An Optimization Algorithm Based on Binary Difference and Gravitational Evolution

Junli Li[1, 2], Yang Lou[2] and Yuhui Shi[3]
1. College of Computer Science, Sichuan Normal University
Chengdu, 610066, China
2. Information Science and Engineering College, Ningbo University
Ningbo, 315211, China
3. Department of Electrical and Electronic Engineering, Xi'an Jiaotong-Liverpool University
Suzhou, 215123, China
E-mail: li.junli@vip.163.com, louyang@mail.nbu.edu.cn, Yuhui.Shi@xjtlu.edu.cn

Received 6 December 2010
Accepted 14 February 2012

Abstract

Universal gravitation is a natural phenomenon. Inspired by Newton's universal gravitation model and based on binary differences strategy, we propose an algorithm for global optimization problems, which is called the binary difference gravitational evolution (BDGE) algorithm. BDGE is a population-based algorithm, and the population is composed of particles. Each particle is treated as a virtual object with two attributes of position and quality. Some of the best objects in the population compose the reference-group and the rest objects compose the floating-group. The BDGE algorithm could find the global optimum solutions through two critical operations: the self-update of reference-group and the interactive-update process between the reference-group and floating-group utilizing the gravitational evolution method. The parameters of BDGE are set by a trial-and-error process and the BDGE is proved that it can converge to the global optimal solution with probability 1. Benchmark functions are used to evaluate the performance of BDGE and to compare it with classic Differential Evolution. The simulation results illustrate the encouraging performance of the BDGE algorithm with regards to computing speed and accuracy.

Keywords: Optimization; Binary Difference; Differential Evolution; Gravitation.

1. Introduction

Evolutionary algorithms are a series of problem-solving methods that based on simulation of the natural evolution system, and its development can be traced back to 1950s. Compared with the classic optimization methods, an evolutionary algorithm has many advantages. For example, it is unconstrained by the search space limitations; it is unconstrained by the function types; and function gradient information is not essential, etc. Evolutionary algorithms have been widely used for optimization problem. Biological evolution theory inspired the emergence and development of bionics, which motivated many types of evolutionary algorithms, such as Genetic Algorithm [1-2], Simulated Annealing [3], Immune Algorithm [4-6], Particle Swarm Optimization [7-9], Ant Colony Optimization [10-11] and Bacterial Foraging Optimization [12], etc. All these contributions have gained great achievements on the field of optimization.

Chris and Tsang firstly proposed the concept and frame of GELS (Gravitational Emulation Local Search) [13] in 1995, which was then further developed by Webster [14]. Balachandar and Kannan [15] proposed RGES (Randomized Gravitational Emulation Search) algorithm in 2007, which overcame some weak points of GELS, such as a relatively slow convergence rate and low quality of solutions. The three gravitational emulation search methods above were initially applied to solve combinatorial optimization problems such as Traveling Salesman Problem (TSP).

Hsiao and Chuang *et. al.* proposed the SGO (Space

Gravitational Optimization) [16], which was based on Einstein's theory and Newton's law of gravitation. The SGO simulated the process that a number of planets shift in the space to search for the planet with the most massive. The geometric transformation of the space generates a force to make the planets shifting faster or slower. In 2007, Chuang and Jiang proposed an algorithm named IRO (Integrated Radiation Optimization) [17], based on the phenomenon that the movements of one planet in the gravitational field were compositively effected by the sum of all the other planets' gravitational radiation forces. Rashedi proposed the GSA (Gravitational Search Algorithm) in 2007 and it was continuously improved thereafter [18-20]. In GSA, a particle's total gravitation was a sum of all other particles' gravitations with random weights, and the total gravitation generated the acceleration to move. GPSO (Particle Swarm Optimization based on Gravitation) [21] was proposed by Kang and Wang *et. al* in 2007, which introduced an acceleration into Particle Swarm Optimization. All the four algorithms above were methods based on the position and displacement, and inspired by gravitation, each of which has different update formulas, respectively.

In this paper, the proposed algorithm Binary-Difference Gravitational Evolution (BDGE) shares the same point with the algorithms above, that is, all these algorithms utilize the concept of gravitation. However, BDGE does not simulate the physical movement processes as above, but via a clustering process based on elitist strategy. In BDGE, objects in the population are clustered into two different groups: the reference-object group and the floating-object group. Objects of different groups are updated independently or cooperatively, and the binary difference [25] strategy is adopted to update objects in the updating processes independently or cooperatively, i.e. the self-update of reference-group and the interactive-update group and interacting process between the two groups.

This paper is organized as follows. Section 2 presents both the details and ensemble of BDGE. Experimental study is given in Section 3, where benchmark functions are used to evaluate the performance. Section 4 gives an analysis on the parameters, and the convergence of algorithm is analyzed in Section 5, followed by the conclusion in Section 6.

2. Binary Difference Gravitational Evolution Algorithm

2.1. *Binary Difference Strategy*

The Differential Evolution (DE) [22-23] algorithm emerged as a very competitive form of evolutionary algorithm more than a decade ago, and was first proposed by R. Storn and K. Price in 1995, which was a simple and efficient heuristic algorithm for global optimization over continuous spaces and its feature is a mutation with a differential strategy. The selection of differential strategies would make significant influence on the performance of the algorithm. In consideration of diversity, there are at least three individuals involved in mutation in DE [23-24]. To simplify the mutation operation, the binary difference strategy was introduced, which is similar to the traditional but with only two individuals involved. As discussed in [25], binary difference using a sorted population could improve the performance of DE, especially in the aspect of convergence speed for low-dimensional problems. Because the objects of the reference-object group in BDGE are sorted by their qualities, binary difference is appropriate to be introduced into BDGE.

In binary difference, two objects are selected as candidates, of which the one with better fitness value is selected as the central object, and the objects with relative worse fitness values are involved in mutation. The process of producing new objects is as follows. A central object is temporarily fixed and the worse-fitness objects are updated one after another according to the central object. The new objects are more likely to be close to the central object.

The binary difference strategy not only is simpler, but also takes advantages of the gravitation clustering. The population is divided into two groups by gravitation clustering, and the binary difference is used for the interactive update of the two groups. The dimensional update of an object with binary difference strategy is as follows:

$$X(new_{kk}) = X(i_{kk}) + csign \cdot c_1 \cdot U(0,1) \cdot \left| X(i_{kk}) - X(j_{kk}) \right|$$

where $X(i)$ and $X(j)$ are two objects in a population, and $X(new)$ represents the newly produced individual. The subscript kk means dimension kk, $csign$ is a random symbol, c_1 is a constant, and $U(0,1)$ is a random value uniformly distributed in $[0,1]$. Binary difference uses a

random and variable factor $csign \cdot c_1 \cdot U(0,1)$ to keep diversity of the population.

2.2. *Gravitational Grouping Mode*

Gravitation is an attractive force between two objects, which exists between any two objects with masses. The value of gravitation is in proportion to the mass of either object, while in inverse proportion to the distance between them. Calculation of gravitation can be described as follows:

$$F = G\frac{m_1 \cdot m_2}{r^2} \tag{1}$$

where $G = 6.672$ is the gravitational constant, m_1 and m_2 are qualities of two objects, and r represents the distance between two objects.

Without loss of generality, we consider only the maximization problem as optimization problem considered in this paper:

$$Z = \max_{X \in S} f(X)$$

The search space is $S = \{X \mid l_i \le x_i \le u_i, i = 1,2,...,n\}$, where $X = (x_1, x_2, \cdots, x_n)^T$. $l_i, u_i (1 \le i \le n)$ are the lower and upper bound. $f(\cdot)$ is the objective function. We treat a particle in the search space S as an object, using gravitation for optimization. For any two objects A_1 and A_2 in the space S, their physical positions are represented by $X(A_1)$ and $X(A_2)$, and their qualities are represented by $m(A_1)$ and $m(A_2)$. Then the gravitation of A_1 and A_2 is simply described as:

$$F_{1,2} = \frac{m(A_1) \cdot m(A_2)}{r_{1,2} + K_0} = \frac{h(f(A_1)) \cdot h(f(A_2))}{r_{1,2} + K_0} \tag{2}$$

where $r_{1,2}$ means distance measurement, and K_0 is a small-valued constant to ensure denominator unequal to zero in Formula (2). $h(\cdot)$ is a one-dimensional scale transformation function satisfied $\forall x \in (-\infty, +\infty)$, $h(x) > 0$, which strictly increase monotonically. For convenience, $h(\cdot)$ is set as the absolute value here, in accordance with the concept that quality is a non-negative value. $m(\cdot) = h(f(\cdot))$ represents the quality of an object, which is scale transformed from objective function. Formula (2) is essentially equal to Formula (1), though the form is changed. In this paper, we calculate the gravitation measurements, but not the true gravitation values.

For convenience, the fitness of an object we mentioned in this paper indicates the quality, which is a scale transformation of the objective function.

2.3. *Binary Difference Gravitational Evolution Summary*

The population denoted by $\overrightarrow{RN}$ is assemble of all objects, which are clustered into two groups, i.e. the reference-object group $\vec{R}$, and the floating-object group $\vec{N}$. Followings are the symbols used in the BDGE descriptions. $X(R_i)$ represents the position of the ith reference-object R_i, $X(R_{i,k})$ represents its kth dimension and $m(R_i)$ represents the quality of the reference-object R_i. Similarly, $X(N_j)$ represents the position, $X(N_{j,k})$ represents its kth dimension and $m(N_j)$ represents the quality of the floating-object N_j. $U(a,b)$ is a random value uniformly distributed in interval $[a,b]$.

Binary Difference Gravitational Evolution algorithm can be summarized as follows:

Step1 The reference-object group $\vec{R}$, and the floating-object group $\vec{N}$ are randomly generated:

$$\begin{cases} \vec{R} = \{R_i, 1 \le i \le n_1\} \\ \vec{N} = \{N_j, 1 \le j \le n_2\} \end{cases}$$

where n_1 and n_2 are two integer numbers, and represent the group sizes of two groups, respectively.

Step2 If the halt conditions are satisfied, then halt. Otherwise resort the population $\overrightarrow{RN}$, which should satisfy the following rules:

$$\begin{cases} m(R_i) \ge m(R_{ii}),\ 1 \le i < ii \le n_1 \\ m(R_i) \ge m(N_j),\ 1 \le i \le n_1,\ 1 \le j \le n_2 \end{cases}$$

Step3 Binary difference strategy is applied to any two reference-objects R_i $(1 \le i \le n_1)$ and R_{ii} $(i < ii \le n_1)$, and a new object $C_{i/ii} (1 \le i < ii \le n_1)$ is generated:

$$\begin{cases} csign_k = \begin{cases} +1, & U(0,1) \le 0.5 \\ -1, & U(0,1) > 0.5 \end{cases}, \ 1 \le k \le n \\ X(C_{i/ii,k}) = \begin{cases} U(l_k, u_k), & \text{if } \left| X(R_{i,k}) - X(R_{ii,k}) \right| < \eta \\ X(R_{i,k}) + csign_{kk} \cdot c_1 \cdot U(0,1) \cdot \left| X(R_{i,k}) - X(R_{ii,k}) \right|, & \\ & \text{otherwise} \end{cases} \end{cases}$$

where $X(C_{i/ii})$, $X(C_{i/ii,k})$ and $m(C_{i/ii})$ represent the position, the kth dimension and the quality of the newly generated object $C_{i/ii}$, respectively. $\left| X(R_{i,k}) - X(R_{ii,k}) \right| < \eta$ represents the distance of two objects. $C_{i/ii}$ replaces R_{n_1} if $m(C_{i/ii}) > m(R_{n_1})$, and then resort the reference-object group $\vec{R}$, which should satisfy the following rules:

$$m(R_i) \ge m(R_{ii}), \ 1 \le i < ii \le n_1 ;$$

Step4 The gravitation measurement $F_{i/j}$ and distance measurement $r_{i/j}$ (Euclidean distance measurement is used here) between any $R_i (1 \le i \le n_1)$ and $N_j (1 \le j \le n_2)$ is calculated as follows:

$$\begin{cases} r_{i/j} = \ \| X(R_i) - X(N_j) \|_2 \\ F_{i/j} = \dfrac{m(R_i) \cdot m(N_j)}{r_{i/j} + K_0} \end{cases}$$

Then a bidirectional selection runs:

(1) For every floating-object $N_j (1 \le j \le n_2)$, to select a reference-object $R_{index\_1(j)}$, which has the maximal gravitation with N_j , where

$$\begin{cases} F_{i/j}^{\max} = \max(F_{i/j}, 1 \le i \le n_1) \\ index\_1(j) = \min(\{ i \mid F_{i/j} = F_{i/j}^{\max}, 1 \le i \le n_1 \}) \end{cases}$$

(2) For every reference-object $R_i (1 \le i \le n_1)$, to select a floating-object $N_{index\_2(i)}$,which has the minimum gravitation with $R_i (1 \le i \le n_1)$, and then it is thorough eliminated, which means it would be replaced and do not participate in generating the next generations, where

$$\begin{cases} set(i) = \{ j \mid i = index\_1(j), \ 1 \le j \le n_2 \} \\ F_{i/j}^{\min} = \min(\{ F_{i/j}, j \in set(i) \}) \\ index\_2(i) = \min(\{ j \mid F_{i/j} = F_{i/j}^{\min}, j \in set(i) \}) \end{cases}$$

The eliminated object $N_{index\_2(i)}$ does not exist, if $set(i) = \phi$.

Step5 There are two situations for update of floating-objects based on ***step4***.

(1) For the floating-objects which have not been eliminated $\{ N_j \mid j \ne index\_2(i), \ 1 \le j \le n_2, \ 1 \le i \le n_1 \}$, binary difference is applied to update them:

$$\begin{cases} csign_k = \begin{cases} +1, & U(0,1) \le 0.5 \\ -1, & U(0,1) > 0.5 \end{cases}, \ 1 \le k \le n \\ X(N_{j,k}) = \begin{cases} U(l_k, u_k), & \text{if } \left| X(R_{index\_1(j),k}) - X(N_{j,k}) \right| < \eta \\ X(R_{index\_1(j),k}) & \\ +csign_k \cdot c_1 \cdot U(0,1) \cdot \left| X(R_{index\_1(j),k}) - X(N_{j,k}) \right|, & \\ & \text{otherwise} \end{cases} \end{cases}$$

(2) For the eliminated floating-objects $\{ N_{index\_2(i)}, \ 1 \le i \le n_1 \}$, we randomly select a reference-object to replace it.

Then go to ***Step2***.

2.4. *Algorithm Illustrations*

2.4.1. *Sorting Operation*

The population is divided into the reference-object group and the floating-object group in BDGE, where the reference-objects have the better fitness values, thus they are objects with bigger qualities. The reference-objects are arranged according to their qualities in the group.

Reference-objects ① ② ③ ④

Ordering Rules $m(①) \ge m(②) \ge m(③) \ge m(④)$

Fig. 1 Ordering of Reference-object Group

The sorting operation is applied to the reference-object group only, and the rules of sorting is illustrated in Figure 1, where $m(\cdot)$ represent quality of an object, and objects are sorted in a proper sequence by their values of qualities. The other part of population, i.e. the floating-object group, does not need to be sorted, and the only requirement we need to ensure is that any floating-object's fitness value should never be better than the worst reference-object's.

2.4.2. *Method of Update*

There are two types of update methods in BDGE, i.e. the self-update of reference-group and the interactive-update process between the reference-group and floating-group.

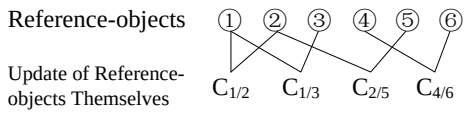


Fig. 2 An Example of Reference-objects Self-update

As it is showed in Figure 2, the self-update process of the reference-objects occurred after objects are sorted. Reference-objects ① and ②, ① and ③, ② and ③ …are used to produce new objects one pair after another by binary difference. Thus, any two reference-objects are selected in the process above. The rule of new objects' selection is that we select the superior with bigger quality and eliminate the inferior. The purpose of a reference-object is to keep a high level on quality as well as fitness, which to a certain degree ignores the diversity of the reference-group but speed up the convergence rate only.

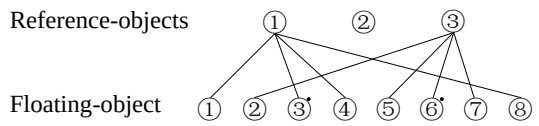


Fig. 3 An Example of Gravitation Clustering

After the self-update process of reference-group, an interactive-update of reference-objects and floating-objects are performed. Figure 3 shows an example of gravitation clustering, the essential of which is the interactive-update. Compared with reference-objects ② and ③, floating-object ① has a bigger gravitation measurement with reference-object ①, so floating-object ① selects reference-object ①, the lines of reference-object ① and floating-object ① shows their relationship. The similar progress occurs on other objects. For the reference-object ①, there has four floating-objects (①, ③, ④ and ⑧) selected, among which floating-object ③ (with a dot) has the minimum gravitation measurement with reference-object ①, then floating-object ③ would be eliminated from population. Similarly, other reference-objects eliminate floating-objects as above to finish the interactive-update process between the reference-group and floating-group.

The positions of eliminated floating-objects are regenerated in a random way as initialization. The regeneration is called forced-update, which means being replaced and not participate in generating the next generations. The forced-update step of interactive-update contributes a lot in the goal of keeping population diversity.

2.4.3. *Physical Meanings*

As showed in both Formula (1) and (2), if two objects have a large value of gravitation, that may be caused by either a small distance in the denominator or big quality of each object in the numerator. The second case is what we expected, while the former may lead to a trap of local optimum. To solve this problem, we introduce a threshold parameter η , the value of which will be discussed in another paragraph.

2.4.4. *Advantages of Gravitation Clustering:*

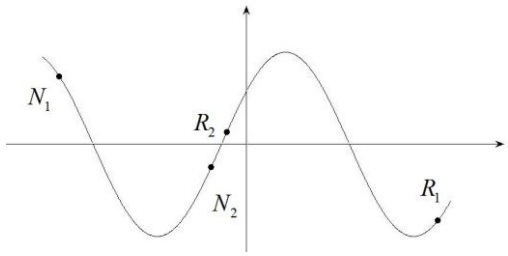


Fig. 4 Searching Method

Figure 4 shows an example of one dimension, as shown reference -objects R_1 , R_2 , and floating-objects N_1 , N_2 , according to Formula (2), $F_1 = m(R_1) \cdot m(N_1) / (r_1 + K_0)$, $F_2 = m(R_2) \cdot m(N_2) / (r_2 + K_0)$. Both F_1 and F_2 are large (when F_i is small, the relationship of R_i and N_i is ignored), where F_1 is caused by big qualities of R_1 , and it is good for exploring on new region, which increases diversity. On the contrary, F_2 may be caused by the small distance r_2 . This is good for exploiting in the current region, which increases accuracy, but may cause a risk of being trapped in local optimums.

3. Experiments

3.1. *Comparison Methods*

We choose Differential Evolution to compare BDGE with because both use difference vector to generate new individuals. The parameters of the two algorithms are set as follows:

DE: The parameters are set as that proposed by Yang *et al*.[26], that is the population size $NP = 10 \times D$, where D is the dimension of variable in solution space, $CR = 0.9$, $F = 0.5$, and a mutation strategy of DE/rand/1. The halting condition is: 1. The number of fitness function evaluations overstepped 2,000,000. 2. Supposed Min$f$ is the global optimum value, f_{best} is the searched optimum value, and $\varepsilon = 10^{-6}$ is the accuracy value, if $|\text{Min}f - f_{best}| < \varepsilon$, the running stops.

BDGE: The reference-object group size $n_1 = 10$ and the floating-object group size $n_2 = 20$ .And $\eta = 10^{-9}$, $c_1 = 2$, $K_0 = 10^{-10}$. The halting condition is set the same as that in DE.

3.2. *Benchmark Functions*

There are 9 benchmark functions presented in table 1, which are used to evaluate the performance of BDGE, and to compare it with the classic Differential Evolution.

Table. 1 Benchmark Functions

Test Function	Domain Range	Optimal Point
Sphere Model $f_1 = \sum_{i=1}^{D} x_i^2$	[-100,100]^D	Min f_1 = $f_1(0,0,\cdots,0) = 0$
Schwefel's Problem 2.22 $f_2 = \sum_{i=1}^{D} \lvert x_i \rvert + \prod_{i=1}^{D} \lvert x_i \rvert$	[-10,10]^D	Min f_2 = $f_2(0,0,\cdots,0) = 0$
Schwefel's Problem 2.21 $f_3 = \max_i \{\lvert x_i \rvert,\ 1 \le i \le D\}$	[-100,100]^D	Min f_3 = $f_3(0,0,\cdots,0) = 0$
Step Function $f_4 = \sum_{i=1}^{D} (\lfloor x_i + 0.5 \rfloor)^2$	[-100,100]^D	Min f_4 = $f_4(0,0,\cdots,0) = 0$
Quartic Function i.e. Noise $f_5 = \sum_{i=1}^{D} i x_i^4 + random[0,1)$	[-1.28,1.28]^D	Min f_5 = $f_5(0,0,\cdots,0) = 0$
Generalized Schwefel's Problem 2.26 $f_6 = -\sum_{i=1}^{D} (x_i \sin(\sqrt{\lvert x_i \rvert}))^2$	[-500,500]^D	Min f_6 = $f_6(420.9687, 420.9687, \cdots, 420.9687) = -12569.5$
Generalized Rastrigin Function $f_7 = \sum_{i=1}^{D} \left(x_i^2 - 10\cos(2\pi x_i) + 10\right)$	[-5.12,5.12]^D	Min f_7 = $f_7(0,0,\cdots,0) = 0$
Generalized Penalized Function I $f_8 = \frac{\pi}{D}\{10\sin^2(\pi y_1) + \sum_{i=1}^{D-1}(y_i - 1)^2 \cdot [1 + 10\sin^2(\pi y_{i+1})] + (y_D - 1)^2\} + \sum_{i=1}^{D} u(x_i, 10, 100, 4)$ where $u(x_i,a,k,m) = \begin{cases} k(x_i - a)^m, & x_i > a \\ 0, & -a \le x_i \le a \\ k(x_i - a)^m, & x_i < -a \end{cases}$ $y_i = 1 + \frac{1}{4}(x_i + 1)$	[-50,50]^D	Min f_8 = $f_8(1,1,\cdots,1) = 0$
Generalized Penalized Function II $f_9 = 0.1\{\sin^2(3\pi x_1) + \sum_{i=1}^{D-1}(x_i - 1)^2 \cdot [1 + \sin^2(3\pi x_{i+1})] + (x_D - 1)^2[1 + \sin^2(2\pi x_D)]\} + \sum_{i=1}^{D} u(x_i, 5, 100, 4)$ where $u(x_i,a,k,m) = \begin{cases} k(x_i - a)^m, & x_i > a \\ 0, & -a \le x_i \le a \\ k(x_i - a)^m, & x_i < -a \end{cases}$	[-50,50]^D	Min f_9 = $f_9(1,1,\cdots,1) = 0$

3.3. *Simulation Results*

We run each algorithm 50 times independently on each benchmark function above. The dimension of problems is set to be 10. Successful run means the difference between the obtained solution and the true value to be less than 10^{-6} before the number of function evaluations reaches the maximum value, which is set to be 2,000,000.

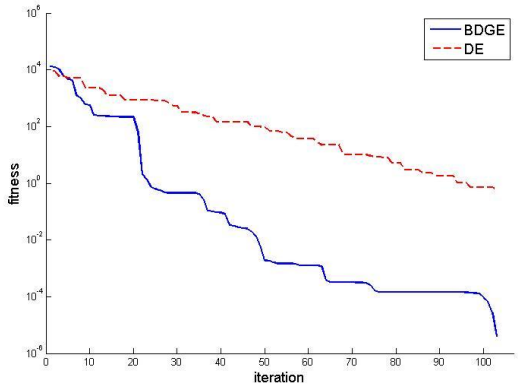
Table. 2 Comparison of results on 50-time independent random testing with benchmark functions

Test Function	BDGE			DE		
	Best/Worst	***Std***	***FEs***	***Best/Worst***	***Std***	***FEs***
f1	3.77e-7/9.86e-6	e-6	6678	3.94e-6/9.98e-6	e-6	22210
f2	2.79e-6/9.97e-6	e-6	7289	7.07e-6/9.94e-6	e-7	32738
f3	0/0	0	180	0/0	0	552
f4	0/0	0	5239	0/0	0	8656
f5	4.04e-6/3.19e-4	e-5	1875528	1.44e-5/1.01e-4	e-5	2000000
f6	-12569.499/-12569.498	e-6	8867	-12569.499/-12569.498	e-6	99408
f7	2.92e-9/9.83e-6	e-6	22491	***4.83e-6/0.99***	***e-1***	***371198***
f8	5.81e-8/9.96e-6	e-6	8002	2.49e-6/9.91e-6	e-6	19870
f9	1.41e-9/9.93e-6	e-6	12478	3.01e-6/9.98e-6	e-6	21092

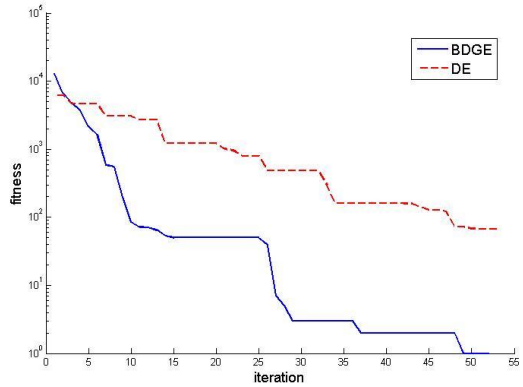
As shown in Table 2, where *Best /Worst* mean the best and worst solutions in the testing based on the average of 50 times of independent runs; *Std* and *FEs* is short for standard deviation and function evaluations. According to the data in Table 2, Differential Evolution cannot always find the optimal value of f_7, for the worst solution in the 50 runs is 0.9949, dissatisfying the precision. We define when the obtained solution which has the precision less then 10^{-4} as a successful run. There are 46 successful runs in 50 total runs while

optimizing f_7 utilizing DE, while for other functions in this test, the successful runs are all 50, i.e. DE can solve the rest problems and the proposed BDGE can optimize all the functions with relative smaller *Std* and *FEs* values, which means a higher precision and fast convergence speed, Table 2 shows BDGE has a better performance than DE due to a smaller number of function evaluations, of which a maximal disparity was that DE got a more than 10-time function evaluations than BDGE, that means the latter is 10 times faster than the former.

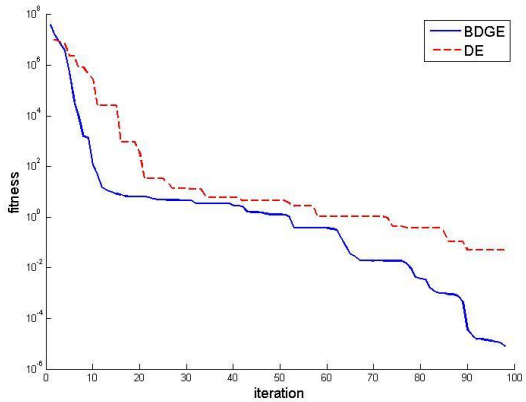
Figure 5 shows the comparison between DE and BDGE, which illustrate the fitness values decreased as iteration increasing for optimizing the functions.



(a)



(b)



(c)

Fig. 5 Comparison of BDGE and DE in Testing of (a)f1, (b)f4, (c)f8.

As can be observed clearly from the figures, BDGE reached the global optimum requiring less number of iteration than DE. During the optimizing process, the blue solid line declines faster than the red dotted line, which illustrates that the BDGE converges faster than DE does.

4. Experimental Study of Parameters

4.1. *The group sizes* n_1 *and* n_2

The values of group size n_1 and n_2 may influence the performance of BDGE a lot. Here we discuss these two parameters, using *Schaffer 1 Problem* (*SF1*) as an example, which was a classical deceptive problem.

$$f_{SF1} = 0.5 + \frac{\left(\sin\sqrt{\sum_{i=1}^{D} x_i^2}\right)^2 - 0.5}{\left(1 + 0.001(\sum_{i=1}^{D} x_i^2)\right)^2}, -500 \le x_i \le 500,\ i = 1, 2, \cdots 10$$

$$\text{Min}\, f_{SF1} = f_{SF1}(0, 0, \cdots 0) = 0$$

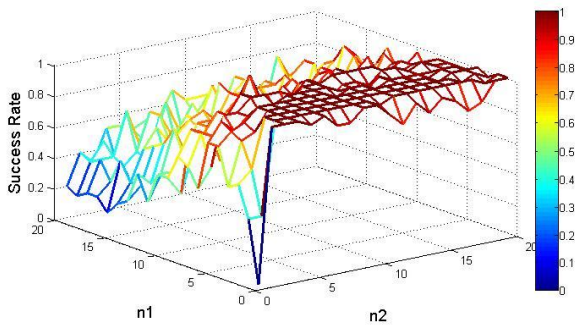
We optimize f_{SF1} with 2-dimension and 5-dimension utilizing BDGE to test different settings of group size n_1 and n_2 . The halt condition is set as either $|f - \text{Min}\, f_{SF1}| < 10^{-6}$ or the number of fitness function evaluations overstepped the *Max-FEs*, which was set to be 150,000 for 2-dimension problem and 2,000,000 for 5-dimension problem, respectively. Other parameters of BDGE are set as follows, $\eta = 10^{-9}$, $c_1 = 2$, $K_0 = 10^{-10}$. n_1 and n_2 are experimentally set to be 1, 2, ..., 20,

respectively. For each pair of n_1 and n_2, 50 runs are conducted independently, and a total of $20\times20\times50$ times of runs are conducted for the two problems. Figure 6 shows the influences on optimization f_{SF1} of 2-dimension and Figure. 7 shows that of 5-dimension. According to the halt condition, we adopt success as operation stopped when $\left|f-\mathrm{Min}\, f_{SF1}\right|<10^{-6}$ other than the number of fitness function exceeds *Max-FEs*.

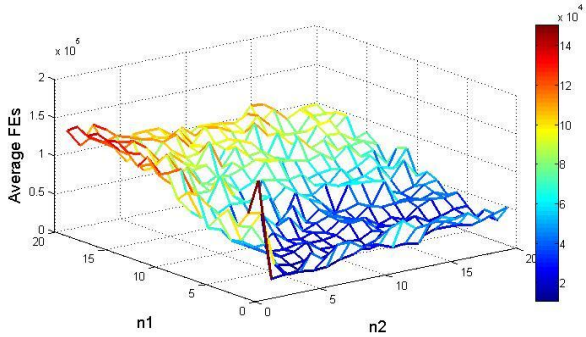
It can be seen from Figure 6 that for a 2-dimension problem, when $n_1<n_2$, it has a higher success rate, while a lower rate when $n_1>n_2$. From Figure 6(b), when $n_1<n_2$, it has a lower number of average function evaluation, while an opposite result when $n_1>n_2$. As an observation from Figure 6, a selection of $n_1<n_2$ is good for the 2-dimension *Schaffer Problem*. The same observation can be obtained from Figure 7 for the 5-dimension problem.

Intuitively according to the algorithm procedure, when $n_1>n_2$, the reference-objects' update takes high percentage of the whole update progress, which leads a fast convergence. However, it will lose diversity especially when dealing with deceptive problem. When $n_1<n_2$, this is another status that the diversity is contented enough but with a little slow convergence. BDGE is an algorithm that can balance between a high convergence rate and a high diversity.

The floating-object group size could be arbitrarily big but a too large group size is time consuming. The conclusion is n_1 should be appropriate small, n_2 could be slightly larger than n_1, and their relationship should meet $n_1\le n_2$. The specific values are defined depending on circumstances.

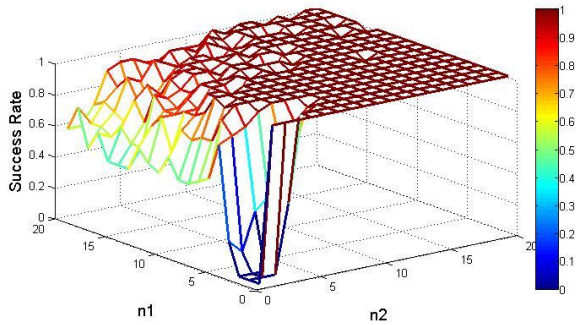


(a)

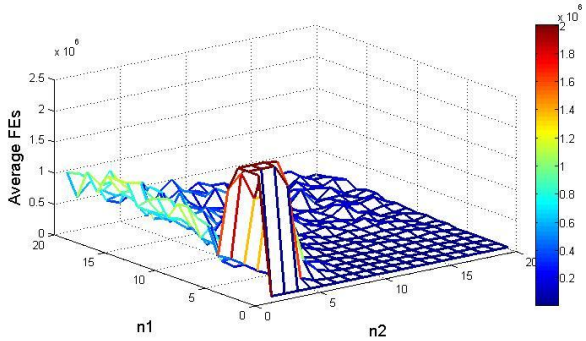


(b)

Fig. 6 The Effect of Alternative Group Sizes on Optimization Schaffer 1 Problem (SF1) of 2-dimension (a) Result of Success Rate, (b) Result of Average Function Evaluations



(a)



(b)

Fig. 7 The Effect of Alternative Group Sizes on Optimization Schaffer 1 Problem (SF1) of 5-dimension (a) Result of Success Rate, (b) Result of Average Function Evaluations

4.2. *The threshold parameter* η

The threshold parameter η is discussed in this part, which influences the algorithm performance a lot as well. BDGE is tested on *Shifted Sphere Function* (f_{SSF}).

$$f_{SSF} = \sum_{i=1}^{D} z_i^2,\ z_i = x_i - o_i,\ -100 \le x_i \le 100$$

$$o_i = \begin{cases} l_i + \frac{u_i - l_i}{2} \times \frac{1}{4} + (u_i - l_i) \times \frac{3}{4} \times \frac{i}{D}, i = 2,4,\cdots \\ l_i + \frac{u_i - l_i}{2} \times \frac{1}{4} + (u_i - l_i) \times \frac{3}{4} \times \frac{D+i-1}{D}, i = 1,3,\cdots \end{cases}$$

$$\text{Min}\, f_{SSF} = f_{SSF}(o_1, o_2, \cdots, o_D) = 0$$

Here we separately test the problem with a dimension of 10, 20, 30 and 100, respectively. The parameters are set as follows: $n_1 = 10$, $n_2 = 20$, $c_1 = 2$, $K_0 = 10^{-10}$. The halt conditions are set as either $|f - \text{Min}\, f_{SSF}| < 10^{-6}$ or the number of fitness function evaluations overstepped 2,000,000. η increases gradually by $\eta = 10^{-1}, 10^{-2}, \cdots, 10^{-10}$, and for each setting of η, 50 runs are conducted independently.

Table. 3 The result of benchmark testing on Shifted Sphere Function with different values of η

Test functions	*Shifted Sphere Function*							
	10-D		*20-D*		*30-D*		*100-D*	
η	*Std*	*FEs*	*Std*	*FEs*	*Std*	*FEs*	*Std*	*FEs*
10^{-1}	e-2	2000000	e-1	2000000	e+0	2000000	e+3	2000000
10^{-2}	e-4	2000000	e-2	2000000	e-2	2000000	e+2	2000000
10^{-3}	e-5	2000000	e-4	2000000	e-4	2000000	e+2	2000000
10^{-4}	e-7	2000000	e-6	2000000	e-5	2000000	e+0	2000000
10^{-5}	e-8	7031	e-8	392908	e-7	2000000	e-1	2000000
10^{-6}	e-8	6194	e-9	24766	e-9	87314	e-2	2000000
10^{-7}	e-8	5952	e-9	21129	e-9	50503	e-4	2000000
10^{-8}	e-8	5877	e-8	20065	e-9	44269	e-6	2000000
10^{-9}	e-8	6184	e-8	20362	e-9	42729	e-6	1501400
10^{-10}	e-8	6202	e-8	21480	e-8	45363	e-9	909876

As shown in Table 3, with the value of η decreasing, the standard deviation *Std* and the average number of fitness function evaluations *FEs* decrease as well, which means a better performance in optimization. From the statistic results shown in Table 3, it can be observed that in general, a setting of $\eta = 10^{-8} \sim 10^{-10}$ preferably fit for most problems, which can be used as a common setting. As for low-dimensional problems, a setting of η nearby the precision value leads to nice performances, while for a high-dimensional problem, the smaller η is set, the better result would be got. For example, for *Shifted Sphere Function* with 100-dimension, when $\eta = 10^{-15}$, the *FEs* value decreases to 459,283, better than 909,876 when $\eta = 10^{-9}$.

5. Analysis of Algorithm Convergence

BDGE is a type of randomized optimization algorithms. The conditions of proving the convergence of a randomized algorithm were firstly proposed by Solis and Wets [27]. They have given the theorems to prove whether an algorithm has converged to the global optimal with probability 1, which can be summarized as follows:

Hypothesis 1 [27] if $f(D(z,\zeta)) \le f(z), \zeta \in S$, then $f(D(z,\zeta)) \le f(\zeta)$.

where D is a function to generate potential solutions, ζ is a random vector generated from the probability space (R^n, B, μ_k), and f is the objective function. S, which is the subspace of R^n, represents the constraint space of the problem. μ_k is probability measurement on B, which is the σ domain of R^n subset.

Hypothesis 2 [27] if A is a Borel subset of S, satisfies $\nu(A) > 0$, then

$$\prod_{k=0}^{\infty}(1 - \mu_k(A)) = 0$$

where $\nu(A)$ is a n-dimensional closure of subset A, and $\mu_k(A)$ is a probability, indicating the rate that μ_k generates A.

Theorem [27] Suppose f is a fathomable function, S is a fathomable subset of R^n, $\{z_k\}_0^\infty$ is a solution sequence generated by the randomized algorithm. If both Hypothesis 1 and Hypothesis 2 are satisfied simultaneously, then

$$\lim_{k \to \infty} P[z_k \in R_\varepsilon] = 1$$

where R_ε represents the set of the global optimal solutions.

According to the theorem, if both Hypothesis 1 and Hypothesis 2 are satisfied simultaneously for BDGE, it can be confirmed that the proposed BDGE algorithm converges to the global optimal solution with probability 1. The convergence proof of BDGE is given as follows:

In the BDGE algorithm, the return value before the tth iteration is the function value of $x_i(t-1)$, i.e. $f(x_i(t-1))$, and $f(x_i(t))$ represents the function value of the tth iteration value $x_i(t)$, where $f(x)$ represents

the objective function. The function D of Hypothesis 1 is defined as:

$$D(x_i(t-1), x_i(t)) = \begin{cases} x_i(t-1), & \text{if } f(x_i(t-1)) \le f(x_i(t)) \\ x_i(t), & \text{if } f(x_i(t-1)) > f(x_i(t)) \end{cases}$$

It can be inferred Hypothesis 1 is satisfied for BDGE. For Hypothesis 2, all that is needed is to prove the S-sized sample space contains S, thus,

$$\bigcup_{i=1}^{S} M_{i,t} \supseteq S$$

where $M_{i,t}$ represents the support set of the ith individual's sample space in tth iteration.

Suppose there are N iterations in the search, and the range of the ith iteration is S_i, which is the support set as well. Therefore, the union space of the population (a set of individuals) is $\bigcup_{i=1}^{N} S_i$. The range of an individual is adjustable, and when range covers the boundary of the solution space, though there are only a few individuals, it can enable $S = \bigcup_{i=1}^{N} S_i$ Then Hypothesis 2 is satisfied for BDGE algorithm.

In conclusion, BDGE converges to the global optimal solution with probability 1, according to the theorem.

6. Conclusion

In this paper, the BDGE algorithm based on two critical models, i.e. binary difference and gravitational evolution for global optimization, was proposed. The proposed algorithm BDGE was compared with the Differential Evolution by testing both algorithms on benchmark functions. Simulation results show that the BDGE can explore the solution space more effectively than DE to obtain the global solution, and the BDGE requires a much smaller size population than DE does. The parameters of BDGE are studied and set by trial-and-error, and the convergence analysis was also conducted to show BDGE can converge to the global optimal solution with probability 1. Certainly, there is still room to further improve BDGE. For instance, the number of parameters should be reduced to simplify the algorithm, and the clustering and gravitational models should be studied to solve high-dimensional problems, which are our future research work.

Acknowledgement

This paper is partially supported by National Natural Science Foundation of China under Grant Numbers 60975080, 60832003; and the Natural Science Foundation of Zhejiang Province under Grant Number Y1100076.

References

1. J. H. Holland. Adaptation in Natural and Artificial Systems, *University of Michigan Press*, Ann Arbor, MI. 1975.
2. Y. G. Xi, T. Y. Chai and W. M. Yun, Survey on Genetic Algorithm, *Control Theory and Applications*, vol.13, no.6, pp. 697–708, 1996.
3. S. Kirkpatrick, C. D. Gelatt, M. P. Vecchi, Optimization by Simulated Annealing, *Science*, vol.220, no.4598, pp.671–680, 1983.
4. K. Mori, M. Tsukiyama and T. Fukuda, Immune Algorithm with Searching Diversity and its Application to Resource Allocation Problem, *T.IEE Japan*, vol. 113-C, no.10, pp.872–878, 1993.
5. M. J. Li, A. Luo and T. S. Tong, Artificial Immune Algorithm and Its Applications, *Control Theory and Applications*, vol.21, no.2, 153–157, 2004.
6. L. Wang, J. Pan and L. C. Jiao, The Immune Algorithm, *Acta Electronica Sinica*, vol. 28, no. 7, 74–78, 2000.
7. R. C. Eberhart and Y. H. Shi, Computational Intelligence: Concepts to Implementations, *Elsevier*, Singapore, 2009.
8. J. Kennedy and R. C. Eberhart, Particle Swarm Optimization, *Proc. of the IEEE Int. Conf. on Neural Networks*, Piscataway, NJ, pp.1942–1948, 1995.
9. X. P. Zhang, Y. P. Du, G. Q. Qin and Z. Qin, Adaptive Particle Swarm Algorithm with Dynamically Changing Inertia Weight, *Journal of Xi'an Jiaotong University*, vol. 39, no. 10, pp.1039–1042, 2005.
10. M. Dorigo and V. Maniezzo, A. Colorni. Ant System: Optimization by a Colony of Cooperating Agents, *IEEE Transactions on Systems, Man and Cybernetics–Part B*, vol.26, no.1, pp.29–41,1996.
11. M. Dorigo, M. Birattari and T. Stützle, Ant Colony Optimization-Artificial Ants as a Computational Intelligence Technique, *IEEE Computational Intelligence Magazine*, 28–39, 2006.
12. K. M. Passino, Bacterial Foraging Optimization, *International Journal of Swarm Intelligence Research*, vol.1, no.1, pp.1–16, 2010.
13. V. Chris, E. Tsang, Guided Local Search, *Technical Report CSM-247*, Department of Computer Science, University of Essex, UK, 1995.
14. B. L. Webster, Solving Combinatorial Optimization Problems Using a New Algorithm Based on Gravitational Attraction, *Ph.D., thesis*, Melbourne, Florida Institute of Technology, 2004.
15. S. R. Balachandar and K. Kannan, Randomized Gravitational Emulation Search Algorithm for Symmetric

Traveling Salesman Problem, *Applied Mathematics and Computation*, vol.192, pp.413–421, 2007.
16. Y. T. Hsiao, C. L. Chuang, J. A. Jiang and C. C. Chien, A Novel Optimization Algorithm: Space Gravitational Optimization, *Proc. of 2005 IEEE International Conference on Systems, Man and Cybernetics*, vol.3, pp.2323–2328, 2005.
17. C. L. Chuang and J. A. Jiang, Integrated Radiation Optimization: Inspired by the Gravitational Radiation in the Curvature of Space-time, *Proc. of 2007 IEEE Congress on Evolutionary Computation*, pp.3157–3164, 2007.
18. E. Rashedi, Gravitational Search Algorithm, *Ph.D., thesis*, Shahid Bahonar University of Kerman, Kerman, Iran, 2007
19. R. Esmat, N. P. Hossein and S. Saeid, GSA: A Gravitational Search Algorithm, *Information Sciences*, vol.179, pp.2232–2248, 2009.
20. E. Rashedi, N. P. Hossein and S. Saeid, BGSA: Binary Gravitational Search Algorithm. *Natural Computing*. vol.9, pp. 727–745, 2010.
21. Q. Kang, L. Wang, and Q. Wu, A Novel Self-organizing Particle Swarm Optimization Based on Gravitation Field Model, *Proc. of the 2007 American Control Conference*, New York, USA, 2007.
22. R. Storn and K. Pcice, Differential Evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces, *Journal of Global Optimization*, vol.11, pp.341–359, 1997.
23. Q. W. Yang, L. Cai and Y. C. Xue, A Survey of Differental Evolution Algorithms, *Pattern Recognition and Artificial Intelligence*, vol.21 no.4, pp.506–513, 2008.
24. R. Mendes and A. S. Mohais, DynDE: a Differential Evolution for Dynamic Optimization Problems, *Proc of the Congress on Evolutionary Optimization*, Edinburgh, UK.vol.3, pp.2808–2815, 2005.
25. Y. Lou, J. L. Li and Y. S. Wang. A Binary-Differential Evolution Algorithm Based on Ordering of Individuals, *Proc. of the International Conference on Natural Computation*. vol.5, pp. 2207–2211, 2010.
26. Z. Y. Yang, K. Tang and X. Yao, Self-adaptive Differential Evolution with Neighborhood Search, *Proc. of the 2008 IEEE Congress on Evolutionary Computation*, pp.1110–1116, 2008.
27. F. J. Solis and R. T. Wets, Minimization by Random Search Techniques. *Mathematics of Operations Research*, vol.6, no.1, pp. 19–30, 1981.