IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received December 11, 2017, accepted February 17, 2018, date of publication February 27, 2018, date of current version March 15, 2018.

*Digital Object Identifier 10.1109/ACCESS.2018.2809445*

# Binary Moth Search Algorithm for Discounted {0-1} Knapsack Problem

**YAN-HONG FENG[1] AND GAI-GE WANG[2,3,4,5,6]**

[1]School of Information Engineering, Hebei GEO University, Shijiazhuang 050031, China
[2]Department of Computer Science and Technology, Ocean University of China, Qingdao 266100, China
[3]School of Computer Science and Technology, Jiangsu Normal University, Xuzhou 221116, China
[4]Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education, Jilin University, Changchun 130012, China
[5]Institute of Algorithm and Big Data Analysis, Northeast Normal University, Changchun 130117, China
[6]School of Computer Science and Information Technology, Northeast Normal University, Changchun 130117, China

Corresponding author: Gai-Ge Wang (gaigewang@gmail.com)

This work was supported in part by the National Natural Science Foundation of China under Grant 61503165, in part by the Natural Science Foundation of Jiangsu Province under Grant BK20150239, and in part by the Key Research and Development Projects of Hebei Province under Grant 17210905.

**ABSTRACT** The discounted {0-1} knapsack problem (DKP) extends the classical 0-1 knapsack problem (0-1 KP) in which a set of item groups is included and each group consists of three items, whereas at most one of the three items can be packed into the knapsack. Therefore, the DKP is more complicated and computationally difficult than 0-1 KP. The DKP has been found many applications in real economic problems and other areas. In this paper, the influence of Lévy flights operator and fly straightly operator in the moth search (MS) algorithm is verified. Nine types of new mutation operator based on the global harmony search are specially devised to replace Lévy flights operator. Then, nine novel MS-based algorithms for DKP are proposed (denoted by MS1–MS9). Extensive experiments on three sets of 30 DKP instances demonstrate the remarkable performance of the proposed nine new MS-based approaches. In particular, it discovers that MS1–MS3 show better comprehensive performance among 10 algorithms. A variety of analyses indicate the important contribution of the individual of memory consideration in MS1–MS9.

**INDEX TERMS** Discounted {0-1} knapsack problem, harmony search, moth search, swarm intelligence.

## I. INTRODUCTION

The discounted {0-1} knapsack problem (DKP), as a generalization of the standard 0-1 knapsack problem (0-1 KP) [1], [2], is a novel and typical discrete optimization problem recently proposed by Guldan [3]. The idea of DKP stems from the promotional discounts activities of real merchants and many practical problems, such as investment decision and resource allocation, can be formulated as the DKP.

As a new variant of the standard 0-1 KP, DKP is more complex and difficult to tackle than the standard 0-1 KP. So far there is not much literature on the topic. Guldan [3] firstly proposed dynamic programming (DP) [4] for solving DKP. Based on the core concept, an effective DP method for the DKP is described by Rong *et al.* [5]. Recently, two new mathematical models of the DKP are firstly shown by He *et al.* [6]. Afterword, a new DP method (NE-DKP) and three approximate algorithms to this difficult problem are presented in [7]. Consequently, these algorithms are mainly exact methods. Very recently, Feng *et al.* [8] proposed a multi-strategy monarch butterfly optimization (MMBO) for DKP and two effective strategies, neighborhood mutation with crowding and Gaussian perturbation, are introduced into MMBO. Feng *et al.* [9] combined monarch butterfly algorithm (MBO) with 7 kinds of differential evolution (DE) [10] mutation strategy and developed a novel DEMBO algorithm for DKP. In view of the significance of solving DKP in practical application and academic research, it is necessary to apply novel algorithm to this difficult problem.

Swarm intelligence (SI) methods, as a branch of modern metaheuristic algorithms, start to demonstrate their power in dealing with tough optimization problems and even *NP*-hard problems. Thereinto, particle swarm optimization (PSO) [11], and ant colony optimization (ACO) [12] are two of the most representative paradigms among the entire family of SI methods. In the past twenty years, more and more effective SI methods are emerging, some of them include artificial bee colony (ABC) [13] algorithm, fruit fly optimization algorithm (FOA) [14], human learning optimization (HLO) [15], krill herd (KH) [16]–[18], elephant herding optimization (EHO) [19], animal

migration optimization (AMO) [20], biogeography-based optimization (BBO) [21], [22], monarch butterfly optimization (MBO) [23], [24], earthworm optimization algorithm (EWA) [25], and moth search (MS) [26] algorithm.

As a novel bio-inspired metaheuristic algorithm, there are two important issues in MS [26] algorithm: Lévy flights and fly straightly of the moths. Owing to the relatively short time of MS being presented, the applications of MS in solving practical problems is rare in literature. For all we know, there is no research work about the MS for solving discrete optimization problems, especially the DKP problem.

This paper presents highly effective MS methods for solving DKP. The main contributions of this paper include the following aspects. Firstly, there are two main operators, Lévy flights and fly straightly. The influence of two different operators on the overall performance of MS for DKP is verified. Secondly, to explore efficiently the comprehensive performance of MS, Lévy flights is replaced by nine types of new mutation operator based on global harmony search (GHS) [27], [28]. Thirdly, as the basic MS was proposed to solve the continuous optimization problem, nine novel binary MS-based (noted MS1-MS9) algorithms and a binary basic MS are specifically designed for the DKP. Finally, two stage repair and optimization operator are employed to repair the infeasible solution and further optimize the feasible solution.

The remainder of the paper is organized as follows. In Section II, the definition and mathematical formulation of the DKP is presented. In Section III, the basic MS is described briefly. Then, binary MS for DKP is presented in Section IV. Section V is dedicated to an extensive investigation of nine MS variants. Conclusions are drawn in Section VI.

## II. PROBLEM DESCRIPTION AND MATHEMATICAL MODELS

When The definition of DKP is as follows: given a set of $n$ item groups $N = \{0, 1, \ldots, n-1\}$ and each group $i$ ($i \in N$) has three items denoted as $3i$, $3i+1$, and $3i+2$, respectively. In each group, item $3i$ and item $3i+1$ have the weights $w_{3i}$, $w_{3i+1}$ and the profits $p_{3i}$, $p_{3i+1}$, respectively. Item $3i+2$ has the profit $p_{3i+2}$ and a discounted weight $w_{3i+2}(w_{3i+2} < w_{3i} + w_{3i+1})$. The number of items that can be packed into the knapsack with capacity $C$ is 0 or 1 in each group. The objective of the DKP is to find a subset of all the items that maximizes the total value on condition that the total weight $\leq C$.

$$\max f(X) = \max \sum_{i=0}^{n-1} (x_{3i}p_{3i} + x_{3i+1}p_{3i+1} + x_{3i+2}p_{3i+2}) \quad (1)$$

$$subject\ to\ x_{3i} + x_{3i+1} + x_{3i+2} \leq 1, \quad \forall i \in N \quad (2)$$

$$\sum_{i=0}^{n-1} (x_{3i}w_{3i} + x_{3i+1}w_{3i+1} + x_{3i+2}w_{3i+2}) \leq C \quad (3)$$

$$x_{3i}, x_{3i+1}, x_{3i+2} \in \{0, 1\}, \quad \forall i \in N \quad (4)$$

where $x_{3i}$, $x_{3i+1}$ and $x_{3i+2}$ are the binary decision variables such that $x_{3i} = 1$ or $x_{3i+1} = 1$ or $x_{3i+2} = 1$ if one of item $3i$, $3i+1$, and $3i+2$ is allocated to the knapsack, $x_{3i} = 0$ and $x_{3i+1} = 0$ and $x_{3i+2} = 0$ otherwise. Constraint (2) guarantees that at most one item in each group can be assigned to the knapsack. Constraint (3) requires that the total weight of items in the knapsack must be no larger than its capacity $C$. Constraint (4) requires that each variable takes the value of 1 or 0.

The DKP generalizes the well-known *NP*-hard 0-1 KP. Notice that DKP contains $n+1$ inequality constraints, in which there are $n$ discount constraints (2) and one knapsack capacity constraint (3), while there is only one inequality constraint in the classical 0-1 KP. Therefore, DKP is more complicated than the classical 0-1 KP. Indeed, the DKP degenerates into the classical 0-1 KP if constraints (2) are removed.

## III. MOTH SEARCH ALGORITHM

Moth search (MS) algorithm [26] is a novel swarm intelligence method inspired by the phototaxis and Lévy flights of the moths. In MS, the whole population is divided into two equal subpopulations according to the fitness, subpopulation 1 and subpopulation 2. There are two main optimization processes: the moths which have a smaller distance from the best one will fly around the best individual by Lévy flights and others will fly towards the best one in line. Therefore, an offspring of subpopulation1 and subpopulation 2 is generated by performing Lévy flights operator and fly straightly operator, respectively.

### A. LÉVY FLIGHTS

For individual $i$ in subpopulation1, the position can be updated by (5).

$$x_i^{t+1} = x_i^t + \alpha L(s) \quad (5)$$

where $x_i^t$ and $x_i^{t+1}$ are the position of individual $i$ at generation $t$ and $t+1$. Parameter $\alpha$ is the scale factor which can be calculated by (6).

$$\alpha = S_{\max}/t^2 \quad (6)$$

where $S_{max}$ is the max walk step and it takes the value 1.0 in this paper.

$L(s)$ is the step drawn from Lévy flights and it can be formulated as (7).

$$L(s) = \frac{(\beta-1)\Gamma(\beta-1)\sin(\frac{\pi(\beta-1)}{2})}{\pi s^{\beta}} \quad (7)$$

where $\Gamma(x)$ is the gamma function, parameter $\beta = 1.5$ and s≥0.

### B. FLY STRAIGHTLY

For individual $i$ in subpopulation2, the position movement from generation $t$ to $t+1$ can be expressed as (8).

$$x_i^{t+1} = \begin{cases} \lambda \times (x_i^t + \varphi \times (x_{best}^t - x_i^t)) & if\ rand > 0.5 \\ \lambda \times (x_i^t + \frac{1}{\varphi} \times (x_{best}^t - x_i^t)) & else \end{cases} \tag{8}$$

where $\lambda$ is a scale factor, $\varphi$ is an acceleration factor and $x_{best}^t$ is the best moth at generation $t$.

## IV. BINARY MS-BASED METHODS FOR DKP

In this section, we present the design of the binary MS-based methods for the DKP. Different from the basic MS, first we outline the global-best harmony search (GHS) algorithm; second nine new MS-based methods which consider GHS as mutation operator are formulated; third we adopt a two-tuples $<\mathbf{X}, \mathbf{Y}>$, in which real vector $\mathbf{X}$ is still involved in evolution and a binary vector $\mathbf{Y}$ represents a solution. Besides, two-stage repair and optimization operator is adopted in the binary MS-based methods, which not only guarantees the feasibility of all solutions, but also further improves the quality of all the feasible solutions.

### A. THE GLOBAL-BEST HARMONY SEARCH

Harmony search (HS) [27] is a relatively effective meta-heuristic algorithm and the optimization process contains three rules: memory consideration, pitch adjustment and random selection. Among many HS-based variants, global-best harmony search (GHS) [28] can perform efficiently on both continuous and discrete problems. Essentially, GHS contains the following three formulas which represents three rules in HS, respectively.

$$x_i^k = x_j^k (j \sim U(1, \ldots, HMS) \quad if\ rand \leq \text{HMCR} \tag{9}$$

$$x_i^k = x_{best}^k \quad if\ rand \leq \text{HMCR} \wedge rand \leq \text{PAR} \tag{10}$$

$$x_i^k = LB^k + rand \times (UB^k - LB^k) \quad if\ rand > \text{HMCR} \tag{11}$$

where HMCR is harmony memory considering rate, PAR is pitch adjusting rate and HMS is the harmony memory size. $x_i^k$ is the $k$th element of individual $i$. Similarly, $x_j^k$, $x_{best}^k$ indicate the $k$th element of individual $j$ and the current global best individual in the entire population, respectively. Rand is a uniformly distributed random number in [0,1], and $LB^k$ and $UB^k$ are the lower and upper limits for the $k$th.

### B. NINE TYPES OF MUTATION OPERATOR BASED ON GHS

As noted earlier, there are two subpopulations in basic MS. Naturally, useful information interaction between two subpopulations should be considered. Additionally, with the aim of improving the overall performance of MS, the excellent gene of global best individual should be kept. Based on the above considerations, nine types of new mutation operator by the modification of the (9)-(11) are devised to be integrated into the basic MS. Firstly, five conditions are as follows:

$$rand \leq \text{HMCR} \tag{12}$$

$$rand \leq \text{HMCR} \wedge rand \leq \text{PAR} \tag{13}$$

$$rand \geq \text{PAR} \wedge r1 \geq 0.5 \wedge n1 \neq n2 \tag{14}$$

$$rand \geq \text{PAR} \wedge r1 < 0.5 \wedge n1 \neq n2 \tag{15}$$

$$rand > \text{HMCR} \tag{16}$$

Then the basic evolution formulas of the nine types of mutation operators are as follows:

Type1:

$$x_i^k = \begin{cases} x_{p,j}^k & if\ (12) \text{ is satisfied} \\ x_{best}^k & if\ (13) \text{ is satisfied} \\ x_{best}^k + r \times (x_{sp2,n1}^k - x_{sp2,n2}^k) & if\ (14) \text{ is satisfied} \\ x_{best}^k - r \times (x_{sp2,n1}^k - x_{sp2,n2}^k) & if\ (15) \text{ is satisfied} \\ LB^k + r \times (UB^k - LB^k) & if\ (16) \text{ is satisfied} \end{cases} \tag{17}$$

where $x_i^k$ is the $k$th element of individual $i$. $x_{p,j}^k$ is the $k$th element of individual $j$ randomly chosen from the entire population. Based on (12), $x_{p,j}^k$ is a newly generated individual in memory consideration process and $x_{p,j}^k$ is called memory consideration individual. $x_{sp2,n1}^k$ and $x_{sp2,n2}^k$ are the $k$th element of individual $n1$ and individual $n2$ in subpopulation 2, respectively.

Type2: differ from Type1, two differential individuals are randomly chosen from subpopulation 1 if (14) or (15) is satisfied.

$$x_i^k = \begin{cases} x_{p,j}^k & if\ (12) \text{ is satisfied} \\ x_{best}^k & if\ (13) \text{ is satisfied} \\ x_{best}^k + r \times (x_{sp1,n1}^k - x_{sp1,n2}^k) & if\ (14) \text{ is satisfied} \\ x_{best}^k - r \times (x_{sp1,n1}^k - x_{sp1,n2}^k) & if\ (15) \text{ is satisfied} \\ LB^k + r \times (UB^k - LB^k) & if\ (16) \text{ is satisfied} \end{cases} \tag{18}$$

where $x_{sp1,n1}^k$ and $x_{sp1,n2}^k$ are the kth element of individual $n1$ and individual $n2$ in subpopulation 1, respectively.

Type3: differ from Type1, two differential individuals are randomly chosen from the entire population if (14) or (15) is satisfied.

$$x_i^k = \begin{cases} x_{p,j}^k & if\ (12) \text{ is satisfied} \\ x_{best}^k & if\ (13) \text{ is satisfied} \\ x_{best}^k + r \times (x_{p,n1}^k - x_{p,n2}^k) & if\ (14) \text{ is satisfied} \\ x_{best}^k - r \times (x_{p,n1}^k - x_{p,n2}^k) & if\ (15) \text{ is satisfied} \\ LB^k + r \times (UB^k - LB^k) & if\ (16) \text{ is satisfied} \end{cases} \tag{19}$$

where $x_{p,n1}^k$ and $x_{p,n2}^k$ are the kth element of individual n1 and individual n2 which comes from the entire population, respectively.

Type4: differ from Type1, two differential individuals are randomly chosen from the entire population if (14) or (15) is satisfied. In addition, the individual of memory consideration are randomly selected from subpopulation 1 if (12) is

satisfied.

$$x_i^k = \begin{cases} x_{sp1,j}^k & if\ (12)\ \text{is satisfied} \\ x_{best}^k & if\ (13)\ \text{is satisfied} \\ x_{best}^k + r \times (x_{p,n1}^k - x_{p,n2}^k) & if\ (14)\ \text{is satisfied} \\ x_{best}^k - r \times (x_{p,n1}^k - x_{p,n2}^k) & if\ (15)\ \text{is satisfied} \\ LB^k + r \times (UB^k - LB^k) & if\ (16)\ \text{is satisfied} \end{cases} \tag{20}$$

Type5: differ from Type1, when (14) or (15) is satisfied, two differential individuals are both randomly chosen from subpopulation 1. The individual of memory consideration is randomly chosen from subpopulation 1 as well when (12) is satisfied.

$$x_i^k = \begin{cases} x_{sp1,j}^k & if\ (12)\ \text{is satisfied} \\ x_{best}^k & if\ (13)\ \text{is satisfied} \\ x_{best}^k + r \times (x_{sp1,n1}^k - x_{sp1,n2}^k) & if\ (14)\ \text{is satisfied} \\ x_{best}^k - r \times (x_{sp1,n1}^k - x_{sp1,n2}^k) & if\ (15)\ \text{is satisfied} \\ LB^k + r \times (UB^k - LB^k) & if\ (16)\ \text{is satisfied} \end{cases} \tag{21}$$

Type6: differ from Type1, the individual of memory consideration is randomly chosen from subpopulation 1 if (12) is satisfied.

$$x_i^k = \begin{cases} x_{sp1,j}^k & if\ (12)\ \text{is satisfied} \\ x_{best}^k & if\ (13)\ \text{is satisfied} \\ x_{best}^k + r \times (x_{sp2,n1}^k - x_{sp2,n2}^k) & if\ (14)\ \text{is satisfied} \\ x_{best}^k - r \times (x_{sp2,n1}^k - x_{sp2,n2}^k) & if\ (15)\ \text{is satisfied} \\ LB^k + r \times (UB^k - LB^k) & if\ (16)\ \text{is satisfied} \end{cases} \tag{22}$$

Type7: differ from Type1, when (12) is satisfied, the individual of memory consideration is randomly chosen from subpopulation 2.

$$x_i^k = \begin{cases} x_{sp2,j}^k & if\ (12)\ \text{is satisfied} \\ x_{best}^k & if\ (13)\ \text{is satisfied} \\ x_{best}^k + r \times (x_{sp2,n1}^k - x_{sp2,n2}^k) & if\ (14)\ \text{is satisfied} \\ x_{best}^k - r \times (x_{sp2,n1}^k - x_{sp2,n2}^k) & if\ (15)\ \text{is satisfied} \\ LB^k + r \times (UB^k - LB^k) & if\ (16)\ \text{is satisfied} \end{cases} \tag{23}$$

Type8: differ from Type1, the individual of memory consideration is randomly chosen from subpopulation 2 if (12) is satisfied and two differential individuals are chosen from subpopulation 1 if (14) or (15) is satisfied.

$$x_i^k = \begin{cases} x_{sp2,j}^k & if\ (12)\ \text{is satisfied} \\ x_{best}^k & if\ (13)\ \text{is satisfied} \\ x_{best}^k + r \times (x_{sp1,n1}^k - x_{sp1,n2}^k) & if\ (14)\ \text{is satisfied} \\ x_{best}^k - r \times (x_{sp1,n1}^k - x_{sp1,n2}^k) & if\ (15)\ \text{is satisfied} \\ LB^k + r \times (UB^k - LB^k) & if\ (16)\ \text{is satisfied} \end{cases} \tag{24}$$

Type9: differ from Type1, the individual of memory consideration is randomly chosen from subpopulation 2 if (12) is satisfied and two differential individuals are chosen from the entire population if (14) or (15) is satisfied.

$$x_i^k = \begin{cases} x_{sp2,j}^k & if\ (12)\ \text{is satisfied} \\ x_{best}^k & if\ (13)\ \text{is satisfied} \\ x_{best}^k + r \times (x_{p,n1}^k - x_{p,n2}^k) & if\ (14)\ \text{is satisfied} \\ x_{best}^k - r \times (x_{p,n1}^k - x_{p,n2}^k) & if\ (15)\ \text{is satisfied} \\ LB^k + r \times (UB^k - LB^k) & if\ (16)\ \text{is satisfied} \end{cases} \tag{25}$$

### C. SOLUTION REPRESENTATION AND EVALUATION FUNCTION

In the basic MS, each moth individual is represented as a real vector. Lévy flights operator and fly straightly operator are defined in continuous space. However, the solution of DKP is a binary vector. To address this problem, a surjection function is devised to realize the mapping relationship from a real vector to a binary vector. Firstly, define a real vector $\mathbf{X} = [x_1, x_2, \ldots, x_n] \in [-a, a]^n$. Here parameter $a$ is a positive real number and $a = 5.0$ in this paper. Parameter n is the number of item groups. Then define $a$ corresponding binary vector $\mathbf{Y} = [y_1, y_2, \ldots, y_n]^n \in \{0, 1\}^n$. Finally, define a correspondence between X and Y as follows:

$$y_i = \begin{cases} 1 & if\ sig(x_i) \geq 0.5 \\ 0 & else \end{cases} \tag{26}$$

where $sig(x) = 1/(1 + e^{-x})$ is the sigmoid function.

The quality of any candidate solution $\mathbf{Y}$ is evaluated directly by the objective function f of the DKP. Given a potential solution $\mathbf{Y} = \{y_0, y_1, \ldots, y_{3n-1}\}$, the objective value $f(\mathbf{Y})$ is calculated by the following formula:

$$f(\mathbf{Y}) = \sum_{i=0}^{3n-1} y_i p_i \tag{27}$$

### D. TWO STAGE REPAIR AND OPTIMIZATION OPERATOR

DKP is a multi-constraint, discrete nonlinear combinatory optimization problem. Usually, the emergence of infeasible solutions is inevitable. For DKP, the probability of a binary vector $\mathbf{Y}$ as an infeasible solution is no less than $1-(1-1/2)^n$. Therefore, traditional penalty function method is not suitable for solving DKP. In this paper, the following two-stage greedy repair operator (D-GRO) and greedy optimization operator (D-GOO) [6], [7] is employed.

In DKP, there exist a weight set $W = \{w_0, w_1, \ldots, w_{3n-1}\}$, a profit set $P = \{p_0, p_1, \ldots, p_{3n-1}\}$ and the knapsack capacity C. Thus, the profit-weight ratio of each item can be defined as follows:

$$r_i = p_i/w_i, \quad i = 0, 1, \ldots, 3n-1 \tag{28}$$

Before performing D-GRO and D-GOO, firstly, all the items are sorted with Quicksort method in a non-ascending

Greedy repair operator for DKP

**Begin**

**Step 1:**

**Input:** $\mathbf{X}=\{x_0,x_1,...,x_{3n-1}\}\in\{0,1\}^n$, $\mathbf{W}=\{w_0,w_2,...,w_{3n-1}\}$, $\mathbf{P}=\{p_0,p_2,...,p_{3n-1}\}$, $H[0...3n-1]$, $C$

**Step 2: Initialization.** The binary vector **Y**=0, Boole array **F**=0, the total weight of items in the knapsack *fw*=0, the total value *fv*=0, variable *i*=0.

**Step 3: Greedy repair stage**

**While** (*fw*<C and $i\leq 3n$-1)

**If** $\begin{pmatrix}(x_{H[i]}=1)\wedge(fw+w_{H[i]}\leq C)\\ \wedge(F[\lfloor H[i]/3\rfloor]=0)\end{pmatrix}$

$fw=fw+w_{H[i]}$.

$y_{H[i]=1}, F[\lfloor H[i]/3\rfloor]=1$.

**End if**

***i*=*i*+1.**

**End while**

**Step 4: Output:** $\mathbf{Y}=\{y_0,y_1,...,y_{3n-1}\}\in\{0,1\}^n$

**End.**

**FIGURE 1.** Greedy repair operator for DKP.

Greedy optimization operator for DKP

**Begin**

**Step 1:**

**Input:** $\mathbf{Y}=\{y_0,y_1,...,y_{3n-1}\}\in\{0,1\}^n$, $\mathbf{W}=\{w_0,w_2,...,w_{3n-1}\}$, $\mathbf{P}=\{p_0,p_2,...,p_{3n-1}\}$, $\mathbf{H}[0...3n-1]$, $C$

**Step 2: Greedy optimization stage**

**For** *i*=0 to 3*n*-1

**If** $(fw+w_{H[i]}\leq C)\,\text{and}\,(F[\lfloor H[i]/3\rfloor]=0)$

$fw=fw+w_{H[i]}$

$y_{H[i]=1}, F[\lfloor H[i]/3\rfloor]=1$

**End if**

**End for**

**Step 3: Calculate the objective function value**

**For** *i*=0 to 3*n*-1

$fv=fv+y_i*p_i$.

**End for**

**Step 4: Output:** $\mathbf{Y}=\{y_0,y_1,...,y_{3n-1}\}\in\{0,1\}^n$

**End.**

**FIGURE 2.** Greedy optimization operator for DKP.

order according to the profit-weight ratio. The index value of items after sorting are recorded in an array $H$ $[0\ldots3n-1]$. Then, a Boolean array $F$ $[0\ldots n-1]$ is initialized to 0, which indicates that no item is packed into the knapsack. Given a binary vector $\mathbf{Y}=[y_0,y_1,\ldots,y_{3n-1}]\in\{0,1\}^{3n}$, the pseudo-code of D-GRO and D-GOO are illustrated in Fig. 1 and Fig.2, respectively.

### E. THE MAIN PROCEDURE OF BINARY MS METHODS FOR DKP

Through the elaborate design above, the main procedure of binary MS for DKP is illustrated in Fig. 3. Firstly, N moth

The main procedure of binary MS for DKP

**Begin**

**Step 1: Sorting.** Sort all $p_i/w_i$ in non-increasing order ($0\leq i\leq 3n-1$), and the indexes of items are recorded in array **H** [0...3*n*-1].

**Step 2: Initialization.** Generate *N* moth individuals randomly {<$\mathbf{X}_1$, $\mathbf{Y}_1$>, <$\mathbf{X}_2$, $\mathbf{Y}_2$>, … , <$\mathbf{X}_N$, $\mathbf{Y}_N$>}. Divide the whole population into subpopulation 1 and subpopulation 2 according to their fitness. Set the maximum iteration number *MaxGen* and iteration counter *G*=1; the max step $S_{\max}$=1.0; the acceleration factor $\varphi$ =0.618, and the index $\beta$ =1.5.

**Step 3: Fitness calculation**. Calculate the initial fitness of each individual, $f(\mathbf{Y}_i)$, $1\leq i\leq N$.

**Step 4: While** *G*<*MaxGen* **do**

Update subpopulation 1 with Lévy flights operator.

Update subpopulation 2 with fly straightly operator.

Perform repair and optimization with D-GRO and D-GOO.

Evaluate the fitness of the population and record the <$\mathbf{X}_{\text{gbest}}$, $\mathbf{Y}_{\text{gbest}}$>.

*G*=*G*+1.

Regroup the two newly-generated subpopulations.

Sort the population by fitness.

Divide the whole population into subpopulation 1 and subpopulation 2.

**Step 5: End while**

**Step 6: Output:** the best results.

**End.**

**FIGURE 3.** The main procedure of binary MS for DKP.

individuals as the initial population are generated randomly and then the population is divided into two subpopulations according to their fitness. Next, repeat the following evolutionary process until a termination condition is met. At each new generation, an offspring individual in subpopulation 1 is generated by Lévy flights operator. Meanwhile, an offspring individual in subpopulation 2 is updated by fly straightly operator. Then D-GRO and D-GOO are performed to repair all the infeasible solution as well as to further optimize all the feasible solution. Finally, the newly generated two subpopulations is recombined into one population.

## V. EXPERIMENTAL ANALYSIS

### A. TEST INSTANCES AND EXPERIMENTS SETTING

In this paper, 30 DKP test instances belong to three different sets: 10 uncorrelated instances (called UDKP1-UDKP10), 10 weakly correlated instances (called WDKP1-WDKP10), and 10 strongly correlated instances (called SDKP1-SDKP10). The above three types of DKP instances were first generated in [6]. The dimensions of 10 instances in each type is $300*n$ ($n=1,2\ldots,10$), respectively. The 30 DKP instances can be available at: DOI. 10.13140/RG.2.2.25166.36160

The proposed algorithms in this paper are coded in C++ and in the Microsoft Visual Studio 2015 environment. All the experiments are run on a PC with Intel (R) Core(TM) i5-2415M CPU, 2.30 GHZ, 4GB RAM.

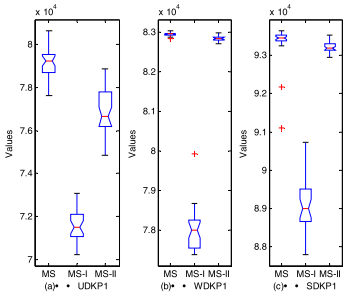


**FIGURE 4.** Boxplot on 300-dimensional DKP instances.

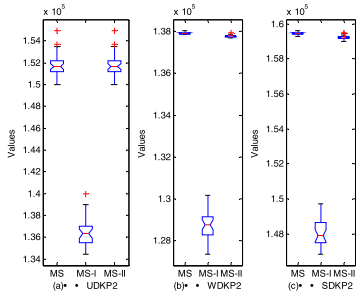


**FIGURE 5.** Boxplot on 600-dimensional DKP instances.

**TABLE 1.** The computational results of MS, MS-I, and MS-II.

| | | UDKP1 | WDKP1 | SDKP1 | UDKP2 | WDKP2 | SDKP2 |
|---|---|---|---|---|---|---|---|
| MS | Best | **80632** | **83034** | **93642** | **154945** | **138045** | **159620** |
| | Mean | **79135** | **82946** | **93339** | **151870** | **137920** | **159464** |
| | Worst | **77615** | **82834** | 91115 | **150015** | **137871** | **159297** |
| MS-I | Best | 73082 | 79936 | 90735 | 139993 | 130179 | 149724 |
| | Mean | 71571 | 78014 | 89059 | 136420 | 128729 | 148043 |
| | Worst | 70231 | 77388 | 87802 | 134447 | 127344 | 146834 |
| MS-II | Best | 78850 | 82984 | 93525 | **154945** | 137948 | 159507 |
| | Mean | 76861 | 82844 | 93208 | **151870** | 137793 | 159236 |
| | Worst | 74843 | 82699 | **92952** | **150015** | 137715 | 159016 |

In this paper, the stopping criteria is maximum number of iterations which equals to the dimension of each instance. For each instance, our algorithms were executed 30 times independently. In addition, the population size of all the algorithms is 50.

For MS, the max step Smax=1.0, acceleration factor $\varphi = 0.618$, and the index $\beta = 1.5$.

### B. ESTIMATE THE EFFECT OF TWO MAIN OPERATOR ON THE PERFORMANCE OF BINARY MS

As previously stated, the whole population is divided equally into two subpopulations based on fitness in the basic MS. Two main operators: Lévy flights operator and fly straightly operator are used to update the individuals in subpopulation 1 and subpopulation 2, respectively. After each iteration, the updated two subpopulations are reorganized into a large population. However, the effect of two main operators on the performance of binary MS was not estimated in basic MS. In this subsection, this issue is verified and then two new methods based on MS: MS-I and MS-II are proposed. In MS-I, the whole population is updated by Lévy flights operator. In MS-II, all the individuals are updated by fly straightly operator. UDKP1, WDKP1, SDKP1, UDKP2, WDKP2, SDKP2 are selected as test instances. The computational result is displayed in Table 1 and two boxplots are also provided in Fig.4 and Fig.5, respectively.

From Table 1, it is clear that the basic MS dominates the MS-I and MS-II for all the six DKP instances, which reveals that the cooperative behavior of two operators can effectively improve the comprehensive performance of MS. In addition, from Fig.4 and Fig.5, we observe that MS and MS-II have greater value and less height than those of MS-I.

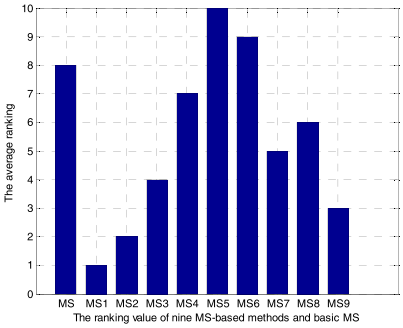


**FIGURE 6.** Comparison of the rank of nine new MS methods with basic MS.

For WDKP1, SDKP1, WDKP2, and SDKP2, MS and MS-II are more robust and effective than MS-I. From this a conclusion can be reached that Lévy flights operator has less influence on MS than fly straightly operator. Based on the above analysis, Lévy flights operator in the basic MS algorithm is replaced with (15) - (23), respectively. Then nine types of new MS-based methods are developed (called MS1, MS2, …, MS9, respectively).

### C. COMPARISONS OF NINE NEW MS APPROACHES WITH BASIC MS

In this section, the comparative results of nine new MS-based methods with basic MS on 30 DKP instances are summarized in Tables 2-4, respectively. For MS1-MS9, HMCR=0.9 and PAR=0.9. In these three tables, three basic evaluation criteria, i.e., ''Best'', ''Mean'', and ''Worst'' are considered to evaluate each method. ''Best'', ''Mean'', and ''Worst'' denote the best value, the average value, and the worst value for each instance reached by each algorithm among 30 times independent running. The values in parentheses in the first column indicate the optimal solution value (Opt) calculated by DP method [5]. The value in bold in each row indicates the best value for each DKP instance obtained by all the algorithms.

From Table 2, we observe that MS1 attains 6 best values, 4 mean values, and 2 worst values on the 10 uncorrelated DKP instances, which is the largest among the results of the 10 algorithms in comparison. MS2 attains 2 best values, while MS3 and MS7 obtain best value once. A conclusion is that MS1 demonstrates the best comprehensive performance when solving uncorrelated DKP.

**TABLE 2.** Comparisons of nine new MS methods with basic MS on uncorrelated DKP instances.

| | | MS | MS1 | MS2 | MS3 | MS4 | MS5 | MS6 | MS7 | MS8 | MS9 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| UDKP1 ( 85740) | Best | 80632 | 84200 | 84286 | 84576 | 84803 | 84416 | 84336 | **85216** | 84951 | 85178 |
| | Mean | 79135 | 82763 | 82356 | 82047 | 83764 | 83279 | 83429 | 84013 | 84070 | **84180** |
| | Worst | 77615 | 81131 | 79710 | 79658 | 81452 | 80032 | 81864 | 81629 | 81280 | **82033** |
| UDKP2 (163744) | Best | 154945 | 161133 | 161118 | **161886** | 158913 | 158350 | 155276 | 156668 | 156059 | 157813 |
| | Mean | 151870 | 158503 | **158663** | 158253 | 157683 | 156882 | 153690 | 154744 | 154538 | 155793 |
| | Worst | 1150015 | 155911 | 154760 | 154918 | **156779** | 155828 | 152546 | 153267 | 152898 | 154232 |
| UDKP3 (269393) | Best | 238755 | 251954 | **252939** | 239173 | 239903 | 239055 | 239489 | 241369 | 239978 | 241751 |
| | Mean | 235192 | **249646** | 249141 | 236276 | 237747 | 236803 | 236978 | 239200 | 238478 | 240340 |
| | Worst | 233184 | 244938 | **246213** | 234510 | 236108 | 235205 | 234874 | 238077 | 237270 | 238986 |
| UDKP4 (347599) | Best | 317591 | **332554** | 327923 | 318371 | 321745 | 319341 | 320423 | 321292 | 321486 | 322317 |
| | Mean | 315365 | **320776** | 320549 | 315380 | 318787 | 317714 | 318758 | 319567 | 319001 | 320641 |
| | Worst | 313614 | 315150 | 315665 | 312962 | 317176 | 315916 | 316450 | 317545 | 316106 | **319276** |
| UDKP5 (442644) | Best | 381323 | **405222** | 402759 | 384025 | 387017 | 385640 | 385495 | 387061 | 389048 | 387343 |
| | Mean | 378195 | **400653** | 397953 | 379639 | 382257 | 382575 | 382403 | 383437 | 382508 | 385104 |
| | Worst | 375865 | **395533** | 389724 | 377091 | 379589 | 378401 | 378812 | 380686 | 380759 | 382929 |
| UDKP6 (536578) | Best | 462859 | **487014** | 484623 | 464864 | 464961 | 462448 | 465223 | 466920 | 466486 | 472212 |
| | Mean | 458733 | **481401** | 480548 | 459774 | 462163 | 460576 | 460946 | 464207 | 463354 | 466851 |
| | Worst | 456086 | **476628** | 474558 | 456869 | 460437 | 458315 | 458849 | 462573 | 461171 | 464922 |
| UDKP7 (635860) | Best | 574229 | **618146** | 614942 | 612998 | 573473 | 570923 | 572037 | 577668 | 577452 | 584711 |
| | Mean | 562736 | 604287 | 604868 | **605584** | 570402 | 566967 | 568261 | 574534 | 572301 | 578133 |
| | Worst | 557849 | 588175 | 590499 | **594060** | 567762 | 564197 | 564793 | 571382 | 569271 | 573996 |
| UDKP8 (650206) | Best | 581456 | **596452** | 586027 | 587251 | 586921 | 584442 | 585345 | 589724 | 587185 | 590636 |
| | Mean | 578208 | 581196 | 581087 | 580856 | 584865 | 582292 | 582558 | 585285 | 584680 | **586467** |
| | Worst | 575686 | 575279 | 576939 | 573465 | 583445 | 579483 | 580631 | 583094 | 583054 | **584135** |
| UDKP9 (718532) | Best | 640669 | **661984** | 660237 | 658067 | 647643 | 646966 | 648973 | 647856 | 649121 | 649431 |
| | Mean | 638165 | **652572** | 652389 | 650236 | 644431 | 642643 | 643381 | 645333 | 645347 | 646441 |
| | Worst | 636382 | 644955 | **645214** | 642465 | 642338 | 640733 | 641007 | 643331 | 643172 | 644648 |
| UDKP10 (779460) | Best | 709352 | 719003 | **724875** | 719692 | 713077 | 712151 | 710615 | 713101 | 715935 | 713133 |
| | Mean | 704405 | 713858 | 714802 | **714992** | 709777 | 708066 | 708305 | 710298 | 710395 | 711297 |
| | Worst | 702757 | 706131 | 708580 | 707831 | 707956 | 706154 | 706532 | 707980 | 708523 | **709584** |

**TABLE 3.** Comparisons of nine new MS methods with basic MS on weakly correlated DKP instances.

| | | MS | MS1 | MS2 | MS3 | MS4 | MS5 | MS6 | MS7 | MS8 | MS9 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| WDKP1 (83098) | Best | 83034 | 83074 | 83074 | 83047 | 83083 | 83086 | 83082 | **83090** | 83087 | 83088 |
| | Mean | 82946 | 82947 | 82926 | 82922 | 82875 | 82772 | 82819 | 82946 | 82944 | **82967** |
| | Worst | **82834** | 82800 | 82814 | 82814 | 81892 | 81399 | 81726 | 82026 | 82068 | 82063 |
| WDKP2 (138215) | Best | 138045 | **138143** | 138106 | 138019 | 138034 | 138040 | 138016 | 138070 | 138070 | 138142 |
| | Mean | 137920 | 137989 | 137993 | 137918 | 137947 | 137913 | 137927 | 137970 | 137970 | **138031** |
| | Worst | 137871 | 137840 | 137830 | 137840 | 137876 | 137829 | 137867 | 137884 | 137884 | **137971** |
| WDKP3 (256616) | Best | **255187** | 248982 | 249734 | 249166 | 246412 | 246165 | 246494 | 247143 | 246966 | 246703 |
| | Mean | **254883** | 248318 | 248237 | 248286 | 245587 | 245380 | 245420 | 246039 | 245998 | 246129 |
| | Worst | **254638** | 247714 | 247486 | 247819 | 244810 | 244561 | 244580 | 245450 | 245399 | 245445 |
| WDKP4 (315657) | Best | 314617 | 314905 | 314901 | **315041** | 314715 | 314612 | 314665 | 314693 | 314653 | 314902 |
| | Mean | 314544 | 314612 | 314600 | 314602 | 314553 | 314516 | 314512 | 314554 | 314544 | **314645** |
| | Worst | 314470 | 314321 | 314379 | 314248 | 314463 | 314451 | 314444 | 314472 | 314457 | **314497** |
| WDKP5 (428490) | Best | 427250 | 427530 | **427847** | 427593 | 427353 | 427245 | 427295 | 427313 | 427241 | 427372 |
| | Mean | 427160 | 427173 | **427193** | 427139 | 427149 | 427084 | 427112 | 427150 | 427104 | 427189 |
| | Worst | **427064** | 426876 | 426914 | 426879 | 426987 | 426986 | 426987 | 427015 | 426988 | 427031 |
| WDKP6 (466050) | Best | 464786 | 464993 | 464953 | **465096** | 464705 | 464641 | 464662 | 464760 | 464683 | 464784 |
| | Mean | 464612 | **464701** | 464656 | 464690 | 464561 | 464540 | 464545 | 464559 | 464555 | 464575 |
| | Worst | **464530** | 464383 | 464401 | 464376 | 464468 | 464437 | 464426 | 464477 | 464432 | 464448 |
| WDKP7 (547683) | Best | 545260 | **545607** | 545528 | 545584 | 545321 | 545319 | 545252 | 545308 | 545311 | 545311 |
| | Mean | 545165 | 545241 | 545251 | **545251** | 545138 | 545109 | 545109 | 545122 | 545132 | 545132 |
| | Worst | **545058** | 544912 | 544986 | 545018 | 545026 | 545023 | 545023 | 545014 | 544994 | 544994 |
| WDKP8 (576959) | Best | 575314 | 575387 | **575457** | 575396 | 575226 | 575255 | 575162 | 575294 | 575229 | 575260 |
| | Mean | **575141** | 575126 | 575137 | 575116 | 575112 | 575094 | 575093 | 575118 | 575112 | 575140 |
| | Worst | **575052** | 574920 | 574931 | 574875 | 575034 | 575011 | 575024 | 575024 | 575024 | 575041 |
| WDKP9 (650660) | Best | 648962 | **649059** | 648977 | 649035 | 648951 | 648947 | 648939 | 649000 | 648918 | 649049 |
| | Mean | **648870** | 648813 | 648799 | 648798 | 648843 | 648844 | 648835 | 648862 | 648841 | 648862 |
| | Worst | **648813** | 648611 | 648621 | 648613 | 648782 | 648777 | 648772 | 648772 | 648769 | 648788 |
| WDKP10 (678967) | Best | 677696 | 677817 | **677887** | 677799 | 677683 | 677721 | 677704 | 677715 | 677715 | 677761 |
| | Mean | 677623 | 677581 | 677585 | 677559 | 677605 | 677603 | 677602 | 677618 | 677613 | **677633** |
| | Worst | **677550** | 677401 | 677392 | 677359 | 677533 | 677533 | 677533 | 677538 | 677533 | 677549 |

From Table 3, the ability of MS1 to find good solutions is similar to that of MS2, because they both obtain 3 best values on the 10 weakly correlated DKP instances. MS3 reaches 2 best values while MS and MS7 both get the best values once. Additionally, MS attains 8 worst values and MS9 obtains 5 mean values on 10 WDKP instances.

**TABLE 4.** Comparisons of nine new MS methods with basic MS on strongly correlated DKP instances.

| | | MS | MS1 | MS2 | MS3 | MS4 | MS5 | MS6 | MS7 | MS8 | MS9 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| SDKP1 (94459) | Best | 93642 | 94030 | 93977 | 93887 | 93923 | 93793 | 93846 | 93969 | 93869 | **94032** |
| | Mean | 93339 | **93695** | 93650 | 93681 | 93608 | 93582 | 93510 | 93506 | 93452 | 93661 |
| | Worst | 91115 | 93376 | 93332 | **93448** | 93006 | 93360 | 92617 | 92570 | 92605 | 92715 |
| SDKP2 (160805) | Best | **159620** | 159019 | 158722 | 158632 | 156746 | 156530 | 156225 | 157108 | 156808 | 157615 |
| | Mean | **159464** | 158082 | 157778 | 157927 | 155777 | 155379 | 155630 | 156308 | 156120 | 156505 |
| | Worst | **159297** | 155574 | 155840 | 155918 | 155311 | 154741 | 155163 | 155805 | 155672 | 155935 |
| SDKP3 (238248) | Best | 236154 | **236634** | 236514 | 236522 | 236174 | 236074 | 236175 | 236270 | 236257 | 236550 |
| | Mean | 235920 | 236070 | 236088 | 236063 | 235960 | 235887 | 235924 | 236064 | 236020 | **236258** |
| | Worst | 235751 | 235765 | 235575 | 235706 | 235847 | 235752 | 235730 | 235892 | 235789 | **236078** |
| SDKP4 (340027) | Best | 337224 | **337954** | 337806 | 337627 | 337290 | 337161 | 337207 | 337377 | 337329 | 337563 |
| | Mean | 337063 | 337248 | 337219 | 337245 | 337091 | 337018 | 337017 | 337123 | 337108 | **337334** |
| | Worst | 336934 | 336834 | 336907 | 336864 | 336922 | 336917 | 336874 | 336928 | 336958 | **336898** |
| SDKP5 (463033) | Best | 454627 | 455491 | 455476 | 455244 | 454930 | 454796 | 454567 | 455144 | 455113 | **455530** |
| | Mean | 453561 | 454026 | 454042 | 453877 | 454471 | 454183 | 454071 | 454579 | 454549 | **454808** |
| | Worst | 452588 | 452553 | 450924 | 451637 | 453840 | 453582 | 453593 | 453893 | **454159** | 453871 |
| SDKP6 (466097) | Best | 460686 | 461242 | **461518** | 461181 | 460855 | 460642 | 460759 | 460805 | 460986 | 461049 |
| | Mean | 460418 | 460729 | **460859** | 460816 | 460484 | 460401 | 460390 | 460489 | 460447 | 460643 |
| | Worst | 460224 | 460178 | 460057 | **460438** | 460265 | 460247 | 460216 | 460313 | 460241 | 460387 |
| SDKP7 (620446) | Best | 609415 | 609852 | **610470** | 609927 | 610254 | 609685 | 609815 | 610086 | 609859 | 610269 |
| | Mean | 608523 | 608712 | 608853 | 608733 | 609397 | 609245 | 609201 | 609450 | 609335 | **609712** |
| | Worst | 607797 | 604763 | 606912 | 606995 | 608666 | 608822 | 608355 | 609027 | 608708 | **609072** |
| SDKP8 (670697) | Best | 663380 | 663804 | 663672 | **663819** | 663273 | 663073 | 663078 | 663192 | 663157 | 663423 |
| | Mean | 663012 | 663103 | **663171** | 663163 | 662942 | 662897 | 662924 | 662964 | 662941 | 663064 |
| | Worst | 662895 | 662574 | 662660 | 662727 | 662781 | 662762 | 662783 | 662812 | 662762 | **662897** |
| SDKP9 (739121) | Best | 730861 | **731439** | 731264 | 731156 | 730590 | 730555 | 730847 | 730935 | 730606 | 731012 |
| | Mean | 730547 | 730654 | 730667 | **730717** | 730455 | 730431 | 730474 | 730526 | 730457 | 730602 |
| | Worst | **730450** | 730204 | 730163 | 730273 | 730333 | 730300 | 730298 | 730347 | 730353 | 730405 |
| SDKP10 (765317) | Best | 757636 | 757821 | **757955** | 757806 | 757570 | 757579 | 757520 | 757498 | 757580 | 757787 |
| | Mean | 757390 | 757466 | 757479 | 757491 | 757372 | 757331 | 757345 | 757400 | 757391 | **757511** |
| | Worst | 757290 | 757158 | 757101 | 757208 | 757250 | 757197 | 757202 | 757291 | 757235 | **757327** |

From Table 4, MS1 and MS2 still remain excellent performance when solving strongly correlated DKP instances, which is deduced from that MS1 and MS2 both obtain 3 best values on 10 SDKP instances. We observe MS9 attains 2 best values while MS and MS1 reach the best values once. In addition, MS9 shows better performance in compared to other 9 methods in terms of the mean value and the worst value.

To further assess the performance of 10 methods clearly, the ranks of each method for 30 DKP instances based on the best values is displayed in Table 5 and Fig. 6, respectively. As can be seen from Table 5 and Fig. 6, it is obvious that MS1 ranks first for all 30 DKP instances among 9 MS-based new methods and basic MS. MS2 and MS9 are the second and the third best among 10 methods. Additionally, basic MS ranks eighth which can be deduced that new mutation operator based GHS can improve the chance of finding the global optima as well as the performance of the MS significantly.

Through an in-depth analysis of (15)- (23), MS1, MS2, and MS3 with better performance have common feature that the individual of memory consideration are all randomly selected from the entire population, which fully explain memory consideration plays an important role in the evolution process of nine MS-based methods. Furthermore, because the individual comes from the whole population, not just subpopulation 1 or subpopulation 2, MS1, MS2, and MS3 can gain more useful information from excellent diversified individuals. As noted earlier, MS1 shows the best overall performance. Compared with MS2 and MS3, it is not difficult to find that two differential individuals are randomly selected from subpopulation 2 in MS1, which can increase the chance of exchanging significant information between two subpopulations. The rankings of 10 methods based on the best values are as follows:

$$\begin{aligned} &\text{MS1} \succ \text{MS2} \succ \text{MS9} \succ \text{MS3} \succ \text{MS7} \\ &\quad \succ \text{MS8} \succ \text{MS4} \succ \text{MS} \succ \text{MS6} \succ \text{MS5} \end{aligned} \tag{29}$$

To evaluate the similar level of the theoretical optimal value to the best value gained by each algorithm, the approximation ratio [29] based on the best value (ARB) is defined as follows:

$$ARB = Opt/Best \tag{30}$$

where *Opt* represents the optimal solution value and *Best* is the best approximate solution value.

For a problem of seeking maximum, *ARB* is obviously a real value of no less than 1.0. Furthermore, the closer that the *ARB* value is to 1.0, the better the quality of a solution. To show the performance of 10 methods more intuitively, the comparative results of *ARB* values of 10 UDKP instances, 10 WDKP instances, and 10 SDKP instances are illustrated in Figs. 7, 8, and 9, respectively. In Fig. 7, we can see that MS1 has the smallest *ARB* for UDKP4-UDKP6 among 10 methods, which is consistent with previous analyses.

**TABLE 5.** Ranks of nine new MS methods with basic MS based on the best values.

| | MS | MS1 | MS2 | MS3 | MS4 | MS5 | MS6 | MS7 | MS8 | MS9 |
|---|---|---|---|---|---|---|---|---|---|---|
| UDKP1 | 10 | 9 | 8 | 5 | 4 | 6 | 7 | 1 | 3 | 2 |
| UDKP2 | 10 | 2 | 3 | 1 | 4 | 5 | 9 | 7 | 8 | 6 |
| UDKP3 | 10 | 2 | 1 | 8 | 6 | 9 | 7 | 4 | 5 | 3 |
| UDKP4 | 10 | 1 | 2 | 9 | 4 | 8 | 7 | 6 | 5 | 3 |
| UDKP5 | 10 | 1 | 2 | 9 | 6 | 7 | 8 | 5 | 3 | 4 |
| UDKP6 | 9 | 1 | 2 | 8 | 7 | 10 | 6 | 4 | 5 | 3 |
| UDKP7 | 7 | 1 | 2 | 3 | 8 | 10 | 9 | 5 | 6 | 4 |
| UDKP8 | 10 | 1 | 7 | 4 | 6 | 9 | 8 | 3 | 5 | 2 |
| UDKP9 | 10 | 1 | 2 | 3 | 8 | 9 | 6 | 7 | 5 | 4 |
| UDKP10 | 10 | 3 | 1 | 2 | 7 | 8 | 9 | 6 | 4 | 5 |
| WDKP1 | 10 | 7 | 7 | 9 | 5 | 4 | 6 | 1 | 3 | 2 |
| WDKP2 | 6 | 1 | 3 | 9 | 8 | 7 | 10 | 4 | 4 | 2 |
| WDKP3 | 1 | 4 | 2 | 3 | 9 | 10 | 8 | 5 | 6 | 7 |
| WDKP4 | 9 | 2 | 4 | 1 | 5 | 10 | 7 | 6 | 8 | 3 |
| WDKP5 | 8 | 3 | 1 | 2 | 5 | 9 | 7 | 6 | 10 | 4 |
| WDKP6 | 4 | 2 | 3 | 1 | 7 | 10 | 9 | 6 | 8 | 5 |
| WDKP7 | 9 | 1 | 3 | 2 | 4 | 5 | 10 | 8 | 6 | 6 |
| WDKP8 | 4 | 3 | 1 | 2 | 9 | 7 | 10 | 5 | 8 | 6 |
| WDKP9 | 6 | 1 | 5 | 3 | 7 | 8 | 9 | 4 | 10 | 2 |
| WDKP10 | 9 | 2 | 1 | 3 | 10 | 5 | 8 | 6 | 6 | 4 |
| SDKP1 | 10 | 2 | 3 | 6 | 5 | 9 | 8 | 4 | 7 | 1 |
| SDKP2 | 1 | 2 | 3 | 4 | 8 | 9 | 10 | 6 | 7 | 5 |
| SDKP3 | 9 | 1 | 4 | 3 | 8 | 10 | 7 | 5 | 6 | 2 |
| SDKP4 | 8 | 1 | 2 | 3 | 7 | 10 | 9 | 5 | 6 | 4 |
| SDKP5 | 9 | 2 | 3 | 4 | 7 | 8 | 10 | 5 | 6 | 1 |
| SDKP6 | 9 | 2 | 1 | 3 | 6 | 10 | 8 | 7 | 5 | 4 |
| SDKP7 | 10 | 7 | 1 | 5 | 3 | 9 | 8 | 4 | 6 | 2 |
| SDKP8 | 5 | 2 | 3 | 1 | 6 | 10 | 9 | 7 | 8 | 4 |
| SDKP9 | 6 | 1 | 2 | 3 | 9 | 10 | 7 | 5 | 8 | 4 |
| SDKP10 | 5 | 2 | 1 | 3 | 8 | 7 | 9 | 10 | 6 | 4 |
| Mean Rank | 7.80 | **2.33** | **2.77** | 4.07 | 6.53 | 8.27 | 8.17 | 5.23 | 6.10 | **3.60** |

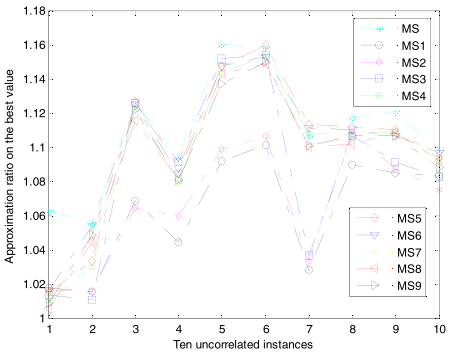


**FIGURE 7.** Comparisons of the approximation ratio on UDKP1-UDKP10.

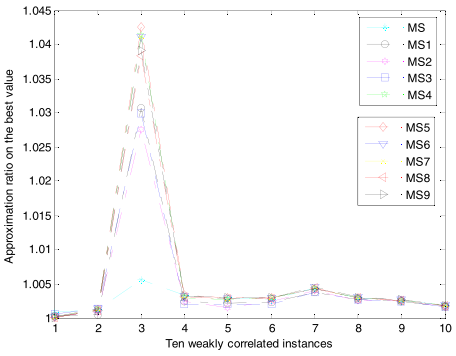


**FIGURE 8.** Comparisons of the approximation ratio on WDKP1-WDKP10.

In general, 10 algorithms can be divided into two groups. MS1, MS2, and MS3 constitute a group with better performance, and the other 7 methods form another group. In Fig.8, all 10 algorithms can obtain very small *ARB* values which are less than 1.005 on all 10 WDKP instances except for WDKP3. It indicates that 9 new MS-based methods and basic MS are very suitable for solving WDKP instances. In Fig.9, *ARB* values of 10 methods have no significant difference for 9 SDKP instances except for SDKP2.

To visualize the experimental results of 10 methods from a statistical perspective, the corresponding boxplots of three higher-dimensional DKP instances (UDKP10, WDKP10, and SDKP10) are provided in Figs. 10, 11, and 12, respectively. From Figs. 10-12, we can make two observations, 1) the

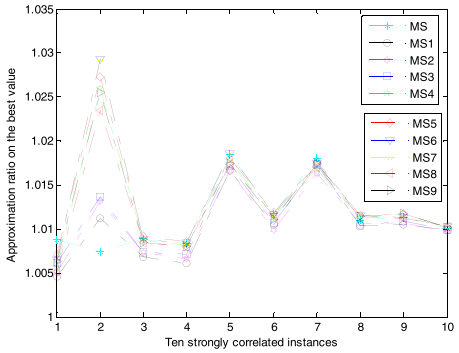


**FIGURE 9.** Comparisons of the approximation ratio on SDKP1-SD KP10.

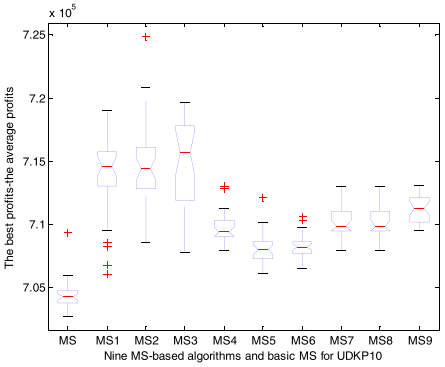


**FIGURE 10.** Boxplot of the best values on UDKP10 in 30 runs.

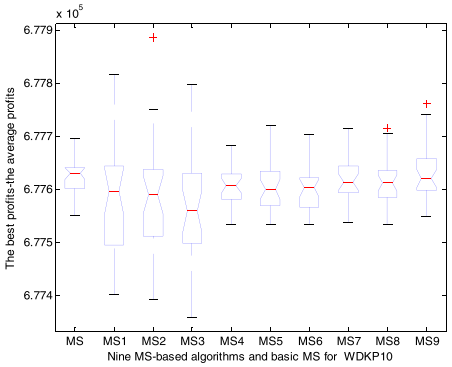


**FIGURE 11.** Boxplot of the best values on WDKP10 in 30 runs.

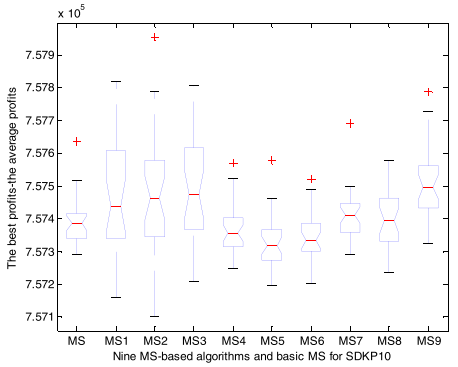


**FIGURE 12.** Boxplot of the best values on SDKP10 in 30 runs.

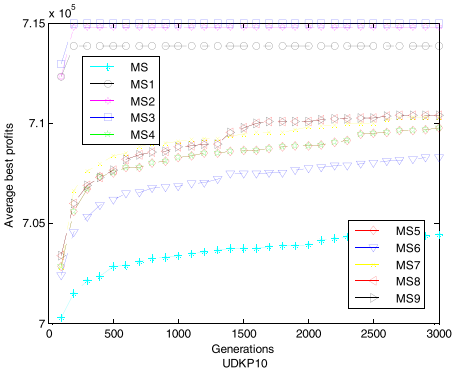


**FIGURE 13.** The convergence graph of ten algorithms on UDKP10.

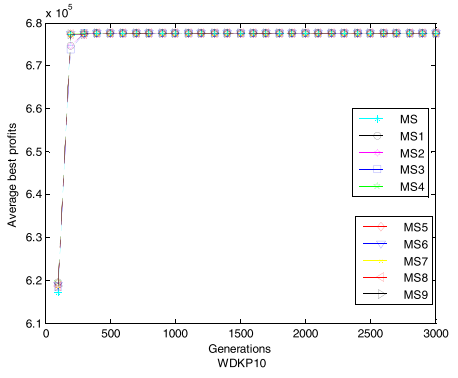


**FIGURE 14.** The convergence graph of ten algorithms on WDKP10.

boxplots of MS1, MS2, and MS3 have greater value but greater height than the other 7 methods, 2) the 10 methods can be roughly divided into 4 groups: group1 (MS), group2 (MS1-MS3), group3 (MS4-MS6), group4 (MS7-MS9). The four groups are exactly corresponding to the above (15)-(23), in which the individuals of memory consideration in group2, group3, and group4 are randomly selected from the entire population, subpopulation1 and subpopulation 2, respectively.

To gain some insights on the optimization process of 10 methods, the evolutionary curves on three representative higher-dimensional DKP instances (UDKP10, WDKP10, and SDKP10) are provided in Figs. 13-15. In these three figures, all the function values are calculated based on the mean best values among 30 independent runs. From Fig. 13, three observations can be obtained: 1) the initial values of MS1, MS2, and MS3 are greater than those of other 7 methods, furthermore, they have fast convergence speed to better global optimum. 2) MS4-MS9 show second good convergence behavior among 10 methods. 3) As the above discussion, MS has the smallest initial value. From Fig. 14, ten methods demonstrate extremely similar convergence behavior when solving weakly correlated WDKP10 instance. The same scene appears in Fig. 15 while the difference is the initial value.

From the above comprehensive analysis, some conclusions can be derived: 1) Nine new MS-based variants show remarkable performance compared with basic MS. 2) MS1, MS2, and MS3 exhibit the best overall performance due to their individuals of memory consideration. 3) The profits

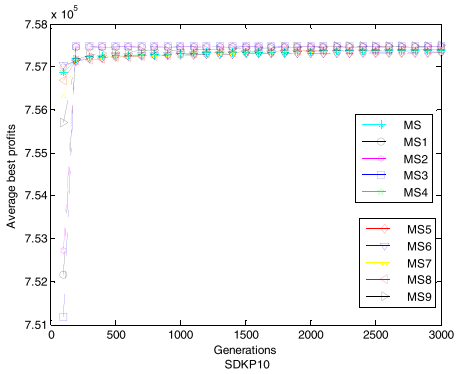


**FIGURE 15.** The convergence graph of ten algorithms on SDKP10.

and weights of three types of DKP instances are randomly generated over a wide range, however, 10 methods in this paper can obtain better approximate solutions with an approximation ratio approaching 1 quickly. Therefore, 10 effective new methods are all suitable for solving large-scale difficult DKP instances.

## VI. CONCLUSION

In this paper, 10 novel effective binary MS-based algorithms (MS1-MS9, MS) are proposed for the discounted {0-1} KP problem (DKP). DKP is a useful model in practice while presenting a real computational challenge. The effect of Lévy flights operator and fly straightly operator on basic MS are firstly evaluated. Then nine new MS-based algorithms combining GHS-based mutation operator are proposed.

Computational assessments on three sets of 30 DKP instances indicate that the proposed nine methods have advanced performance both in solution quality and computational efficiency compared to the basic MS. For all 30 DKP instances, MS1 reveal the best performance for it can attain 12 best values and 11 second best values.

Finally, it is expected that the ideas behind the GHS-based mutation operator developed in this work would be useful to other constrained knapsack problems, such as multiple-choice multidimensional knapsack problem (MMKP) [30], [31], set-union knapsack problem (SUKP) [32], and more generally combination optimization problems.

## REFERENCES

[1] T. H. Cormen, *Introduction to Algorithms*. Cambridge, MA, USA: MIT Press, 2009.
[2] D. Z. Du and K. I. Ko, *Theory of Computational Complexity*. Hoboken, NJ, USA: Wiley, 2011.
[3] H. Zhu, Y. He, X. Wang, and E. C. C. Tsang, "Discrete differential evolutions for the discounted 0-1 knapsack problem," *Int. J. Bio-Inspired Comput.*, vol. 10, no. 4, pp. 219–238, 2017.
[4] R. Bellman, "Dynamic programming and Lagrange multipliers," *Proc. Nat. Acad. Sci. USA*, vol. 42, no. 10, pp. 767–769, 1956.
[5] A. Rong, J. R. Figueira, and K. Klamroth, "Dynamic programming based algorithms for the discounted 0-1 knapsack problem," *Appl. Math. Comput.*, vol. 218, no. 12, pp. 6921–6933, 2012.
[6] Y. C. He, X. Z. Wang, W. B. Li, X. L. Zhang, and Y. Y. Chen, "Research on genetic algorithms for the discounted 0-1 knapsack problem," *Chin. J. Comput.*, vol. 39, no. 12, pp. 2614–2630, 2016.
[7] Y. C. He, X.-Z. Wang, Y.-L. He, S.-L. Zhao, and W.-B. Li, "Exact and approximate algorithms for discounted 0-1 knapsack problem," *Inf. Sci.*, vol. 369, pp. 634–647, Nov. 2016.
[8] Y. H. Feng, G.-G. Wang, W. B. Li, and N. Li, "Multi-strategy monarch butterfly optimization algorithm for discounted {0-1} knapsack problem," *Neural Comput. Appl.*, to be published, doi: 10.1007/s00521-017-2903-1.
[9] Y. H. Feng, J. Yang, Y. C. He, and G.-G. Wang, "Monarch butterfly optimization algorithm with differential evolution for the discounted 0-1 knapsack problem," *Acta Electronica Sinica*, to be published.
[10] A. K. Qin, V. L. Huang, and P. N. Suganthan, "Differential evolution algorithm with strategy adaptation for global numerical optimization," *IEEE Trans. Evol. Comput.*, vol. 13, no. 2, pp. 398–417, Apr. 2009.
[11] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. IEEE Int. Conf. Neural Netw.*, Dec. 1995, pp. 1942–1948.
[12] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: Optimization by a colony of cooperating agents," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 26, no. 1, pp. 29–41, Feb. 1996.
[13] Y. Xue, J. Jiang, B. Zhao, and T. Ma, "A self-adaptive artificial bee colony algorithm based on global best for global optimization," *Soft Comput.*, to be published, doi: 10.1007/s00500-017-2547-1.
[14] L. Wang, X. L. Zheng, and S. Y. Wang, "A novel binary fruit fly optimization algorithm for solving the multidimensional knapsack problem," *Knowl.-Based Syst.*, vol. 48, no. 2, pp. 17–23, 2013.
[15] L. Wang, R. Yang, H. Ni, W. Ye, M. Fei, and P. M. Pardalos, "A human learning optimization algorithm and its application to multi-dimensional knapsack problems," *Appl. Soft Comput.*, vol. 34, pp. 736–743, Sep. 2015.
[16] G.-G. Wang, L. H. Guo, A. H. Gandomi, G.-S. Hao, and H. Q. Wang, "Chaotic krill herd algorithm," *Inf. Sci.*, vol. 274, pp. 17–34, Aug. 2014.
[17] G.-G. Wang, A. H. Gandomi, and A. H. Alavi, "Stud krill herd algorithm," *Neurocomputing*, vol. 128, pp. 363–370, Mar. 2014.
[18] G.-G. Wang, S. Deb, A. H. Gandomi, and A. H. Alavi, "Opposition-based krill herd algorithm with Cauchy mutation and position clamping," *Neurocomputing*, vol. 177, pp. 147–157, Feb. 2016.
[19] G.-G. Wang, S. Deb, and L. dos S. Coelho, "Elephant herding optimization," in *Proc. 3rd Int. Symp. Comput. Bus. Intell. (ISCBI)*, Bali, Indonesia, 2015, pp. 1–5.
[20] X. Li, J. Zhang, and M. Yin, "Animal migration optimization: An optimization algorithm inspired by animal migration behavior," *Neural Comput. Appl.*, vol. 24, nos. 7–8, pp. 1867–1877, 2014.
[21] D. Simon, "Biogeography-based optimization," *IEEE Trans. Evol. Comput.*, vol. 12, no. 6, pp. 702–713, Dec. 2008.
[22] X. Li, J. Wang, J. Zhou, and M. Yin, "A perturb biogeography based optimization with mutation for global numerical optimization," *Appl. Mathe. Comput.*, vol. 218, no. 2, pp. 598–609, 2011.
[23] G.-G. Wang, S. Deb, and Z. H. Cui, "Monarch butterfly optimization," *Neural Comput. Appl.*, to be published, doi: 10.1007/s00521-015-1923-y.
[24] Y. H. Feng, G.-G. Wang, S. Deb, M. Lu, and X. J. Zhao, "Solving 0-1 knapsack problem by a novel binary monarch butterfly optimization," *Neural Comput. Appl.*, vol. 28, no. 7, pp. 1619–1634, 2017.
[25] G.-G. Wang, S. Deb, and L. D. S. Coelho, "Earthworm optimization algorithm: A bio-inspired metaheuristic algorithm for global optimization problems," *Int. J. Bio-Inspired Comput.*, to be published, doi: 10.1504/IJBIC.2015.10004283.
[26] G.-G. Wang, "Moth search algorithm: A bio-inspired metaheuristic algorithm for global optimization problems," *Memetic Comput.*, to be published, doi: 10.1007/s12293-016-0212-3.
[27] Z. W. Geem, J. H. Kim, and G. V. Loganathan, "A new heuristic optimization algorithm: Harmony search," *J. Simul.*, vol. 76, no. 2, pp. 60–68, Feb. 2001.
[28] M. G. H. Omran and M. Mahdavi, "Global-best harmony search," *Appl. Math. Comput.*, vol. 198, no. 2, pp. 643–656, 2008.
[29] M. H. Alsuwaiyel, *Algorithms Design Techniques and Analysis*. Singapore: World Scientific, 2009.
[30] B. Han, J. Leblet, and G. Simon, "Hard multidimensional multiple choice knapsack problems, an empirical study," *Comput. Oper. Res.*, vol. 37, no. 1, pp. 172–181, 2010.
[31] Z. Ren, Z. Feng, and A. Zhang, "Fusing ant colony optimization with Lagrangian relaxation for the multiple-choice multidimensional knapsack problem," *Inf. Sci.*, vol. 182, no. 1, pp. 15–29, 2012.
[32] Y. C. He, H. Xie, T.-L. Wong, and X. Wang, "A novel binary artificial bee colony algorithm for the set-union knapsack problem," *Future Generat. Comput. Syst.*, vol. 78, pp. 77–86, Jan. 2018.

**YAN-HONG FENG** received the bachelor's degree from the School of Information Science and Engineering, Hebei University of Science and Technology, Shijiazhuang, in 2001, and the M.E. degree from the Department of Computer Science, North China Electronic Power University, Baoding, in 2006. She is currently an Associate Professor with the School of Information Engineering, Hebei GEO University. Her major research interests include swarm intelligence, evolutionary computation, and approximation algorithm.

**GAI-GE WANG** is currently an Associate Professor with the Ocean University of China, China. He just proposed five bio-inspired algorithms (monarch butterfly optimization, earthworm optimization algorithm, elephant herding optimization, moth search algorithm, and rhino herd). His research interests are swarm intelligence, evolutionary computation, and big data optimization. He has been an Associate Editor of IJCISIM and an Editorial Board Member of IJBIC since 2016.