Neural Comput & Applic (2017) 28:1619–1634
DOI 10.1007/s00521-015-2135-1

ORIGINAL ARTICLE

Solving 0–1 knapsack problem by a novel binary monarch butterfly optimization

Yanhong Feng[1] · Gai-Ge Wang[2,3,4] · Suash Deb[5] · Mei Lu[2] · Xiang-Jun Zhao[2]

Received: 20 September 2015 / Accepted: 16 December 2015 / Published online: 28 December 2015

Abstract This paper presents a novel binary monarch butterfly optimization (BMBO) method, intended for addressing the 0–1 knapsack problem (0–1 KP). Two tuples, consisting of real-valued vectors and binary vectors, are used to represent the monarch butterfly individuals in BMBO. Real-valued vectors constitute the search space, whereas binary vectors form the solution space. In other words, monarch butterfly optimization works directly on real-valued vectors, while solutions are represented by binary vectors. Three kinds of individual allocation schemes are tested in order to achieve better performance. Toward revising the infeasible solutions and optimizing the feasible ones, a novel repair operator, based on greedy strategy, is employed. Comprehensive numerical experimentations on three types of 0–1 KP instances are carried out. The comparative study of the BMBO with four state-of-the-art classical algorithms clearly points toward the superiority of the former in terms of search accuracy, convergent capability and stability in solving the 0–1 KP, especially for the high-dimensional instances.

Keywords Evolutionary computation · Monarch butterfly optimization · Knapsack problems · Greedy optimization algorithm

✉ Gai-Ge Wang
gaigewang@163.com; gaigewang@gmail.com

Yanhong Feng
qinfyh@163.com

Suash Deb
suashdeb@gmail.com

Mei Lu
lumei@jsnu.edu.cn

Xiang-Jun Zhao
xjzhao@jsnu.edu.cn

[1] School of Information Engineering, Shijiazhuang University of Economics, Shijiazhuang 050031, Hebei, China

[2] School of Computer Science and Technology, Jiangsu Normal University, Xuzhou 221116, Jiangsu, China

[3] Institute of Algorithm and Big Data Analysis, Northeast Normal University, Changchun 130117, China

[4] School of Computer Science and Information Technology, Northeast Normal University, Changchun 130117, China

[5] IT & Educational Consultant, Ranchi, Jharkhand, India

1 Introduction

Knapsack problem (KP) is one of the well-known combinatorial optimization and NP-complete problems [1]. The 0–1 or binary KP has been the most important among the entire set of KP. The problem can be described as stated below:

Given a set of items with its own profit and weight and a knapsack, our goal is to find an item subset so as to maximize the profit without exceeding the knapsack capacity. KP is one of the most widely studied and experimented discrete programming problems. It offers many practical applications in diverse areas, such as project selection [2], resource distribution [3], investment decision making [4] and network interdiction problem [5].

Different methods have been proposed to solve the 0–1 KP since the groundbreaking work of Dantzig's [6]. At the early periods, deterministic optimization algorithms were mainly used to address this problem, such as branch and bound algorithm (B&B) [7], dynamic programming (DP) [8] as well as a hybrid of B&B and DP method [9]. The advancement of computational intelligence in general and development of advanced optimization methodologies in particular for addressing various complex situations, had

led to the endeavors for applying of metaheuristic algorithms in solving 0–1 KP, especially for the large-scale problems. These modern metaheuristic algorithms include genetic algorithm (GA) [10], differential evolution (DE) [11], shuffled frog-leaping algorithm (SFLA) [12], cuckoo search (CS) [13, 14], artificial bee colony (ABC) [15], harmony search (HS) [16, 17] and bat algorithm (BA) [18].

Although many methods have been successfully used in dealing with 0–1 KP, it is still significant in theoretical research and practical application. Hitherto, the existing algorithms had mainly focused on solving 0–1 KP with lower dimension. In comparison, studies, concentrating on the effects of high-dimensional sizes, are few and far between. Also, the subject of the correlation between the weight and the value of the items has not received adequate attention, so far. Since solving 0–1 KP efficiently can have a significant impact on scholarly research and real-world applications, it is worth addressing this problem with a novel algorithm.

In recent years, a variety of new nature-inspired metaheuristic algorithms have been reported, e.g., ant lion optimizer (ALO) [19], BA [20], biogeography-based optimization (BBO) [21, 22], charged system search (CSS) [23], CS [24–26], animal migration optimization (AMO) [27], krill herd (KH) [28–32], wolf search algorithm (WSA) [33], artificial plant optimization algorithm (APOA) [34], human learning optimization (HLO) [35], swarm search [36], earthworm optimization algorithm (EWA) [37], magnetic optimization algorithm (MOA) [38], monarch butterfly optimization (MBO) [39] and others. In fact, MBO is inspired by the migration behavior of the monarch butterflies in nature. The related investigations [39, 40] have opined that MBO is simple, effective and easy to implement. The original MBO was intended for addressing continuous optimization problems, i.e., it cannot be used directly in dealing with discrete optimization problems. This algorithm is relatively a new one, and endeavors toward solving the 0–1 KP with MBO have not yet been reported. Since 0–1 KP falls under the constrained discrete combinatorial optimization problems category, several factors need to be considered, such as individual encoding scheme and constraint-handling method. Hence, for solving the 0–1 KP effectively, it is imperative to specifically design a binary monarch butterfly optimization (BMBO). Toward that, in this paper, each individual is represented as two tuples with real-valued vector and binary vector, which is called hybrid encoding mechanism [13, 14]. Meanwhile, a novel and efficient repair operator, namely greedy optimization algorithm (GOA) [41], is employed. Furthermore, the original MBO [39] consists of two subpopulations—subpopulation1, which acts as migration operator, and subpopulation2, which is the butterfly adjusting operator. As against this, three different individual distribution methods are designed for the BMBO in order to test its effects on the performance. Finally, a large number of experiments and comparisons on 0–1 KP are carried out in order to demonstrate the effectiveness of the proposed BMBO.

The rest of the paper is organized as follows. Section 2 introduces the mathematical model of the 0–1 KP, while Sect. 3 provides a snapshot of the MBO algorithm. These are followed by Sect. 4 which presents the proposed BMBO for 0–1 KP. Section 5 deals with a series of simulation experiments as well as comparisons of the outcomes. Finally, the paper ends with Sect. 6 after sharing the conclusions and throwing some lights on scope of further research.

2 0–1 KP

The 0–1 KP [6] is a typical combination optimal problem with inequality constraints. Formally, we can describe the 0–1 KP as follows:

Given a set U of n items $u_1, u_2, \ldots, u_n$, we write $U = \{u_1, u_2, \ldots, u_n\}$, a weight w_i and a profit v_i for each item u_i, $1 \le i \le n$, and an integer C (the total capacity of the knapsack), the objective is to find a subset A of items that maximizes the total value $\sum_{u_i \in A} v_i$, subject to the condition that the total weight of all the items $\sum_{u_i \in A} w_i$ does not exceed C.

Generally speaking, the 0–1 KP can be defined as:

$$\begin{aligned} &\text{Maximize} && f(x) = \sum_{i=1}^{n} v_i x_i \\ &\text{subject to} && \sum_{i=1}^{n} w_i x_i \le C, \\ &x_i = 0 \text{ or } 1, && i = 1, \ldots, n, \end{aligned} \tag{1}$$

where the 0–1 variable x_i has the following meaning:

$$x_i = \begin{cases} 0, & \text{if } u_i \notin A, \\ 1, & \text{if } u_i \in A. \end{cases} \tag{2}$$

3 Monarch butterfly optimization

Monarch butterfly optimization (MBO) [39] is a new nature-inspired metaheuristic algorithm. It is inspired by the migration behavior of the monarch butterflies in nature. In MBO, the search direction of the monarch butterfly individuals mainly depends on two factors: the migration operator and butterfly adjusting operator. For clarity, we give a brief introduction to the foundations and evolutionary process of MBO.

3.1 Migration operator

We show a simple explanation about migration operator. Firstly, the entire population consists of two subpopulations, called subpopulation1 and subpopulation2, which lived in

Land1 and Land2, respectively. Secondly, NP, NP_1 and NP_2 denote the total number of individuals in the entire population, the number of individuals in subpopulation1 and the number of individuals in subpopulation2, respectively. Additionally, $NP_1 = p * NP$ and $NP_2 = NP - NP_1$, where p is the ratio of monarch butterflies in Land1. Thirdly, the parameter peri in MBO denotes migration period. Then, the migration operator can be described in Fig. 1.

From Fig. 1, it is clear that each individual in two subpopulations has a chance to generate offspring. Moreover, the probability can be adjusted by the parameter p. In addition, information, together with individual, is exchanged. Again, migration operator can be considered to be a unary search operation, identical to that of mutation in genetic algorithm. Thus, the main goal of the migration operator is to preserve the diversity by incorporating small, random changes into MBO.

3.2 Butterfly adjusting operator

Based on the careful analysis in the evolution principle of the migration operator, we can see that the offspring is generated by randomly selecting the individuals from the two subpopulations. The best monarch butterfly in Land1 and Land2 is not automatically passed on. Hence, in butterfly adjusting operator, some elements of newly produced offspring inherit directly from the global best individual to hold the beneficial information. Besides, Levy flights stemmed from Levy distribution are an efficient random walk pattern especially in exploring unknown, high-dimensional search space [42]. In MBO, Levy flight is introduced for its outstanding properties, such as self-similarity and fractal behavior. In the following, the butterfly adjusting operator can be shown in Fig. 2.

In Fig. 2, dx can be obtained by performing Levy flights [39] in which the step size is chosen as the follows.

$$\text{StepSize} = \text{ceil}(\text{exprnd}(2 * \text{Maxgen})) \quad (3)$$

where function exprnd(x) generates exponential random numbers with mean x and function ceil(x) returns a value which is the nearest integer greater than or equal to x. Maxgen is the maximum number of iteration.

The parameter ω can be regarded as the weighting factor which is inversely proportional to the current generation [39].

3.3 Main procedure of MBO algorithm

After migration operator and butterfly adjusting operator are designed, MBO method can be presented in Fig. 3.

Based on this structure of MBO, there exist four main steps in the evolutionary process. At first, the entire population is divided into two subpopulations according to fitness. Afterward, subpopulation1 is updated by migration operator. It is followed by subpopulation2 being updated by butterfly adjusting operator. Finally, subpopulation1 is combined with subpopulation2 into one population. This procedure is repeated until termination criterion is arrived at. We will elaborate steps 1 and 4 in Sect. 4.3.

4 A binary monarch butterfly optimization for 0–1 KP

In this section, we present the detailed design of the BMBO for solving 0–1 KP. Firstly, we employ the hybrid encoding scheme. Secondly, a novel and efficient repair operator GOA is used in BMBO. Thirdly, three kinds of individual distribution methods in two subpopulations are designed and verified. Fourthly, the basic sketch of BMBO for 0–1 KP is given. Lastly, the time complexity is analyzed.

Fig. 1 Migration operator

Migration operator

Begin
  for i=1 to NP_1 **do**
    for j=1 to d **do**
      r=$rand$*$peri$, where $rand$~U(0,1)
      if $r \le p$ **then**
        $x_{i,j} = x_{r1,j}$, where $r1$~U[1, 2,..., NP_1]
      else
        $x_{i,j} = x_{r2,j}$, where $r2$~U[1, 2,..., NP_2]
      endif
    end for
  end for
End.

Fig. 2 Butterfly adjusting operator

Butterfly adjusting operator

Begin
for i=1 to NP_2
for j=1 to d
if $rand \leq p$ **then** where $rand \sim U(0,1)$

$$x_{i,j} = x_{best,k}$$

else

$$x_{i,j} = x_{r3,j} \quad \text{where } r3 \sim U[1,2,..., NP_2]$$

if $rand > BAR$ **then**

$$x_{i,j} = x_{i,j} + \omega \times (dx_j - 0.5)$$

end if
end if
end for
end for
End.

Fig. 3 Monarch butterfly optimization algorithm

Monarch Butterfly Optimization algorithm

Begin
Step 1: Initialization. Initialize the population of NP monarch butterfly individuals randomly; set migration period $peri$, the migration ratio p, butterfly adjusting rate BAR and the max walk step of Levy flights S_{max}; set the maximum generation $MaxGen$.
Step 2: While (stopping criterion)
Sort by fitness of all monarch butterfly individuals in non-ascending order.
Construct subpopulation1 by better NP_1 individuals and construct subpopulation2 by the rest.
Generate new subpopulation1 by performing Algorithm 1.
Generate new subpopulation2 by performing Algorithm 2.
Recombine the newly-generated subpopulations into one whole population.
Find the current best.
end while
Step 3: Find the current best
End.

4.1 Encoding scheme

From the nature of the 0–1 KP, it would be natural to represent each individual by an n-bit binary string. Here, n is the dimension of the problem to be solved. Concretely speaking, n is item number. But the standard MBO operates in continuous space, or to be more specific, migration operator and butterfly adjusting operator work directly on n-dimensional real-valued vectors. In order to keep the excellent evolution mechanism of the basic MBO, hybrid encoding scheme [13, 14] is used in BMBO for solving 0–1 KP. As already mentioned, each monarch butterfly individual is represented by two tuples $\langle \mathbf{X}, \mathbf{Y} \rangle$ specifying the search space $\mathbf{X}$ and the solution space $\mathbf{Y}$, and the coding form is shown in Table 1.

It leads to a one-to-one, onto mapping g from $\mathbf{X}$ to $\mathbf{Y}$ such that

$$g(x) = \begin{cases} 1 & \text{if } \mathrm{sig}(x) \geq 0.5 \\ 0 & \text{else} \end{cases} \tag{4}$$

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

Table 1 Hybrid encoding scheme

Real-valued vector X	x_1	x_2	…	x_i	…	x_d
Binary vector Y	y_1	y_2	…	y_i	…	y_d

4.2 Repair operator

Usually, in solving 0–1 KP, some newly generated individuals may violate the capacity constraints given above. There are two approaches that are generally used to deal with the constraints: changing the objective function by penalty function method (PFM) [43, 44] and individual optimization method based on greedy strategy (IOM) [45, 46]. Though the penalty function method is feasible, it has been found to be inefficient while encountering large-scale problem. In addition, it is difficult to choose an efficient penalty function. Therefore, we choose IOM as repair operator. The basic idea of IOM is provided below:

For non-normal coding individual, the item with the greater density v_i/w_i and $x_i = 1$ is firstly put in the knapsack until the total weight does not exceed the knapsack capacity C. Meanwhile, let $x_i = 0$ for those items which cannot be packed into the knapsack. For its

Fig. 4 Greedy optimization algorithm

Greedy Optimization Algorithm

Begin

Input: $X = \{x_1, x_2, ..., x_n\} \in \{0,1\}^n$, $W = \{w_1, w_2, ..., w_n\}$, $V = \{v_1, v_2, ..., v_n\}$, $H[1...n]$, C.

Step 1: repair

weight=0; *value*=0; *temp*=0;

for *i*=1 to *n*

$weight = weight + x_i * w_i$

end for

if ($weight > C$)

for *i*=1 to *n*

$temp = temp + x_{H[i]} * w_{H[i]}$

if $temp > c$

$temp = temp - x_{H[i]} * w_{H[i]}$

$x_{H[i]} = 0$

end if

end for

weight=*temp*

end if

Step 2: optimize

for *i*=1 to *n*

if $x_{H[i]} = 0$ *and* $weight + w_{H[i]} \le C$

$x_{H[i]} = 1$

end if

end for

Step 3: compute

For *i*=1 to *n*

$value = value + x_i * v_i$

end for

Output: $X = \{x_1, x_2, ..., x_n\} \in \{0,1\}^n$, *value*

End.

simplicity and effectiveness, IOM has been widely applied to 0–1 KP.

In this paper, a novel and efficient greedy optimization algorithm (GOA) [41] is employed. It must be pointed out that all items are sorted in the non-increasing order by v_i/w_i and the index of items is stored in an array $H[1\ldots n]$ first. This means that $v_{H[1]}/w_{H[1]} \geq v_{H[2]}/w_{H[2]} \geq \cdots \geq v_{H[n]}/w_{H[n]}$. The pseudo-code of GOA can be shown in Fig. 4.

4.3 The performance of BMBO with different population strategies

In the basic MBO, the whole population is divided into subpopulation1 (in Land1) and subpopulation2 (in Land2). Moreover, from our observation, after each recombination, we see that the relatively better NP_1 individuals are assigned to the subpopulation1. Additionally, two subpopulations are combined after each generation, as already mentioned in Sect. 3.3. In order to estimate that the individual allocation strategy may have influence on the performance, we consider three kinds of individual allocation schemes in two subpopulations and compare their performances on some typical 0–1 KP instances. For convenience, we use three methods, viz. BMBO-1, BMBO-2 and BMBO-3, respectively. Let us now introduce these three methodologies below:

4.3.1 Three kinds of population strategies

1. BMBO-1: Here the entire population is randomly divided into subpopulation1 and subpopulation2. Additionally, during the entire evolution process, the newly generated subpopulations are not recombined. In principle, there are two populations which search for the optimal value by different search operations. Furthermore, little information exchange can happen. In theory, this method may not yield desired result and may have a worse performance.
2. BMBO-2: It differs from the BMBO-1. After initialization, the better NP_1 individuals are assigned to subpopulation1 and the rest of the whole population belongs to subpopulation2. Similarly, the newly generated subpopulations are also not recombined throughout the evolution.
3. BMBO-3: This population strategy is similar to that of the basic MBO. The difference is that the subpopulations are recombined at certain generations rather than with each generation. Obviously, this method can improve the evolution ability, and hence, the chance of finding the optimal value is enhanced to a large extent. Our previous experience suggests that this method should show better performance.

4.3.2 Comparison among three kinds of methods on 0–1 KP instances

In this subsection, an overall evaluation of the performance of BMBO-1, BMBO-2 and BMBO-3 is carried out. Firstly, ten standard 0–1 KP instances are used here (see Table 2, f_1–f_{10}). In Table 2, the column f^* denotes the theoretical optimum. More detailed information about these test problems can be obtained from [12, 16]. Since the optimum solutions for these low-dimensional test problems can be obtained easily, six medium-dimensional test instances (f_{11}–f_{16}) are considered subsequently, which are listed in Table 3. The column "Capacity" in this table denotes the knapsack capacity, and "Total Values" signifies the total values of all items. The generation method of 0–1 KP

Table 2 Standard ten low-dimensional 0–1 KP instances

f	Dim	Capacity	f^*
f_1	10	269	295
f_2	20	878	1024
f_3	4	20	35
f_4	4	11	23
f_5	15	375	481.0694
f_6	10	60	52
f_7	7	50	107
f_8	23	10,000	9767
f_9	5	80	130
f_{10}	20	879	1025

Table 3 Randomly generating eighteen 0–1 KP instances

f	Correlation	Dim	Capacity	Total values
f_{11}	Uncorrelated	150	6471	8111
f_{12}	Uncorrelated	200	8328	10,865
f_{13}	Weakly correlated	150	6403	8504
f_{14}	Weakly correlated	200	8358	11,051
f_{15}	Strongly correlated	150	5893	9357
f_{16}	Strongly correlated	200	8006	12,675
f_{17}	Uncorrelated	800	33,367	44,005
f_{18}	Uncorrelated	1000	41,948	54,764
f_{19}	Uncorrelated	1200	49,485	66,816
f_{20}	Uncorrelated	1500	62,015	83,572
f_{21}	Weakly correlated	800	33,367	44,491
f_{22}	Weakly correlated	1000	41,849	55,683
f_{23}	Weakly correlated	1200	49,808	56,811
f_{24}	Weakly correlated	1500	62,145	83,382
f_{25}	Strongly correlated	800	33,367	52,489
f_{26}	Strongly correlated	1000	40,883	64,510
f_{27}	Strongly correlated	1200	50,430	79,240
f_{28}	Strongly correlated	1500	62,142	97,856

instances in Table 3 is the same as shown in [14]. It should be noted that the weight and the value of the items are correlative, called uncorrelated instance, weakly correlated instance and strongly correlated instance, respectively. The last two kinds of instances are common in management.

The same parameters for three methods are set as follows: the migration ratio $p = 5/12$, migration period peri = 1.2, butterfly adjusting rate BAR = 5/12, max step $S_{\max} = 1.0$, population size NP = 50, maximum generation MaxGen = 50. Then again, in BMBO-3, the generation interval between recombination is set to 5.

We use C++ as the programming language and run the codes on a PC with Intel (R) Core(TM) i5-2415M CPU, 2.30 GHz, 4 GB RAM. Each instance is implemented for 50 times independently.

The comparison results of BMBO-1, BMBO-2 and BMBO-3 on ten standard test problems are presented in Table 4. The best solution, the worst solution, the mean, the median and the standard deviation (SD) for all the solutions are listed here, along with average minimal generation (AMG) and average minimal time (AMT) for getting the optimum results. The best value obtained by each method are in bold. From Table 4, it is clear that all three methods perform similarly for f_1, f_3–f_7 and f_9. For f_2, BMBO-1 has a slight advantage over the other two methods. For f_8, BMBO-3 shows a superior performance as compared to that of the other two methods. For f_{10}, the worst and the mean obtained by BMBO-2 are better than that of BMBO-1 and BMBO-3. In general, for standard ten low-dimensional test instances, all the three methods can reach the optimum solutions and show subtle performance differences.

The comparison results on six medium-dimensional test problems are recorded in Table 5. For the best solutions, Table 5 demonstrates that BMBO-3 can get the maximum total value on five functions. BMBO-1 and BMBO-2 can find the best solutions on three functions. By further observing the other four performance evaluation standards

Table 4 Performance comparison on standard ten low-dimensional test problems

Fun	Algorithm	Best	Worst	Mean	SD	ATG	ATT
f_1	BMBO-1	295	295	295	0.00	0	0.00
	BMBO-2	295	295	295	0.00	0	0.00
	BMBO-3	295	295	295	0.00	0	0.00
f_2	BMBO-1	1024	1024	1024.00	0.00	0	0.00
	BMBO-2	1024	1024	1024.00	0.00	0.44	0.0018
	BMBO-3	1024	1018	1023.76	1.17	0	0.00
f_3	BMBO-1	35	35	35	0.00	0	0.00
	BMBO-2	35	35	35	0.00	0	0.00
	BMBO-3	35	35	35	0.00	0	0.00
f_4	BMBO-1	23	23	23	0.00	0	0.00
	BMBO-2	23	23	23	0.00	0	0.00
	BMBO-3	23	23	23	0.00	0	0.00
f_5	BMBO-1	481.0694	481.0694	481.0694	0.00	0	0.00
	BMBO-2	481.0694	481.0694	481.0694	0.00	0	0.00
	BMBO-3	481.0694	481.0694	481.0694	0.00	0	0.00
f_6	BMBO-1	52	52	52	0.00	0	0.00
	BMBO-2	52	52	52	0.00	0	0.00
	BMBO-3	52	52	52	0.00	0	0.00
f_7	BMBO-1	107	107	107	0.00	0	0.00
	BMBO-2	107	107	107	0.00	0	0.00
	BMBO-3	107	107	107	0.00	0	0.00
f_8	BMBO-1	9767	9761	9763.56	2.19	4.24	0.0044
	BMBO-2	9767	9765	9766.12	0.99	2.12	0.0024
	BMBO-3	9767	9766	9766.56	0.49	2.88	0.0019
f_9	BMBO-1	130	130	130	0.00	0	0.00
	BMBO-2	130	130	130	0.00	0	0.00
	BMBO-3	130	130	130	0.00	0	0.00
f_{10}	BMBO-1	1025	1019	1023.56	0.00	0	0.00
	BMBO-2	1025	1025	1025.00	0.00	0	0.00
	BMBO-3	1025	1019	1021.28	2.91	0	0.00

Table 5 Performance comparison on six medium-dimensional test problems

Fun	Algorithm	Best	Worst	Mean	Median	SD
f_{11}	BMBO-1	7472	7395	**7442**	**7445**	**19.81**
	BMBO-2	**7474**	**7377**	7433	7434	25.46
	BMBO-3	**7474**	7350	**7442**	**7445**	23.55
f_{12}	BMBO-1	**9865**	**9764**	**9816**	**9819**	27.11
	BMBO-2	9862	9741	9809	9814	27.57
	BMBO-3	9851	9727	9811	9815	**23.56**
f_{13}	BMBO-1	6671	**6649**	**6662**	**6663**	**5.82**
	BMBO-2	6673	6640	6660	6662	6.82
	BMBO-3	**6676**	6644	6660	6659	7.17
f_{14}	BMBO-1	8726	8673	8699	8699	12.97
	BMBO-2	8723	8674	8699	8697	11.63
	BMBO-3	**8727**	**8682**	**8701**	**8701**	**11.37**
f_{15}	BMBO-1	**7173**	7163	7172	**7173**	2.21
	BMBO-2	**7173**	**7173**	**7173**	**7173**	**0.00**
	BMBO-3	**7173**	7163	**7173**	**7173**	1.98
f_{16}	BMBO-1	**9716**	**9706**	**9712**	**9716**	4.94
	BMBO-2	**9716**	**9706**	**9712**	**9716**	4.92
	BMBO-3	**9716**	**9706**	**9712**	**9716**	**4.79**
Total	BMBO-1	3	**3**	**4**	**5**	2
	BMBO-2	3	**3**	2	2	1
	BMBO-3	**5**	2	**4**	4	**3**

The best value obtained by each method are in bold

(columns 4–7), BMBO-1 and BMBO-3 exhibit almost identical performance and BMBO-2 performs the worst, as compared to the other two methods.

Taken together, it can be seen that the average performance of BMBO-3 is better than BMBO-1 and BMBO-2. The experimental results coincide with the theoretical analysis, as mentioned previously. Thus, we choose it to form the hybrid encoding MBO that will be described in detail in Sect. 4.4.

4.4 The procedure of BMBO for 0–1 KP

After the careful design as outlined above, it is natural to present the particular procedure of BMBO for 0–1 KP. The pseudo-code is described in Fig. 5. In addition to the process of initialization, essentially, it consists of three main processes. In migration process, the new monarch butterfly individuals of subpopulation1 are generated. This process can be viewed as exploitation around the local neighbors of subpopulation1 or subpopulation2. Moreover, we can say that the individuals of subpopulation1 can be influenced by the individuals of subpopulation2. In butterfly adjusting process, different information, including the best monarch butterfly in two lands and the individual selected randomly from subpopulation2, is used to influence generation of new individuals of subpopulation2. In addition, Levy flights are employed for its superior traits, such as longer step length in exploring the search space, which can accelerate convergence rate. In population recombination process, information, together with individual, can be exchanged. Thus, as a critical issue in the field of meta-heuristics, the trade-off between the exploitation and exploration, also called attraction and diffusion [51], is both considered in BMBO. That is why the probability of obtaining truly optimal solution is sufficiently enhanced.

4.5 Algorithm complexity

In this subsection, the time complexity of BMBO for 0–1 KP (Fig. 5) is to be estimated. From Algorithm 5, it can be seen that the computing time is mainly determined by step 1–3. In step 1, the time complexity of Quicksort algorithm is $O(n \log n)$. In step 2, the initialization of NP monarch butterfly individuals has time complexity $O(\mathrm{NP} \times n) = O(n^2)$. In step 3, the migration operator costs time $O(\mathrm{NP}_1 \times n) = O(n^2)$, the butterfly adjusting operator costs time $O(\mathrm{NP}_2 \times n) = O(n^2)$, and the greedy optimization algorithm costs time $O(n)$. Thus, BMBO for 0–1 KP runs in time complexity $O(n \log n) + O(n^2) + O(n^2) + O(n^2) + O(n) = O(n^2)$.

5 Simulation experiments

In this section, in order to evaluate the comprehensive performance of BMBO, twelve high-dimensional 0–1 KP test instances are adopted in which the correlation between the weight and the value of the items is considered, as previously mentioned. The following subsections will provide an overview of the related comparison of algorithms at first, followed by comparison with experimental results along with the experiment analysis.

5.1 An overview of the related comparison algorithms

In order to test its overall optimization performance, comparison of the BMBO with four optimization methods is carried out, which are ABC [47], CS [48], DE [49] and GA [50]. It should be noted that ABC, CS and DE all employ hybrid encoding mechanism, whereas ABC, CS, DE and GA all use greedy optimization algorithm (GOA) as repair operator. For consistency in expressions, we use BABC, BCS and BDE to represent binary-coding algorithms, respectively.

In our experiments, the parameter settings are listed in Table 6. It should be noted that the generation interval between recombination is set to 50 in BMBO. In addition,

Fig. 5 Binary monarch butterfly optimization algorithm for 0–1 KP

Monarch Butterfly Optimization algorithm for 0-1 KP

Begin

Step 1: Sorting. Perform *QuickSort* algorithm to sort all items in the non-increasing order by v_i/w_i, $1 \le i \le n$ and the index of items is stored in array $\boldsymbol{H}[1...n]$.

Step 2: Initialization. Set the generation counter g=1; set migration period *peri*, the migration ratio p, butterfly adjusting rate *BAR* and the max walk step of Lévy flights S_{max}; set the maximum generation *MaxGen*.Set the generation interval between recombination *RG*. Generate *NP* monarch butterfly individuals randomly $\{<\mathbf{X_1}, \mathbf{Y_1}>, <\mathbf{X_2}, \mathbf{Y_2}>, \ldots, <\mathbf{X_{NP}}, \mathbf{Y_{NP}}>\}$. Calculate the fitness for each individual, $f(\mathbf{Y}_i)$, $1 \le i \le NP$.

Step 3: While(stopping criterion)

Divide the whole population into subpopulation1 (NP_1 individuals) and subpopulation2 (NP_2 individuals) according to their fitness;

Determine the global optimal individual $<\mathbf{X_{best}}, \mathbf{Y_{best}}>$.

Generate new subpopulation1 by performing Algorithm 1;

Generate new subpopulation2 by performing Algorithm 2;

Repair and optimize the individuals by performing GOA.

Keep best solutions.

Find the current best (Y_{best}, $f(Y_{best})$).

g=g+1.

Step 4: Recombine.

if g=*RG*

Recombine the newly-generated subpopulations into one whole population.

end if

Step 5: end while

Step 6: Output the best results.

End.

Table 6 The parameter settings of five algorithms on 0–1 KP

Algorithm	Parameter	Value
BABC	Population size	50
	Number of food sources	25
	Maximum search times	100
BCS	Population size	40
	P_a	0.25
BDE	Population size	50
	Crossover probability	0.9
	Amplification factor	0.3
GA	Population size	50
	Crossover probability	0.6
	Mutation probability	0.001
BMBO	Migration ratio	5/12
	Migration period	1.2
	Butterfly adjusting rate	5/12
	Max step	1.0

the maximum computation time is used as termination criterion which is set to 10 s for 1500-dimensional instances and 8 s for others, respectively.

5.2 The experimental results

We verify the performance of the BMBO on four uncorrelated instances, four weakly correlated instances and four strongly correlated instances. The experimental results obtained by five algorithms from 50 runs are recorded in Tables 7, 8 and 9, respectively.

Table 7 presents the experimental results on four uncorrelated instances from 50 runs and which clearly points out the best comprehensive performance of the BMBO. For the best solution, BCS can find the optimal solutions on three out of four uncorrelated instances and the BMBO gets one. For the worst values, the mean values and the median values, BMBO clearly demonstrates absolute advantage over the four other methods. With regard to the algorithm

Table 7 Experimental results of five algorithms on uncorrelated KP instances

Fun	Algorithm	Best	Worst	Mean	Median	SD
f_{17}	BABC	39,816	39,542	39,639	39,641	**55.50**
	BCS	**40,445**	39,411	39,602	39,527	218.11
	BDE	39,486	39,154	39,323	39,326	80.60
	GA	39,190	38,274	38,838	38,867	196.70
	BMBO	40,232	**39,765**	**40,035**	**40,036**	105.83
f_{18}	BABC	49,374	49,150	49,256	49,256	**58.56**
	BCS	**50,104**	49,056	49,211	49,154	205.08
	BDE	49,303	48,696	48,945	48,940	111.08
	GA	48,955	47,809	48,384	48,416	256.69
	BMBO	50,024	**49,336**	**49,699**	**49,689**	135.33
f_{19}	BABC	60,222	59,867	60,059	60,046	**82.28**
	BCS	60,490	59,764	59,938	59,926	120.76
	BDE	59,921	59,435	59,645	59,643	114.17
	GA	59,578	58,106	58,996	59,061	362.53
	BMBO	**61,109**	**60,214**	**60,677**	**60,660**	165.83
f_{20}	BABC	74,959	74,571	74,742	74,723	**90.07**
	BCS	**75,828**	74,472	74,666	74,583	245.28
	BDE	74,671	74,077	74,319	74,310	113.92
	GA	74,372	72,477	73,584	73,615	414.01
	BMBO	75,761	**75,062**	**75,464**	**75,482**	193.25
Total	BABC	0	0	0	0	**4**
	BCS	**3**	0	0	0	0
	BDE	0	0	0	0	0
	GA	0	0	0	0	0
	BMBO	1	**4**	**4**	**4**	0

The best value obtained by each method are in bold

Table 8 Experimental results of five algorithms on weakly correlated KP instances

Fun	Algorithm	Best	Worst	Mean	Median	SD
f_{21}	BABC	34,706	34,650	34,675	34,672	**16.00**
	BCS	**34,975**	34,621	34,676	34,663	65.25
	BDE	34,629	34,549	34,588	34,586	20.93
	GA	34,585	34,361	34,476	34,481	60.91
	BMBO	34,860	**34,681**	**34,786**	**34,784**	35.01
f_{22}	BABC	43,321	43,243	43,275	43,271	**18.74**
	BCS	**43,708**	43,215	43,326	43,266	143.50
	BDE	43,251	43,140	43,187	43,181	23.94
	GA	43,172	42,901	43,049	43,067	74.36
	BMBO	43,491	**43,359**	**43,412**	**43,413**	31.36
f_{23}	BABC	52,061	51,711	51,876	51,873	**79.72**
	BCS	**52,848**	51,617	51,838	51,768	260.46
	BDE	51,900	51,289	51,547	51,538	123.67
	GA	51,460	50,112	50,945	50,991	281.41
	BMBO	52,774	**52,110**	**52,425**	**52,390**	158.19
f_{24}	BABC	64,864	64,752	64,806	64,802	**25.45**
	BCS	**65,549**	64,749	64,932	64,844	245.03
	BDE	64,770	64,620	64,692	64,689	35.66
	GA	64,769	64,315	64,535	64,543	85.75
	BMBO	65,123	**64,916**	**65,022**	**65,012**	56.38
Total	BABC	0	0	0	0	**4**
	BCS	**4**	0	0	0	0
	BDE	0	0	0	0	0
	GA	0	0	0	0	0
	BMBO	0	**4**	**4**	**4**	0

The best value obtained by each method are in bold

stability, BABC ranks the first and the "SD" of BABC is much smaller than that of the other four methods. Unfortunately, BDE and GA failed to reach the optimum and other performances are not prominent either.

The experimental results on four weakly correlated instances from 50 runs are listed in Table 8. For the best solution, BCS still ranks no. 1. For the worst value, the mean value and the median value, BMBO still ranks 1. According to the last column, BABC gets the best ranking. It can be seen that the rank of five methods in Tables 7 and 8 is identical.

Table 9 presents the experimental results on four strongly correlated instances from 50 runs. It is clear that the BMBO still maintains the absolute best position. Compared with the previous two kinds of cases, BMBO can obtain all the optimal solutions. Additionally, for the worst values, BABC shows the best performance rather than BMBO.

Based on the above analyses, we get the conclusion that BMBO is superior to, or at last competitive with the other four algorithms. BCS has the advantage of high precision, whereas BABC has good stability. However, in three different occasions, BDE and GA have the worst performance when compared with the other three methods.

To clearly demonstrate the power of BMBO, six of the most representative instances, including f_{19}, f_{20}, f_{23}, f_{24}, f_{27} and f_{28}, are chosen and then the comparisons of average best profits and convergence curves with 50 runs are illustrated in this work. They are demonstrated in Figs. 6, 7, 8, 9, 10 and 11, in Figs. 12, 13, 14, 15, 16 and 17, respectively.

Figure 6 shows the values obtained by five methods on 1200-dimensional uncorrelated KP instance in 50 runs. Figure 6 depicts that all the profits obtained by BMBO in 50 runs are clearly superior to the other methods. Here we should note that there exists only one best value obtained by BCS, which is bigger than the worst value and near-worst values obtained by BMBO. In addition, BABC fluctuates around certain values, as is expected. This again shows that BABC has good numerical stability, as well as consistent with the previous analysis. BDE has an ordinary performance for this case. For GA, the results clearly point out that the solution precision and stability are unsatisfactory.

Table 9 Experimental results of five algorithms on strongly correlated KP instances

Fun	Algorithm	Best	Worst	Mean	Median	SD
f_{25}	BABC	40,127	**40,107**	40,116	**40,117**	**4.52**
	BCS	40,127	40,096	40,108	40,107	6.59
	BDE	**40,137**	40,087	**40,119**	**40,117**	10.19
	GA	40,069	39,930	40,023	40,025	31.12
	BMBO	**40,137**	40,087	**40,119**	**40,117**	10.19
f_{26}	BABC	49,390	**49,363**	49,376	49,373	**5.61**
	BCS	**49,393**	49,353	49,364	49,363	6.80
	BDE	49,363	49,323	49,340	49,339	8.31
	GA	49,333	49,231	49,287	49,294	29.76
	BMBO	**49,393**	49,353	**49,378**	**49,382**	10.12
f_{27}	BABC	60,567	**60,540**	60,554	60,554	5.54
	BCS	60,559	60,533	60,543	60,541	**5.39**
	BDE	60,545	60,498	60,518	60,517	10.16
	GA	60,520	60,391	60,451	60,449	29.87
	BMBO	**60,588**	60,530	**60,562**	**60,560**	11.98
f_{28}	BABC	74,822	**74,792**	74,805	74,802	**6.85**
	BCS	74,837	74,779	74,794	74,792	9.19
	BDE	74,778	74,737	74,759	74,759	10.23
	GA	74,766	74,606	74,689	74,694	37.20
	BMBO	**74,842**	74,772	**74,818**	**74,821**	15.80
Total	BABC	0	**4**	0	1	**3**
	BCS	1	0	0	0	1
	BDE	1	0	1	1	0
	GA	0	0	0	0	0
	BMBO	**4**	0	**4**	**4**	0

The best value obtained by each method are in bold

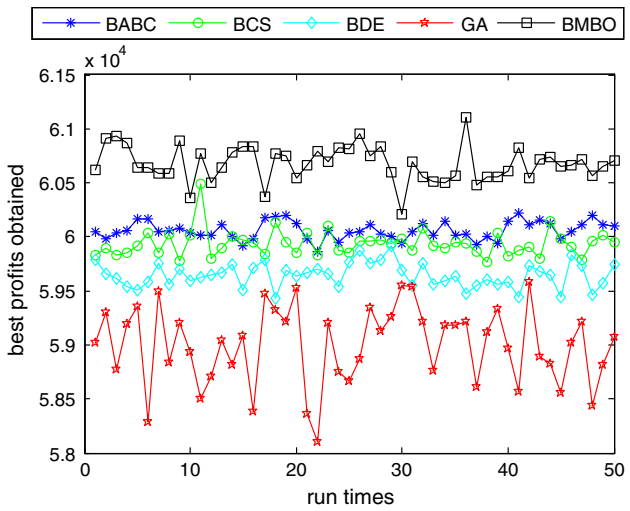

Fig. 6 The comparisons of average best profits on f_{19} in 50 runs

Figure 7 demonstrates the values obtained by five methods on 1500-dimensional uncorrelated KP instance in 50 runs. We can easily see that the whole black curve of

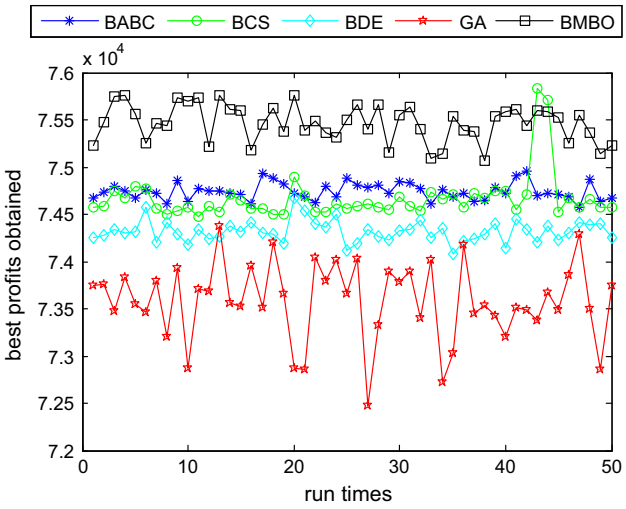

Fig. 7 The comparisons of average best profits on f_{20} in 50 runs

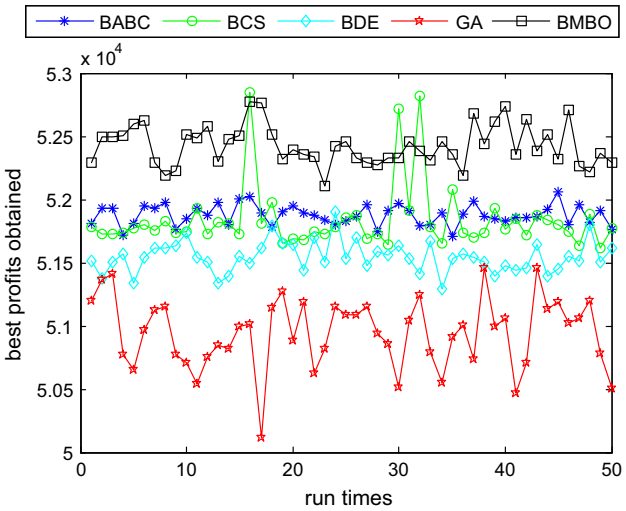

Fig. 8 The comparisons of average best profits on f_{23} in 50 runs

BMBO is in the top-most position in Fig. 7 which directly reflects that BMBO greatly outperforms all the other methods. Obviously, two values obtained by BCS are bigger than the best value obtained by BMBO. In summary, the curve trends in Fig. 6 and in Fig. 7 are rather similar and the main difference revolves around which method, BMBO or BCS, can obtain the best value.

Figure 8 illustrates the values obtained by five methods on 1200-dimensional weakly correlated KP instance in 50 runs. When taking a glance at this figure, three values obtained by BCS are most prominent and two values are bigger than the best value obtained by BMBO. Compared with Figs. 6 and 7, five methods have little change in performance when solving uncorrelated KP instances and weakly correlated KP instances.

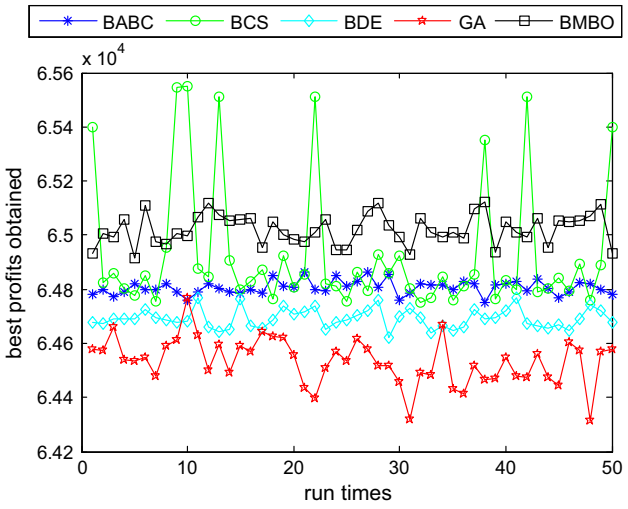

Fig. 9 The comparisons of average best profits on f_{24} in 50 runs

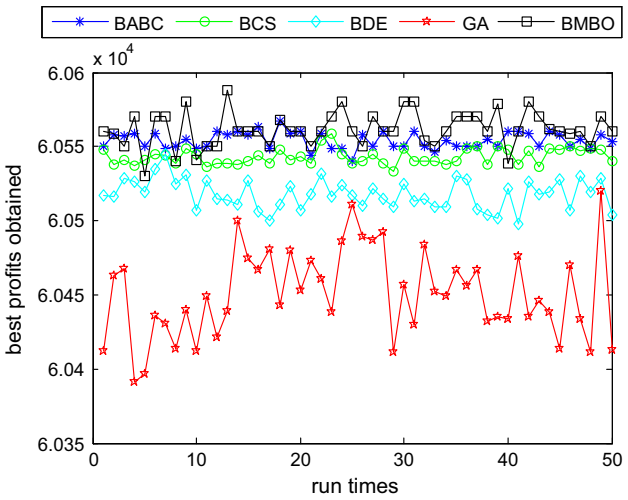

Fig. 10 The comparisons of average best profits on f_{27} in 50 runs

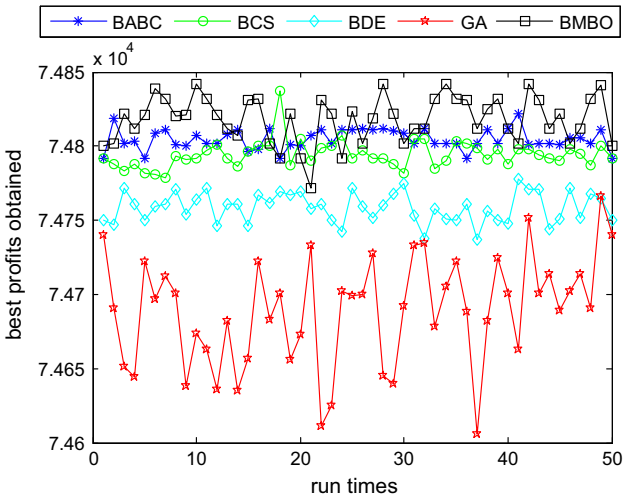

Fig. 11 The comparisons of average best profits on f_{28} in 50 runs

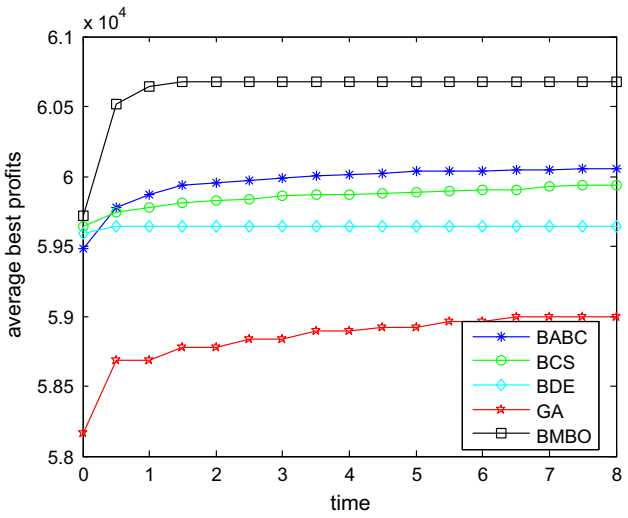

Fig. 12 The convergence graph of five methods on f_{19} in 8 s

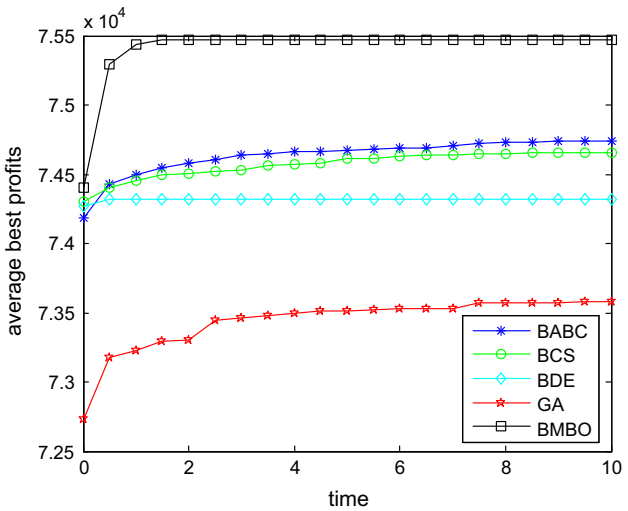

Fig. 13 The convergence graph of five methods on f_{20} in 10 s

Figure 9 displays the values obtained by five methods on 1500-dimensional weakly correlated KP instance in 50 runs. At first glance, the best values found by BCS are bigger than BMBO eight times and the other values are smaller than BMBO. Regrettably, although weakly correlated KP is closer to the real-world situations, BMBO was not able to overtake BCS in terms of the optimal values.

Taking consideration of the results as shown in Figs. 6, 7, 8 and 9, an important observation is that the best profits obtained by BCS can exceed BMBO occasionally.

Figure 10 reveals the values obtained by five methods on 1200-dimensional strongly correlated KP instance in 50 runs. Different from the previous two kinds of cases, it can be seen that most of the values obtained by BMBO are better than almost all values found by the other four

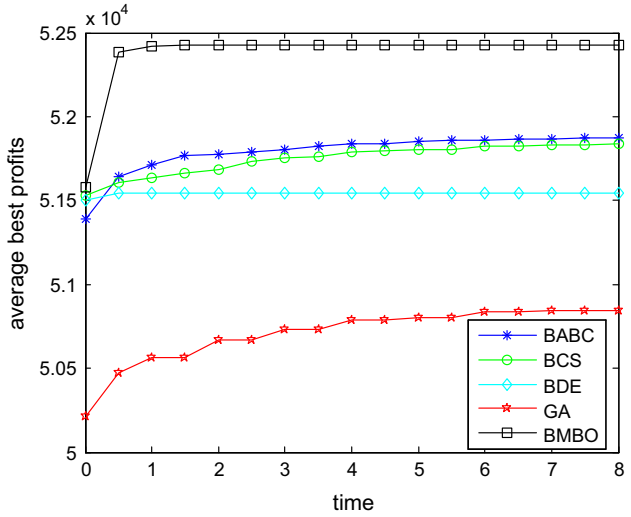

Fig. 14 The convergence graph of five methods on f_{23} in 8 s

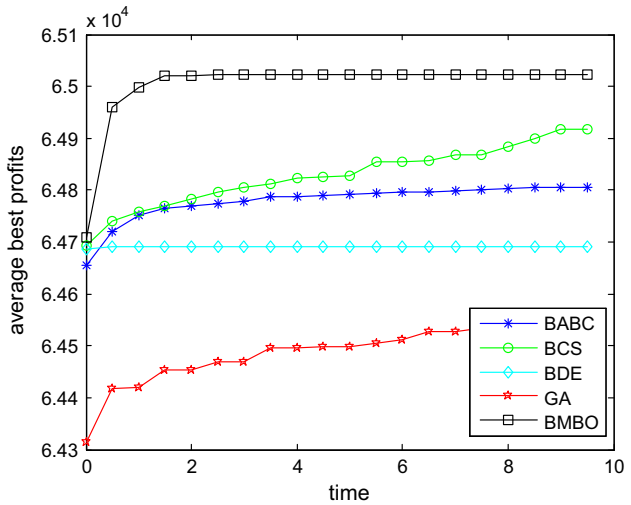

Fig. 15 The convergence graph of five methods on f_{24} in 10 s

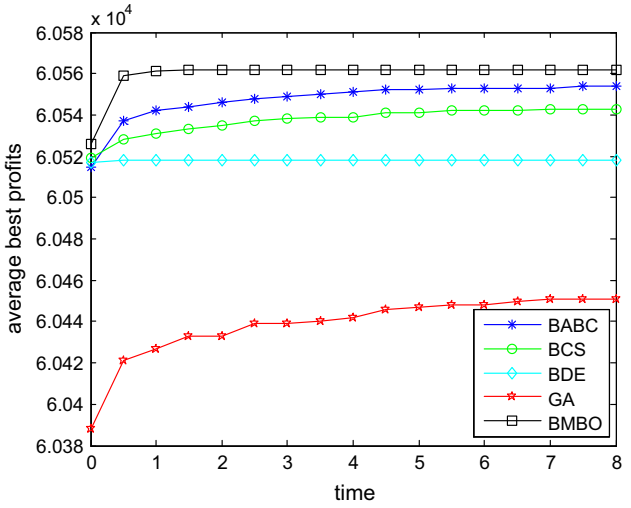

Fig. 16 The convergence graph of five methods on f_{27} in 8 s

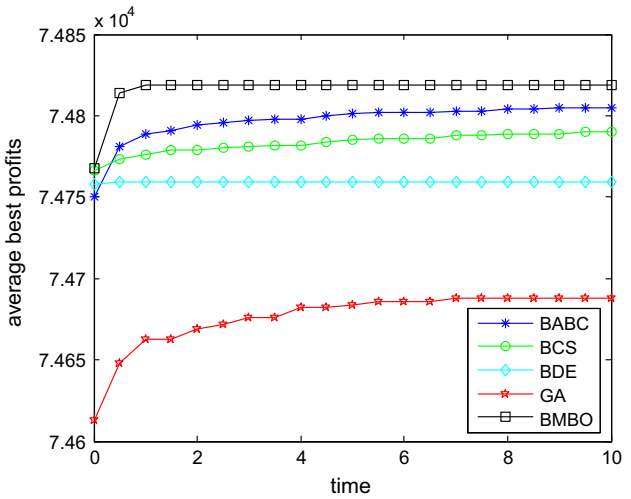

Fig. 17 The convergence graph of five methods on f_{28} in 10 s

methods. In other words, BMBO clearly outperforms the other four methods. Moreover, the curves of BABC, BCS and BMBO partially overlap and interweave, implying that BABC, BCS and BMBO can compete when solving strongly correlated 0–1 KP instances.

Figure 11 presents the values obtained by five methods on 1500-dimensional strongly correlated KP instance in 50 runs. For this case, similar to the f_{28} in Fig. 10, BMBO outperforms the other four methods.

Based on the above analysis of Figs. 6, 7, 8, 9, 10 and 11, some observations are summarized below:

(1) BMBO outperforms the other four methods in performance on average. (2) Although BCS can reach the optimum in some test problems, it also bears an important weakness of poor stabilization. (3) The curves of BABC change gently, indicating that good stabilization is a trait for BABC. (4) The performance of BDE is very poor for the accuracy of solutions. (5) GA shows the worst performance not only in accuracy of the results but also in terms of the stabilization.

Figures 12, 13, 14, 15, 16 and 17 show that the convergence trajectories of five methods while solving six high-dimensional 0–1 KP instances. As depicted in Figs. 12, 13, 14, 15, 16 and 17, at the beginning of evolution, BMBO performs well. Next, BMBO converges rapidly and the convergent traces of BMBO are above the other four methods. In addition, for the other four methods, it is obvious that BCS, BDE and BABC have similar initial function values, which is better than GA, though worse than MBO. However, this situation does not maintain during evolution process. BABC has better performance,

and BCS takes the third place. Strangely, BDE gets trapped into the local optimum and can be viewed as the fourth best performance. Without any ambiguity, the performance of GA can be visualized as the worst.

From the above analysis of Figs. 12, 13, 14, 15, 16 and 17, we conclude that BMBO is superior to BABC, BCS, BDE and GA in terms of the accuracy of the solution and convergence speed.

Based on the analysis of Tables 6, 7 and 8 and Figs. 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16 and 17, BMBO performs better optimization ability as compared to the other four methods. In addition, BMBO has better stability than BCS and GA.

6 Conclusions

This paper reports the first ever applications of monarch butterfly optimization algorithm to the 0–1 KP. In order to deal with the binary-coded optimization problems efficiently and effectively, a hybrid encoding monarch butterfly optimization was proposed. Moreover, the original MBO is divided into two subpopulations. To ensure the best performance, three kinds of individual allocation strategies were investigated. Besides, an efficient repair operator, called greedy optimization algorithm, was used, which can not only revise the infeasible solutions, but also optimize the feasible ones. Finally, BMBO was verified and compared with BABC, BCS, BDE and GA on three types of 0–1 KP instances. The experimental results show that BMBO outperforms the other four methods in terms of the search accuracy, convergent speed and numerical stability. It certainly gives credence to the fact that the binary-coded MBO is an effective optimization method.

Though MBO has displayed huge advantage in solving 0–1 KP, there are some points to explain which can aid the future research and direction of this algorithm.

Firstly, it is vitally important to choose effective parameters which are always independent to specific problems for a metaheuristic method. To inquire into the influence of these key parameters on BMBO, we may be applied Taguchi method [52] to design the experiment.

Secondly, in this work, three types of large-scale 0–1 KP instances are randomly generated in which the values of weight and profit have different relevance, respectively. In future, other complex KP, such as multidimensional KP, quadratic KP, knapsack sharing problem and randomized time-varying knapsack problems (RTVKP) [53], should be addressed to demonstrate more effectiveness of BMBO.

Thirdly, due to the fact that BMBO is simple and easy to use in numerical function optimization, it can be extended to other combinatorial optimization problems, such as job scheduling problem and set covering problems.

Fourthly, the main advantage of BMBO, as compared to the other four optimization methods, is that it has higher convergence speed and better efficiency and precision. However, efforts should be initiated toward its improvement in other performance, such as stabilization. Toward that, different strategies, such as hybridization with other popular methods, can be introduced.

Fifthly, some researchers try to combine two algorithms for complementing each other, such as the hybridization of DE with particle swarm optimization (PSO) [54]. In future research work, the hybrid algorithm based on MBO and other methods can be designed.

At last, greedy optimization algorithm (GOA) and greedy transform method (GTM) [13]—both are novel and effective repair operators to handle the infeasible solutions. However, which of the two approaches is better has not been examined through comparative experiments. We plan to address this in our successive research.

Acknowledgments This work was supported by National Natural Science Foundation of China (Nos. 61272297, 61402207, 61503165), Jiangsu Province Science Foundation for Youths (No. BK20150239) and R&D Program for Science and Technology of Shijiazhuang (No. 155790215).

References

1. Du DZ, Ko KI, Hu X (2011) Design and analysis of approximation algorithms. Springer, Berlin
2. Mavrotas G, Diakoulaki D, Kourentzis A (2008) Selection among ranked projects under segmentation, policy and logical constraints. Eur J Oper Res 187(1):177–192. doi:10.1016/j.ejor.2007.03.010
3. Vanderster DC, Dimopoulos NJ, Parra-Hernandez R et al (2009) Resource allocation on computational grids using a utility model and the knapsack problem. Future Gener Comput Syst 25(1):35–50. doi:10.1016/j.future.2008.07.006
4. PeetaS Salman FS, Gunnec D et al (2010) Pre-disaster investment decisions for strengthening a highway network. Comput Oper Res 37(10):1708–1719. doi:10.1016/j.cor.2009.12.006
5. Yates J, Lakshmanan K (2011) A constrained binary knapsack approximation for shortest path network interdiction. Comput Ind Eng 61(4):981–992. doi:10.1016/j.cie.2011.06.011
6. Dantzig GB (1957) Discrete-variable extremum problems. Oper Res 5(2):266–288
7. Shih W (1979) A branch and bound method for the multi-constraint zero-one knapsack problem. J Oper Res Soc. doi:10.2307/3009639
8. Toth P (1980) Dynamic programing algorithms for the zero-one knapsack problem. Computing 25(1):29–45. doi:10.1007/BF02243880
9. Plateau G, Elkihel M (1985) A hybrid method for the 0–1 knapsack problem. Methods Oper Res 49:277–293
10. Thiel J, Voss S (1994) Some experiences on solving multi constraint zero-one knapsack problems with genetic algorithms. INFOR 32(4):226–242
11. Chen P, Li J, Liu ZM (2008) Solving 0–1 knapsack problems by a discrete binary version of differential evolution. In: Second international symposium on intelligent information technology application, vol 2, pp 513–516. doi:10.1109/IITA.2008.538

12. Bhattacharjee KK, Sarmah SP (2014) Shuffled frog leaping algorithm and its application to 0/1 knapsack problem. Appl Soft Comput 19:252–263. doi:10.1016/j.asoc.2014.02.010
13. Feng YH, Jia K, and He YC (2014) An improved hybrid encoding cuckoo search algorithm for 0–1 knapsack problems. Comput Intell Neurosci 2014:970456. doi:10.1155/2014/970456
14. Feng YH, Wang GG, Feng QJ, Zhao XJ (2014) An effective hybrid cuckoo search algorithm with improved shuffled frog leaping algorithm for 0–1 knapsack problems. Comput Intell Neurosci. doi:10.1155/2014/857254
15. Kashan MH, Nahavandi N, Kashan AH (2012) DisABC: a new artificial bee colony algorithm for binary optimization. Appl Soft Comput 12(1):342–352. doi:10.1016/j.asoc.2011.08.038
16. Zou D, Gao L, Li S, Wu J (2011) Solving 0–1 knapsack problem by a novel global harmony search algorithm. Appl Soft Comput 11(2):1556–1564. doi:10.1016/j.asoc.2010.07.019
17. Kong X, Gao L, Ouyang H, Li S (2015) A simplified binary harmony search algorithm for large scale 0–1 knapsack problems. Expert Syst Appl 42(12):5337–5355. doi:10.1016/j.eswa.2015.02.015
18. Zhou Y, Li L, Ma M (2015) A complex-valued encoding bat algorithm for solving 0–1 knapsack problem. Neural Process Lett. doi:10.1007/s11063-015-9465-y
19. Mirjalili S (2015) The ant lion optimizer. Adv Eng Softw 83:80–98
20. Yang XS, Deb S, Fong S (2014) Bat algorithm is better than intermittent search strategy. J Multi-Valued Log Soft Comput 22(3):223–237
21. Simon D (2008) Biogeography-based optimization. IEEE Trans Evol Comput 12(6):702–713. doi:10.1109/TEVC.2008.919004
22. Li X, Wang J, Zhou J, Yin M (2011) A perturb biogeography based optimization with mutation for global numerical optimization. Appl Math Comput 218(2):598–609. doi:10.1016/j.amc.2011.05.110
23. Kaveh A, Talatahari S (2010) A novel heuristic optimization method: charged system search. Acta Mech 213(3–4):267–289. doi:10.1007/s00707-009-0270-4
24. Srivastava PR, Chis M, Deb S et al (2012) An efficient optimization algorithm for structural software testing. Int J Artif Intell^{TM} 8(S12):68–77
25. Wang G-G, Guo LH, Duan H et al (2012) A hybrid meta-heuristic DE/CS algorithm for UCAV path planning. J Inform Comput Sci 9(16):4811–4818
26. Wang G-G, Gandomi AH, Zhao XJ et al (2014) Hybridizing harmony search algorithm with cuckoo search for global numerical optimization. Soft Comput. doi:10.1007/s00500-014-1502-7
27. Li X, Zhang J, Yin M (2014) Animal migration optimization: an optimization algorithm inspired by animal migration behavior. Neural Comput Appl 24(7–8):1867–1877. doi:10.1007/s00521-013-1433-8
28. Wang G-G, Guo LH, Wang HQ et al (2014) Incorporating mutation scheme into krill herd algorithm for global numerical optimization. Neural Comput Appl 24(3–4):853–871. doi:10.1007/s00521-012-1304-8
29. Wang GG, Gandomi AH, Yang XS, et al (2012) A new hybrid method based on krill herd and cuckoo search for global optimization tasks. Int J Bio-Inspired Comput
30. Wang G-G, Gandomi AH, Alavi AH et al (2015) A hybrid method based on krill herd and quantum-behaved particle swarm optimization. Neural Comput Appl. doi:10.1007/s00521-015-1914-z
31. Wang G-G, Guo LH, Gandomi AH et al (2014) Chaotic krill herd algorithm. Inf Sci 274:17–34
32. Wang G-G, Gandomi AH, Alavi AH (2014) Stud krill herd algorithm. Neurocomputing 128:363–370
33. Fong S, Deb S, Yang XS (2015) A heuristic optimization method inspired by wolf preying behavior. Neural Comput Appl. doi:10.1007/s00521-015-1836-9
34. Cui Z, Fan S, Zeng J et al (2013) Artificial plant optimization algorithm with three-period photosynthesis. Int J Bio-Inspired Comput 5(2):133–139. doi:10.1504/IJBIC.2013.053507
35. Wang L, Yang R, Ni H et al (2015) A human learning optimization algorithm and its application to multi-dimensional knapsack problems. Appl Soft Comput 34:736–743. doi:10.1016/j.asoc.2015.06.004
36. Fong S, Yang XS, Deb S (2013) Swarm search for feature selection in classification. In: Computational science and engineering (CSE), 2013 IEEE 16th international conference on. IEEE, pp 902–909
37. Wang G-G, Deb S, Coelho LDS (2015) Earthworm optimization algorithm: a bio-inspired metaheuristic algorithm for global optimization problems. Int J Bio-Inspired Comput in press
38. Mirjalili SA, Hashim SZM (2011). BMOA: binary magnetic optimization algorithm. In: 2011 3rd international conference on machine learning and computing (ICMLC 2011), Singapore, pp 201–206
39. Wang G-G, Deb S, Cui Z (2015) Monarch butterfly optimization. Neural Comput Appl. doi:10.1007/s00521-015-1923-y
40. Wang G-G, Zhao XC, Deb S (2015). A novel monarch butterfly optimization with greedy strategy and self-adaptive crossover operator. In: the 2015 2nd international conference on soft computing & machine intelligence (ISCMI 2015), Hong Kong. IEEE
41. He Y, Zhang X, Li W et al (2014) Algorithms for randomized time-varying knapsack problems. J Comb Optim. doi:10.1007/s10878-014-9717-1
42. Yang XS (2010) Nature-inspired metaheuristic algorithms. Luniver Press, Frome
43. Joines JA, Houck CR (1994) On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with GA's. In: Evolutionary computation, 1994. IEEE World Congress on computational intelligence. Proceedings of the first IEEE conference on. IEEE, pp 579–584. doi:10.1109/ICEC.1994.349995
44. Olsen AL (1994) Penalty functions and the knapsack problem. In: Evolutionary computation, 1994. IEEE World congress on computational intelligence. Proceedings of the first IEEE conference on. IEEE, pp 554–558. doi:10.1109/ICEC.1994.350000
45. Goldberg DE (1989) Genetic algorithms in search, optimization, and machine learning. Addison-Wesley, Boston
46. Simon D (2013) Evolutionary optimization algorithms. Wiley, New York
47. Karaboga D, Basturk B (2007) A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. J Glob Optim 39(3):459–471. doi:10.1007/s10898-007-9149-x
48. Yang XS, Deb S (2009) Cuckoo search via Levy flights. In: Nature & biologically inspired computing, 2009. NaBIC 2009. World Congress on. IEEE, pp 210–214. doi:10.1109/NABIC.2009.5393690
49. Storn R, Price K (1997) Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. J Glob Optim 11(4):341–359. doi:10.1023/A:1008202821328
50. Michalewicz Z (1996) Genetic algorithms + data structures = evolution programs. Springer, Berlin
51. Yang XS, Deb S, Hanne T, He XS (2015) Attraction and diffusion in nature-inspired optimization algorithms. Neural Comput Appl. doi:10.1007/s00521-015-1925-9

52. Montgomery DC (2005) Design and analysis of experiments. Wiley, Arizona
53. Feng YH, Wang G-G (2015) An Improved hybrid encoding firefly algorithm for randomized time-varying knapsack problems. In: The 2015 2nd international conference on soft computing & machine intelligence (ISCMI 2015), Hong Kong. IEEE
54. Wang G-G, Hossein Gandomi A, Yang XS et al (2014) A novel improved accelerated particle swarm optimization algorithm for global numerical optimization. Eng Comput 31(7):1198–1220