# Elite-Guiding Binary Differential Evolution

Gening Xu[1, a], Xingfeng Wang[2,b*]

[1]Taiyuan University of Science and Technology, Taiyuan 030024, Shanxi, China

[2]Taiyuan University of Science and Technology, Taiyuan 030024, Shanxi, China

[a]xugening@sina.com, [b]wangxingfeng08@126.com

**Keywords:** differential evolution; discrete variables; elite-guiding.

**Abstract.** Differential evolution (DE) is an algorithm highly effective in solving problems with continuous variables while not applicable to problems with discrete variables. To overcome this disadvantage an elite-guiding binary differential evolution (EGBDE) is proposed. Inspired by the mutation with GA which could be dominated by the genes of the fittest chromosomes, a similar process is added to replace the original mutation of DE. Through simulations of test functions and Knapsack Problem (KP) EGBDE is proved to be both feasible and efficient in optimizations of problems with continuous and discrete variables.

## Introduction

Differential evolution (DE) is an evolutionary algorithm that operates with differences between individuals. Through the cooperation and competence of individuals an optimal solution of problems is found. Compared with other evolutionary algorithms, DE outstands not only in its extreme robustness in optimizations of non-convex problems, multimodal problems and nonlinear problems, but also in its fast convergence speed and simplicity in programming.

Currently researches of DE focus mainly on continuous optimization problems. Considering the many advantages of DE and the massive existence of discrete optimization problems, such as Knapsack Problem(KP), travelling salesman problems(TSP) and vehicle route planning, it definitely would be great if DE were applicable to the optimization of discrete problems.

DE is essentially an elite-oriented greedy genetic algorithm. The mutant vector of DE functions as a disturbance to individuals and with disturbances added to individuals solutions with better fitness might be expected. So the crux of DE is in formulating an appropriate mutant vector. And the difficulty of applying DE to discrete problems is also in maintaining the disturbance ability of the mutant vector. Yichao He[1] proposed a hybrid-encoding method by representing an individual with both a real code and a binary code. The real code is used in producing the mutant vector before mapping the vector with the sigmoid function to the binary code. Such a strategy adds to the complexity of the algorithm and an optimal solution is not always available. Zhifeng Wu[2] put forward a binary-encoding DE which regards the Hamming distances of individuals as the mutant vector. Since Hamming distance is not exactly the real-code distance, it is inappropriate to represent the distinction of individuals with the Hamming distance. Zhigang Wang[3] raised a binary differential evolution in which the majority-voting rule is adopted while mutation. Yet this approach is flawed in its randomness while selecting disturbances.

This paper comes up with an elite-guiding binary differential evolution (EGBDE), which features a unique mutation strategy. In genetic algorithms, the mutant vector could be achieved by comparing every gene of the fittest individual to the corresponding gene of the selected individuals and the bits from the first different bit to the end of the gene are then randomly set. The whole process is named elite guiding. The elite guiding mutation is adopted in EGBDE. With such a strategy the individuals could be drawn to the finest solutions. Through simulation of test functions and Knapsack Problem, the effectiveness of EGBDE is confirmed.

 

**Differential Evolution**

Differential evolution is a real-code evolutionary algorithm. The initial individuals are randomly distributed, and each of them being a vector in the searching space. Suppose $x_i(g)$ denotes the *ith* individual from the *gth* iteration, and $x_i(g)$ falls between the range $\left[x_i^L, x_i^U\right]$, then

$$x_i(g)=\left(x_{i1}(g), x_{i1}(g),\ldots, x_{in}(g)\right), i=1,2,\ldots,NP; g=1,2,\ldots,T_{\max}. \quad (1)$$

where $x_i^U$, $x_i^L$ represent the upper and lower limits of the searching space, respectively, and *NP* is the size of the population and $T_{max}$ is the maximum iteration.

The basic procedures of DE is as follows:

(1) Generate the initial population. The initial individuals are selected from the searching space according to

$$x_{ij}(0)=x_i^L+rand(0,1)\left(x_i^U-x_i^L\right). \quad (2)$$

(2) Mutation. The mode of mutation varies with different DE algorithms. The most common one is DE/rand/1/bin, which goes by randomly choosing three individuals out of the population $x_{p1}$, $x_{p2}$, $x_{p3}$ and $p_1 \neq p_2 \neq p_3 \neq i$ and then the mutant is derived

$$h_{ij}(g)=x_{p1j}+F\left(x_{p2j}-x_{p3j}\right). \quad (3)$$

where F is a scaling factor.

(3) Cross-over. Through cross-over, a population with more diversities than previous generation could be expected:

$$v_{ij}(g+1)=\begin{cases} h_{ij}(g), rand(0,1)\le p_c \ or \ j=rand(1,n) \\ x_{ij}(g), rand(0,1)< p_c \ or \ j\neq rand(1,n) \end{cases}. \quad (4)$$

where $p_c$ is the chance of individuals being crossed over and $p_c \in [0,1]$; rand(1,n) signals an integer between 1 and n, which is a guarantee that at least one element from the mutant could be chosen.

(4) Selection. The fitness function is used in the judgment of a finer offspring. Assume that the problem in question is a minimum problem, and then the selection can be expressed:

$$x_i(g+1)=\begin{cases} v_i(g+1), \ f\left(v_i(g+1)\right)< f\left(x_i(g)\right) \\ x_i(g), \ f\left(v_i(g+1)\right)\ge f\left(x_i(g)\right) \end{cases}. \quad (5)$$

The iteration loops until the maximum iteration is reached or the allowed tolerance is satisfied.

**Elite-Guiding Binary differential evolution**

The elite-guiding binary differential evolution (EGBDE) is unique in its mutation, which is a reference to the mutation in genetic algorithms. The mutation steps of EGBDE are:

(1) Calculate the fitness of the population and pick out the *elite(g)*;
(2) Randomly choose an individual $x_i(g)$;
(3) Compare the homologous genes of *elite(g)* and $x_i(g)$ from the first bit to the end till the first different bit nth is encountered;

(4) Randomly set the values of the bits from the nth bit to the end of the gene;
(5) Repeat the steps 3)~4) for the remaining homologous genes in the *elite(g)* and $x_i(g)$.

The elite-guiding mutation is essentially a search in an expanded space that covers the values of the genes in *elite(g)* and $x_i(g)$. For example, the genes 11001100 and 11011000, which represent 204 and 216 in real code. When processing by the proposed mutation method, the first three bits remains unchanged and the remaining five bits are randomly set. The result of the mutation is a gene ranging from 11000000 and 11011111, or a number between 192 and 255, which is apparently a range that contains 204 and 216.

The distinction between the elite-guiding mutation and the Hamming-distance based mutation can be found in a next example. Consider the two pairs of genes, the first pair 1100 and 1000 and the second pair 1100 and 0100. With the hamming-distance based mutation, an exactly the same process will be done to the two pairs. Yet with the elite-guiding mutation, a search between 8 and 15 will be made for the first pair and a search between 0 and 15 for the second pair. Obviously the elite-guiding mutation is a strategy that makes more sense.

**Simulation of test functions**

To explore the efficiency of EGBDE, simulations of test functions are carried out using EGBDE, BDE[2] and a binary particle swarm optimization[3](BPSO). These test functions, whose optimal values are all zeros and these values are met only when the variables are set zeros, are as follows:
(1) Sphere function:

$$f_1(x)=\sum_{i=1}^{n} x_i^2,\ -2.048\le x_i\le 2.048,\ n=3. \tag{6}$$

(2) Rosenbrock function

$$f_2(x)=\sum_{i=1}^{n}\left(100\left(x_{i+1}-x_i^2\right)^2+\left(1-x_i\right)^2\right),\ -2.048\le x_i\le 2.048,\ n=3. \tag{7}$$

(3) Schaffer function：

$$f_3(x_1,x_2)=0.5+\frac{\sin^2\sqrt{x_1^2+x_2^2}-0.5}{\left(1+0.001\left(x_1^2+x_2^2\right)\right)^2},\ -100\le x_i\le 100,\ n=2. \tag{8}$$

(4) Griewank function

$$f_4(x)=\sum_{i=1}^{n}\frac{x_i^2}{4000}-\prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right),\ -300\le x_i\le 300,\ n=30. \tag{9}$$

For EGBDE and BDE, the computing accuracy is set 0.01, and the size of the swarm is 100, with a maximum iteration of 150. The arguments for BPSO is as given by Kennedy J.Eberhart[4].

Each function is run 30 times. Thus the mean best fitness (MBF) and the standard deviation (SD) can be derived to precisely depict the efficiencies of these algorithms. Table 1 is an illustration of the comparison.

Table 1 Comparison of EGBDE, BDE and BPSO

| Test Functions | Algorithms | MBF | SD |
|---|---|---|---|
| Sphere | EGBDE | 0.000 000 | 0.000 000 |
| | BDE | 0.001 033 | 0.000 003 |
| | BPSO | 0.000 006 | 0.000 007 |
| Rosenbrock | EGBDE | 0.000 059 | 0.000 000 |
| | BDE | 0.026 133 | 0.002 499 |
| | BPSO | 0.000 256 | 0.000 275 |
| Schaffer | EGBDE | 0.002 743 | 0.000 018 |
| | BDE | 0.013 449 | 0.000 224 |
| | BPSO | 0.000 001 | 0.000 003 |
| Griewank | EGBDE | 1.470 977 | 0.105 469 |
| | BDE | 9.413 467 | 3.074 090 |
| | BPSO | 37.033 964 | 4.886 305 |

According to Table 1, the performance of EGBDE is overwhelmingly better than that of BDE, which is a best proof of the superiority of the elite-guiding mutation to the majority voting rules. And compared with BPSO, EGBDE demonstrates a higher searching accuracy and stability in all these test functions, with the only exception of Schaffer.

**Simulation of Knapsack Problems**

The Knapsack Problem (KP) is a maximization of the total value of the items packed in the knapsack while constrained by the knapsack's capacity. The mathematical model is as follows:

$$\max\ f(x_1, x_2, \ldots, x_n) = \sum_{i=1}^{n} x_j p_j$$

$$s.t. \begin{cases} \sum_{i=1}^{n} s_j x_j \le C \\ x_j \in \{0,1\},\ j = 1,2,\ldots,n \end{cases}. \tag{10}$$

An greedy algorithm is applied in the processing of the constrains. All items are ranked according to their value-volume rate, and for the solutions that exceed the given capacity, drop the item with the least value-volume rate until the total volume comes down below the limit.

Sample data, provided by Zhang Lin[5], are adopted in the simulation. These data are listed as below:

$P = \{220, 208, 198, 192, 180, 180, 165, 162, 160, 158, 155, 130, 125, 122, 120, 118, 115, 110, 105, 101, 100, 100, 98, 96, 95, 90, 88, 82, 80, 77, 75, 73, 70, 69, 66, 65, 63, 60, 58, 56, 50, 30, 20, 15, 10, 8, 5, 3, 1, 1\}$

$S = \{80, 82, 85, 70, 72, 70, 66, 50, 55, 25, 50, 55, 40, 48, 50, 32, 22, 60, 30, 32, 40, 38, 35, 32, 25, 28, 30, 22, 50, 30, 45, 30, 60, 50, 20, 65, 20, 25, 30, 10, 20, 25, 15, 10, 10, 10, 4, 4, 2, 1\}$

$C = 1000$

For comparison, BDE and the classical genetic algorithm are also included in the simulation. Figure 1 is the convergence curves of EGBDE, BDE and GA.

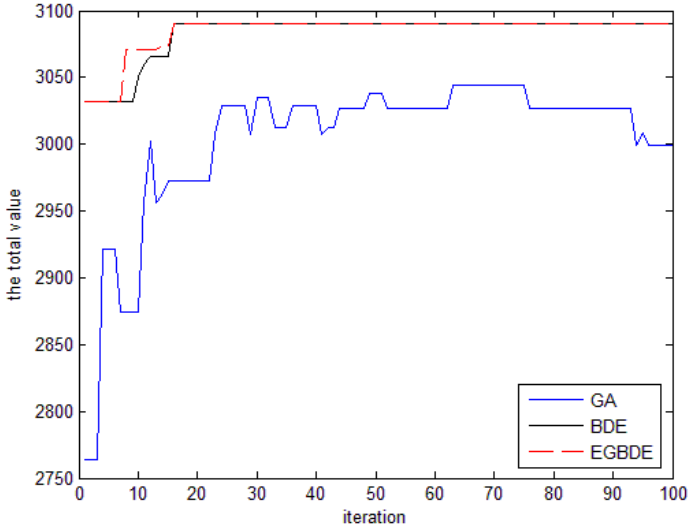


Figure 1 convergence curves of GA, BDE and EGBDE

Table 2 Comparison of GA, BDE and EGBDE

| Test Function | Max Iteration | MBF | SD | Best solution of 50 times ( total value/total volume ) |
|---|---|---|---|---|
| GA | 100 | 3049.7 | 335.8547 | 3090/1000 |
| BDE | 20 | 3082.9 | 80.9963 | 3090/1000 |
| EGBDE | 20 | 3080.6 | 99.8269 | 3090/1000 |

Table 2 is based on performances of the above three algorithms, with each run 50 times. Judging from Figure 1, BDE and EGBDE converge faster than GA, which is very reasonable. And the total values produced by BDE and EGBDE are remarkably higher than GA does. Yet EGBDE is slightly less efficient than BDE in solving Knapsack Problems, which is due to the fact that every gene is only represented by a single bit, zero or one, and with the elite-guiding mutation, the genes would stay unchanged or be set randomly. The power of elite-guiding simply vanishes here.

## Conclusion

An elite-guiding binary differential evolution (EGBDE) is proposed in the paper and simulations of test functions and Knapsack Problems are conducted. Through comparisons with traditional binary-code algorithms such as BDE, BPSO and GA, the efficiency and stability of EGBDE is fully proven. Yet similar to the real-code differential evolution, EGBDE is flawed in prematurity and a slow convergence speed in later iterations.

## References

[1] Yichao He, A hybrid-encoding binary differential evolution, Research and Development of Computer. 9 (2007) pp.1476-1484.

[2] Zhifeng Wu, Differential evolutions and their applications, Doctor's thesis, Beijing Jiaotong University (2009).

[3] Zhigang Wang, Binary differential evolution and its applications, Research and Application of Computer Engineering, Vol.18 (2008), pp. 48–50.

[4] Kennedy J.Eberhart, A discrete binary version of the particle swarm optimizer [C]. IEEE International Conference on Computational Cybernetics and Simulation, 1997,5:4104-4108.

[5] Lin Zhang, Good-point set genetic algorithm, Chinese Journal of Computers, 24 (2001), pp. 917–922.