Hindawi
Computational Intelligence and Neuroscience
Volume 2022, Article ID 1266529, 11 pages
https://doi.org/10.1155/2022/1266529

Research Article

Fast Polynomial Time Approximate Solution for 0-1 Knapsack Problem

Zhengyuan Wang, Hui Zhang, and Yali Li

Xi'an Research Institute of Hi-Tech, Xi'an, Shaanxi 710025, China

Correspondence should be addressed to Zhengyuan Wang; zh53054958@163.com

Received 16 May 2022; Revised 14 September 2022; Accepted 15 September 2022; Published 22 October 2022

Academic Editor: Maciej Lawrynczuk

0-1 Knapsack problem (KP) is NP-hard. Approximate solution is vital for solving KP exactly. In this paper, a fast polynomial time approximate solution (FPTAS) is proposed for KP. FPTAS is a local search algorithm. The best approximate solution to KP can be found in the neighborhood of the solution of upper bound for exact k-item knapsack problem (E-kKP) where k is near to the critical item s. FPTAS, in practice, often achieves high accuracy with high speed in solving KP. The computational experiments show that the approximate algorithm for KP is valid.

1. Introduction

0-1 Knapsack problem (KP) is a classical combinatorial optimization problem. KP is NP-hard. Knapsack problems were used to model capital budgeting problems in which investment projects are to be selected subject to expenditure limitations. Additionally, the knapsack problem has been used to model loading problems. Knapsack problems, moreover, arise as suboptimization problems in solving larger optimization problem [1]. KP can be formulated as follows:

$$\begin{gathered}\max f(x)=\sum_{i=1}^{n} p_i x_i,\\ \text{s.t,}\\ \sum_{i=1}^{n} w_i x_i \le c,\\ x_i = 0, 1. i = 1, 2, \cdots, n,\end{gathered} \tag{1}$$

where p_i is the profit of item i, w_i is the weight of item i, c is the capacity of the knapsack, n is the number of items in KP, and variable $x_i = 0 \text{ or } 1$ indicates whether item i is selected or not. Without loss of generality, it is assumed that all items are arranged in nonincreasing order of efficiency.

$$\left(\frac{p_i}{w_i}\right) \ge \left(\frac{p_{i+1}}{w_{i+1}}\right) (i = 1, 2, \cdots, n-1). \tag{2}$$

It is very important to find a fast polynomial approximation for KP in practice. Greedy algorithm [1], an approximate algorithm for KP, is on the basis of certain rules, such as the item being selected with priority over larger efficiency, larger profit, or smaller weight. Greedy algorithm is $O(n)$ time complexity, but it is not an ε-approximate algorithm [1]. Approximation algorithm is often k-neighborhood local search method. Increasing the radius k of the neighborhood can improve the accuracy of the approximate algorithm [1, 2]. While PTAS for KP typically require only $O(n)$ storage, all FPTAS are based on dynamic programming and their memory requirement increases rapidly with the accuracy ε, which makes them impractical even for relatively big values of ε [3]. Heuristics rules are adopted to decrease the calculation in searching accurate approximation, such as harmony search algorithm [4, 5], amoeboid organism algorithm [6], cuckoo search algorithm [5, 7], binary monarch butterfly optimization [8], cognitive discrete gravitational search algorithm [9], bat algorithm [10], and wind driven optimization [11]. Nowadays, it is tend to combine different heuristics together in solving combinatorial optimization problem, such as mixed-variable differentiate evolution [12], self-adaptive differential evolution algorithm [13], two-stage cooperative evolutionary algorithm [14], cooperative water wave optimization algorithm with reinforcement learning [15], and cooperative multi-stage hyper-heuristic algorithm [16]. However, these algorithms cannot guarantee the accuracy of the solution of KP. Meanwhile, these methods for the solution to KP are generally time-consuming. More domain knowledge is necessary for better algorithm. Pisinger gave an exact algorithm for KP [17] which is based on an expanding core. The items not included in the core are certain to be selected or not in the optimal solution, while the items in the core are uncertain to be selected or not in the optimal solution. He found that algorithms solving some kinds of core problem may be stuck by difficult cores [18]. For example, KP is determined by

$$w_i = i, p_i = 10i^2 + 10^7, c = 200020, i = 1, 2, \cdots, 10^4. \tag{3}$$

It is easy to find that the core of KP is [1, 10000], so this problem is difficult to tackle by the exact algorithm in [17]. In this paper, it is easy to get an approximate solution of KP which objective value is 6395122580. The solution can be proved to be the optimal solution.

In this paper, a fast polynomial approximate solution is proposed for 0-1 knapsack problems based on the solution of its upper bound. Firstly, an upper bound is presented based on the exact k-item knapsack problem E-kKP in Section 2. Secondly, an initial solution of KP is constructed on the basis of the solution of upper bound of E-kKP in Section 3.1. Thirdly, the approximate solution is proposed to find the best solution in the neighborhood of the initial solution in Section 3.2. In Section 3.3, the best approximation solution is achieved by comparison with number of items changing. In Section 3.4, the calculation for the approximate solution of KP is analyzed. In Section 4, computational experiments of KP are implemented. The results show that the approximate solution proposed in this paper can achieve high accuracy in general. It implicates that the exact solution to KP is similar to the solution to the upper bound. The algorithm proposed here is a fast polynomial approximate solution to KP.

2. Upper Bound for KP

If there are exact k objects selected in the knapsack, then KP is an exact k-item knapsack problem E-kKP formulated as follows [3]:

$$\begin{aligned} &\max f(x) = \sum_{i=1}^{n} p_i x_i, \\ &\text{s.t,} \\ &\sum_{i=1}^{n} w_i x_i \le c, \\ &\sum_{i=1}^{n} x_i = k, \\ &x_i = 0, 1. i = 1, 2, \cdots, n. \end{aligned} \tag{4}$$

The upper bound of E-kKP may be achieved by Lagrangian relaxation of capacity constraint:

$$\begin{aligned} &L(k,\lambda) = \lambda c + \max \sum_{i=1}^{n} (p_i - \lambda w_i) u_i, \\ &\text{s.t,} \\ &\sum_{i=1}^{n} u_i = k, \\ &u_i = 0, 1. i = 1, 2, \cdots, n. \end{aligned} \tag{5}$$

Suppose that

$$p_{d_{k1}} - \lambda w_{d_{k1}} \ge p_{d_{k2}} - \lambda w_{d_{k2}} \ge \cdots \ge p_{d_{kn}} - \lambda w_{d_{kn}}, w_{d_{kk}} \le w_{d_{kk+1}}. \tag{6}$$

Then

$$L(k,\lambda) = \lambda c + \sum\nolimits_{i=1}^{k} \left(p_{d_{ki}} - \lambda w_{d_{ki}}\right). \tag{7}$$

So $L(k,\lambda)$ can be solved by sorting $p_i - \lambda w_i$ in descending order in O ($n\ln n$) time if λ and k are fixed. The upper bound of E-kKP is formulated as follows:

$$B_k = \min_{\lambda \ge 0} L(k,\lambda). \tag{8}$$

It can be proved that $L(k,\lambda)$ is a unimodal function of λ if k is fixed. B_k can be quickly solved by linear search algorithm [19]. So B_k can be solved in O ($n\ln n$) time.

The upper bound of KP is the maximum value of B_k:

$$B = \max_{0 \le k \le n} B_k. \tag{9}$$

It can be proved that B_k increases when $k < s - 1$ and decreases when $k > s$, where the critical item s satisfies.

$$0 \le c - w_1 - w_2 - \cdots - w_{s-1} < w_s. \tag{10}$$

So the upper bound of KP is

$$B = \min \{B_{s-1}, B_s\}. \tag{11}$$

Example 1. Consider the instance of KP listed in Table 1. The capacity is 467.8435.

Here the critical item s is 20. There is no feasible solution with 20 items included in the knapsack. So the upper bound $B_{20}=0$. If k is 19, we have the optimal ratio $\lambda_{19}=1.0003251$ and the upper bound $B_{19}=486.9565$ by (8). The solution of $L(19,\lambda_{19})$ is $u^{(19)}=(u_{d_{19i}})$as follows:

$$u_{d_{19i}}=\begin{cases}1, & i\leq 19,\\ 0, & i>19,\end{cases}$$
$$(d_{19i})=(2,1,4,7,5,3,10,6,9,16,8,13,18,11,26,20,21,17,14,27,28,23,12,15,19,30,25,24,22,29). \tag{12}$$

So the upper bound of KP is 486.9565.

We test different upper bounds of KP from Pisinger's paper [20], where the weights and profits of items are randomized. The instances listed in Table2 are tested to compare our proposed method with the existing upper bounds, such as the upper bound U_{MT2} proposed by Martello and Toth [21], the improved upper bound U_{MTM} [22] and the upper bound $U_{k\max}$ with maximum cardinality [23]. The difficult instances can be constructed as follows [20]:

(i) Uncorrelated instances with similar weights: Weights w_i are distributed in $[R, R+100]$ and the profits p_i in [1, 1000].

(ii) Uncorrelated data instances: p_i and w_i are chosen randomly in $[1, R]$.

(iii) Weakly correlated instances: Weights w_i are chosen randomly in $[1, R]$ and the profits p_i in $[w_i-0.1\,R,\ w_i+0.1\,R]$ such that $p_i\geq 1$.

(iv) Strongly correlated instances: Weights w_i are distributed in $[1, R]$ and $p_i=w_i+0.1R$.

(v) Inverse strongly correlated instances: Profits p_i are distributed in $[1, R]$ and $w_i=p_i$+0.1R.

(iv) Almost strongly correlated instances: Weights w_i are distributed in $[1, R]$ and the profits p_i in $[w_i+0.098R,\ w_i+0.102R]$.

(vii) Subset sum instances: Weights w_j are randomly distributed in $[1, R]$ and $p_i=w_i$.

(viii) Circle instances circle (d): The weights are uniformly distributed in $[1, R]$ and for each weight w the corresponding profit is chosen as $p=d\sqrt{4wR-w^2}$ where d is 2/3.

(ix) Profit ceiling instances pceil (d): The weights of the n items are randomly distributed in $[1, R]$, and the profits are set to $p_i=d\lceil w_i/d\rceil$. The parameter d was chosen as $d=3$.

(x) Multiple strongly correlated instances mstr (k_1, k_2, d): The weights of the n items are randomly distributed in $[1, R]$. If the weight w_i is divisible by d, then we set the profit $p_i:=w_i+k_1$; otherwise set it to $p_i:=w_i+k_2$. We set $d:=6$ here.

For each instance type, a series of $K=100$ instances is performed, and the capacity is determined by (13). All the instances above are generated with data range $R=10^3$, 10^4, 10^5, 10^6 or 10^7.

$$c=\frac{k}{1+K}\sum_{i=1}^{n}w_i,k=1,2,\cdots,K. \tag{13}$$

There are 100 instances for each type of KP where the capability is described as follows:

$$C=\left[\frac{k}{101}\sum_{i=1}^{n}w_i\right],k=1,2,\cdots,100,n=10^4. \tag{14}$$

The mean relative error of upper bounds to the best upper bound of KP is listed in Table 3.

From Table 3, we find that the relative error of the upper bound B is the minimum in general. The upper bounds of 5000 instances are carried out with different methods. More details are listed in Table 4.

From Table 4, we find that $U_{k\max}$ is the best upper bound in 2025 instances. The upper bound B plays an important role in obtaining the best upper bound even if $s\leq k_{\max}$.

The upper bound can gather most items selected in the optimal solution all together. For example, KP is determined by

$$w_i=i,p_i=10i^2+10^7,c=200020,i=1,2,\cdots,10^4. \tag{15}$$

A maximum of 631 items can be selected in the knapsack. The upper bound B_{s-1} ($s=632$) and the solution $u^{(s-1)}$ are as follows:

$$\begin{aligned}B_{s-1}&=\min_{\lambda\geq 0}\left\{\max\left\{\lambda c+\sum_{i=1}^{10000}(p_i-\lambda w_i)x_i\,\middle|\,\sum_{i=1}^{10000}x_i=s-1,x_i=0,1\right\}\right\}\\ &=106310c+\sum_{i=1}^{631}(p_i-106310w_i)=7215794600,u_i=\begin{cases}1, & i=1,2,\ldots,631,\\ 0, & 0,631<i\leq 10000.\end{cases}\end{aligned} \tag{16}$$

The initial solution $z^{(s-1)}$ is equal to $u^{(s-1)}$. The best solution $y_a^{(s-1)}$ in the neighborhood $\cup_{h=1}^{4} N(z^{(s-1)}, 2h)$ can be described as follows:

$$y_i = \begin{cases} 1, & i = 1, 2, \ldots, 630, 1225, \\ 0, & i \neq 1225, 630 < i \leq 10000. \end{cases} \tag{17}$$

It is obviously that there is little difference between $z^{(s-1)}$ and $y_a^{(s-1)}$. Relation between $(p_i - 106310w_i)$ and w_i is displayed in Figure 1. Relation between p_i/w_i and w_i is displayed in Figure 2. It is found that $(p_i - 106310w_i)$ changes more dramatically than p_i/w_i. It makes solution to KP easy.

From Figure 1, we can see that $(p_i - 106310w_i)$ of all items selected are larger, while the others are smaller. It is easy to get a better solution on the basis of $(p_i - 106310w_i)$ than that on the basis of efficiency p_i/w_i.

3. Approximate Solutions

The approximate solution to E-kKP is a solution to KP. We obtain the best approximate solution to KP by comparison to approximate solutions for E-kKP where k is near to the critical item s. In order to achieve a better solution to E-kKP, we firstly obtain an initial solution on the basis of the solution to the upper bound of E-kKP. Then the initial solution is developed by local search in the neighborhood of the initial solution. At last, the approximate solution to KP is the best approximate solution to E-kKP with various k. The upper bound of E-kKP is used to decrease calculation. The solution to the upper bound of E-kKP and the upper bound make key contribution in FPTAS to KP.

3.1. Initial Solution to E-kKP. Let $u^{(k)} = (u_{d_{ki}})$ be an optimal solution to the upper bound B_k. We may obtain an initial solution to E-kKP on the basis of $u^{(k)}$. The initial solution $z^{(k)}$ to E-kKP is determined by

Table 1: Weight and profit of items.

i	w_i	p_i
1	1.7856	2.7924
2	4.877	5.8853
3	6.3493	7.3521
4	7.0943	8.0992
5	7.8807	8.885
6	8.5593	9.5616
7	13.9249	14.9319
8	19.6114	20.6148
9	21.0881	22.0925
10	24.2688	25.2767
11	27.3441	28.3472
12	31.618	32.6189
13	32.7739	33.7797
14	32.787	33.7898
15	33.9368	34.9379
16	37.1566	38.1662
17	37.887	38.892
18	39.6104	40.6175
19	40.014	41.0159
20	40.7362	41.7432
21	42.4565	43.463
22	45.2896	46.2899
23	45.6688	46.6693
24	45.7868	46.7932
25	46.6997	47.7013
26	47.8583	48.866
27	47.8753	48.8848
28	47.9746	48.9822
29	48.2444	49.2448
30	48.5296	49.5335

$$u_{d_{ki}} = \begin{cases} 1, & i \leq k, \\ 0, & k < i \leq n, \end{cases}$$
$$z_{d_{ki}} = \begin{cases} u_{d_{ki}}, i \neq k, k+1, \\ u_{d_{ki}}, w_{d_{kk+1}} > c - \sum_{j=1}^{k-1} w_{d_{kj}}, i = k, k+1, \\ 1 - u_{d_{ki}}, w_{d_{kk}} > c - \sum_{j=1}^{k-1} w_{d_{kj}}, i = k, k+1, \\ u_{d_{ki}}, c - \sum_{j=1}^{k-1} w_{d_{kj}} \geq \max\left\{w_{d_{kk}}, w_{d_{kk+1}}\right\}, p_{d_{kk}} \geq p_{d_{kk+1}}, i = k, k+1, \\ 1 - u_{d_{ki}}, c - \sum_{j=1}^{k-1} w_{d_{kj}} \geq \max\left\{w_{d_{kk}}, w_{d_{kk+1}}\right\}, p_{d_{kk}} < p_{d_{kk+1}}, i = k, k+1. \end{cases} \tag{18}$$

In Example 1, $z^{(19)}$ is an initial solution and its objective value is

Table 2: Weights and profits of knapsack problems from Pisinger's paper [20].

No.	(w_i, p_i) $(i = 1, 2, \ldots, 10^4)$	No.	(w_i, p_i) $(i = 1, 2, \ldots, 10^4)$
1	$([R + 100u_i], \lceil 1000v_i \rceil)$	2	$(\lceil Ru_i \rceil, \lceil Rv_i \rceil)$
3	$([Ru_i], [R(u_i + 0.2v_i - 0.1)])$	4	$([Ru_i], [Ru_i + 0.1R])$
5	$([Rv_i + 0.1R], [Rv_i])$	6	$([Ru_i + 0.098R], [R(u_i + 0.102R)])$
7	$(\lceil Ru_i \rceil, \lceil Ru_i \rceil)$	8	$([Ru_i], [2R/3\sqrt{u_i(4 - u_i)}])$
9	$([Ru_i], \lceil Ru_i/3 \rceil)$	10	$([Ru_i], [Ru_i + 0.3R]), [Ru_i]\%6 = 0;$ $([Ru_i], [Ru_i + 0.2R]), [Ru_i]\%6 \neq 0$

*Note. u_i and v_i are uniformly distributed in [0,1]. $R = 10^3$, 10^4, 10^5, 10^6 or 10^7.

Table 3: Mean relative error of upper bounds to the best upper bound of (KP) (ppm).

(No, R)	U_{MT2}	U_{MTM}	$U_{k\text{max}}$	B	(No, R)	U_{MT2}	U_{MTM}	$U_{k\text{max}}$	B
(1, 10^3)	4.6696	4.6696	4.6869	0	(2, 10^3)	0.0138	0.0031	0.0421	0.0174
(1, 10^4)	91.413	91.3869	81.4422	0	(2, 10^4)	0.0345	0.0075	0.0465	0.0189
(1, 10^5)	60.2607	60.2542	43.2765	0	(2, 10^5)	0.0194	0.005	0.0351	0.0164
(1, 10^6)	61.0366	61.0366	0	0	(2, 10^6)	0.0258	0.0057	0.0391	0.0152
(1, 10^7)	61.3372	61.3372	0	0	(2, 10^7)	0.0274	0.0069	0.0466	0.018
(3, 10^3)	0.0116	0.0049	0.0179	0	(4, 10^3)	28.7876	28.7876	0	0
(3, 10^4)	0.0031	0	0.0196	0.0044	(4, 10^4)	34.656	34.6518	0	0
(3, 10^5)	0.0148	0.0018	0.027	0.016	(4, 10^5)	31.5535	31.5456	0	0
(3, 10^6)	0.0114	0.0016	0.0166	0.0071	(4, 10^6)	27.5082	27.5001	0	0
(3, 10^7)	0.0066	0.0022	0.0105	0.0031	(4, 10^7)	25.7493	25.7467	0	0
(5, 10^3)	38.9054	38.9054	38.9054	0	(6, 10^3)	0	0	0	0
(5, 10^4)	0	0	0	0	(6, 10^4)	0	0	0	0
(5, 10^5)	0	0	0	0	(6, 10^5)	0	0	0	0
(5, 10^6)	0	0	0	0	(6, 10^6)	0	0	0	0
(5, 10^7)	0	0	0	0	(6, 10^7)	0	0	0	0
(7, 10^3)	0	0	0	0	(8, 10^3)	0	0	0	0
(7, 10^4)	0	0	0	0	(8, 10^4)	0	0	0	0
(7, 10^5)	0	0	0	0	(8, 10^5)	22.8602	22.8602	0	0
(7, 10^6)	0	0	0	0	(8, 10^6)	0	0	0	0
(7, 10^7)	0	0	0	0	(8, 10^7)	0	0	0	0
(9, 10^3)	0	0	0	0	(10, 10^3)	0	0	0	0
(9, 10^4)	0	0	0	0	(10, 10^4)	0	0	0	0
(9, 10^5)	16.1344	16.1344	0	0	(10, 10^5)	0	0	0	0
(9, 10^6)	30.1447	30.1447	0	0	(10, 10^6)	0	0	0	0
(9, 10^7)	8.9156	8.9156	8.9156	0	(10, 10^7)	0	0	0	0

*The best upper bound is $\min\{U_{\text{MT2}}, U_{\text{MTM}}, U_{\text{kmax}}, B\}$ in this paper.

Table 4: Sum of the best upper bounds in 5000 instances.

Upper bound	Sum of the best upper bound	Percentage of the best upper bound (%)	Sum of the best upper bounds equal to B
U_{MT2}	1520	30.40	1495
U_{MTM}	1925	38.50	1548
$U_{k\text{max}}$	2025	40.50	2025
B	4623	92.46	4623

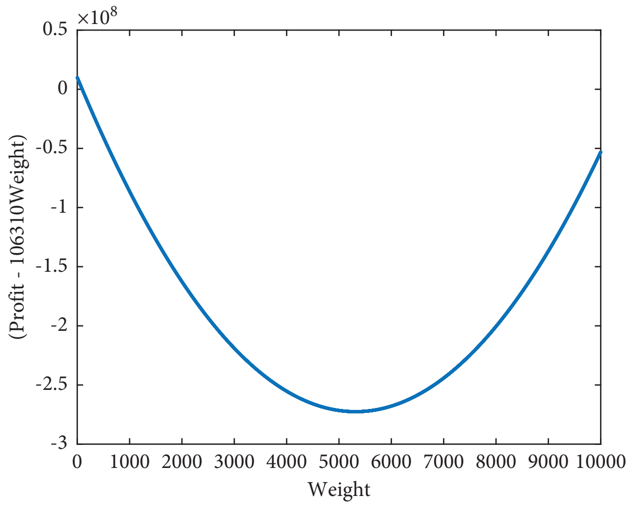

Figure 1: Relation between $(p_i - 106310w_i)$ and w_i.

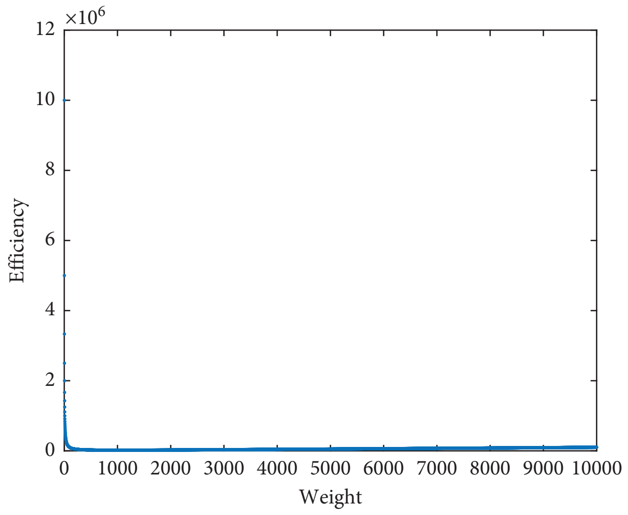

Figure 2: Relation between (p_i/w_i) and w_i.

$$f\left(z^{(19)}\right) = 473.1749$$
$$Zd19i = \begin{cases} 1, i \le 19 \\ 0, i > 19 \end{cases} \tag{19}$$
$$\left(d_{19i}\right) = (2, 1, 4, 7, 5, 3, 10, 6, 9, 16, 8, 13, 18, 11, 26, 20, 21, 17, 14, 27, 28, 23, 12, 15, 19, 30, 25, 24, 22, 29).$$

3.2. Approximate Solution to E-kKP. Approximate solution to E-kKP is the best solution in the neighborhood of $z^{(k)}$. Let $N(z^{(k)}, 2h)$ be a neighborhood of $z^{(k)}$ that is defined by

$$N\left(z^{(k)}, 2h\right) = \left\{ y^{(k)} = \left(y_{d_{ki}}\right) \left| \sum_{i=1}^{n} \left| y_{d_{ki}} - z_{d_{ki}} \right| = 2h, \sum_{i=1}^{n} y_{d_{ki}} = k, y_{d_{ki}} = 0, 1 \right. \right\}. \tag{20}$$

It is obvious that the size of $N(z^{(k)}, 2h)$ increases with h. But it is unnecessary to take into account all elements of $N(z^{(k)}, 2h)$. Algorithm 1 is a fast algorithm for searching the best solution in $N(z^{(k)}, 2h)$.

In order to decrease the calculation for the approximate solution to KP, $N(z^{(k)}, 2h)$ is redefined by (21) and Step 1 in Algorithm 1 is modified correspondingly.

$$N\left(z^{(k)}, 2h\right) = \left\{ y^{(k)} = \left(y_{d_{ki}}\right) \left| \sum_{i=L_{kh}}^{U_{kh}} \left| y_{d_{ki}} - z_{d_{ki}} \right| = 2h, \sum_{i=1}^{n} y_{d_{ki}} = k, y_{d_{ki}} = 0, 1 \right. \right\}, \tag{21}$$

where

```
Step 1: Obtain the neighborhood $N(z^{(k)}, 2h)$ by (20), $h = 1, 2, 3, 4$.
Step 2: Calculate the objective value of feasible solution $y^{(k)} = (y_{d_{ki}})$ in $N(z^{(k)}, 2h)$ by equation as follows:
    $f(y^{(k)}) = \begin{cases} \sum_{i=1}^{n} p_i y_{d_{ki}}, & \sum_{i=1}^{n} w_i y_{d_{ki}} \le c \\ 0, & \sum_{i=1}^{n} w_i y_{d_{ki}} > c \end{cases}$
Step 3: Obtain the best solution $y_a^{(k)}$ in $N(z^{(k)}, 2h)(h = 1,2,3,4)$ by (Algorithm 1). $f(y_a^{(k)}) = \max\{f(y^{(k)}) \mid y^{(k)} \in \cup_{h=1}^{4} N(z^{(k)}, 2h)\}$
```

Algorithm 1: Approximate solution to (E-kKP).

$$L_{kh} = \begin{cases} 1, h = 1 \\ \max\{k-100, 1\}, h = 2 \\ \max\{1, k-50\}, h = 3 \\ \max\{k-30, 1\}, h = 4 \end{cases}$$
$$U_{kh} = \begin{cases} n, h = 1 \\ \min\{99+k, n\}, h = 2 \\ \min\{49+k, n\}, h = 3 \\ \min\{29+k, n\}, h = 4 \end{cases} \tag{22}$$
$$s - 5 \le k \le s + 4.$$

From (21), we know that the size of $N(z^{(k)}, 2h)(h = 1,2,3,4)$ is limited. In order to search for the best solution, it is unnecessary to seek all solutions in $N(z^{(k)}, 2h)$. Let

$$\begin{gathered} r_w = c - w_{d_{k1}} - w_{d_{k2}} - \cdots - w_{d_{kk}} \\ \left\{(p_{s1}, w_{s1}) | p_{s1} = p_{d_{ki_1}} + \cdots + p_{ki_h}, w_{s1} = w_{d_{ki_1}} + \cdots + w_{d_{ki_h}} + r_w, \max\{1, L_{kh}\} \le i_1 < \cdots < i_h \le k\right\} \\ \left\{(p_{s2}, w_{s2}) | p_{s2} = p_{d_{kj_1}} + \cdots + p_{d_{kj_h}}, w_{s2} = w_{d_{kj_1}} + \cdots + w_{d_{kj_h}}, k < j_1 < \cdots < j_h \le \min\{n, U_{kh}\}\right\}. \end{gathered} \tag{23}$$

The best approximate solution corresponds to $\max\{p_{s2} - p_{s1} \mid w_{s2} \le w_{s1}\}$, so the calculation is $O\ (n\ln n)$ time when h equals to 1. A better solution is generated if the profit sum of h items selected in the knapsack is less than that of h items not selected in the knapsack, and the capacity constraint still holds at the same time. On the other hand, the calculation decreases via variable reduction in practice.

In Example 1, let the approximate solution x_a equal to $z^{(19)}$ and objective value f_a equal to $f(z^{(19)})$ firstly, and then the approximate solution x_a is updated by the best solution $y_a^{(19)}$ in $N(z^{(19)}, 2)$ if possible.

$$\begin{gathered} f\left(y_a^{(19)}\right) = 486.1884 > f_a = 473.1749 \\ yd_{19i} = \begin{cases} z_{d_{19i}}, i \le 30, i \ne 12, 22 \\ 1 - z_{d_{19i}}, i = 12, 22 \end{cases} \\ f_a \leftarrow f\left(y_a^{(19)}\right), x_a \leftarrow y_a^{(19)}. \end{gathered} \tag{24}$$

Similarly, x_a is replaced by the best solution $y_a^{(19)}$ in $N(z^{(19)}, 4)$ if possible.

$$\begin{gathered} f\left(y_a^{(19)}\right) = 486.9426 > f_a = 486.1884 \\ yd_{19i} = \begin{cases} 1, i = 1, 2, \ldots, 21, i \ne 13, 17 \\ 0, i = 13, 17, 22, \ldots, 30 \end{cases} \\ f_a \leftarrow f\left(y_a^{(19)}\right), x_a \leftarrow y_a^{(19)}. \end{gathered} \tag{25}$$

There is no better solution of KP in $N(z^{(19)}, 6)$ and $N(z^{(19)}, 8)$. We get an approximate solution $y_a^{(19)}$ with 19 items selected and the approximate objective value is 486.9426.

3.3. Approximation Algorithm of KP.

Let x_a describe the best approximate solution to KP in (26).

```
Step 1 Input the capacity $c$, size $n$, weights $w_i$ and profits $p_i$ of items.
Step 2 Obtain the critical item $s$ by (10).
Step 3 Let $= (0, 0, ..., 0), f_a = 0, k_1 = s - 1, k_2 = s$.
Step 4 Obtain the upper bound $B_k$ and its solution $u^{(k)}$ $(k = k_1, k_2)$ by (8).
Step 5 If $B_{k_1} \geq B_{k_2}$, then $k = k_1$. Otherwise, $k = k_2$.
Step 6 If $B_k \leq f_a$, then exit.
Step 7 Obtain the initial solution $z^{(k)}$ by (18).
Step 8 Obtain the best solution $y_a^{(k)}$ in the neighborhood $N(z^{(k)}, 2h)(h = 1, 2, 3, 4)$ and its objective value $f(y_a^{(k)})$ by Algorithm 1.
Step 9 If $f_a < f(y_a^{(k)})$, then $f_a \leftarrow f(y_a^{(k)}), x_a \leftarrow y_a^{(k)}$.
Step 10 If $k = k_1$, then $k_1 \leftarrow k_1 - 1$, get $B_{k_1}$ and its solution $u^{(k_1)}$ by (8); else, $k_2 \leftarrow k_2 + 1$, get $B_{k_2}$ and its solution $u^{(k_2)}$ by (8). Go to Step 5.
```

Algorithm 2: Approximate Algorithm for (KP).

Table 5: 0/1 Knapsack problems.

No.	(w_i, p_i) $(i = 1, 2, \ldots, 10000)$	C
1	$(i, \sqrt{i})$	$2.6e+07$
2	$(i, \ln i + 1)$	$2.7e+07$
3	$(i, \tan(\pi i/30000))$	$2.8e+07$
4	$(i, 100 + 10^{-4}i^2)$	$2.9e+07$
5	$(i, 0.01i - 5600)^2 + 1)$	$3.0e+07$
6	$(i, i + \tan(\pi i/30000))$	$3.1e+07$
7	$(i, \tan(\pi i/30000) + 0.001i^2)$	$3.2e+07$
8	$(i, 5000 + 10^{-6}i^3)$	$3.3e+07$
9	$(i, \tan(\pi i/30000) + 10^{-6}i^3)$	$3.4e+07$
10	$(i, \arctan(0.01i) + 1)$	$3.5e+07$
11	$(i, 3 - \ln\cos(10^{-4}i))$	$3.6e+07$
12	$(i, \arctan(10^{-4}i^2) + 1)$	$3.7e+07$
13	$(i, \sin e^{10^{-4}i} + 10^{-4}i^2)$	$3.8e+07$
14	$(i, \sin e^{10^{-4}i} + 10^{-2}i)$	$3.9e+07$
15	$(i, (1/i))$	$4.0e+07$
16	$(i, (e^{10^{-4}i} + e^{-10^{-4}i}/\sin(10^{-4}i)))$	$4.1e+07$
17	(i, i^2)	$4.2e+07$
18	$(i, i + 2 + \varepsilon_i), \varepsilon_i \sim [0, 0.003]$	$4.3e+07$

Table 6: Approximate value and optimization value of 0/1 Knapsack problems.

No.	Approximate value	Optimal value
1	$4.082159596963170e+05$	$4.082159596963174e+05$
2	$6.541219347026785e+04$	$6.541219347026781e+04$
3	$4.102160742890480e+03$	$4.102160742890481e+03$
4	$2.462154058619999e+07$	$2.462154058620000e+07$
5	$8.338180285999998e+08$	$8.338180286000001e+08$
6	$3.100447382723671e+07$	$3.100447382723672e+07$
7	$2.613912845586523e+07$	$2.613912845586526e+07$
8	$2.232528710024500e+09$	$2.232528710024500e+09$
9	$2.244286940564110e+09$	$2.244286940564110e+09$
10	$2.096538351264514e+04$	$2.096538351264514e+04$
11	$2.655627458756987e+04$	$2.655627458756988e+04$
12	$2.189122733748200e+04$	$2.189122733748204e+04$
13	$2.942021969836180e+07$	$2.942021969836181e+07$
14	$3.981146835799920e+05$	$3.981146835799920e+05$
15	9.675897955855723	9.675897955855723
16	$1.997554888239176e+05$	$1.997554888239177e+05$
17	$3.120269950000000e+11$	$3.120269950000000e+11$
18	$4.301855990805434e+07$	$4.301855990837396e+07$

Table 7: Relative average error to the upper bound (ppm).

R	No. 1	No. 2	No. 3	No. 4	No. 5	No. 6	No. 7	No. 8	No. 9	No. 10
10^3	5.6579	2.7082	1.6469	0.4003	0	1.2883	0.0573	100.5351	41.0547	24.8855
10^4	0.3227	1.5208	0.7258	0.0157	0.0132	0.4878	0	46.7007	0.131	21.4483
10^5	0.0591	1.1447	0.4877	0.0141	0.0444	0.2357	0.0044	43.6594	0.0148	19.1698
10^6	0.0577	1.1524	0.6731	0.0185	0.0328	0.246	0.0047	50.0126	0.0014	19.21
10^7	0.0510	1.205	0.5800	0.0229	0.0192	0.1975	0.0034	49.9764	0.0001	14.8858

Table 8: Relation between (p_i-1.0003121w_i) and the optimal solution (x_i^{opt}) in Example 1.

i	w_i	p_i	(p_i/w_i)	e_j	$p_i - 1.0003121w_i$	d_{19l}	x_i^{opt}	x_i^{ei}	$z_i^{(19)}$
2	4.877	5.8853	1.20674595	2	1.006714385	1	1	1	1
1	1.7856	2.7924	1.563844086	1	1.006219464	2	1	1	1
4	7.0943	8.0992	1.141648929	4	1.002593494	3	1	1	1
7	13.9249	14.9319	1.072316498	7	1.002472723	4	1	1	1
5	7.8807	8.885	1.127437918	5	1.001737819	5	1	1	1
3	6.3493	7.3521	1.15793867	3	1.000735709	6	1	1	1
10	24.2688	25.2767	1.04153069	10	1.000009703	7	1	1	1
6	8.5593	9.5616	1.117100697	6	0.999517192	8	1	1	1
9	21.0881	22.0925	1.047628757	9	0.997543816	9	1	1	1
16	37.1566	38.1662	1.027171485	16	0.997519609	10	1	1	1
8	19.6114	20.6148	1.051164119	8	0.997023922	11	1	1	1
13	32.7739	33.7797	1.030689054	13	0.995144517	12	1	1	1
18	39.6104	40.6175	1.025425141	18	0.994221827	13	0	1	1
11	27.3441	28.3472	1.03668433	11	0.994209859	14	1	1	1
27	47.8753	48.8848	1.02108603	26	0.993934735	15	1	0	1
20	40.7362	41.7432	1.024720028	20	0.993755806	16	1	0	1
21	42.4565	43.463	1.023706617	21	0.9926965	17	0	0	1
17	37.887	38.892	1.026526249	17	0.992682141	18	1	1	1
26	47.8583	48.866	1.021055909	27	0.992140262	19	1	0	0
14	32.787	33.7898	1.030585293	14	0.992140258	20	1	1	1
28	47.9746	48.9822	1.021002781	28	0.99200245	21	1	0	0
24	45.7868	46.7932	1.021980134	23	0.99151375	22	0	0	0
12	31.618	32.6189	1.031656019	12	0.990620324	23	0	1	0
15	33.9368	34.9379	1.029498951	15	0.990066434	24	0	1	0
19	40.014	41.0159	1.025038736	19	0.988890608	25	0	1	0
30	48.5296	49.5335	1.020686344	30	0.988122008	26	0	0	0
25	46.6997	47.7013	1.021447675	25	0.986416947	27	0	0	0
23	45.6688	46.6693	1.021907736	24	0.985652114	28	0	0	0
22	45.2896	46.2899	1.022086748	22	0.9855754	29	0	0	0
29	48.2444	49.2448	1.020736085	29	0.984714732	30	0	0	0

* $p_{d_{19l}} - 1.0003121w_{d_{19l}} \ge p_{d_{19l+1}} - 1.0003121w_{d_{19l+1}}$, $(p_{e_j}/w_{e_j}) \ge (p_{e_{j+1}}/w_{e_{j+1}})$, $j, l = 1, 2, \cdots, 29$. (x_i^{ei}) is an initial solution constructed in descending order of efficiency.

$$x_a = \underset{k}{\operatorname{argmax}}\, f\left(y_a^{(k)}\right) = \underset{k}{\operatorname{argmax}} \max_{1 \le h \le 4} \left\{ f\left(y^{(k)}\right) \middle| y^{(k)} \in N\left(z^{(k)}, 2h\right) \right\}. \tag{26}$$

We can obtain the approximate solution x_a of KP by Algorithm 2.

In Example 1, we obtain $f_a = 486.9426$ when $k = 19$.

$$B_{20} = 0, B_{18} = 485.9582 < f_a = 486.9426. \tag{27}$$

The best approximate solution $x_a = (x_{d_{19i}})$ for KP and its objective value are as follows:

$$f_a = 486.9426. \tag{28}$$

It may be proved that x_a is the optimal solution to KP in Example 1.

3.4. Calculation Analysis. Approximation algorithms for KP based on upper bound of E-kKP runs in polynomial time. Calculation for the upper bound of E-kKP is less than O ($n\ln n$) for a given E-kKP. We obtain the approximate solution to KP on the basis of the upper bound of E-kKP where k is an integer close to s. For a given k, the initial solution is determined by (13). It takes $O(n\ln n)$ time to develop the initial solution in the neighborhood $N(z^{(k)}, h)$ $(h = 1, 2, \ldots, 4)$ defined by (21). Hence, it takes at most $O(n\ln n)$ time to achieve the approximate solution to E-kKP. Generally, we obtain the best approximate solution for KP among the approximate solutions for E-kKP where k is near the critical item s. Hence, the calculation for approximate solution to KP is less than $O(n\ln n)$.

In Algorithm 1, we search for the best solution in $N(z^{(k)}, h)$ defined by (21). In order to develop the approximate solution quickly, the weight sum of h items is sorted in ascending order firstly, and then the profit sum of h items selected in the knapsack is compared with the profit sum of h items not selected. So the storage needed in Algorithm 1 is $O(n)$ when h is fixed at 1, 2, 3, or 4. In Algorithm 2, we have the best solution to KP by comparison to the best approximate solution to E-kKP where k is an integer close to s. So the storage of the approximate algorithm for KP is $O(n)$.

3.5. Accuracy Analysis. All feasible solutions to KP are in the search scope of approximation algorithm with changing k and h. So approximation algorithm for KP is an ε-approximation algorithm. From one aspect, weights, profits, capacity and size of KP have influence on the accuracy of an approximation algorithm. From another aspect, the search scope of the approximation algorithm has influence on its accuracy as well. Better solution is usually with more calculation. The exact algorithm for KP may be explored on the basis of the branch and bound algorithm here. Intensive research will be carried out in the future.

4. Computational Experiments

By Algorithm 2, we get the approximate solutions to KP listed in Table 5. The results are listed in Table 6. The optimal value listed in Table 6 is carried out by combo [24].

From Table 6, we find that the approximate solutions are almost the optimal solutions in 18 instances. It implicates that the approximate algorithm proposed here achieves high precision in solving KP.

In Table 7 lists the experimental results of the solutions for KP listed in Table 2 by Algorithm 2. The upper bound listed in Table 4 is carried out by equation (11) in Section 2.

From Table 7, we see that the relative average error of 100 instances is almost less than 0.0001. The experiment result shows that the approximate algorithm proposed here can achieve high accuracy.

From Table 8, we find that the upper bound of E-kKP makes key contribution in FPTAS. Firstly, the initial solution constructed on the basis of the solution to the upper bound of E-kKP is similar with the optimal solution. For example, there are only 4 elements different between the initial solution and the optimal solution to E-kKP where k equals to 19, while there are 8 elements different between the initial solution constructed by efficiency and the optimal solution to KP in Table 8. Secondly, the differences between the initial solution and the solution to the upper bound of E-kKP are near to the item d_{kk}. It is to say, we search the optimal solution to KP in a core with small size on the basis of the solution to the upper bound of E-kKP. While the optimal solution to KP in a core with large size on the basis of the solution to the upper bound of Dantzig. So, algorithms proposed here is easy to get approximate solution to KP.

Furthermore, no better solution of E-kKP exists when the objective value f_a of the approximate solution achieved before is larger than the upper bound B_k. So we only search solution of E-kKP with upper bound larger than f_a which is in the neighborhood of the optimal solution to the upper bound of E-kKP. The candidate strategy makes the search in polynomial time and the solution with high accuracy in Algorithm 1. The upper bound of E-kKP plays important role in decreasing calculation of the approximate solution of E-kKP in Algorithm 2 as well.

5. Conclusion

It is still difficult to obtain the exact solution for large scale 0-1 knapsack problem directly. Here a fast polynomial approximate solution is proposed on the basis of the upper bound for KP. The exact solution to KP is in the neighborhood of the solution to the upper bound for E-kKP. Therefore, it is possible to find an approximation with high accuracy in the neighborhood of the solution to the upper bound for E-kKP where k is near to the critical item s. All in all, as the basis of fast exact algorithm for KP, it is important to obtain an approximate solution and the upper bound for KP. In order to obtain a fast exact solution to KP, more intensive research on variables reduction need be conducted in the future.

Data Availability

All data inside the manuscript have been specified clearly in the manuscript.

Conflicts of Interest

The authors declare that they have no known conflicts financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] S. Sahni, "Approximate algorithms for the 0/1 knapsack problem," *Journal of the ACM*, vol. 22, no. 1, pp. 115–124, 1975.

[2] O. H. Ibarra and C. Kim, "Fast approximation algorithms for the knapsack and sum of Subset problems," *Journal of the ACM*, vol. 22, no. 4, pp. 463–468, 1975.

[3] A. Caprara, H. Kellerer, U. Pferschy, and D. Pisinger, "Approximation algorithms for knapsack problems with cardinality constraints," *European Journal of Operational Research*, vol. 123, no. 2, pp. 333–345, 2000.

[4] D. Zou, L. Gao, S. Li, and J. Wu, “Solving 0–1 knapsack problem by a novel global harmony search algorithm,” *Applied Soft Computing*, vol. 11, no. 2, pp. 1556–1564, 2011.

[5] Y. Feng, G.-Ge Wang, and X.-Z. Gao, “A novel hybrid cuckoo search algorithm with global harmony search for 0-1 knapsack problems,” *International Journal of Computational Intelligence Systems*, vol. 9, no. 6, pp. 1174–1190, 2016.

[6] X. Zhang, S. Huang, Y. Hu, Y. Zhang, S. Mahadevan, and Y. Deng, “Solving 0-1 knapsack problems based on amoeboid organism algorithm,” *Applied Mathematics and Computation*, vol. 219, no. 19, pp. 9959–9970, 2013.

[7] Y. Feng, Ke Jia, and Y. He, “An Improved Hybrid Encoding Cuckoo Search Algorithm for 0-1 Knapsack Problems,” *Computational Intelligence and Neuroscience*, vol. 2014, 2014.

[8] Y. Feng, G.-Ge Wang, S. Deb, M. Lu, and X.-J. Zhao, “Solving 0-1 Knapsack Problem by a Novel Binary Monarch Butterflfly Optimization,” *Neural Computing & Applications*, vol. 28, 2015.

[9] S. F. Razavi and H. Sajedi, “Cognitive discrete gravitational search algorithm for solving 0-1 knapsack problem,” *Journal of Intelligent and Fuzzy Systems*, vol. 29, no. 5, pp. 2247–2258, 2015.

[10] Y. Zhou, L. Li, and M. Ma, “A Complex-Valued Encoding Bat Algorithm for Solving 0–1 Knapsack Problem,” *Neural Processing Letters*, vol. 44, 2015.

[11] Y. Zhou, Z. Bao, Q. Luo, and S. Zhang, “A Complex-Valued Encoding Wind Driven Optimization for the 0-1 Knapsack Problem,” *Applied Intelligence*, vol. 46, 2016.

[12] W.-L. Liu, Y.-J. Gong, W.-N. Chen, Z. Liu, H. Wang, and J Zhang, “Coordinated charging scheduling of electric vehicles: a mixed-variable differential evolution approach,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 12, pp. 5094–5109, 2020.

[13] S. Zhou, L. Xing, X. Zheng, N. Du, L. Wang, and Q. Zhang, “A self-adaptive differential evolution algorithm for scheduling a single batch-processing machine with arbitrary job sizes and release times,” *IEEE Transactions on Cybernetics*, vol. 51, no. 3, pp. 1430–1442, 2021.

[14] F. Zhao, X. He, and L. Wang, “A two-stage cooperative evolutionary algorithm with problem-specific knowledge for energy-efficient scheduling of No-wait flow-shop problem,” *IEEE Transactions on Cybernetics*, vol. 51, no. 11, pp. 5291–5303, 2021.

[15] F. Zhao, L. Zhang, J. Cao, and J Tang, “A cooperative water wave optimization algorithm with reinforcement learning for the distributed assembly no-idle flowshop scheduling problem,” *Computers & Industrial Engineering*, vol. 153, no. 10, Article ID 107082, 2021.

[16] F. Zhao, S. Di, J. Cao, and J. Tang, “A novel cooperative multistage hyper-heuristic for combination optimization problems,” *Complex System Modeling and Simulation*, vol. 1, no. 2, pp. 91–108, 2021.

[17] D. Pisiginger, “An Expanding Core Algorithm for the Exact 0-1 Knapsack Problem,” *European Journal of Operational Research*, vol. 87, 1995.

[18] D. Pisiginger, *Core problems in knapsack algorithms*, 1999.

[19] Z. Wang, Li Gao, and H. Wang, “A hybrid one dimensional optimization,” in *Proceedings of the 4th International Conference on Electronics, Communications and Networks (CECNET IV)*, pp. 409–414, Beijing, China, December 2014.

[20] D. Pisinger, “Where are the hard knapsack problems?” *Computers & Operations Research*, vol. 32, no. 9, pp. 2271–2284, 2005.

[21] S. Martello and P. Toth, “An upper bound for the zero-one knapsack problem and a branch and bound algorithm,” *European Journal of Operational Research*, vol. 1, no. 3, pp. 169–175, 1977.

[22] S. Martello and P. Toth, “A new algorithm for the 0-1 knapsack problem,” *Management Science*, vol. 34, no. 5, pp. 633–644, 1988.

[23] S. Martello and P. Toth, “Upper bounds and algorithms for hard 0-1 knapsack problems,” *Operations Research*, vol. 45, no. 5, pp. 768–778, 1997.

[24] S. Martello, D. Pisinger, and P. Toth, “Dynamic programming and strong bounds for the 0-1 knapsack problem,” *Management Science*, vol. 45, no. 3, pp. 414–424, 1999.