

[20] R. Storn and K. Price, "Differential evolution – A simple and efficient adaptive scheme for global optimization over continuous space," *J. Global Optimization*, vol. 11, no. 4, pp. 341-359, 1997.

[21] Z. H. Zhan, J. Zhang, Y. Li, and H. S. H. Chung, "Adaptive particle swarm optimization," *IEEE Trans. Syst., Man, and Cybern. B.*, vol. 39, no. 6, pp. 1362-1381, Dec. 2009.

[22] J. Q. Zhang and A. C. Sanderson, "JADE: Adaptive differential evolution with optional external archive," *IEEE Trans. Evol. Comput.*, vol. 13, no. 5, pp. 945–958, Oct. 2009.