Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2016, Article ID 5732489, 12 pages
http://dx.doi.org/10.1155/2016/5732489

Research Article

Dichotomous Binary Differential Evolution for Knapsack Problems

Hu Peng,[1] Zhijian Wu,[2] Peng Shao,[3] and Changshou Deng[1]

[1]*School of Information Science and Technology, Jiujiang University, Jiujiang 332005, China*
[2]*State Key Lab of Software Engineering, School of Computer, Wuhan University, Wuhan 430072, China*
[3]*School of Computer and Information Engineering, Jiangxi Agricultural University, Nanchang 330045, China*

Correspondence should be addressed to Zhijian Wu; zhijianwu@whu.edu.cn

Received 12 June 2016; Revised 12 November 2016; Accepted 21 November 2016

Academic Editor: Dan Simon

Differential evolution (DE) is one of the most popular and powerful evolutionary algorithms for the real-parameter global continuous optimization problems. However, how to adapt into combinatorial optimization problems without sacrificing the original evolution mechanism of DE is harder work to the researchers to design an efficient binary differential evolution (BDE). To tackle this problem, this paper presents a novel BDE based on dichotomous mechanism for knapsack problems, called DBDE, in which two new proposed methods (i.e., dichotomous mutation and dichotomous crossover) are employed. DBDE almost has any difference with original DE and no additional module or computation has been introduced. The experimental studies have been conducted on a suite of 0-1 knapsack problems and multidimensional knapsack problems. Experimental results have verified the quality and effectiveness of DBDE. Comparison with three state-of-the-art BDE variants and other two state-of-the-art binary particle swarm optimization (PSO) algorithms has proved that DBDE is a new competitive algorithm.

1. Introduction

The knapsack problems, as one of the classical NP-hard combinational optimization problem, have a lot of immediate applications in project selection, resource distribution, investment decision-making, financial management, and so on. The 0-1 knapsack problem and multidimensional knapsack problem are the most common and important in the family of knapsack problems and have been extensively studied [1]. In recent decades, evolutionary algorithms, such as genetic algorithms (GA) [2–4], particle swarm optimization (PSO) [5, 6], and differential evolution (DE) [7, 8], have been well-adopted for solving the knapsack problems. Although many knapsack problems have been tackled by these approaches, some new and more difficult knapsack problems (such as high dimension knapsack problems) hidden in the real world have not been properly solved [9, 10]. So the research on them is still important and necessary.

DE, proposed by Storn and Price [11], has drawn the attention of many researchers all over the world. As one of the most popular and powerful optimization techniques, DE has many advantages, for example, fast, simple, easy to use, and efficient for the real-parameter global continuous optimization problems. Initially, DE was designed as an optimization technique for used in real-number spaces. As the emergence of DE, many researchers around the world study from different aspects and proposed a lot of improved DE variants [12–17]. However, many combinational optimization problems are set in a space featuring binary and the classical DE is not suitable for it. Therefore, a number of binary versions of DE (BDE) have been proposed. Some work is mainly dedicated to design a mapping from continuous-space to binary space. For example, in the early days of BDE research, Pampara et al. [18] proposed an angle modulated DE (AMDE), in which a trigonometric function was employed as a bit string generator. Thus AMDE can operate in binary spaces without deviating from the basic search mechanism of the classical DE. Hota and Pat [7] proposed an adaptive quantum-inspired DE (AQDE), in which mutation operator is similar to that of classical DE and operates directly on the Q-bit (θ) where θ is defined in $[0, 2\pi]$. Besides the abovementioned methods, many attempts as well focused on operating

on binary space directly. For example, He and Han [19] first use logical operators to replace the original operations of the mutation mechanism in DE, and thus the individual can be evolved directly using binary string. Gong and Tuson [20] proposed binary-adapted DE operators based on forma analysis to directly manipulate bit strings. Additionally, some BDE variants with new binary mutation operators, such as MBDE [21], NMBDE [8], and BLDE [22], are proposed one after another. However, we have noted that it is a challenging task to design an efficient BDE. If the BDE still evolves in the real-number spaces, the expensive computational cost of transforming the real-coded individuals into binary strings is a hard problem. Moreover, if the BDE evolves in the binary spaces directly, the weak exploration ability of directly operating bit-stings has definitely hindered the BDE to solve the difficult and high-dimensional knapsack problems.

In psychology, dichotomous thinking, also known as "black or white thinking," is a tendency to only see extremes. Based on dichotomous thinking, the result of executing XOR operation on any two binary strings can be divided into two types, that is, common and difference, and the mutation and crossover operations of BDE can also be divided into two types. If the result is common then we can perform one operation, else if it is difference then we can perform another heterogeneous operation. Motivated by these observations, we proposed a novel dichotomous binary differential evolution (DBDE), in which dichotomous mechanism is fused into the mutation and crossover operations. As the main contribution in DBDE, dichotomous mechanism is learning from the dichotomous thinking of psychology. The primary idea of dichotomous mechanism is that each of the two features (i.e., 0 and 1) after the XOR operation corresponds to two different strategies, the result increases the diversity of population and enhances the exploration ability in the binary search space. Comprehensive experiments have been conducted on a suite of 0-1 knapsack problems and multidimensional knapsack problems to verify the effectiveness and efficiency of the proposed approach.

The rest of the paper is organized as follows. In Section 2, the 0-1 knapsack problems and multidimensional knapsack problems are introduced. Section 3 gives a brief review of the DE algorithm. Our proposed DE variant, called DBDE, is described in Section 4. The analysis of solution quality of DBDE is presented in Section 5. Experimental results and discussions are presented in Section 6. Finally, the work is concluded in Section 7.

2. Knapsack Problems

The knapsack problem is one of typical NP-hard combinatorial optimization problems in operation research, which includes a variety of knapsack problems such as the 0-1 knapsack problems and multidimensional knapsack problems.

2.1. The 0-1 Knapsack Problems. Generally, in an instance of the 0-1 knapsack problem, given a set of n items, each item i having an integer profit p_i and an integer weight w_i, the task is to maximise the sum of profits of items packed in the knapsack without exceeding a given capacity C. Mathematically, the 0-1 knapsack problem can be formulated as follows:

$$\begin{aligned} \text{Maximize} \quad & f(X) = \sum_{j=1}^{n} p_j x_j \\ \text{s.t.} \quad & \sum_{j=1}^{n} w_j x_j \le C \\ & x_j \in \{0,1\}, \quad j = 1,2,\ldots,n, \end{aligned} \tag{1}$$

where $x_j = 1$ or 0 is used to indicate whether item j is included in the knapsack or not. It may be assumed that all profits and weights are positive, and that all weights are smaller than the capacity C but the total weight of all items exceeds C.

2.2. The 0-1 Multidimensional Knapsack Problems. The 0-1 multidimensional knapsack problem is similar to 0-1 knapsack problem, but where a set of knapsack constraints are satisfied rather than one. Therefore, it is more complex and more difficult to be solved. A comprehensive overview of practical and theoretical results for the 0-1 multidimensional knapsack problem can be found in [10]. The 0-1 multidimensional knapsack problem can be formulated as follows:

$$\begin{aligned} \text{Maximize} \quad & f(X) = \sum_{j=1}^{n} p_j x_j \\ \text{s.t.} \quad & \sum_{j=1}^{n} w_{k,j} x_j \le C_k, \quad k = 1,2,\ldots,m \\ & x_j \in \{0,1\}, \quad j = 1,2,\ldots,n, \end{aligned} \tag{2}$$

where m is the number of knapsack constraints with maximum capacities.

3. DE Algorithm

Since the emergence of DE algorithm, much effort has been made to improve the performance and these works can be classified into three categories. The first is hybridization with other techniques [23–25], the second is modification of the mutation strategy [26–28], and the third is adaptation of mutation strategy and parameter settings (i.e., NP, F, and CR) [29–31]. Additionally, some comparative study has shown that the performance of DE and its variants is better than the PSO variants over a wide variety of problems [16, 32]. As a very competitive form of evolutionary algorithm, DE has been successfully applied to diverse fields such as electrical power systems [33, 34], bioinformatics [35, 36], pattern and recognition, and image processing [37, 38]. Recently, Das et al. [16, 17] presented a comprehensive survey on the state-of-the-art of DE variants, so more information can be referred to it.

DE is a population-based heuristic search algorithm which maintains a population with NP floating-point encoded individuals $X_{1,G}, X_{2,G}, \ldots, X_{\mathrm{NP},G}$, where NP is the

population size, G is the generation number, and each member represents a candidate solution, randomly sampled from the search space. It starts with an initial population $X_{1,0}, X_{2,0}, \ldots, X_{\mathrm{NP},0}$ and then conducts mutation, crossover, and selection operators to improve its population generation by generation until the preset stopping criterion, that is, target error accuracy level (ε) or maximum number of function evaluates (MaxFEs), is satisfied.

At each generation G, DE executes mutation operation firstly. The mutant vector $V_{i,G}$ for each individual $X_{i,G}$ (or, namely, target vector) will be created by the mutation strategy. The well-known and widely used DE mutation strategies "DE/rand/1" are shown as follows:

$$V_{i,G} = X_{r1,G} + F \cdot \left(X_{r2,G} - X_{r3,G}\right), \tag{3}$$

where $i = 1, 2, \ldots, \mathrm{NP}$, $r1$, $r2$, and $r3$ are different indices randomly chosen from the set $\{1, 2, \ldots, \mathrm{NP}\}$ and all are different from i. The control parameter F is a scale factor that amplifies the vector difference.

Then, DE commonly conducts binomial crossover operator to recombine the target vector $X_{i,G}$ and mutant vector $V_{i,G}$. The trial vector $U_{i,G}$ is generated as follows:

$$u_{i,j,G} = \begin{cases} v_{i,j,G}, & \text{if } \mathrm{rand}_j \le \mathrm{CR} \text{ or } j = j_{\mathrm{rand}} \\ x_{i,j,G}, & \text{otherwise,} \end{cases} \tag{4}$$

where $j = 1, 2, \ldots, D$, $j_{\mathrm{rand}} \in \{1, 2, \ldots, D\}$ is a randomly selected integer, $\mathrm{rand}_j \in [0, 1]$ is a uniformly distributed random number for the jth dimension, and the control parameter $\mathrm{CR} \in [0, 1]$ is the crossover probability.

Finally, the selection operator is used to choose the better one among the target vector $X_{i,G}$ and the trial vector $U_{i,G}$ in terms of their fitness value to enter the next generation:

$$X_{i,G+1} = \begin{cases} U_{i,G}, & \text{if } f\left(U_{i,G}\right) > f\left(X_{i,G}\right) \\ X_{i,G}, & \text{otherwise.} \end{cases} \tag{5}$$

To alleviate the classical DE cannot be directly applied to binary-valued optimization problem, many versions of BDE were put forward around the world; especially logical operations were first introduced by He and Han [19] to replace the original operations of the mutation mechanism in DE and the binary mutation equation as follows:

$$V_{i,G} = X_{r1,G} \odot F \otimes \left(X_{r2,G} \oplus X_{r3,G}\right), \tag{6}$$

where $\otimes$ denotes the AND operator, $\odot$ denotes the OR operator, and $\oplus$ denotes the XOR operator.

4. Dichotomous Binary Differential Evolution

Dichotomous binary differential evolution (DBDE) as proposed in this paper is based on both binary string representation and the dichotomous mechanism. The details are as follows.

4.1. Dichotomous Mutation. The idea of using logical operations to replace the original operations of the mutation, such as subtraction, multiplication, and addition, was first introduced by He and Han [19]. This approach can evolve the bit string individual directly. However, by using the binary strings, there are only two different codes "0" and "1" in the population; the differential of two individuals on the same location is too negligible to operate the complicated mutation like the real coding individuals. The novel binary mutation mechanism developed in this paper is based on XOR logical operation mainly. As we all know, the bit coded as "0" after the XOR operation represents the common between the two selected bits; otherwise the "1" represents difference. According to the common and difference feature patterns of two randomly selected individuals, dichotomous mutation executes difference operations and the new mutation equation is as follows:

$$\begin{aligned} v_{i,j,G} = &\left(\left(x_{r1,j,G} \oplus x_{r2,j,G}\right) \otimes \mathrm{rand}\right) \\ &\odot \left(!\left(x_{r1,j,G} \oplus x_{r2,j,G}\right) \otimes x_{r1,j,G}\right), \end{aligned} \tag{7}$$

where ! denotes the NOT operator. In dichotomous mutation, the scale factor F disappeared and we no longer have to worry about its value. If the bits between $x_{r1,j,G}$ and $x_{r2,j,G}$ are difference, then the mutation value is randomly chosen from "0" or "1"; otherwise, if the bits between $x_{r1,j,G}$ and $x_{r2,j,G}$ are common, then the mutation value is determined by the value of $x_{r1,j,G}$.

As a simple example, as depicted in Figure 1, two individuals $X_{r1,G}$ and $X_{r2,G}$ are randomly selected at generation G for ith individual $X_{i,G}$ to execute the dichotomous mutation. Then a binary string $(X_{r1,G} \oplus X_{r2,G})$ is produced after the XOR operation of $X_{r1,G}$ and $X_{r2,G}$. On this basis, the mutation binary string $V_{i,G}$ is obtained by the dichotomous mutation mechanism. In Figure 1, $V_{i,G}$ composes rand, $x_{r1,1,G}$, rand, and $x_{r1,n,G}$ on the 1st, 2nd, 3rd, and nth location, respectively, in which "rand" represents that the value is randomly chosen from "0" or "1" with equal probability.

4.2. Dichotomous Crossover. The new binary crossover operator, that is, dichotomous crossover, is used to produce the trial individual $U_{i,G}$ by mixing the target individual $X_{i,G}$ and mutant individual $V_{i,G}$. The trial individual $U_{i,G}$ can be obtained according to the following equation:

$$\begin{aligned} u_{i,j,G} &= \begin{cases} v_{i,j,G}, & \text{if } \mathrm{rand}_j \le \mathrm{CR}_j \\ x_{i,j,G}, & \text{otherwise,} \end{cases} \\ \mathrm{CR}_j &= \begin{cases} \mathrm{CR1}, & \text{if } \left(x_{r1,j,G} \oplus x_{r2,j,G}\right) == 0 \\ \mathrm{CR2}, & \text{if } \left(x_{r1,j,G} \oplus x_{r2,j,G}\right) == 1, \end{cases} \end{aligned} \tag{8}$$

where $j = 1, 2, \ldots, D$. The mechanism of dichotomous crossover is similar to the crossover of the original DE, but there exists a main difference between them. That is, dichotomous crossover uses two crossover probabilities, that is, CR1 and CR2, rather than only one CR in classical DE. For jth bit of the trial individual, if the bits between $x_{r1,j,G}$ and

Input: Infeasible individual X; Weights w_i of each item i; Capacity C of a knapsack
(1) **while** X is infeasible **do**
(2) i = ratio-greedily select an item from the knapsack;
(3) $x_i = 0$;
(4) **if** $\sum_{i=1}^{n} x_i \cdot w_i \leq C$ **then**
(5) Return X;
(6) **end if**
(7) **end while**
Output: Feasible individual X

Algorithm 1: Ratio-greedy repair.

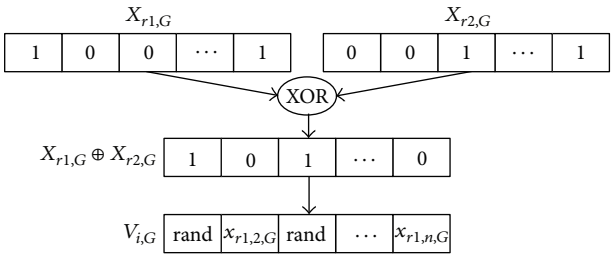

Figure 1: Illustration of the dichotomous mutation.

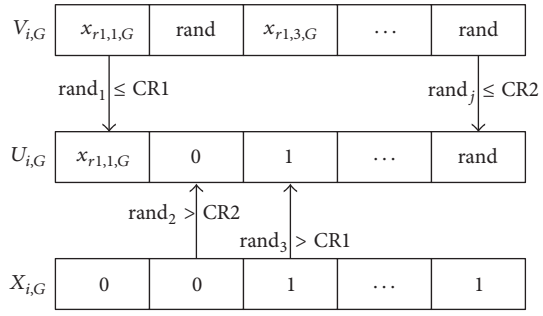

Figure 2: Illustration of the dichotomous crossover.

$x_{r2,j,G}$ are common, then CR_j is equal to CR1; otherwise, if the bits between $x_{r1,j,G}$ and $x_{r2,j,G}$ are difference, then CR_j is equal to CR2.

As demonstrated in Figure 2, the elements of trial individual $U_{i,G}$ come from mutate individual $v_{i,G}$ and target individual $x_{i,G}$ with difference crossover probabilities. From Figure 2, we can find that the decisions of the crossover operation for the 1st, 2nd, 3rd, and nth bit of $U_{i,G}$ are based on CR1, CR2, CR1, and CR2, respectively.

4.3. DBDE for Knapsack Problems. Handling the constraint in the knapsack problems and considering some empirical results indicate that repair method is the most efficient for the knapsack problems [3, 4]. Thus, only repair method with ratio-greedy manner is used in this paper to tackle the knapsack problem. The steps of the ratio-greedy repair are described in Algorithm 1, in which if the solution X is infeasible, we sort the items according to the descending order of the corresponding profit-to-weight ratios firstly and then choose an item with the smallest ratio and remove it from the knapsack, and the program will loop continuously until a feasible one is generated.

The framework of DBDE for knapsack problem is described in Algorithm 2. Compared to the original DE, there is merely two major differences in the DBDE. One is the using of ratio-greedy repair for the infeasible solution after the new individual generated. Another is that the new dichotomous mutation and dichotomous crossover replace the old one.

Table 1: Truth table of dichotomous mutation.

$x_{r1,j}$	$x_{r2,j}$	rand	$x_{r1,j} \oplus x_{r2,j}$	$v_{i,j}$
0	0	0	0	0
0	0	1	0	0
0	1	0	1	0
0	1	1	1	1
1	0	0	1	0
1	0	1	1	1
1	1	0	0	1
1	1	1	0	1

5. Analysis of Solution Quality of DBDE

In this section, we analyze the solution quality of DBDE on the 0-1 knapsack problem in terms of the approximation ratio [39, 41]. In DBDE, the dichotomous mutation, dichotomous crossover, and selection are performed one by one at each generation to guide the population to the global optimum. From the demonstration of truth table in Table 1, we can observe that the probabilities of bit "1" and bit "0" generated by dichotomous mutation are equal to 0.5. Meanwhile, based on dichotomous crossover probabilities CR1 and CR2 the mutant bit will be accepted as a trial bit in the trial vector and the probabilities of using CR1 and CR2 are equal to 0.5. Thus, dichotomous mutation and crossover flip bit from "0" to "1" with a probability $0.5 * 0.5 * (\text{CR1} + \text{CR2})$ and the same as from "1" to "0". For the purposes of simplification, let $q = 0.5 * 0.5 * (\text{CR1} + \text{CR2})$.

Theorem 1. *For any constant $\alpha \in (0, 1)$, the DBDE needs $\Omega(q^{-\alpha n})$ running time to find α-approximation solution in the worst case.*

Input: Population size, NP; Crossover probability, CR1 and CR2; Maximum number of objective function evaluations, MaxFEs
(1) Randomly initialize population P_0 with NP individuals
(2) **for** $i = 1 : \text{NP}$ **do**
(3) **if** $X_{i,0}$ is an infeasible individual **then**
(4) Execute Algorithm 1 for ratio-greedy repair
(5) **end if**
(6) Evaluate the objective function value $f(X_{i,0})$
(7) **end for**
(8) FEs = NP
(9) **while** FEs < MaxFEs **do**
(10) **for** $i = 1 : \text{NP}$ **do**
(11) Randomly select two individuals $x_{r1,G}$ and $x_{r2,G}$ from population P_G
(12) Execute the dichotomous mutation to generate a mutate individual $V_{i,G}$
(13) Execute the dichotomous crossover to generate a trial individual $U_{i,G}$
(14) **if** $U_{i,G}$ is an infeasible individual **then**
(15) Execute Algorithm 1 for ratio-greedy repair
(16) **end if**
(17) Evaluate the objective function value $f(U_{i,G})$
(18) **if** $f(U_{i,G}) > f(X_{i,G})$ **then**
(19) $X_{i,G} = U_{i,G}$
(20) **end if**
(21) **end for**
(22) FEs = FEs + NP
(23) **end while**
Output: Optimal individual with the maximum profit value

Algorithm 2: DBDE for knapsack problem.

Table 2: Instance 1 for analysis [39].

Item i	1	$2, \ldots, \alpha n$	$\alpha n + 1, \ldots, n$
Profit p_i	n	1	$1/n$
Weight w_i	n	$1/\alpha n$	n
Capacity		n	

Proof. According to the definition of an evolutionary approximation algorithm (see [41] for a detailed exposition), it suffices to consider the instance of the 0-1 knapsack problem, which is described in Table 2. In Table 2, αn is a large positive integer for a sufficiently large n. As seen, the global optimum for the instance is $(10\cdots0)$, $f(10\cdots0) = n$ and the global optimum is unique. A local optimum is

$$0\overbrace{1\cdots1}^{\alpha n-1}0\cdots0, f\left(0\overbrace{1\cdots1}^{\alpha n-1}0\cdots0\right) = \alpha n - 1. \quad (9)$$

Notice that the local optimum is the second largest objective function value among feasible solution. The ratio of fitness between the local optimum and the global optimum is

$$\frac{\alpha n - 1}{n} < \alpha. \quad (10)$$

Suppose that the DBDE starts at the local optimum $(01\cdots10\cdots0)$. For the individual X_i, the dichotomous mutation and crossover operators are conducted and a trial vector $U_{i,G}$ is generated. At the same time, the greedy selection operator with the ratio-greedy repair will prevent $U_{i,G}$ from entering into the next generation unless the trial vector is the global optimum itself. Thus, the event happens only if the first bit of X_i is flipped from $x_1 = 0$ to $x_1 = 1$ and $\alpha n-1$ one-valued bits are flipped into zero-valued ones while other zero-valued bits remain unchanged. The probability of this event is

$$q \times (q)^{\alpha n-1} \times (1-q)^{n-\alpha n} = O\left(q^{\alpha n}\right). \quad (11)$$

From the above analysis, we now deduce that the expected runtime to reach the global □

6. Experimental Study

6.1. Experimental Setting. Two types of knapsack problems, that is, 0-1 knapsack and multidimensional knapsack problems, are used in the following experimental studies to extensively investigate the performance of the proposed DBDE algorithm. Meanwhile, DBDE compares with five other state-of-the-art binary evolutionary algorithms, including three binary DE variants, that is, BLDE [22], BinDE [20], and AQDE [7], and two binary PSO variants, that is, BPSO [40] and MBPSO [6]. For all the contestant algorithms, the control parameter settings are the same as their original literature, but the MaxFEs is set to $100 * D$ and $\text{NP} = 100$ for fairness consideration. The parameters are used in DBDE, that is, $\text{CR1} = 0.2$ and $\text{CR2} = 0.5$. The details of the control parameters of these contestant algorithms are summarized in Table 3.

All the experiments are done on a computer with 2.8 GHz Dual-core Processor and 4 GB RAM under Windows 7 platform and all algorithms are implemented in Matlab 2010b.

Table 3: Parameter settings for the contestant algorithms.

Algorithm	Parameter settings
BLDE [22]	$p = \max(0.05, \min(0.15, 10/n))$
BinDE [20]	$F = 0.8$, CR = 0.5, DE/res/bin
AQDE [7]	$F = 0.1 * r_1 * r_2$, CR = $0.5 + 0.0375 * r_3$, $r_1, r_2 \sim U(0,1)$, $r_3 \sim N(0,1)$
BPSO [40]	$C = 2$, $V_{\max} = 6$
MBPSO [6]	$C = 2$, $V_{\max} = 4$
DBDE	CR1 = 0.2, CR2 = 0.5

Table 4: The detailed information of the 0-1 knapsack instances.

Number	Problem	Type	D
1	kp_uc_100	Uncorrelated	100
2	kp_uc_200	Uncorrelated	200
3	kp_uc_300	Uncorrelated	300
4	kp_uc_500	Uncorrelated	500
5	kp_uc_1000	Uncorrelated	1000
6	kp_wc_100	Weakly correlated	100
7	kp_wc_200	Weakly correlated	200
8	kp_wc_300	Weakly correlated	300
9	kp_wc_500	Weakly correlated	500
10	kp_wc_1000	Weakly correlated	1000
11	kp_sc_100	Strongly correlated	100
12	kp_sc_200	Strongly correlated	200
13	kp_sc_300	Strongly correlated	300
14	kp_sc_500	Strongly correlated	500
15	kp_sc_1000	Strongly correlated	1000
16	kp_ss_100	Subset sum	100
17	kp_ss_200	Subset sum	200
18	kp_ss_300	Subset sum	300
19	kp_ss_500	Subset sum	500
20	kp_ss_1000	Subset sum	1000

Thirty independent runs are carried out for each algorithm on each instance. The best results are shown in boldface. In order to have statistically sound conclusions, Wilcoxon's rank sum test at a 0.05 significance level is conducted on the experimental results.

6.2. 0-1 Knapsack Problem. A suite of twenty randomly generated 0-1 knapsack problems is used to verify the efficacy of DBDE. An algorithm for generating the instances is available from http://www.diku.dk/~pisinger/generator.c. The detailed descriptions of these instances are summarized in Table 4.

Table 5: Experimental results of DBDE and other five competitors on twenty 0-1 knapsack instances.

Problem	BPSO	MBPSO	BLDE	BinDE	AQDE	DBDE
kp_uc_100	**1807**	**1807**	**1807**	1732	**1807**	**1807**
kp_uc_200	3402	3378	3401	3146	3390	**3403**
kp_uc_300	5443	5344	5441	4836	5401	**5444**
kp_uc_500	9492	9145	9484	8058	9381	**9495**
kp_uc_1000	18829	17600	18492	15380	18467	**18843**
kp_wc_100	**659**	**659**	658	656	**659**	**659**
kp_wc_200	**1332**	1331	1329	1304	1328	**1332**
kp_wc_300	1961	1957	1961	1922	1958	**1963**
kp_wc_500	3246	3230	**3247**	3163	3236	**3247**
kp_wc_1000	**6478**	6401	6458	6252	6446	6463
kp_sc_100	**813**	**813**	812	786	812	**813**
kp_sc_200	1629	1617	1626	1552	1624	**1631**
kp_sc_300	**2433**	2402	2426	2307	2422	**2433**
kp_sc_500	4069	3982	4051	3807	4045	**4070**
kp_sc_1000	**8212**	7936	8073	7590	8137	8109
kp_ss_100	**493**	**493**	**493**	**493**	**493**	**493**
kp_ss_200	**1001**	**1001**	**1001**	**1001**	**1001**	**1001**
kp_ss_300	**1523**	**1523**	**1523**	**1523**	**1523**	**1523**
kp_ss_500	**2518**	**2518**	**2518**	**2518**	**2518**	**2518**
kp_ss_1000	**5068**	**5068**	**5068**	**5068**	**5068**	**5068**
$w/t/l$	7/2/11	13/0/7	13/0/7	15/0/5	12/1/7	—

The optimum solutions of these instances are not known. These instances can be divided into four groups, including uncorrelated, weakly correlated, strongly correlated, and subset-sum data instances. In all instances, the weights w_j are randomly distributed in $[1, R]$. The profits p_j are expressed as a function of the weights w_j, yielding the specific properties of each group and as follows: (1) uncorrelated data instances—the profits p_j are randomly chosen in $[1, R]$, there is no correlation between the profit and the weight of an item; (2) weakly correlated data instances—p_j randomly distributed in $[w_j - R/10, w_j + R/10]$ such that $p_j \geq 1$. Despite their name, weakly correlated instances have a very high correlation between the profit and weight of an item. Typically, the profit differs from the weight by only a few percent; (3) strongly correlated data instances—profits are set to $p_j = w_j + 10$; and (4) subset-sum data instances—the profit of each item is equal to the weight, that is, $p_j = w_j$. The number of items N is set to 100, 200, 300, 500, and 1000, respectively, to test the algorithms with different problem scales. Moreover, the used 0-1 knapsack instances in this paper are available at https://github.com/whuph/KP_data, so that readers are able to test these instances in their algorithms.

Table 5 presents the mean maximum profit achieved by DBDE and other five competitive algorithms over 30 runs on each 0-1 knapsack instance. At the bottom of the table, Wilcoxon's rank sum test results between DBDE and others are summarized as "$w/t/l$", which means that DBDE wins in w functions, ties in t functions, and loses in l functions. Based on the results, our algorithm achieves better results than others on the majority of test instances. However, on

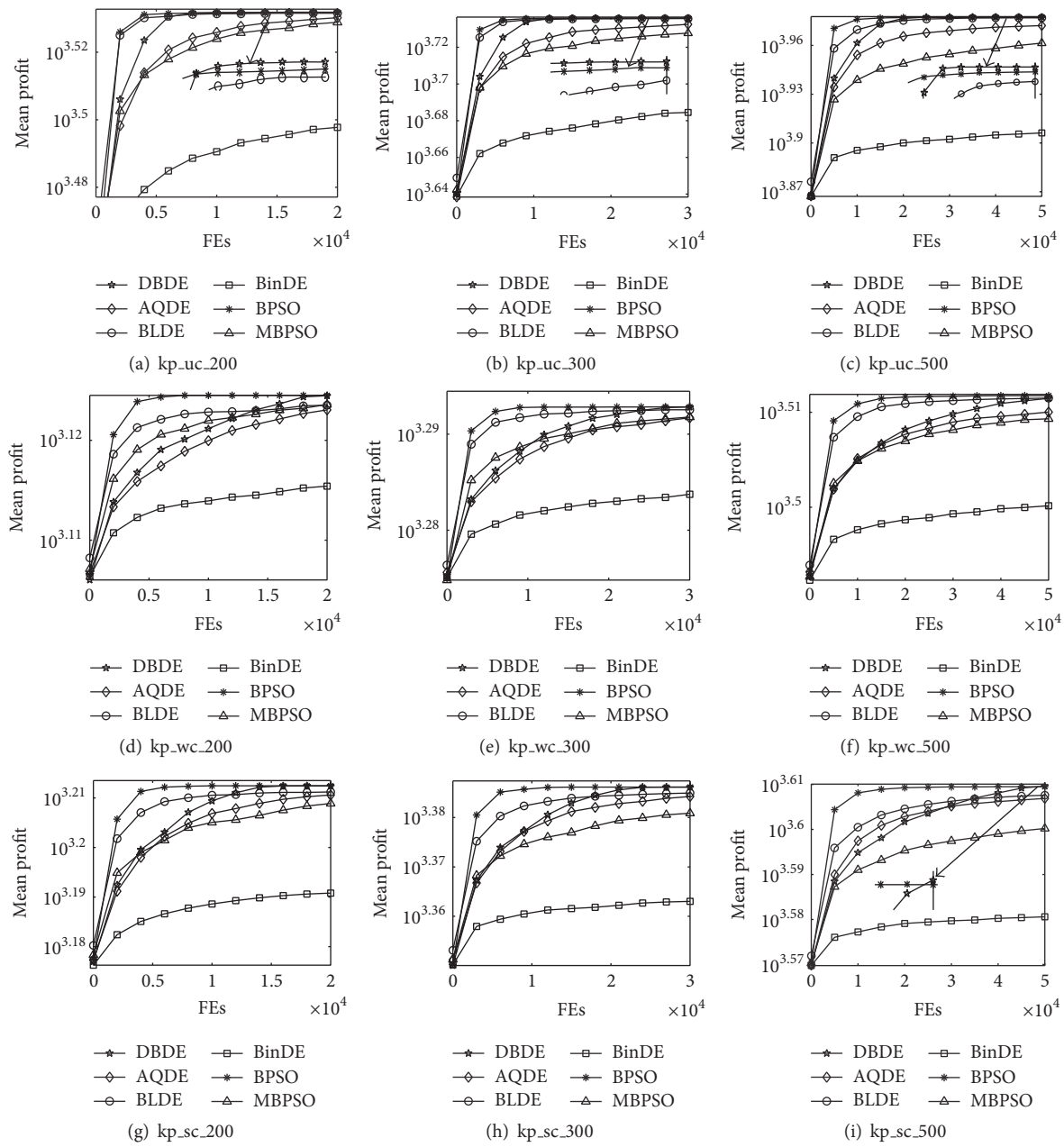

Figure 3: Convergence curves of the six contestant algorithms on nine representative 0-1 knapsack instances.

the subset-sum data instances all of the competitors get the same results. Because subset-sum instances are generally easy to solve, the maximum sum of weight indicates the optimal solution. A statistical view in terms of Wilcoxon's rank sum test in the last line of Table 5 is more appropriate to see the superiority of DBDE over other five competitors. We observe from that DBDE is significantly better than BPSO, MBPSO, BLDE, BinDE, and AQDE on 7, 13, 15, 15, and 12 test instances, respectively, but only worse than BPSO and AQDE on 2 and 1 test instances, respectively. Figure 3 shows the convergence curves of the six contestant algorithms on nine representative 0-1 knapsack instances.

Apart from the above analysis, one may come up with a question: what are the differences of computational cost between DBDE and other competitive algorithms on 0-1 knapsack problems? In order to respond to this, beside the maximum profit results, the runtime was also recorded to compare the computational cost of six algorithms on the test suite. Each algorithm is run 30 times per test instance and the mean runtime is recorded. Table 6 presents the statistical results of runtime. At the bottom of the table, the total runtime on all test instances is summarized.

From Table 6, we can see that the runtime of DBDE is higher than BinDE, but this difference is slight. In addition to

Table 6: Runtime (in seconds) of DBDE and other five competitors on twenty knapsack instances.

Problem	BPSO	MBPSO	BLDE	BinDE	AQDE	DBDE
kp_uc_100	1.65	1.52	1.25	0.53	1.71	0.48
kp_uc_200	6.53	6.07	4.81	1.81	6.74	2.32
kp_uc_300	15.32	14.11	11.72	3.43	15.53	5.18
kp_uc_500	45.39	43.34	37.05	8.65	47.47	13.99
kp_uc_1000	225.82	218.68	192.97	30.98	239.39	67.52
kp_wc_100	1.82	1.90	1.42	0.82	1.87	0.89
kp_wc_200	7.24	7.22	5.36	2.86	7.24	3.26
kp_wc_300	17.17	17.33	13.69	5.98	17.17	7.10
kp_wc_500	52.88	51.06	44.16	14.39	53.39	18.87
kp_wc_1000	264.33	255.35	227.96	53.96	268.16	68.24
kp_sc_100	1.51	1.42	1.18	0.68	1.56	0.74
kp_sc_200	6.04	5.63	4.47	2.06	6.17	2.47
kp_sc_300	14.22	13.39	11.22	4.15	14.68	5.19
kp_sc_500	42.67	40.39	34.85	9.79	44.80	12.13
kp_sc_1000	210.27	197.74	175.21	34.82	223.75	34.55
kp_ss_100	1.83	1.81	1.53	1.05	1.94	1.02
kp_ss_200	7.52	7.44	5.87	4.46	7.78	4.09
kp_ss_300	17.48	17.40	14.31	9.33	17.91	9.05
kp_ss_500	52.77	52.75	45.63	28.43	54.75	26.94
kp_ss_1000	267.82	271.25	242.57	145.57	280.47	135.37
Total	1260.29	1225.82	1077.23	**363.73**	1312.48	419.40

this, DBDE is significant faster than BPSO, MBPSO, BLDE, and AQDE. The possible reasons are as follows. First, as pointed out in Algorithm 2, the DBDE almost have any difference with classical DE and no additional module or computation has been introduced. Second, to benefit from dichotomous mutation and crossover, the rate of producing feasible individuals is much higher than other competitors. So the computation cost of repairing infeasible individual is less than others. Taking into account these experiments and analyses, we can conclude that DBDE is effective and efficient for 0-1 knapsack problems.

6.3. Multidimensional Knapsack Problem. Three groups of well-known multidimensional knapsack instances selected from OR-library [42] are used in the experimental studies to verify the performance of DBDE. The detailed descriptions of these multidimensional knapsack instances are summarized in Table 7. The optimum solutions of these instances are known. The first group with 2 multidimensional knapsack instances corresponds to "sento" [43], which is characterized by large constraints and the number of constraints is 30. In contrast to the first group, the second group with 8 multidimensional knapsack instances corresponds to "weing" [44], which is characterized by small constraints. The last group with 30 multidimensional knapsack instances corresponds to "weish" [45], which is a moderate test suit. Among the "weish" multidimensional knapsack instances, the number of constraints is 5 and the number of items ranges between 20 and 90. The used parameters are the same as in Section 6.1. The results of the mean maximum profit achieved by the DBDE and other competitors are presented in Table 8, where "$w/t/l$" summarizes the competition results between DBDE and others algorithms.

Table 7: The detailed information of the multidimensional knapsack problems.

Number	Problem	D	Constraints	Optimal value
1	Sento1	60	30	7772
2	Sento2	60	30	8722
3	Weing1	28	2	141278
4	Weing2	28	2	130883
5	Weing3	28	2	95677
6	Weing4	28	2	119337
7	Weing5	28	2	98796
8	Weing6	28	2	130623
9	Weing7	105	2	1095445
10	Weing8	105	2	624319
11	Weish1	30	5	4554
12	Weish2	30	5	4536
13	Weish3	30	5	4115
14	Weish4	30	5	4561
15	Weish5	30	5	4514
16	Weish6	40	5	5557
17	Weish7	40	5	5567
18	Weish8	40	5	5605
19	Weish9	40	5	5246
20	Weish10	50	5	6339
21	Weish11	50	5	5643
22	Weish12	50	5	6339
23	Weish13	50	5	6159
24	Weish14	60	5	6954
25	Weish15	60	5	7486
26	Weish16	60	5	7289
27	Weish17	60	5	8633
28	Weish18	70	5	9580
29	Weish19	70	5	7698
30	Weish20	70	5	9450
31	Weish21	70	5	9074
32	Weish22	80	5	8947
33	Weish23	80	5	8344
34	Weish24	80	5	10220
35	Weish25	80	5	9939
36	Weish26	90	5	9584
37	Weish27	90	5	9819
38	Weish28	90	5	9492
39	Weish29	90	5	9410
40	Weish30	90	5	11191

From Table 9, it can be seen from the boldface that DBDE achieves best results among the six contestants on 34 out of 40 test instances. Meanwhile, based on Wilcoxon's rank sum test results, DBDE is significantly better than BPSO, MBPSO, BLDE, BinDE, and AQDE on 22, 32, 23, 38, and 38 test instances, respectively, but BPSO, MBPSO, BLDE,

Table 8: Results of mean maximum profits on the multidimensional knapsack problems.

Problem	BPSO Mean	MBPSO Mean	BLDE Mean	BinDE Mean	AQDE Mean	DBDE Mean
Sento1	7744	7715	7750	7661	7670	**7753**
Sento2	**8700**	8630	8693	8476	8584	8696
Weing1	141224	141251	141129	140724	140948	**141278**
Weing2	**130883**	130874	**130883**	130593	130434	130878
Weing3	95600	95641	**95677**	95658	95192	95617
Weing4	**119337**	**119337**	119204	118364	119163	**119337**
Weing5	98774	98725	**98785**	98607	98440	98626
Weing6	130480	130519	130493	130305	130419	**130610**
Weing7	1092872	1043323	1084627	935904	1056826	**1093085**
Weing8	623630	622248	622121	610320	619660	**623945**
Weish1	**4554**	**4554**	**4554**	4542	**4554**	**4554**
Weish2	4534	4532	4527	4496	4528	**4536**
Weish3	4095	4097	**4115**	4095	4065	**4115**
Weish4	4560	4554	4559	4532	4538	**4561**
Weish5	**4514**	**4514**	**4514**	4494	**4514**	**4514**
Weish6	5543	5537	5533	5467	5527	**5547**
Weish7	**5566**	5552	5557	5475	5542	5556
Weish8	5600	5598	5599	5488	5590	**5604**
Weish9	**5246**	**5246**	**5246**	5211	5225	**5246**
Weish10	6332	6319	6336	6209	6281	**6339**
Weish11	**5641**	5638	5629	5550	5614	5637
Weish12	**6339**	6324	**6339**	6187	6270	**6339**
Weish13	6148	6118	6157	6031	6086	**6159**
Weish14	6903	6894	6925	6745	6875	**6954**
Weish15	7481	7446	7484	7167	7400	**7486**
Weish16	7285	7271	7287	7076	7242	**7289**
Weish17	8626	8535	8617	8035	8522	**8630**
Weish18	9574	9507	9561	9002	9479	**9580**
Weish19	7693	7686	7693	7439	7655	**7698**
Weish20	9446	9368	9442	8809	9308	**9450**
Weish21	9069	9029	9068	8547	8996	**9074**
Weish22	8918	8882	8938	8496	8866	**8947**
Weish23	8338	8298	8329	7997	8240	**8342**
Weish24	10219	10087	10208	9484	10096	**10220**
Weish25	9916	9830	9916	9246	9792	**9939**
Weish26	**9584**	9538	9570	9048	9513	**9584**
Weish27	**9819**	9769	9813	9143	9738	**9819**
Weish28	9481	9403	9481	8939	9372	**9492**
Weish29	**9410**	9373	9408	8864	9343	**9410**
Weish30	11186	11021	11177	10221	11053	**11190**
$w/t/l$	22/4/14	32/2/6	23/2/15	38/0/2	38/0/2	—

BinDE, and AQDE beat DBDE on 4, 2, 2, 0, and 0 test instances, respectively. Overall, the performance of DBDE on the multidimensional knapsack instances is better than that of the five competitors.

In order to compare the convergence rate of different algorithms, the mean number of FEs required to converge to the optimum solution (MFEs) and the successful rate (SR) are recorded in Table 9. The stopping criterion is that each algorithm is terminated when the optimum solution is found or the number of FEs reaches to its maximum value ($100 * D$). As seen, it is clear that DBDE is more reliable than other competitors as successful rate (SR) for DBDE reaches 100 on 22 multidimensional knapsack instances out of 40. Figure 4 shows the average successful rates of the six contestant algorithms on the multidimensional knapsack problems.

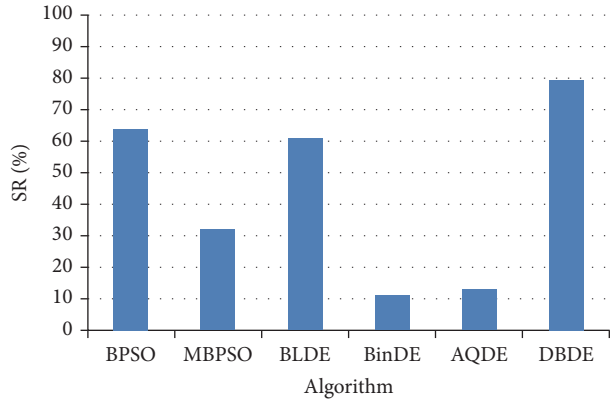

Figure 4: The successful rates of six algorithms on the multidimensional knapsack problems.

Moreover, DBDE shows faster convergence speed on the majority of multidimensional knapsack instances. Although, from Table 9, the MFEs of Weing1, Weing4, Weish1, Weish3, Weish4, Weish5, Weish6, Weish8, Weish9, Weish17, Weish18, Weish23, Weish24, Weish26, Weish27, and Weish29 achieved by DBDE are not the best, but DBDE exhibits the best SR on this test multidimensional knapsack instances among the six algorithms. From the last row of Table 9, DBDE costs the lowest average MFEs to reach the threshold and gets the highest average SR. Therefore, the above experimental results and analysis verify the superior performance of DBDE on multidimensional knapsack problem.

6.4. Validity of Population Size and Crossover Probability Settings. In order to investigate the validity of population size (NP) and crossover probability (CR1 and CR2) settings on the performance of DBDE, three scalable tests of DBDE have been carried out on the multidimensional knapsack problems with different NP, CR1, and CR2 values. The test suit of multidimensional knapsack problem is the same as Section 6.3 used. The experimental results of DBDE with different NP, CR1, and CR2 values are summarized in Tables 10, 11, and 12, respectively.

In the case of different NP values, experiments have been done on NP = 50, 100, 150, and D. It is clear from Table 10 that the performance of NP = 100 is significantly better than that of others. When NP = 100, DBDE achieved best SR results (boldface). These observations demonstrate that NP = 100 is efficient and suitable.

As shown in Table 11, for the CR1 test, CR1 is set to 0.1, 0.2, 0.3, 0.4, and 0.5. By carefully examining the results, we find that value 0.2 gets best results. Therefore, on the whole, CR1

Table 9: Results of mean number of FEs and successful rate on the multidimensional knapsack problems.

Problem	BPSO		MBPSO		BLDE		BinDE		AQDE		DBDE	
	MFEs	SR (%)	MFEs	SR (%)	MFEs	SR (%)	MFEs	SR (%)	MFEs	SR (%)	MFEs	SR (%)
Sento1	6000	0	6000	0	5536	13	6000	0	6000	0	**5234**	**43**
Sento2	**6000**	**0**	**6000**	**0**	**6000**	**0**	**6000**	**0**	**6000**	**0**	**6000**	**0**
Weing1	1471	87	1951	73	**1401**	70	2789	3	2724	10	1642	**100**
Weing2	1047	**100**	1819	93	**617**	**100**	2623	17	2733	13	1507	97
Weing3	1133	90	1406	90	**340**	**100**	1588	73	2428	27	1719	60
Weing4	**888**	**100**	1396	**100**	982	93	2576	23	2316	63	1390	**100**
Weing5	1578	87	2347	60	**818**	**93**	2704	23	2800	0	2478	30
Weing6	1818	63	1942	73	1490	67	2500	23	2309	53	**1346**	**97**
Weing7	**10432**	**3**	10500	0	10500	0	10500	0	10500	0	10500	0
Weing8	10500	0	10500	0	**7401**	**47**	10500	0	10500	0	10500	0
Weish1	625	**100**	801	**100**	**579**	**100**	2077	53	1269	**100**	828	**100**
Weish2	1836	60	2264	53	1821	53	2966	13	2677	33	**1708**	**100**
Weish3	2518	43	2540	47	**960**	**100**	2687	40	3000	0	1200	**100**
Weish4	1653	97	2131	77	**874**	97	2651	50	2728	23	997	**100**
Weish5	483	**100**	610	**100**	**465**	**100**	1961	63	742	**100**	714	**100**
Weish6	**3455**	23	3915	7	3650	17	4000	0	4000	0	3649	**30**
Weish7	**2202**	**93**	3703	30	2555	67	4000	0	3943	10	3629	33
Weish8	**2677**	57	3457	30	2755	43	4000	0	3808	13	2928	**87**
Weish9	891	**100**	1658	**100**	**710**	**100**	3299	47	3279	50	1120	**100**
Weish10	3093	77	4751	20	2353	87	5000	0	5000	0	**2092**	**100**
Weish11	**2136**	**90**	3905	70	4143	27	4981	3	4726	20	3538	63
Weish12	2882	90	4455	47	1872	97	5000	0	5000	0	**2100**	**100**
Weish13	3345	83	4961	10	2104	97	5000	0	5000	0	**2227**	**100**
Weish14	5977	3	6000	0	4601	40	6000	0	6000	0	**3362**	**100**
Weish15	3667	83	5908	7	2997	93	6000	0	6000	0	**2683**	**100**
Weish16	4392	57	6000	0	4215	57	6000	0	6000	0	**3980**	**87**
Weish17	**4937**	37	5986	3	5806	10	6000	0	5902	3	5384	**67**
Weish18	**4347**	67	7000	0	6519	20	7000	0	7000	0	4848	**100**
Weish19	4322	67	6230	37	4191	63	7000	0	7000	0	**3084**	**100**
Weish20	5187	63	7000	0	5811	47	7000	0	7000	0	**3678**	**100**
Weish21	4772	80	6863	7	4470	80	7000	0	7000	0	**3431**	**100**
Weish22	6784	27	8000	0	5148	60	8000	0	8000	0	**3651**	**100**
Weish23	**7172**	20	8000	0	7641	10	8000	0	8000	0	7354	**23**
Weish24	**3953**	97	8000	0	7200	47	8000	0	8000	0	5278	**100**
Weish25	7981	7	8000	0	7572	23	8000	0	8000	0	**5703**	**97**
Weish26	**3987**	97	8721	10	6333	67	9000	0	9000	0	4800	**100**
Weish27	**3579**	**97**	8946	7	6881	43	9000	0	9000	0	5084	**97**
Weish28	4481	83	8992	3	3951	87	9000	0	9000	0	**3892**	**100**
Weish29	**3514**	**100**	8709	23	4604	97	9000	0	9000	0	4340	**100**
Weish30	8107	33	9000	0	8449	17	9000	0	9000	0	**7253**	**83**
Average	3896	64	5259	32	3908	61	5610	11	5560	13	**3671**	**80**

= 0.2 is the best choice for DBDE. In similar circumstances, observed from Table 12, CR2 = 0.5 also is the best option for DBDE.

7. Conclusion

DE is a powerful population-based random optimization algorithm. However, its operators are based on floating-point representation only and are not suitable for the combinational optimization problems. In order to solve these problems, a novel binary DE with dichotomous mechanism (DBDE) is proposed, in which dichotomous mutation and dichotomous crossover are employed to enhance the exploration ability of DE in the binary search space. Dichotomous mechanism is learned from the dichotomous thinking of psychology. The primary idea of dichotomous mechanism is that each

Table 10: Experimental results of DBDE with different NP values.

NP	50	100	150	n
Mean SR (%)	73.17	**79.83**	70.00	73.42

Table 11: Experimental results of DBDE with different CR1 values.

CR1 (CR2 = 0.5)	0.1	0.2	0.3	0.4	0.5
Mean SR (%)	67.25	**79.83**	76.83	73.75	66.92

Table 12: Experimental results of DBDE with different CR2 values.

CR2 (CR1 = 0.2)	0.3	0.4	0.5	0.6	0.7
Mean SR (%)	77.92	78.75	**79.83**	76.58	76.83

of the two features (i.e., 0 and 1) after the XOR operation corresponds to two different strategies. Compared with the classical DE, DBDE almost has any difference and no additional computation. Moreover, we analyze the solution quality of DBDE on the 0-1 knapsack problem in terms of the approximation ratio and conclude that the expected runtime of DBDE to reach the global optimum is $\Omega(q^{-\alpha n})$.

The experimental studies in this paper are performed on a suite of 0-1 knapsack problems and multidimensional knapsack problems. The quality of DBDE is verified by comparing with three start-of-the-art binary DE variants (i.e., BLDE, BinDE, and AQDE) and two binary PSO variants (i.e., BPSO and MBPSO) and the results show that DHDE is the best among the six algorithms. On the 0-1 knapsack problems, DBDE is significantly better than BPSO, MBPSO, BLDE, BinDE, and AQDE on 7, 13, 15, 15, and 12 test instances, respectively, but only worse than BPSO and AQDE on 2 and 1 test instances, respectively. On the 0-1 multidimensional knapsack problems, DBDE achieves best results among the six contestants on 34 out of 40 test instances. Furthermore, DBDE costs the lowest average MFEs to reach the threshold and gets the highest average SR. For the computational cost, DBDE is significant faster than BPSO, MBPSO, BLDE, and AQDE but slightly higher than BinDE. In addition, the effect of the population size (NP) and crossover probability settings (CR1 and CR2) are experimentally studied.

The DBDE can effectively improve the performance of DE on knapsack problems, so it may also work well on other combinatorial optimization problems, such as traveling salesman problem and inventory routing problem. In the future, how to generalize our work to solve other combinational optimization problems remains an attractive topic. Additionally, Local Search (LS) is a technique that guides the search to the most promising solution area, and the hybridization of DBDE with other LS methods can be attempted to further enhancing performance.

Competing Interests

The authors declare that they have no competing interests.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (no. 61364025, no. 61662029), the Foundation of State Key Laboratory of Software Engineering (no. SKLSE2012-09-39), and the Science and Technology Foundation of Jiangxi Province, China (no. GJJ13729) as well.

References

[1] S. Martello and P. Toth, *Knapsack Problems*, John Wiley & Sons, Chichester, UK, 1990.

[2] Z. Michalewicz, *Genetic algorithms + data structures = evolution programs*, Springer-Verlag, Berlin, Second edition, 1994.

[3] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, Springer Science & Business Media, 1996.

[4] J. He and Y. Zhou, "A comparison of GAs using penalizing infeasible solutions and repairing infeasible solutions on average capacity knapsack," in *Advances in Computation and Intelligence*, vol. 4683 of *Lecture Notes in Computer Science*, pp. 100–109, Springer, Berlin, Germany, 2007.

[5] F. Hembecker, H. S. Lopes, and W. Godoy Jr., "Particle swarm optimization for the multidimensional knapsack problem," in *Adaptive and Natural Computing Algorithms: 8th International Conference, ICANNGA 2007, Warsaw, Poland, April 11–14, 2007, Proceedings, Part I*, vol. 4431 of *Lecture Notes in Computer Science*, pp. 358–365, Springer, Berlin, Germany, 2007.

[6] J. C. Bansal and K. Deep, "A modified binary particle swarm optimization for knapsack problems," *Applied Mathematics and Computation*, vol. 218, no. 22, pp. 11042–11061, 2012.

[7] A. R. Hota and A. Pat, "An adaptive quantum-inspired differentiale volutional gorithm for 0–1 knap sackproble," in *Proceedings of the IEEE 2nd World Congress on Nature and Biologically Inspired Computing (NaBIC '10)*, pp. 703–708, Kitakyushu, Japan, December 2010.

[8] L. Wang, X. Fu, Y. Mao, M. I. Menhas, and M. Fei, "A novel modified binary differential evolution algorithm and its applications," *Neurocomputing*, vol. 98, pp. 55–75, 2012.

[9] A. Fréville, "The multidimensional 0-1 knapsack problem: an overview," *European Journal of Operational Research*, vol. 155, no. 1, pp. 1–21, 2004.

[10] J. Puchinger, G. R. Raidl, and U. Pferschy, "The multidimensional knapsack problem: structure and algorithms," *INFORMS Journal on Computing*, vol. 22, no. 2, pp. 250–265, 2010.

[11] R. Storn and K. Price, "Differential evolution: a simple and efficient adaptive scheme for global optimization over continuous spaces," Tech. Rep. TR-95–012, International Computer Science Institute, Berkeley, Calif, USA, 1995.

[12] J. Zhang and A. C. Sanderson, "JADE: adaptive differential evolution with optional external archive," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 5, pp. 945–958, 2009.

[13] H. Peng and Z. Wu, "Heterozygous differential evolution with Taguchi local search," *Soft Computing*, vol. 19, no. 11, pp. 3273–3291, 2015.

[14] Q. Ding and G. Zheng, "The cellular differential evolution based on chaotic local search," *Mathematical Problems in Engineering*, vol. 2015, Article ID 128902, 15 pages, 2015.

[15] Z. Guo, G. Liu, D. Li, and S. Wang, "Self-adaptive differential evolution with global neighborhood search," *Soft Computing*, 2016.

[16] S. Das and P. N. Suganthan, "Differential evolution: a survey of the state-of-the-art," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 1, pp. 4–31, 2011.

[17] S. Das, S. S. Mullick, and P. Suganthan, "Recent advances in differential evolution—an updated survey," *Swarm and Evolutionary Computation*, vol. 27, pp. 1–30, 2016.

[18] G. Pampara, A. P. Engelbrecht, and N. Franken, "Binary differential evolution," in *Proceedings of the IEEE Congress on Evolutionary Computation*, pp. 1873–1879, Vancouver, Canada, July 2006.

[19] X. He and L. Han, "A novel binary differential evolution algorithm based on artificial immune system," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '07)*, pp. 2267–2272, IEEE, Singapore, September 2007.

[20] T. Gong and A. L. Tuson, "Differential evolution for binary encoding," in *Soft Computing in Industrial Applications*, pp. 251–262, Springer, Berlin, Germany, 2007.

[21] C.-Y. Wu and K.-Y. Tseng, "Topology optimization of structures using modified binary differential evolution," *Structural and Multidisciplinary Optimization*, vol. 42, no. 6, pp. 939–953, 2010.

[22] Y. Chen, W. Xie, and X. Zou, "Abinary differential evolutional gorithm learning from explored solutions," *Neurocomputing B*, vol. 149, pp. 1038–1047, 2015.

[23] J. Sun, Q. Zhang, and E. P. Tsang, "DE/EDA: a new evolutionary algorithm for global optimization," *Information Sciences. An International Journal*, vol. 169, no. 3-4, pp. 249–262, 2005.

[24] W. Gong, Z. Cai, and C. X. Ling, "DE/BBO: a hybrid differential evolution with biogeography-based optimization for global numerical optimization," *Soft Computing*, vol. 15, no. 4, pp. 645–665, 2011.

[25] A. R. Yildiz, "A new hybrid differential evolution algorithm for the selection of optimal machining parameters in milling operations," *Applied Soft Computing Journal*, vol. 13, no. 3, pp. 1561–1566, 2013.

[26] H.-Y. Fan and J. Lampinen, "A trigonometric mutation operation to differential evolution," *Journal of Global Optimization*, vol. 27, no. 1, pp. 105–129, 2003.

[27] S. Das, A. Abraham, U. K. Chakraborty, and A. Konar, "Differential evolution using a neighborhood-based mutation operator," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 3, pp. 526–553, 2009.

[28] W. Gong and Z. Cai, "Differential evolution with ranking-based mutation operators," *IEEE Transactions on Cybernetics*, vol. 43, no. 6, pp. 2066–2081, 2013.

[29] A. K. Qin, V. L. Huang, and P. N. Suganthan, "Differential evolution algorithm with strategy adaptation for global numerical optimization," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 2, pp. 398–417, 2009.

[30] R. Mallipeddi, P. N. Suganthan, Q. K. Pan, and M. F. Tasgetiren, "Differential evolution algorithm with ensemble of parameters and mutation strategies," *Applied Soft Computing Journal*, vol. 11, no. 2, pp. 1679–1696, 2011.

[31] Y. Wang, H.-X. Li, T. Huang, and L. Li, "Differential evolution based on covariance matrix learning and bimodal distribution parameter setting," *Applied Soft Computing*, vol. 18, no. 5, pp. 232–247, 2014.

[32] J. Vesterstrom and R. Thomsen, "A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '04)*, vol. 2, pp. 1980–1987, Portland, Ore, USA, June 2004.

[33] Y. Wang, B. Li, and T. Weise, "DE/BBO: a hybrid differential evolution with biogeography-based optimization for global numerical optimization," *Information Sciences*, vol. 180, no. 12, pp. 2405–2420, 2010.

[34] J. Zhao, Y. Xu, F. Luo, Z. Dong, and Y. Peng, "Power system fault diagnosis based on history driven differential evolution and stochastic time domain simulation," *Information Sciences*, vol. 275, pp. 13–29, 2014.

[35] N. Noman and H. Iba, "Inferring gene regulatory networks using differential evolution with local search heuristics," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 4, no. 4, pp. 634–647, 2007.

[36] C. Zhan, W. Situ, L. F. Yeung, P. W.-M. Tsang, and G. Yang, "A parameter estimation method for biological systems modelled by ODE/DDE models using spline approximation and differential evolution algorithm," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 11, no. 6, pp. 1066–1076, 2014.

[37] I. De Falco, A. Della Cioppa, D. Maisto, and E. Tarantino, "Differential evolution as a viable tool for satellite image registration," *Applied Soft Computing*, vol. 8, no. 4, pp. 1453–1462, 2008.

[38] S. Paul and S. Das, "Simultaneous feature selection and weighting—an evolutionary multi-objective optimization approach," *Pattern Recognition Letters*, vol. 65, pp. 51–59, 2015.

[39] J. He, B. Mitavskiy, and Y. Zhou, "A theoretical assessment of solution quality in evolutionary algorithms for the knapsack problem," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '14)*, pp. 141–148, Beijing, China, July 2014.

[40] J. Kennedy and R. C. Eberhart, "A discrete binary version of the particle swarm algorithm," in *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation*, vol. 5, pp. 4104–4108, Orlando, Fla, USA, October 1997.

[41] D. P. Williamson and D. B. Shmoys, *The design of approximation algorithms*, Cambridge University Press, Cambridge, 2011.

[42] J. Beasley, "Orlib-operations research library," 2005, http://people.brunel.ac.uk/~mastjjb/jeb/orlib/mknapinfo.html.

[43] S. Senju and Y. Toyoda, "An approach to linear programming with 0–1 variables," *Management Science*, vol. 15, no. 4, pp. B-196–B-207, 1968.

[44] H. M. Weingartner and D. N. Ness, "Methods for the solution of the multidimensional 0/1 knapsack problem," *Operations Research*, vol. 15, no. 1, pp. 83–103, 1967.

[45] W. Shih, "A branch and bound method for the multiconstraint zero-one knapsack problem," *Journal of the Operational Research Society*, vol. 30, no. 4, pp. 369–378, 1979.