Applied Mathematics & Information Sciences
An International Journal

http://dx.doi.org/10.18576/amis/100538

A New Modified Binary Differential Evolution Algorithm and its Applications

Ali Wagdy Mohamed[1,2,*]

[1] Statistics Department, Faculty of Sciences, King Abdulaziz University, P.O.Box80203, Jeddah 21589, Saudi Arabia.
[2] Operations Research Department, Institute of Statistical Studies and Research, Cairo University, Giza 12613, Egypt.

Received: 20 Feb. 2015, Revised: 21 Dec. 2015, Accepted: 24 Dec. 2015
Published online: 1 Sep. 2016

Abstract: This paper proposes a novel discrete version of Differential Evolution (NBDE) algorithm to solve combinatorial optimization problems with binary variables. A new binary mutation rule is introduced derived from the table of the basic DE mutation strategy and the value of scaling factor F is 1. The eight different combinations of the three randomly selected individuals using binary encoding are deduced. The developed mutation operator enables NBDE to explore and exploit the search space efficiently and effectively which are verified in applications to discrete optimization problems. Numerical experiments and comparisons on One-Max problem and Knapsack problem with two different sizes demonstrate that NBDE outperforms other existing algorithms in terms of final solution quality, search process efficiency and robustness.

Keywords: differential evolution, discrete optimization, binary mutation rule

1 Introduction

Differential Evolution (DE), first proposed by Storn and Price in 1995 [1], is a population-based stochastic optimization technique. The standard DE algorithm is simple, reliable, and robust and is relatively considered a new optimization method [2]. DE uses a simple and effective mechanism that works well on a wide continuous optimization problems. During the evolution, DE uses the difference between different individuals with certain scale factor to generate the new offspring. And it uses greedy selection criterion to determine which of the rivals to remain in the next generation.Therefore, DE has received wide attention in recent years. Besides, many different improved versions have been developed [3,4,5, 6]. However, DE, just as most popular evolutionary algorithms, has been originally designed to solve real-parameter optimization problems. Hence, it is not able to solve discrete combinatorial optimization problems. As a result, to counter this deficiency, many researchers have proposed several binary DE variants[7,8, 9,10,11]. The remainder of this paper is organized as follows: In section 2, the standard DE algorithm is introduced. Next, in section 3, the new BDE algorithm is described in detail. The Max-One problem and two knapsack problems are used to evaluate the DDE in section 4. Section 5 concldues the paper.

2 The differential evolution algorithm

A bound constrained global optimization problem can be defined as follows [12]:

$$\min f(x),\ X = [x_1, ..., x_n];\ S.t. x_j \in [a_j, b_j];\ j = 1, 2, ...n \quad (1)$$

Where f is the objective function,X is the decision vector consisting of n variables, and a_j and b_j are the lower and upper bounds for each decision variable, respectively. Virtually, there are several variants of DE [1]. In this paper, we use the scheme which can be classified using the notation as DE/rand/1/bin strategy [1, 13]. This strategy is the most often used in practice. A set of D optimization parameters is called an individual, which is represented by a D-dimensional parameter vector. A population consists of NP parameter vectors x_i^G, $i = 1, 2, ..., NP$. G denotes one generation. NP is the number of members in a population. It is not changed during the evolution process. The initial population is

* Corresponding author e-mail: aliwagdy@gmail.com

chosen randomly with uniform distribution in the search space. DE has three operators: mutation, crossover and selection. The crucial idea behind DE is a scheme for generating trial vectors. Mutation and crossover operators are used to generate trial vectors, and the selection operator then determines which of the vectors will survive into the next generation [14].

2.1 Initialization

In order to establish a starting point for the optimization process, an initial population must be created. Typically, each decision parameter in every vector of the initial population is assigned a randomly chosen value from the boundary constraints:

$$x_{ij}^0 = a_j + rand_j \cdot (b_j - a_j) \quad (2)$$

Where i indexes the population,$rand_j$ denotes a uniformly distributed number between [0, 1], generating a new value for each decision parameter. a_j and b_j are the lower and upper bounds for the jth decision parameter, respectively.

2.2 Mutation

For each target vector x_i^G, a mutant vector v_i^{G+1} is generated according to the following:

$$v_i^{G+1} = x_{r_1}^G + F * (x_{r_2}^G - x_{r_3}^G), r_1 \neq r_2 \neq r_3 \neq i \quad (3)$$

with randomly chosen indices and $r_1, r_2, r_3 \in \{1, 2, ..., NP\}$. Note that these indices must be different from each other and from the running index i so that NP must be at least four. F is a real number to control the amplification of the difference vector $(x_{r_2}^G - x_{r_3}^G)$. According to [14], the range of F is in [0, 2]. If a component of a mutant vector goes off the search space, then the value of this component is generated a new using (2).

2.3 Crossover

The target vector is mixed with the mutated vector, using the following scheme, to yield the trial vector u_i^{G+1}

$$u_{ij}^{G+1} = \begin{cases} v_{ij}^{G+1}, rand(j) \leqslant CR or j = randn(i), \\ x_{ij}^G, rand(j) > CR and j \neq randn(i), \end{cases} \quad (4)$$

Where, $j = 1, 2, ..., D$, $rand(j) \in [0, 1]$ is the jth evaluation of a uniform random generator number. $CR \in [0, 1]$ is the crossover probability constant, which has to be determined by the user. $randn(i) \in \{1, 2, ..., D\}$ is a randomly chosen index which ensures that u_i^{G+1} gets at least one element from v_i^{G+1}; otherwise no new parent vector would be produced and the population would not alter.

2.4 Selection

DE adapts a greedy selection strategy. If and only if the trial vector u_i^{G+1} yields as good as or a better fitness function value than x_i^G, then u_i^{G+1} is set to x_i^{G+1}. Otherwise, the old vector x_i^G is retained. The selection scheme is as follows (for a minimization problem):

$$x_i^{G+1} = \begin{cases} u_i^{G+1}, f(u_i^{G+1}) \leqslant f(x_i^G) \\ x_i^G \quad , otherwise. \end{cases} \quad (5)$$

3 Binary Differential Evolution Algorithm

In fact, it can be obviously seen mutation operator, from equation (3), that it can only keep the closing in the field of real-parameter optimization problems [11]. Nonetheless, it cannot keep the closure when it is applied in the discrete domain. Accordingly, in order to solve discrete-parameter optimization problems, a new mutation scheme must be proposed to expand DE into discrete binary space. In the proposed discrete binary differential evolution algorithm (NBDE), the cross over operations and the selection operations are the same as the original DE algorithm. However, the new generation of initial population and the proposed novel mutation operation defined by formulas (2) and (3), respectively, are as follows.

3.1 Initialization of binary population

In order to establish a starting point for the optimization process in the binary space, an initial population must be created by the following way $x_{ij}^0 = round(rand(0, 1))$. The function round (rand(0,1)) is used to generate a binary number.

3.2 Binary mutation operation

Really, the novel binary mutation operator is based on the original mutation of basic DE algorithm and the value of F is 1. As can be seen from mutation operation defined by formulas (2), there are three different vectors and each one of them can take two possible values 0 or 1 in the binary space. Thus there are 3^2 possible results of the original DE mutation operator in binary space directly are listed in table 1. Simply, there are eight kinds of different combinantions of the mutually three different vectors in the mutation operation. These eight combinantions can be classified into two cases.Each case has four kinds of combinations. The results of two cases can be derived as follows.

(a) Case (1): If x_{r2} equals x_{r3}, then the result of mutation equals to x_{r1}. All results of case (1) are shown in table 2.

(b) Case (2): If x_{r2} is different from x_{r3}, then the result of mutation equals to x_{r2}. Taking into consideration that -1 and 2 values are rounded independently to 0 and 1,respectively , as they are the nearest binary values for each value. all results of case (2) are shown in table 3.

Consequantly, with the above mentioned two cases, the proposed binary mutation operator is depicated as follows.

$$v_i^{G+1} = \begin{cases} x_{r1}^G \,, \; if \; x_{r2}^G = x_{r3}^G \\ x_{r2}^G \,, \; if \; x_{r2}^G \neq x_{r3}^G \end{cases} \tag{6}$$

Table 1: All results of original DE operator on binary level when the value of is 1.

x_{r1}	x_{r2}	x_{r3}	F	results
0	0	0	1	0
0	0	1	1	-1
0	1	0	1	1
0	1	1	1	0
1	0	0	1	1
1	0	1	1	0
1	1	0	1	2
1	1	1	1	1

Table 2: All results of case 1 when x_{r2} equals x_{r3}.

x_{r1}	x_{r2}	x_{r3}	F	results
0	0	0	1	0
0	1	1	1	0
1	0	0	1	1
1	1	1	1	1

Table 3: All results of case 2 when x_{r2} is different from x_{r3}.

x_{r1}	x_{r2}	x_{r3}	F	results
0	0	1	1	-1
0	1	0	1	1
1	0	1	1	0
1	1	0	1	2

4 Numerical Experiments and Comparisons

4.1 Benchmark functions

In order to evaluate the performance and show the efficiency and superiority of the proposed algorithm (NBDE), MAX-ONE problem [15] and two knapsack problems with different sizes[16] are used. All these functions are maximization problems. Definitions of the Benchmark Problems are as follows: 1) MAX-ONE problem

The aim of a MAX-ONE problem [15] is simply to maximize the ones in a binary string. The fitness of a string is the number of ones it has. The string length 100 is used for this study, with optimum 100.

2) ZERO-ONE Knapsack problem

The 0-1 knapsack problem is always defined as follows [1]: Suppose there are n objects and a knapsack. Each object i has a weight w_i, and a profit p_i, and the knapsack has a weight limit C. The problem is to choose a subset of the objects such that their overall profit is maximized, while the overall weight does not exceed a given capacity C. Then, the knapsack problem can be formulated as [16]:

$$Max\, f(x_1,\, x_2\,, ...,\, x_n) = \sum_{i=1}^{n} p_i\, x_i$$

$$s.t. \begin{cases} \sum\limits_{i=1}^{n} w_i x_i \leqslant C \\ x_i \in \{0,\, 1\}\, (i = 1, 2, ...n) \end{cases}$$

Where x_i is the variable denoting whether object i is chosen or not. Without loss of generality it may be assumed that all profits and weight are positive, that all weights are smaller than the capacity C, and that the overall weight of the objects exceeds C.

The data of the two selected 0-1 knapsack problems are as fellows.

KPl: n=20

P=92, 4, 43, 83, 84,68,92,82,6,44,32, 18,56,83,25,96,70,48, 14, 58

W=44,46, 90, 72, 91,40, 75, 35, 8,54, 78,40, 77,15,61, 17, 75, 29, 75, 63

C=878.

KP2: n=50

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, 72, 70, 69, 66, 65, 63, 60, 58,56,50,30,20,15,10,8,5,3, I

W=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, 25, 30, 45,30,60,50,20,65,20,25,30, 10,20,25, 15, 10, 10, 10,4,4,2, I

C=1000.

4.2 Algorithms for comparisons

The parameters of the novel NBDE is presented in Table 4.

Table 4: Parameters of NBDE for the three problems.

Examples	Population Size	CR	Maximum Generation
MAX-ONE	40	0.5	125
Kp1	40	0.5	75
Kp2	40	0.5	750

In order to evaluate the benefits of the proposed modifications ,a comparison of NBDE with two DE methods is done. These approaches are BDE [11] and

bdeTA [8].The parameters of the BDE and bdeTA algorithm are presented in Table 5.

Table 5: Parameters of BDE and bdeTA for the three problems.

Examples	Population Size	CR	Maximum Generation
MAX-ONE	50	0.15	500
Kp1	50	0.15	100
Kp2	50	0.15	600

4.3 Experimental results and discussions

For each problem, 50 independent runs are performed and statistical results are provided including The best results (Best), average results (Avg), worst results (Worst) , the standard deviation (SD) and Total Number of Function Evaluations(TNFE) of NBDE , BDE and bdeTA for the MAX-ONE and two Knapsack problems are presented in tables 6 , 7 and 8, respectively. The results provided by these two approaches were directly taken from [11].

Table 6: The Statistical Results of MAX-ONE problem.

Algorithms	Best	Avg	Worst	SD	TNFE
NBDE	100	100	100	0	5000
BDE	50	99.84	99	0.3703	25000
bdeTA	71	68.86	67	1.1608	25000

Table 7: The Statistical Results of Kp1.

Algorithms	Best	Avg	Worst	SD	TNFE
NBDE	1042	1042	1042	0	3000
BDE	1042	1040.9	1037	2.0923	5000
bdeTA	1042	1017.2	970	19.5809	5000

Table 8: The Statistical Results of Kp2.

Algorithms	Best	Avg	Worst	SD	TNFE
NBDE	3119	3117.92	3112	2.0388	30000
BDE	3119	3118.3	3113	1.9672	30000
bdeTA	3097	3062	3023	15.6616	30000

The presented results in Tables 6, 7 and 8 show that NBDE is able to find the global optimal solution consistently in MAX-ONE and Kp1 over 50 runs. Moreover, it is superior to the BDE and bdeTA in Best, Avg , Worst and SD performances. Additionally, it can be observed that the NBDE algorithm requires less computational effort than the other two algorithms. Therefore, it is considered the fastest one with the least (TNFE) in addition to its extreme efficiency and robustness with the smallest standard deviations. For Kp2, the statistical results provided by NBDE are similar to the statistical results obtained by BDE. However, both of them are superior to bdeTA in Best, Avg, Worst and SD performances with the same used (TNFE). For the MAX-ONE and two Knapsack problems, the convergence graphs of the average best fitness against Total Number of Function Evolutions (TNFE) of the NBDE are shown in Fig.1.

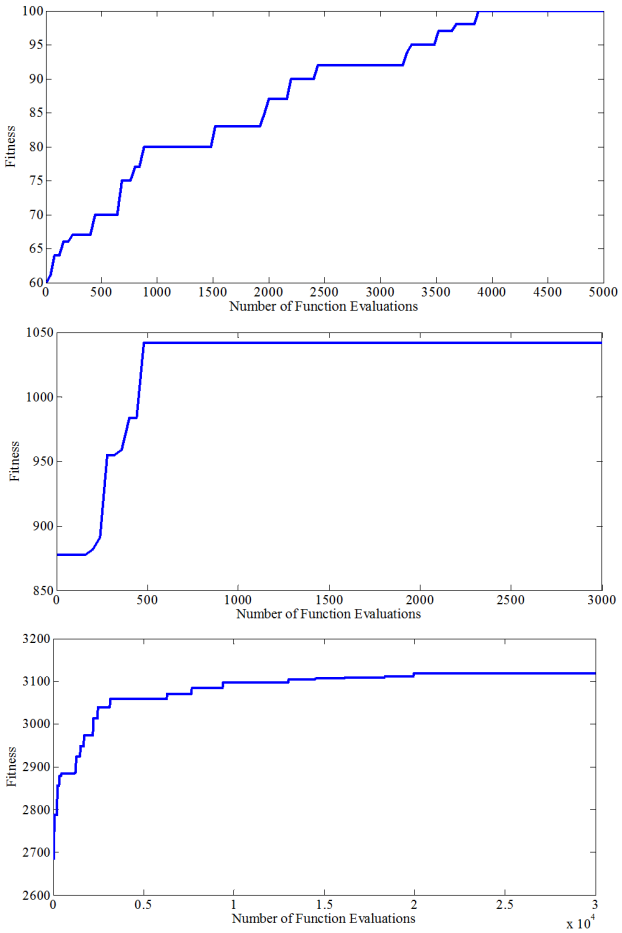

Fig. 1: Convergence graph of NBDE algorithm on MAX-ONE, Kp1 and Kp2 problems.

5 Conclusion

This study represents a significant step and a considerable trend towards the progress of existing intelligent approaches to solve combinatorial optimization problems in binary search space. In this paper, a novel discrete version of Differential Evolution (NBDE) algorithm to solve combinatorial optimization problems with binary variables is proposed. A new binary mutation rule is introduced derived from the table of the basic DE mutation strategy and the value of scaling factor F is 1.

The binary benchmark problems of MAX-ONE and Knapsack problems are used to test this new binary DE algorithm. The experimental results and comparisons have shown that the DDE algorithm performs better in combinatorial optimization problems with different types, complexity and dimensionality; it performs better with regard to the search process efficiency, the final solution quality and robustness, when compared with other algorithms. Finally, the performance of the NBDE algorithm is superior to and/or competitive with other binary DE (BDE and bdeTA) algorithms. Current research effort focuses on how to control the cross over factor by self-adaptive mechanism that may improve the performance. Additionally, future research will investigate the performance of the NBDE algorithm in solving different combinatorial optimization problems with discrete, integer and binary variables as well as real world applications with more comparative works to provide a full view and possible extensions of the proposed NBDE algorithm.

Acknowledgements

This work was funded by the Deanship of Scientific Research (DSR), King Abdulaziz University , Jeddah, under grant No.(965-012-D1434). The author, therefore , acknowledge with thanks DSR technical and financial support.

References

[1] R. Storn, K. Price, Differential evolution a simple and efficient adaptive scheme for global optimization over continuous spaces, Technical report TR-95-012, ICSI, 1995.

[2] K. Price, R. Storn, J. Lampinen. Differential Evolution: A Practical Approach to Global Optimization. Berlin, Germany: Springer-Verlag; 2005.

[3] Mohamed, A. W. (2015). An improved differential evolution algorithm with triangular mutation for global numerical optimization. Computers & Industrial Engineering, 85, 359-375.

[4] A. W. Mohamed, H.Z. Sabry, An alternative differential evolution algorithm for global optimization, Journal of advanced research 3 (2012) 149-165.

[5] A.W. Mohamed, H.Z. Sabry, Constrained optimization based on modified differential evolution algorithm, information Sciences 194 (2012) 171-208.

[6] A. W. Mohamed, An Efficient Modified Differential Evolution Algorithm for Solving Constrained Non-linear Integer and Mixed-Integer Global Optimization Problems, International Journal of Machine Learning and Cybernetics (Springer). DOI 10.1007/s13042-015-0479-6.

[7] G. Pampar, A. P. Engelbrecht, and N. Franken, Binary Differential Evolution, IEEE Congress on Evolutionary Computation, Sheraton Vancouver Wall Centre, Vancouver, BC, Canada, 2006, pp. 18731879.

[8] T. Gong, and L. Tusion, "Differential Evolutionfor Binary Encoding," Soft Computing in Industrial Applications, 2007, pp.251-262.

[9] Q. Y. Yang, A comparative stude of discrete differential evolution on binary constraint satisfaction problems, Proc. of the Congress on Evolutionary Computation(CEC08) , pp.330-335.

[10] E. Zio, L. R. Golea, and G. Sansavini, "Optimizing protections against cascades in network systems: A modified binary differential evolution algorithm," Reliability Engineering and System Safety, vol.103, pp. 72-83, 2011.

[11] C. Deng; C. Liang; Y. Yang; B. Zhao; H. Zhang, "Binary Differential Evolution algorithm with new mutation operator," Intelligent Computing and Intelligent Systems (ICIS), 2010 IEEE International Conference on , vol.1, no., pp.498,501, 29-31 Oct. 2010

[12] Y. Xu , L.Wang ,L. Li , Emerging intelligent computing technology and applications. with aspects of artificial intelligence, in: D.-S. Huang, K.-H. Jo , H. H. Lee H.-J. Kang (Eds.), An Effective Hybrid Algorithm Based on Simplex Search and Differential Evolution for Global Optimization. Lecture Notes on Computer Science, vol. 5755, Springer-Verlag, Berlin-Heidelberg, (2009), pp. 341-350.

[13] A. W. Mohamed, Zaher, H., Real parameter optimization by An Effective Differential Evolution Algorithm, Egyptian Informatics Journal (Elsevier). 14 (2013). 37-53.

[14] A. W. Mohamed, RDEL: Restart Differential Evolution Algorithm with Local Search Mutation for Global Numerical optimization, Egyptian Informatics Journal (Elsevier). 3 (2014).

[15] Ackley, D.: A connectionist Machine for Genetic Hillclimbing.Kluwer Academic Publishers, Boston, MA (1987)

[16] Van Liu and Chao Liu.: A Schema-Guiding Evolutionary Algorithm for 0-1 Knapsack Problem. Iacsit-sc. In: 2009 International Association of Computer Science and Information Technology - Spring Conference, IEEE Press . 2009,160-164.

Ali Wagdy Mohamed is an Assistant Professor with Statistics Department, King Abdulaziz University, Jeddah, Saudi Arabia and he is an Assistant Professor with Operations Research Department, Institute of Statistical Studies and Research, Cairo University, Egypt. His prime research area is evolutionary computing and its application to various domains of engineering, constrained and unconstrained optimization and multi-objective optimization.