A Binary Particle Swarm Optimization for Solving the Bounded Knapsack Problem

Ya Li, Yichao He(✉), Huanzhe Li, Xiaohu Guo, and Zewen Li

College of Information and Engineering, Hebei GEO University,
Shijiazhuang 050031, China
921045813@qq.com, heyichao@hgu.edu.cn

Abstract. Bounded knapsack problem (BKP) is a classical knapsack problem. At present, methods for solving the BKP are mainly deterministic algorithms. The literature that using evolutionary algorithms solve this problem has not been reported. Therefore, this paper uses a binary particle swarm optimization (BPSO) to solve the BKP. On the basis of using the repair and optimization method to deal with the infeasible solutions, an effective method of using BPSO to solve the BKP is given. For three kinds of large-scale BKP instances, the feasibility and efficiency of BPSO are verified by comparing the results with whale optimization algorithm and genetic algorithm. The experimental results show that BPSO is not only more stable, but also can obtain the approximation ratio closer to 1.

Keywords: Bounded knapsack problem · Evolutionary algorithm · Binary particle swarm optimization · Repair and optimization method

1 Introduction

Knapsack problem (KP) [1–3] is a class of combination optimization problem, and it is also a kind of NP-hard problem. It has important theoretical significance and application value in the fields of industry, economy, and finance. The KP includes different expanded forms, such as the classic 0-1 knapsack problem (0-1KP) [4], the multidimensional knapsack problem (MDKP) [5], the multiple knapsack problem (MKP) [6], the bounded knapsack problem (BKP) [7], the unbounded knapsack problem (UKP) [8], the quadratic knapsack problem (QKP) [9], the randomized time-varying knapsack problem (RTVKP) [10] and the set-union knapsack problem (SUKP) [11] etc., and most of them have been successfully applied in various fields.

Because the time complexity of the deterministic algorithms for solving the KP is pseudo polynomial time, it is not suitable for solving the large-scale KP instances. Therefore, one often uses the evolutionary algorithms (EAs) to solve KP [12]. At present, many effective evolutionary algorithms have been proposed successively, such as genetic algorithm (GA) [13], particle swarm optimization (PSO) [14], differential evolution (DE) [15], ant colony optimization (ACO) [16], artificial bee colony (ABC) [17] and whale optimization algorithm (WOA) [18] etc. Among them PSO is a famous evolutionary algorithm proposed by Kennedy and Eberhart in 1995, and they proposed the binary particle swarm optimization (BPSO) in 1997 [19]. Since then,

H. Peng et al. (Eds.): ISICA 2018, CCIS 986, pp. 50–60, 2019.
https://doi.org/10.1007/978-981-13-6473-0_5

several versions of discrete PSO have been proposed. For example, Clerc [20] proposed an improved discrete PSO for solving TSP problem. Van Den Bergh [21] introduced a new construction model of cooperative PSO and used it to solve IP problem. Liu et al. [22] presented a hybrid PSO for solving pipeline scheduling problem. Li et al. [23] proposed a binary particle swarm algorithm based on multiple mutation strategy to solve 0-1 KP. Bansal et al. [24] proposed an improved BPSO by limiting the update equation of position and used it to solve 0-1KP. He et al. [10, 11] solved RTVKP and SUKP respectively by BPSO, and obtained good results. Therefore, it is not difficult to see that BPSO is very suitable for solving combination optimization problems in discrete domain.

BKP is a classical KP problem, which has not been solved by evolutionary algorithms. Therefore, this paper uses BPSO to solve the BKP, and verifies the efficiency by comparing with other algorithms. The rest of this paper is organized as follows: Sect. 2 introduces the mathematical model of BKP. In Sect. 3, we firstly introduce the binary particle swarm optimization (BPSO), the repair optimization method is given to handle the infeasible solution, and the pseudo-code of BPSO to solve the BKP is given at the end. In Sect. 4, the feasibility and efficiency of this method are verified according to the calculation results of BPSO, improved whale optimization algorithm (IWOA) [25] and GA on three kinds of large-scale BKP instances. Finally, we summarize the whole paper and look forward to the future research directions.

2 Definition and Mathematical Model of BKP

BKP is defined as: Given a set of m items, each item i has a profit p_{i}, a weight w_{i}, and a bound b_{i}. The target is to select a number of each item i such that the sum of the profit is maximized, and the sum of weight is not exceed C.

BKP is an expanded form of the 0-1 KP, it can be converted to the 0-1KP. In the BKP, each item i has a bound b_{i}. Set the sum of the quantity of each item is $n = \sum b_{\mathrm{i}}$, it can be regarded as a 0-1 KP with n items. Let the value set is $P = \{p_1, p_2, \ldots, p_{\mathrm{n}}\}$, the weight set is $W = \{w_1, w_2, \ldots, w_n\}$, and the knapsack capacity is C. $p_{\mathrm{i}}, w_{\mathrm{i}} (1 \le i \le n)$, C are positive integers. According to the definition, a mathematical model of the 0-1KP form of BKP can be established. $X = [x_1, x_2, \ldots, x_{\mathrm{n}}] \in \{0, 1\}^n$ stand for a feasible solution of the BKP. $x_i = 1$ means that item i is included in the knapsack, and $x_i = 0$ that it is not. The mathematical model of the 0-1KP form of BKP is as follows:

$$Max. \sum\nolimits_{\mathrm{i}=1}^{\mathrm{n}} p_{\mathrm{i}} x_{\mathrm{i}} \tag{1}$$

$$s.t. \sum\nolimits_{\mathrm{i}=1}^{\mathrm{n}} w_{\mathrm{i}} x_{\mathrm{i}} \le C \; x_i \in \{0, 1\} \tag{2}$$

3 Solve BKP with BPSO

3.1 BPSO

In BPSO, each individual is treated as a particle in the *n*-dimensional space. $X_{\mathrm{i}} = (x_{\mathrm{i}1}, x_{\mathrm{i}2}, \ldots, x_{\mathrm{in}})$ and $V_{\mathrm{i}} = (v_{\mathrm{i}1}, v_{\mathrm{i}2}, \ldots, v_{\mathrm{in}})$ stand for the current position and velocity of the *i*th particle. $P_{\mathrm{i}} = (p_{\mathrm{i}1}, p_{\mathrm{i}2,} \ldots, p_{\mathrm{in}})$ is the local best position. Assuming that the problem to be solved is a minimum optimization problem, the P_i is determined by the formula (3):

$$P_{\mathrm{i}}(t+1) = \begin{cases} P_i(t), \text{if} f(X_{\mathrm{i}}(t+1) \geq f(P_{\mathrm{i}}(t)) \\ X_{\mathrm{i}}(t+1), \text{if} f(X_{\mathrm{i}}(t+1) < f(P_{\mathrm{i}}(t)) \end{cases} \tag{3}$$

The size of population is *N*. $P_{\mathrm{g}}(t)$ is the global best position, and determined by the following formula:

$$P_{\mathrm{g}}(t) \in \{P_0(t), P_1(t), \ldots, P_{\mathrm{N}}(t)\} \,|f\big(P_{\mathrm{g}}(t)\big) = min\{f(P_0(t)), f(P_1(t)), \ldots, f(P_{\mathrm{N}}(t))\} \tag{4}$$

the evolution equation of BPSO can be described as:

$$v_{\mathrm{ij}}(t+1) = v_{\mathrm{ij}}(t) + c_1 r_{1\mathrm{j}}(t)\big(p_{\mathrm{ij}}(t) - x_{\mathrm{ij}}(t)\big) + c_2 r_{2\mathrm{j}}(t)\big(p_{\mathrm{gj}}(t) - x_{\mathrm{ij}}(t)\big), \tag{5}$$

$$x_{\mathrm{ij}}(t+1) = \begin{cases} 0, \mathrm{sig}\big(v_{\mathrm{ij}}(t+1)\big) \leq r_3(t) \\ 1, \mathrm{sig}\big(v_{\mathrm{ij}}(t+1)\big) > r_3(t) \end{cases}. \tag{6}$$

where c_1 and c_2 are acceleration constants, and the values are usually between 0 and 2. $r_1 \sim \mathrm{U}(0, 1)$, $r_2 \sim \mathrm{U}(0, 1)$, $r_3 \sim \mathrm{U}(0, 1)$ are three independent random functions. $sig(x) = 1/(1 + e^{-x})$ is a fuzzy function, $1 \leq i \leq N, 1 \leq j \leq n$.

The initialization process of BPSO is as follows:

(1) Set the size of population is *N*.
(2) v_{ij} is subject to uniform distribution in $[-v_{\max}, v_{\max}]$.
(3) Randomly generate $x_{\mathrm{ij}} = 0$ or $x_{\mathrm{ij}} = 1$.

3.2 Repairing and Optimization Method

Because the BKP is a constrained optimization problem, the infeasible solution may be generated when solving it by BPSO. Therefore, the infeasible solution needs to be processed. There are three main methods to deal with this problem: penalty function method [26, 27], repair method [26] and repairing and optimization method [28, 29]. This paper refers to the idea of the literature [29, 30], and uses the repairing and optimization method to solve the BKP problems. Let $p_{\mathrm{i}}/w_{\mathrm{i}}$ be the density of item *i*. In the repair phase, the items with less density corresponding to the infeasible solution are removed from the knapsack one by one to ensure that the sum of the weight is within C, that is, the infeasible solution is repaired as a feasible solution. In the optimization phase, the items that are not loaded with relatively large density are loaded into the knapsack as much as possible, and will not be overweight after loading.

According to the density, all items are sorted in descending order. The index of each item are stored in an array $H[1...n]$. The BKP-GROA is shown in Algorithm 1.

Algorithm 1. BKP-GROA

```
Input: A potential solution X=[x_1,...,x_n]and array H[1…n]
Output: A feasible solution X=[x_1,...,x_n] and f(X)
fweight←∑_{j=1}^{n} w_j x_j, j←n
WHILE (fweight>C) DO
    IF (x_H[j]=1) THEN
        x_H[j]←0, fweight←fweight-w_H[j]
    END IF
    j←j-1
END WHILE
FOR j←1 to N do
    IF (x_H[j]=0) AND(fweight+w_H[j]≤C) THEN
      x_H[j]←1, fweight←fweight+w_H[j]
   END IF
END FOR
RETURN (X, f(X))
```

3.3 Application of BPSO to BKP

The main steps to solve BKP by using BPSO are as follows: firstly, randomly initialize N particles, calculate the fitness of each particle, and determine P_g. Then, the following process is repeated until the termination condition is met: update the velocity and position according to formulas (5) and (6), and calculate the fitness of each particle; For each particle, if its fitness is better than the fitness of P_i, it will be the local best position at present. Then, the P_g can be determined. Finally, output the optimal solution and optimal value of BKP.

$H[1...n] \leftarrow QuickSort(\{p_j/w_j \mid p_j \in P, w_j \in W, 1 \le j \le n\})$, where $QuickSort$ is used for sorting all items to descending order according to the density, and all items' index are stored in an array $H[1...n]$. Then, the pseudo-code description of BPSO for the BKP is shown in Algorithm 2.

Algorithm 2. BPSO

Input: The population size N, the number of iterations $MaxIter$, and c_1, c_2
Output: Optimal position X^* and $f(X^*)$
1. $H[1 \ldots n] \leftarrow QuickSort(\{p_j/w_j \mid p_j \in P, w_j \in W, 1 \le j \le n\})$
2. Generate initial population X_i (i=1…N) randomly, calculate the fitness of each particle and set $P_i = X_i$;
3. $t \leftarrow 0$;
4. WHILE ($t \le MaxIter$)
5. FOR $i \leftarrow 1$ TO N
6. Update position and velocity of the particles by formula(5)(6)
7. $(X_i, f(X_i)) \leftarrow$ BKP-GROA $(X_i, H[1 \ldots n])$;
8. END FOR
9. FOR $i \leftarrow 1$ TO N
10. IF $f(Xi(t+1)) > f(Pi(t))$ then $P_i(t+1) \leftarrow X_i(t+1)$
11. Else $Pi(t+1) \leftarrow Pi(t)$
12. END IF
13. $P_g = max\{P_i \mid 1 \le i \le N\}$
14. END FOR
15. $t \leftarrow t+1$;
16. END WHILE

4 Experimental Results and Discussions

4.1 BKP Instance and Experimental Environment

In this section, we tested three different types of the BKP: Uncorrelated instances of BKP (UBKP), Weakly correlated instances of BKP (WBKP), and Strongly correlated BKP instances (SBKP), each of which contains 10 BKP instances of size 100, 200, . . ., 1000, namely UBKP1 ~ UBKP10, WBKP1 ~ WBKP10 and SBKP1 ~ SBKP10. For specific data of all instances, please refer to the document from http://xxgc.hgu.edu.cn/uploads/heyichao/ThreekindsofBKPInstances.rar.

The HP 280 Pro G3 MT desktop computer is used for all the calculations in this paper. The hardware configuration is Intel (R) Core (TM) i5-7500 CPU@3.40 GHz with 4 GB. Programming with C language, the compiler environment is VC++ 6.0; The line charts are drawn with Python in JetBrains PyCharm.

4.2 Parameter Settings of Algorithms and Comparison of Calculation Results

In order to verify the effectiveness of BPSO, it is compared with IWOA and GA, the detailed parameters of each algorithm are set as follows: In IWOA, $N = 50$, $b = 0.5$. In GA, $N = 50$, the crossover probability $P_c = 0.8$, and the mutation probability $P_m = 0.001$. In BPSO, $N = 50$, $w = 1.0$, $c_1 = c_2 = 1.8$. The number of iterations of each algorithm is twice the size of the instance.

The calculation results of solving the BKP instances are shown in Tables 1, 2 and 3, where OPT is the optimal value of the instance calculated by the dynamic programming method (DP), and Best denote the best values by using all algorithms among 50 times. Mean and Std denote the average values and the standard deviations.

Table 1. Comparison of 3 algorithms for solving UBKP instances

Instance	Results	DP	IWOA	GA	BPSO
UBKP1	Best	201616	**201616**	**201616**	**201616**
	Mean		201609.16	**201616**	201615.18
	Std		23.4635	0	4.165
UBKP2	Best	414114	**414114**	**414114**	**414114**
	Mean		**414114**	413995.12	**414114**
	Std		0	69.4964	0
UBKP3	Best	594613	594586	**594610**	594603
	Mean		594580.98	**594610**	594602.12
	Std		6.562	0	3.5757
UBKP4	Best	831629	831612	831611	**831614**
	Mean		831601.72	831594.3	**831613.68**
	Std		11.2855	69.4067	0.7332
UBKP5	Best	1003643	1003628	1003602	**1003633**
	Mean		1003619.98	1003589.74	**1003633**
	Std		6.6588	71.1347	0
UBKP6	Best	1228085	1228083	1228073	**1228085**
	Mean		1228075.68	1227988.58	**1228085**
	Std		3.5465	254.3668	0
UBKP7	Best	1524770	**1524759**	1524739	**1524759**
	Mean		1524753.82	1524703.04	**1524757.88**
	Std		4.9018	121.1483	1.796
UBKP8	Best	1692853	1692835	1692835	**1692844**
	Mean		1692835	1692684.64	**1692841.78**
	Std		0	431.4943	2.8865
UBKP9	Best	1869142	1869131	1869095	**1869138**
	Mean		1869122.44	1868982.32	**1869135**
	Std		8.8253	360.0176	3.3941
UBKP10	Best	2066060	**2066060**	2066025	**2066060**
	Mean		2066043.78	2065995.86	**2066060**
	Std		10.6645	66.85	0

From Table 1, we can see that when using BPSO to solve the UBKP instances, the best values are better than IWOA and GA except for the instance UBKP3; Except for the instance UBKP1 and UBKP3, the average values of BPSO are better than IWOA and GA.

As can be seen from Table 2, when BPSO solves the WBKP instances, the best values and average values are better than IWOA and GA.

It can be seen from Table 3 that when BPSO solves the SBKP instances, all instances can reach the optimal value obtained by the deterministic algorithm. Except that the average values of the instance SBKP4 and SBKP5 are not as good as GA, the calculation results of other instances are better than IWOA and GA.

Table 2. Comparison of 3 algorithms for solving WBKP instances

Instance	Results	DP	IWOA	GA	BPSO
WBKP1	Best	119312	119309	119308	**119312**
	Mean		119306.68	119308	**119312**
	Std		3.1333	0	0
WBKP2	Best	297700	**297700**	**297700**	**297700**
	Mean		**297700**	**297700**	**297700**
	Std		0	0	0
WBKP3	Best	444156	444147	444147	**444156**
	Mean		444144.42	444145.36	**444155.82**
	Std		2.3246	7.375	1.26
WBKP4	Best	605678	605668	605653	**605676**
	Mean		605660.6	605652.68	**605675.76**
	Std		6.5635	1.0852	1.1926
WBKP5	Best	772191	772187	772168	**772188**
	Mean		772184.88	772168	**772188**
	Std		2.8889	0	0
WBKP6	Best	890314	890307	890303	**890313**
	Mean		890303.32	890300.6	**890313**
	Std		2.6566	6.3119	0
WSBKP7	Best	1045302	**1045297**	1045291	**1045297**
	Mean		1045294.38	1045272.3	**1045297**
	Std		2.125	56.1	0
WBKP8	Best	1210947	**1210944**	1210936	**1210944**
	Mean		1210941.14	1210936	**1210944**
	Std		2.9121	0	0
WBKP9	Best	1407365	**1407364**	**1407364**	**1407364**
	Mean		**1407364**	1407318.18	**1407364**
	Std		0	168.6493	0
WBKP10	Best	1574079	1574074	1574066	**1574075**
	Mean		1574071.92	1574000.56	**1574074.4**
	Std		2.2076	183.7961	0.4899

In order to compare the performance of IWOA, BPSO and GA more intuitively, the average approximation ratio and the best approximation ratio are used to verify the performance. The average approximation ratio is defined as OPT/Mean, and the best approximation ratio is defined as OPT/Best. Figures 1, 2 and 3 shows the comparison of the average approximation ratio and the best approximation ratio of IWOA, GA and BPSO for three types of BKP instances.

As can be seen from Fig. 1(a), the average approximation ratio of BPSO is better than GA except UBKP1 and UBKP3, and BPSO is better than IWOA in solving all UBKP instances. From Fig. 1(b), we can see that the best approximation ratio obtained by BPSO is better than IWOA and GA except UBKP3. As can be seen from Fig. 2, the

Table 3. Comparison of 3 algorithms for solving SBKP instances

Instance	Results	DP	IWOA	GA	BPSO
SBKP1	Best	144822	144821	**144822**	**144822**
	Mean		144815.12	**144822**	**144822**
	Std		7.8426	0	0
SBKP2	Best	259853	**259853**	**259853**	**259853**
	Mean		259844.92	**259853**	**259853**
	Std		5.5239	0	0
SBKP3	Best	433414	**433414**	**433414**	**433414**
	Mean		433406.42	**433414**	**433414**
	Std		6.2421	0	0
SBKP4	Best	493847	**493847**	**493847**	**493847**
	Mean		493841.46	**493847**	493846.94
	Std		4.8092	0	0.2375
SBKP5	Best	688246	**688246**	**688246**	**688246**
	Mean		688240.14	**688246**	688245.938
	Std		5.0991	0	0.1972
SBKP6	Best	849526	**849526**	**849526**	**849526**
	Mean		849523.98	849523.08	**849526**
	Std		3.3555	4.6897	0
SBKP7	Best	1060106	**1060106**	1060105	**1060106**
	Mean		1060104.38	1060100.82	**1060106**
	Std		1.7192	5.2486	0
SBKP8	Best	1171576	**1171576**	1171566	**1171576**
	Mean		1171570.18	1171554.18	**1171576**
	Std		5.2103	9.5408	0
SBKP9	Best	1263609	**1263609**	1263597	**1263609**
	Mean		1263606.02	1263591.72	**1263609**
	Std		3.1968	15.3467	0
SBKP10	Best	1412095	**1412095**	1412085	**1412095**
	Mean		1412089.16	1412074.88	**1412095**
	Std		3.7382	13.3396	0

average approximation ratio and the best approximation ratio of BPSO are better than IWOA and GA in solving WBKP instances, and the performance of BPSO is very stable for ten different scales of WBKP instances.

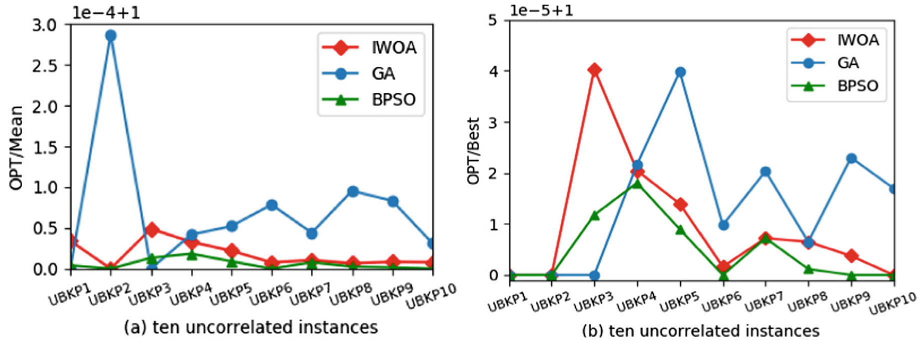

Fig. 1. The approximation ratio of 3 algorithms for solving UBKP instances

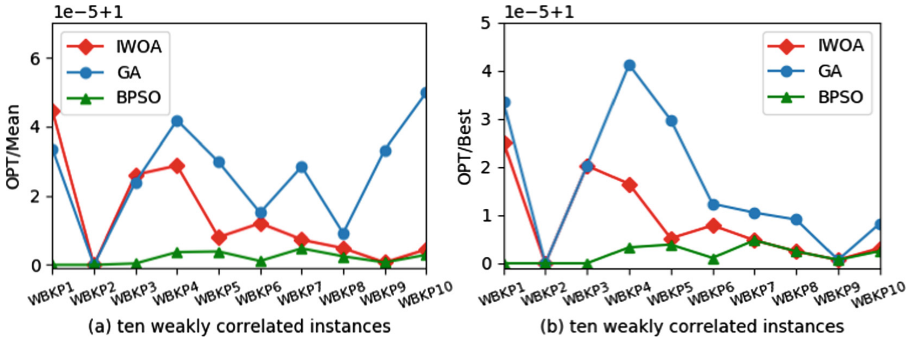

Fig. 2. The approximation ratio of 3 algorithms for solving WBKP instances

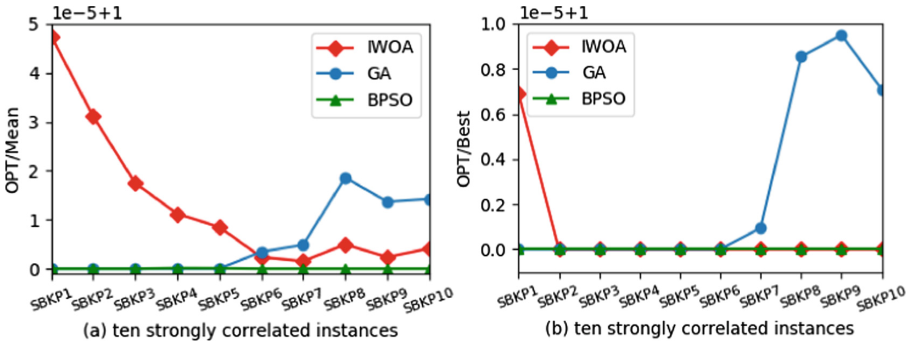

Fig. 3. The approximation ratio of 3 algorithms for solving SBKP instances

As can be seen from Fig. 3, the average approximation ratio and the best approximation ratio of BPSO are almost all 1 when solving SBKP instances, and the performance of BPSO is very stable for ten different scale SBKP instances; the performance of IWOA and GA is not as good as BPSO and the stability is inferior to BPSO when solving few instances.

5 Conclusion

In this paper, BPSO is used to solve the BKP problem, and the performance of BPSO is verified by three kinds of large-scale BKP instances. The comparison with the experimental results of IWOA and GA shows that the best values and average values obtained by BPSO are better when solving BKP instances. In addition, by comparing the average approximation ratio and the best approximation ratio, it is not difficult to see that BPSO not only has good stability, but also has the approximation ratio closer to 1, so the calculation effect is optimal. Although BKP is a classical combination optimization problem, the research of its solution by using evolutionary algorithms is relatively weak. Therefore, it is worthy of further research to explore the performance of BKP using other EAs.

Acknowledgments. This work was supported by the Scientific Research Project Program of Colleges and Universities in Hebei Province (ZD2016005), and the Natural Science Foundation of Hebei Province (F2016403055).

References

1. Karp, R.M.: Reducibility among combinatorial problems. In: Miller, R.E., Thatcher, J.W. (eds.) Proceedings of the Complexity of Computer Computations, pp. 110–137. Plenum Press, New York (1972)
2. Martello, S., Toth, P.: Knapsack Problems: Algorithms and Computer Implementations. Wiley, New York (1990)
3. Kellerer, H., Pferschy, U., Pisinger, D.: Knapsack Problems, pp. 55–75. Springer, Berlin (2004). https://doi.org/10.1007/978-3-540-24777-7
4. Mathews, G.B.: On the partition of numbers. Proc. Lond. Math. Soc. **28**, 486–490 (1897). https://doi.org/10.1112/plms/s1-28.1.486
5. Chu, P.C., Beasley, J.E.: A genetic algorithm for the multidimensional knapsack problem. J. Heuristics **4**(1), 63–86 (1998)
6. Khuri, S.L.: The zero/one multiple knapsack problem and genetic algorithms. In: Proceedings of the 1994 ACM Symposium of Applied Computing, pp. 188–193 (1994)
7. Pisinger, D.: A minimal algorithm for the bounded knapsack problem. In: Balas, E., Clausen, J. (eds.) IPCO 1995. LNCS, vol. 920, pp. 95–109. Springer, Heidelberg (1995). https://doi.org/10.1007/3-540-59408-6_44
8. Martello, S., Toth, P.: An exact algorithm for large unbounded knapsack problems. Oper. Res. Lett. **9**(1), 15–20 (1990)
9. Caprara, A., Pisinger, D., Toth, P.: Exact solution of the quadratic knapsack problem. Informs J. Comput. **11**(2), 125–137 (1999)

10. He, Y.C., Wang, X.Z., Li, W.B., Zhao, S.L.: Exact algorithms and evolutionary algorithms for randomized time-varying knapsack problem. Ruan Jian Xue Bao/J. Softw. (2016). (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4937.htm. https://doi.org/10.13328/j.cnki.jos.004937
11. He, Y., Xie, H., Wong, T.-L., Wang, X.: A novel binary artificial bee colony algorithm for the set-union knapsack problem. Future Gener. Comput. Syst. **87**(1), 77–86 (2018)
12. Wang, X.Z., He, Y.-C.: Evolutionary algorithms for knapsack problems. J. Softw. **28**, 1–16 (2017)
13. Goldberg, D.E.: Genetic Algorithm in Search, Optimization, and Machine Learning. Addison-Wesley, Reading (1989)
14. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: 1995 Proceedings of the IEEE International Conference on Neural Networks, vol. 4, pp. 1942–1948 (1995)
15. Storn, R., Price, K.: Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. J. Global Optim. **11**(4), 341–359 (1997)
16. Dorigo, M., Birattari, M., Stutzle, T.: Ant colony optimization. IEEE Comput. Intell. Mag. **1**(4), 28–39 (2007)
17. Karaboga, D., Basturk, B.: A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. J. Global Optim. **39**(3), 459–471 (2007)
18. Mirjalili, S., Lewis, A.: The whale optimization algorithm. Adv. Eng. Softw. **95**, 51–67 (2016)
19. Kennedy, J., Eberhart, R.C.: A discrete binary version of the particle swarm algorithm. In: Proceedings 1997 Conference on Systems, Man, and Cybernetics, pp. 4104–4109. IEEE Service Center, Piscataway (1997)
20. Clerc, M.: Discrete particle swarm optimization, illustrated by the traveling salesman problem. In: Clerc, M. (ed.) New Optimization Techniques in Engineering. STUDFUZZ, vol. 141, pp. 219–239. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-39930-8_8
21. Van Den Bergh, F.: An analysis of particle swarm optimizers. Ph.D. thesis (2007)
22. Liu, B., Wang, L., Jin, Y.H.: An effective PSO-based memetic algorithm for flow shop scheduling. IEEE Trans. Syst. Man Cybern. Part B Cybern. **37**(1), 18 (2007). A Publication of the IEEE Systems Man & Cybernetics Society
23. Li, Z., Li, N.: A novel multi-mutation binary particle swarm optimization for 0/1 knapsack problem. In: 2009 Chinese Control and Decision Conference, no. 2, pp. 3090–3095 (2009)
24. Bansal, J.C., Deep, K.: A modified binary particle swarm optimization for knapsack problems. Appl. Math. Comput. **218**(22), 11042–11061 (2012)
25. Hussien, A.G., Houssein, E.H., Hassanien, A.E.: A binary whale optimization algorithm with hyperbolic tangent fitness function for feature selection. In: Eighth International Conference on Intelligent Computing and Information Systems. IEEE (2018)
26. Michalewicz, Z.: Genetic Algorithm + Data Structure = Evolution Programs, pp. 13–103. Springer, Berlin (1996). https://doi.org/10.1007/978-3-662-03315-9
27. Zou, D.X., Gao, L.Q., Li, S., Wu, J.H.: Solving 0-1 knapsack problem by a novel global harmony search algorithm. Appl. Soft Comput. **11**, 1556–1564 (2011). https://doi.org/10.1016/j.asoc.2010.07.019
28. He, Y.C., Zhang, X.L., Li, X., Wu, W.L., Gao, S.G.: Algorithms for randomized time-varying knapsack problems. J. Comb. Optim. **31**(1), 95–117 (2016). https://doi.org/10.1007/s10878-014-9717-1
29. He, Y.C., Wang, X.Z., Li, W.B., Zhao, S.L.: Exact algorithms and evolutionary algorithms for randomized time-varying knapsack problem. Ruan Jian Xue Bao/J. Softw. (2016). (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4937.htm. https://doi.org/10.13328/j.cnki.jos.004937
30. He, Y.-C., Song, J.-M., Zhang, J.-M., et al.: Research on genetic algorithm for solving static and dynamic knapsack problems. Appl. Res. Comput. **32**(4), 1011–1015 (2015). (in Chinese)