International Journal of Innovative
Computing, Information and Control
Volume **14**, Number **5**, October **2018**

 ISSN 1349-4198
pp. 1833–1854

A HEURISTIC ALGORITHM BASED ON EXPECTATION EFFICIENCY FOR 0-1 KNAPSACK PROBLEM

Lingling Zhang[1] and Jianhui Lv[2]

[1]Department of Computer Science and Technology
Beihua University
No. 3999, East Binjiang Road, Jilin 132013, P. R. China
373752782@qq.com

[2]College of Computer Science and Engineering
Northeastern University
No. 3-11, Wenhua Road, Heping District, Shenyang 110819, P. R. China
lvjianhui2012@163.com

Received December 2017; revised April 2018

Abstract. *0-1 Knapsack Problem (KP01) is a classical NP-hard problem and it plays an important role in real-world applications. In this paper, we propose a heuristic algorithm based on Expectation Efficiency, Increasing rate of profit and Improved greedy strategy (EEII) to get approximate solution of KP01. (i) Some items are selected with greedy strategy and put into knapsack. (ii) We propose expectation efficiency strategy and increasing rate of profit strategy to determine which items are loaded into knapsack from the remaining items. (iii) We limit the size of items to improve calculation efficiency of expectation efficiency and increasing rate of profit. (iv) An improved greedy strategy is devised to solve the special KP01 from the perspective of variation coefficient. The computational experiments are done by comparing EEII with (i) Greedy Approximate Algorithm (GAA) and (ii) Greedy Degree and Expectation Efficiency (GDEE) algorithm, and the results show that EEII has good performance to solve KP01.*

Keywords: Heuristics, Combinatorial optimization, Expectation efficiency, Increasing rate of profit, 0-1 knapsack problem

1. Introduction.

1.1. **Background.** Knapsack Problem (KP) is a classical NP-hard problem [1] and it is very unlikely that a polynomial algorithm can be designed. KP was proposed by Dantzig [2] in 1950s, as a famous subset problem, and it has been studied by many related researchers during more than half a century. As we know, KP has played an important role in real-world applications such as combinatorial mathematics, cryptology, operational research and scheduling, and it is usually divided into Separable KP (SKP), 0-1 KP (KP01), Multi-objective KP (MKP) and Hybrid KP (HKP). In particular, KP01 is one of the most common KPs and others can be translated into KP01 to be solved. Mathematically, KP01 is described as follows.

$$\begin{cases} \text{Maximize } optp = \sum_{i=1}^{n} p_i x_i \\ \text{Subject to } \sum_{i=1}^{n} w_i x_i \leq M, \end{cases} \tag{1}$$

DOI: 10.24507/ijicic.14.05.1833

where w_i is the weight of item i, p_i is the profit of item i, M is the capacity of knapsack, and *optp* is the objective function value. In addition, $x_i \in \{0, 1\}$, if x_i is 1, item i is in knapsack; otherwise, item i is not in knapsack. Consequently, the ultimate goal of KP01 is to find the solution $\mathbf{X}^* = (x_1, x_2, \ldots, x_n)$ from n items with $\mathbf{W} = (w_1, w_2, \ldots, w_n)$ and $\mathbf{P} = (p_1, p_2, \ldots, p_n)$ which makes *optp* reach maximum.

1.2. **Related work.** In recent years, a number of efficient methods have been proposed for the solution of KP01. From the perspective of whether the solution is exact, these methods are usually classified into traditional exact ones and heuristic approximate ones.

1.2.1. *Exact methods.* Regarding the traditional exact methods, Dynamic Programming Algorithm (DPA) and Branch-and-Bound Algorithm (BBA) are two classical representatives [3]. In particular, Toth who was the pioneer on this matter proposed DPA to solve KP01 in 1980 [4]. Besides, some researchers also leveraged DPA to solve Bi-objective KP01 (BKP01). For example, in [5], a fully polynomial time approximation algorithm was proposed to solve BKP01 by making full use of dominance relations in DP. In [6], a new DPA was proposed for the solution of BKP01 by combining two state reduction techniques (i.e., generating a backward reduced-state DP space and reducing the number of states further). In [7], two new DPAs were proposed to find exact Pareto frontier for BKP01.

However, during the process of using DPA, the temporary elements are stored as a matrix, which consumes large time as well as storage space. Compared with DPA, these elements are stored in a tree structure when BBA is used, which saves more computation time and storage space. With respect to BBA, the first one to exact solution of KP01 was proposed by Kolesar [8]. Thereafter, Horowitz and Sahni presented BBA based on computing partitions [9]; Martello and Toth designed BBA by adopting particular tree-search technique [10]; Pisinger proposed BBA based on core idea [11]. Similar to DPA, in [12], BBA was also used to solve BKP01. In order to exploit the advantages of DPA and BBA, some hybridizations were proposed, such as [13,14].

In spite of obtaining the optimal solution of DPA and BBA, the convergence speed is so slow that the optimal solution cannot be obtained within the acceptable time in terms of large size KP01. For that reason, a lot of heuristic methods have been proposed to find approximate solution of KP01.

1.2.2. *Approximate methods.* The heuristic approximate methods are usually divided into two categories, that is, intelligent heuristics inspired from the natural phenomenon and general heuristics generated from the methodology.

About the intelligent heuristic methods, in [15], genetic algorithm and particle swarm optimization were combined to solve KP01. In [16], a hybrid of rough set and genetic algorithm was proposed by selecting the crossover points randomly. In [17], a chemical reaction optimization which had good searching ability with greedy strategy for KP01 was proposed. In [18], an algorithm based on amoeboid organism was proposed. In [19], an effective grasp and tabu search for KP01 was proposed. In [20], a simplified binary artificial fish swarm algorithm was proposed. In [21], a novel binary social spider algorithm was proposed, which was composed of discrete process and constraint handling process. Furthermore, there are a few intelligent optimization approaches to solve BKP01. For example, in [22], an adaptive population multi-objective quantum evolutionary algorithm was proposed. In [23], an algorithm based on self-assembly of DNA tiles was proposed. Although the intelligent heuristics can solve KP01 effectively, they have to perform the complex process of iteration. In particular, it requires to set different number of populations for different examples. In other words, the intelligent heuristic methods are not

usually adaptive to the optimization of engineering problem. Given these considerations, the general heuristics have attracted much more attention from researchers.

Regarding the general heuristic methods, in [24], the modified DPA and limited BBA were presented. In [25], an iterative scheme based on the dynamic fixation of variables was proposed. In [26], an α-approximation algorithm was proposed to solve KP01. In [27], a heuristic approach based on core problem was proposed. In [28], a conic approximation method was proposed. However, [24,25-28] still had high time complexity and could not obtain the optimal solution within the linear time. To this end, in [29], the expectation efficiency theory based on economic principle was first proposed to solve KP01, which showed good performance and low time complexity. In fact, [29] still had some limitations to be improved and enhanced, which motivated our design.

1.3. **Contributions.** This paper improves the previous work [29] and presents a new heuristic algorithm based on Expectation Efficiency, Increasing rate of profit and Improved greedy strategy (EEII) to get approximate solution of KP01. The major contributions are summarized as follows.

- We design a threshold determination algorithm with two variable constraint parameters to put some items into knapsack by only considering the weight of item, instead of considering both the weight and profit of item in [29], which can decrease the complexity.
- We propose an expectation efficiency model and an increasing rate of profit model inspired by economics, and their combination is used to determine which items are loaded into knapsack from the remaining items, instead of only considering expectation efficiency model in [29], which can enhance the obtaining of optimal solution.
- We design a size limitation algorithm which limits the range of the best profit to improve calculation efficiency of expectation efficiency model and increasing rate of profit model. However, the size limitation is not designed in [29]. In particular, [29] introduces the dynamic expectation efficiency model and thus decreases the overall performance to some extent brought by the iterative phenomenon.
- We design a size expansion algorithm to modify the limited size properly; in particular, we design an improved greedy strategy to solve the special KP01, in which the variation coefficient of elements is considerably large. However, the two schemes are not designed in [29].

The rest of this paper is organized as follows. Section 2 presents EEII ideas. The algorithm design is introduced in Section 3. Experimental results are reported in Section 4. Finally, Section 5 concludes this paper.

2. Algorithm Ideas.

2.1. **Greedy strategy.** Dantzig [2] proposed a greedy algorithm to get approximate solution of KP01, and his version rearranged items in descending order of profit per unit of weight, i.e., $r_i = p_i/w_i$. In this paper, r_i is called as the cost-performance ratio of item i. Although the got solution with greedy algorithm is uncertain, the operation of rearranging items according to cost-performance ratio in descending order is a nice method. Based on this, both DPA and BBA rearrange items according to cost-performance ratio in descending order before performing their unique operations. That is to say, DPA and BBA also adopt the idea of greedy strategy to some extent. Thus, leveraging the idea of greedy strategy for the solution of KP01 is important. Here, we clarify that the following statements concerning the design of EEII algorithm are based on the fact that items have been rearranged according to cost-performance ratio in descending order, and the rearranged weight and profit are denoted by $\mathbf{W}'$ and $\mathbf{P}'$ respectively.

However, greedy strategy only obtains the local optimal solution rather than the optimal solution; in other words, *optp* is unlikely equal to the best profit. In fact, the first few elements of the optimal solution are the same as those of the solution with greedy strategy. For example, assume that the optimal solution is $(\underline{\mathbf{1, 1, 1, 1,}}\, 0, 1, 1, 0)$ while the solution with greedy strategy is $(\underline{\mathbf{1, 1, 1, 1,}}\, 1, 1, 0, 0)$. We observe that the first 4 elements $(\underline{\mathbf{1, 1, 1, 1}})$ are common. In other words, the first 4 items can be loaded into knapsack and be as an important part of the optimal solution. Based on this, we put the first 4 items into knapsack, and then manage the remaining 4 items with another method. Inspired by the above, we put the first few items into knapsack with greedy strategy, and then use another method to load some items from the remaining items into knapsack. It is obvious that KP01 can be converted into a problem for managing the remaining items.

The EXample 1 (EX1) is given in Table 1 to illustrate the idea of putting the first few items into knapsack, where $\mathbf{R} = (r_1, r_2, \ldots, r_n)$. If greedy strategy is used to solve EX1, then $\mathbf{X}^* = (\underline{\mathbf{1, 1, 1,}}\, 0, 0, 0)$. In fact, the optimal solution of EX1 is $(\underline{\mathbf{1, 1, 1,}}\, 0, 1, 0)$. Here, the common elements are $(\underline{\mathbf{1, 1, 1}})$, thus the first 3 items should be put into knapsack early while the following is to decide which items are loaded into knapsack from the remaining 3 items.

Table 1. EX1 and EX1-1 examples

Example	M	n	*optp*	**W**	**P**	**R**	$\mathbf{X}^*$
EX1	87	6	159	(12, 20, 25, 31, 30, 62)	(29, 41, 49, 49, 40, 51)	(2.4167, 2.05, 1.96, 1.5806, 1.3333, 0.8226)	(1, 1, 1, 0, 1, 0)
EX1-1	87	6	159	(12, 20, 25, 31, 30, 62)	(29, 41, 49, 52, 40, 51)	(2.4167, 2.05, 1.96, 1.6774, 1.3333, 0.8226)	(1, 1, 1, 0, 1, 0)

2.2. **Expectation efficiency strategy.** The expectation efficiency, a classical theory in economics, was proposed by Kahneman and Tversky [30] in 1979 and its corresponding model could solve uncertain problems (e.g., stock prediction, psychoanalysis and livehood economy) with relatively acceptable expectation results. For solving KP01, its mathematical expression can be described as follows.

$$\eta_i = \frac{r_i}{r_{i-1}}, \tag{2}$$

where η_i means the expectation efficiency value of items i against $(i-1)$ and $\eta_i \in (0, 1]$.

Assume that (2) is considered as determination function to decide which items are loaded into knapsack from the remaining items, and that an item with large expectation efficiency value has high priority. In EX1, the first 3 items are loaded into knapsack, and then (2) is used to determine which items are loaded into knapsack from the remaining 3 items. To calculate η_4, η_5 and η_6, they are 0.8065, 0.8435 and 0.6170 respectively. Since $\eta_5 > \eta_4 > \eta_6$, item 5 has the highest priority to be loaded into knapsack, followed by items 4 and 6. The total weight of items is 87 in case of putting the 5th item into knapsack; however, the total weight of items is larger than the capacity of knapsack (i.e., 87) when the 4th item is loaded into knapsack; thus, only the 5th item is loaded into knapsack. The solution got with (2) is $(1, 1, 1, 0, 1, 0)$ and *optp* is 159. Indeed, the optimal solution of EX1 is also $(1, 1, 1, 0, 1, 0)$, which suggests that (2) may be regarded as determination function.

However, (2) cannot be used to get the optimal solution for all cases. For example, suppose that p_4 in EX1 is 52 not 49, and EX1 changes to EX1-1 in Table 1. To calculate η_4,

η_5 and η_6, they are 0.8558, 0.7949 and 0.6170 respectively. In this case, since $\eta_4 > \eta_5 > \eta_6$, item 4 has the highest priority to be loaded into knapsack, followed by items 5 and 6. However, the total weight of items is larger than the capacity of knapsack (i.e., 87) when the 4th item is loaded into knapsack; thus, only the first 3 items are loaded into knapsack. The solution got with (2) is $(1, 1, 1, 0, 0, 0)$ and *optp* is 119. Indeed, the optimal solution remains $(1, 1, 1, 0, 1, 0)$ and *optp* remains 159. EX1-1 shows that (2) cannot be regarded as determination function directly; thus, (2) needs to be improved.

As a matter of fact, the optimal solution depends on the number of items, the weight of item, the profit of item, the cost-performance ratio of item and the capacity of knapsack. Five examples, i.e., EX2-EX6, are given to verify this idea, as shown in Table 2. Here, for EX2 and EX3, their optimal solutions are different with different M. For EX3 and EX6, their optimal solutions are different with different $\mathbf{W}$ and $\mathbf{P}$. For EX2 and EX5, their optimal solutions are different with different n. For EX2 and EX4, their optimal solutions are different with different r_{17}. The results from EX2-EX6 show that the optimal solution is related to n, w, p, r and M, and the expectation efficiency model can be described as

Table 2. EX2-EX6 examples

Example	M	n	*optp*	$\mathbf{W}$	$\mathbf{P}$	$\mathbf{R}$	$\mathbf{X}^*$
EX2	660	20	821	(36, 27, 19, 30, 24, 32, 17, 35, 9, 47, 90, 99, 78, 75, 52, 89, **12**, 42, 80, 10)	(90, 57, 38, 54, 37, 48, 24, 49, 12, 56, 100, 95, 68, 63, 43, 72, **9**, 30, 56, 1)	(2.5, 2.1111, 2, 1.8, 1.5417, 1.5, 1.4118, 1.4, 1.3333, 1.1915, 1.1111, 0.9596, 0.8718, 0.84, 0.8269, 0.809, **0.75**, 0.7143, 0.7, 0.1)	(1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0)
EX3	680	20	835	(36, 27, 19, 30, 24, 32, 17, 35, 9, 47, 90, 99, 78, 75, 52, 89, 12, 42, 80, 10)	(90, 57, 38, 54, 37, 48, 24, 49, 12, 56, 100, 95, 68, 63, 43, 72, 9, 30, 56, 1)	(2.5, 2.1111, 2, 1.8, 1.5417, 1.5, 1.4118, 1.4, 1.3333, 1.1915, 1.1111, 0.9596, 0.8718, 0.84, 0.8269, 0.809, 0.75, 0.7143, 0.7, 0.1)	(1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1)
EX4	660	20	826	(36, 27, 19, 30, 24, 32, 17, 35, 9, 47, 90, 99, 78, 75, 52, 89, **42**, 42, 80, 10)	(90, 57, 38, 54, 37, 48, 24, 49, 12, 56, 100, 95, 68, 63, 43, 72, **33**, 30, 56, 1)	(2.5, 2.1111, 2, 1.8, 1.5417, 1.5, 1.4118, 1.4, 1.3333, 1.1915, 1.1111, 0.9596, 0.8718, 0.84, 0.8269, 0.809, **0.7857**, 0.7143, 0.7, 0.1)	(1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0)
EX5	660	15	810	(36, 27, 19, 30, 24, 32, 17, 35, 9, 47, 90, 99, 78, 75, 52)	(90, 57, 38, 54, 37, 48, 24, 49, 12, 56, 100, 95, 68, 63, 43)	(2.5, 2.1111, 2, 1.8, 1.5417, 1.5, 1.4118, 1.4, 1.3333, 1.1915, 1.1111, 0.9596, 0.8718, 0.84, 0.8269)	(1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1)
EX6	680	20	843	(27, 36, 75, 42, 48, 32, 60, 12, 70, 90, 82, 120, 18, 24, 78, 95, 38, 135, 65, 56)	(54, 58, 120, 65, 70, 40, 72, 14, 80, 100, 90, 105, 15, 20, 60, 70, 28, 90, 40, 30)	(2, 1.6111, 1.6, 1.5476, 1.4583, 1.25, 1.2, 1.1667, 1.1429, 1.1111, 1.0976, 0.875, 0.8333, 0.8333, 0.7692, 0.7368, 0.7368, 0.6667, 0.6154, 0.5357)	(1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0)

follows.

$$f(n, w, p, r, M) \propto n, w, p, r, M, \tag{3}$$

where $f(n, w, p, r, M)$ is an abstract function to determine which items are loaded into knapsack from the remaining items. Therefore, the concretization of $f(n, w, p, r, M)$ is the key point (see Section 3.3 for details).

2.3. **Increasing rate of profit strategy.** In this section, we introduce another idea, i.e., the increasing rate of profit, to be combined with the expectation efficiency model to manage the remaining items. According to (1), KP01 pays attention to the ultimate objective function value (i.e., the profit of the invested items). To some extent, the item with high profit has large probability to be loaded into knapsack. For example, for two items i ($p_i = 27$) and j ($p_j = 30$), if they can be loaded into knapsack independently (i.e., the current total weight of items is smaller than or equal to M), item i has higher priority than item j. The above statements mean that the profit of item has an important impact on the best profit of KP01. In this paper, however, the profit of item does not be considered as determination model to manage the remaining items directly; instead, the increasing rate of profit of item is introduced. Let *irp* denote it, and the corresponding model can be described as follows.

$$irp \propto p, \tag{4}$$

where *irp* is also an abstract function to determine which items are loaded into knapsack from the remaining items. Therefore, the concretization of *irp* is the key point (see Section 3.4 for details).

The ideas of EEII algorithm are summarized as follows. At first, some items are put into knapsack with greedy strategy. Then, both expectation efficiency model and increasing rate of profit model are used to determine which items are loaded into knapsack from the remaining items.

3. Algorithm Design.

3.1. **Threshold determination.** According to Section 2, the first few items were selected to be put into knapsack and they are never removed from knapsack. Suppose that the number of items loaded into knapsack with greedy strategy is m, and the corresponding total weight is M'. In this way, it is very important to determine m. For convenience, let $m = n/2$ at first. Based on this, suppose that the total weight of items actually loaded into knapsack from m items is M'' when the solution is optimal. It is obvious that $M'' \leq M'$. For example, any item j is considered, here $j \leq m$. If the optimal solution does not include item j, the actual total weight of items from m items is $M'' = M' - w_j$; if the optimal solution includes item j, the actual total weight of items from m items is $M'' = M'$. Two examples, i.e., EX7 and EX8, are given to discuss m, M' and M'', as shown in Table 3. Here, $m = 7$, $M'' = M' = 205$ in EX7 since the optimal solution includes the first 7 items, and $175 = M'' \neq M' = 205$ in EX8 since the optimal solution does not include the 7th item. Thus, $m = n/2$ is acceptable in EX7 while it is unacceptable in EX8. In fact, the setting of m is acceptable when and only when $M'' = M'$.

In this paper, let $m = n/2 \pm \lambda$, here $\lambda \in N_+$, so that $M'' = M'$, the setting of λ is described as follows. Consider φ, α and β, here $\alpha < \beta$. Initially, let φ represent the ratio of the total weight of the first $n/2$ items to M, and $\varphi \in [0, 1]$, where $\varphi = 0$ if and only if $n = 1$, and $\varphi = 1$ if and only if the total weight of the first $n/2$ items is equal to M.

(i) If $\alpha \leq \varphi \leq \beta$, the first $n/2$ items are loaded into knapsack in advance, m is equal to $n/2$ and $\lambda = 0$.

TABLE 3. EX7 and EX8 examples

Example	M	M'	n	$\mathbf{W}$	$\mathbf{P}$	$\mathbf{X}^*$	M''
EX7	480	205	15	$(10, 35, 45, 18, 12,$ $55, 30, 48, 77, 24,$ $90, 90, 65, 28, 84)$	$(20, 70, 80, 30, 20,$ $90, 48, 68, 105, 32,$ $120, 100, 70, 30, 90)$	$(\mathbf{1, 1, 1, 1, 1, 1, 1,}$ $1, 1, 1, 1, 0, 0, 1, 0)$	205
EX8	223	205	15	$(10, 35, 45, 18, 12,$ $55, 30, 48, 77, 24,$ $90, 90, 65, 28, 84)$	$(20, 70, 80, 30, 20,$ $90, 48, 68, 105, 32,$ $120, 100, 70, 30, 90)$	$(\mathbf{1, 1, 1, 1, 1, 1, 0,}$ $1, 0, 0, 0, 0, 0, 0, 0)$	175

(ii) If $\varphi > \beta$, remove item from knapsack in the sequence of items $(n/2-1)$, $(n/2-2)$, ... until the first Constraint Condition (CC1) is met, m is smaller than $n/2$ and $\lambda = n/2-m$.

$$CC1 : \begin{cases} \dfrac{\sum_{i=1}^{j} w_i}{M} \leq \beta \\ \dfrac{\sum_{i=1}^{j+1} w_i}{M} > \beta. \end{cases} \tag{5}$$

(iii) If $\varphi < \alpha$, put item into knapsack in the sequence of items $(n/2+1)$, $(n/2+2)$, ... until CC2 is met, m is larger than $n/2$ and $\lambda = m - n/2$.

$$CC2 : \begin{cases} \dfrac{\sum_{i=1}^{n/2} w_i + \sum_{i=n/2+1}^{k} w_i}{M} < \alpha \\ \dfrac{\sum_{i=1}^{n/2} w_i + \sum_{i=n/2+1}^{k+1} w_i}{M} \geq \alpha. \end{cases} \tag{6}$$

The threshold determination algorithm concerning m is described in Algorithm 1, and the related operations are depicted in Figure 1.

Algorithm 1: Threshold determination algorithm

Input: $\mathbf{W}$, $\mathbf{P}$, n and M
Initially: $wq = 0$;
for $i = 1$ to $n/2$, **do**
 $wq = wq + w_i$;
end for
Compute $\varphi = wq/M$;
if $\varphi > \beta$ and CC1 met, **then**
 $m = j$;
else if $\varphi < \alpha$ and CC2 met, **then**
 $m = k$;
else
 $m = n/2$;
end if
Output: m

3.2. **Size limitation.** When the first m items are put into knapsack, the number of remaining items is $(n-m)$. Each one from $(n-m)$ items corresponds to one expectation efficiency value and one increasing rate of profit value; thus, $(n-m)$ expectation efficiency values and $(n-m)$ increasing rate of profit values need to be calculated. However, not all

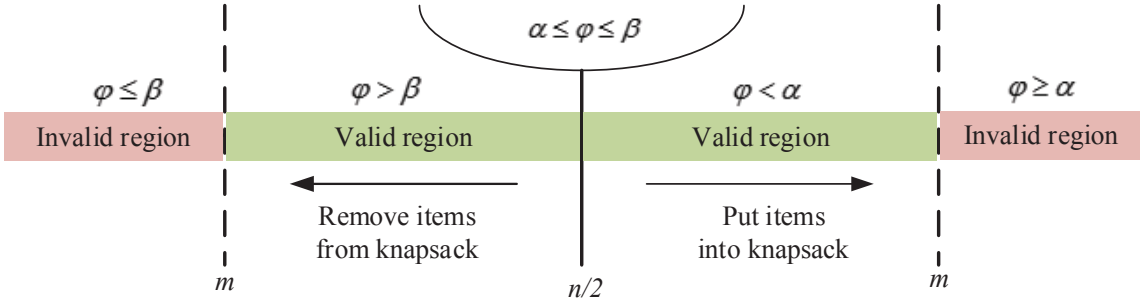

FIGURE 1. Illustration of Algorithm 1

these $(n-m)$ items can be loaded into knapsack. Given this, a scheme was proposed to limit the size of items in order to improve calculation efficiency in terms of the remaining $(n-m)$ items, i.e., reducing computation times. For example, 10 expectation efficiency values and 10 increasing rate of profit values need to be calculated in case of $n = 20$ and $m = 10$. However, if the last 2 items are unlikely loaded into knapsack, that is, the optimal solution does not include the last 2 items, the size is limited to 18; in this case, we only need to calculate 8 expectation efficiency values and 8 increasing rate of profit values for items 11 to 18.

If greedy strategy is used to solve KP01, the corresponding *optp* is not larger than the best profit. At this moment, if the next item is loaded into knapsack sequentially, it is obvious that the total weight of items is larger than M and the corresponding total profit is larger than *optp*. CC3 is presented to reflect the above statements, as follows.

$$CC3: \begin{cases} \sum_{i=1}^{m} w_i + \sum_{i=m+1}^{Q} w_i < M \\ \sum_{i=1}^{m} w_i + \sum_{i=m+1}^{Q+1} w_i > M. \end{cases} \tag{7}$$

Here, the solution with greedy strategy includes the first Q items while item $(Q+1)$ cannot be loaded into knapsack. Furthermore,

$$optp(Q) = \sum_{i=1}^{m} p_i + \sum_{i=m+1}^{Q} p_i, \tag{8}$$

$$optp(Q+1) = \sum_{i=1}^{m} p_i + \sum_{i=m+1}^{Q+1} p_i, \tag{9}$$

where $optp(Q)$ and $optp(Q+1)$ mean the total profits of the first Q and $(Q+1)$ items respectively, and $optp(Q)$ is the total profit with greedy strategy. Especially when

$$\sum_{i=1}^{m} w_i + \sum_{i=m+1}^{Q} w_i = M, \tag{10}$$

the best profit can be obtained with greedy strategy, and $optp(Q+1)$ is insignificant.

In fact, for any KP01, $optp(Q)$ is the lower bound of *optp* while $optp(Q+1)$ is the corresponding upper bound, which is proved in Theorem 3.1. If so, the size of items is limited to $(Q+1)$ according to (8)-(10), and we only need to calculate $(Q-m+1)$ expectation efficiency values and $(Q-m+1)$ increasing rate of profit values. Then, the size limitation algorithm is described in Algorithm 2.

Algorithm 2: Size limitation algorithm

Input: **W**, **P**, n and M
Initially: $pq = 0$;
while $i \leq Q$ and CC3 met, **do**
$pq = pq + p_i$;
end while
$optp(Q) = pq$;
$optp(Q+1) = pq + p_{Q+1}$;
Output: $Q + 1$

Theorem 3.1. *For $\forall Q$, consider $m \leq Q \leq n$, and $optp(Q) \leq optp < optp(Q+1)$. The related proof is found in Appendix A.*

3.3. **Expectation efficiency model.** The expectation efficiency model is used to select and load some other items from the remaining $(Q - m + 1)$ items into knapsack, and the modeling process for (3) is presented as follows. With the items loaded into knapsack one by one, the remaining capacity of knapsack (denoted by T) is gradually reduced and approaches zero eventually. In theory, T always expects to respond to more profit from the remaining $(Q - m + 1)$ items. When the cost-performance ratio of item $(i-1)$ is set as the reference, we get the expectation profit of T, denoted by $optp'$ as follows.

$$T = M - \sum_{k=1}^{m} w_k - \sum_{k=m+1}^{i-1} w_k x_k, \tag{11}$$

$$\frac{p_{i-1}}{w_{i-1}} = \frac{optp'}{M - \sum_{k=1}^{m} w_k - \sum_{k=m+1}^{i-1} w_k x_k} = \frac{optp'}{T}, \tag{12}$$

$$optp' = r_{i-1} T. \tag{13}$$

In general, the difference between $optp'$ and p_i is very large; thus, we consider the difference between the ratio of $optp'$ to $(n - i + 1)$ and p_i in order to balance the related expectation profit. Let Δp represent it, as follows.

$$\Delta p = \frac{r_{i-1} T}{n - i + 1} - p_i. \tag{14}$$

We consider (2), (3) and (14) comprehensively and heuristically, and the concrete expectation efficiency model is denoted by $f_i(n, w, p, r, m, M)$ as follows.

$$f_i(n, w, p, r, m, M) = \frac{r_i}{r_{i-1}} * \frac{\frac{r_{i-1} T}{n-i+1} - p_i}{\frac{T}{n-i+1} - w_i} = \frac{r_i}{r_{i-1}} * \frac{r_{i-1} T - (n - i + 1) p_i}{T - (n - i + 1) w_i}, \tag{15}$$

where $i \in [m + 1, Q + 1]$.

The first m items are loaded into knapsack with Algorithm 1, and the size is limited to $(Q+1)$ with Algorithm 2. Thus, the domain of (15) is $[m+1, Q+1]$. If Algorithm 2 is not used, the domain of (15) is $[m+1, n]$. It is obvious that Algorithm 2 improves calculation efficiency from the domain perspective. The usage of (15) to determine which items are loaded into knapsack from the remaining $(Q - m + 1)$ items is described as follows.

If $T = 0$, it means that the current total weight of items in knapsack is equal to M, which suggests that the solution is optimal and the remaining items must not be loaded into knapsack; meanwhile, (15) should not be performed.

If $T < 0$, check whether expectation efficiency value of item $(Q+1)$ has been obtained: if yes, obtain $(Q - m + 1)$ expectation efficiency values, denoted by $f_{m+1}, f_{m+2}, \ldots, f_{Q+1}$;

meanwhile, (15) should not be performed; otherwise, obtain $(i - m - 1)$ expectation efficiency values, denoted by $f_{m+1}, f_{m+2}, \ldots, f_{i-1}$, and then remove the item with the minimal expectation efficiency value in $f_{m+1}, f_{m+2}, \ldots, f_{i-1}$ in turn until $T \geqslant 0$.

If $T > 0$, perform (15) for the current item and go on performing (15) until $T \leqslant 0$ or $i = Q + 1$.

Furthermore, when (15) is not performed after obtaining expectation efficiency value of item i, put the item with the maximal expectation efficiency value into knapsack. Then, check whether the total weight of items in knapsack is smaller than M: if yes, continually put the item with the secondary maximal expectation efficiency value into knapsack until it cannot accommodate any item; otherwise, the remaining items must not be loaded into knapsack. Finally, obtain the total profit of items in knapsack, denoted by *foptp*, and it is calculated by Algorithm 3.

Algorithm 3: Expectation efficiency algorithm

Input: **W**, **P**, n, m and M
Initially: $x_k = 0$, $m + 1 \leq k \leq Q + 1$ and $foptp = \sum_{k=1}^{m} p_k$;
for $i = m + 1$ to $Q + 1$, **do**
if $T = 0$ or $i = Q + 1$, **then**
Perform (15); **break**;
end if
while $T < 0$, **do**
while $i \neq Q + 1$, **do**
$f_l = \min\{f_{m+1}, f_{m+2}, \ldots, f_{i-1}\}$ $//m + 1 \leq l \leq i - 1$;
$x_l = 0$ //Remove item l from knapsack;
$f_l = +\infty$ $//f_l$ has no chance in the next iteration;
end while
end while
while $T > 0$, **do**
Perform (15);
end while
end for
for $i = m + 1$ to $Q + 1$, **do**
while the total weight of items in knapsack $< M$, **do**
$f_\tau = \max\{f_{m+1}, f_{m+2}, \ldots, f_{Q+1}\}$ $//m + 1 \leq \tau \leq Q + 1$;
Put item τ into knapsack;
$f_\tau = 0$ $//f_\tau$ has no chance in the next iteration;
$foptp = foptp + p_\tau$;
end while
end for
Output: *foptp*

3.4. **Increasing rate of profit model.** The increasing rate of profit model is combined with the expectation efficiency model to determine which items are loaded into knapsack from the remaining $(Q - m + 1)$ items; thus, the concretization of (4) needs to be done. Let $irp_i(p)$ denote the increasing rate of profit for item i, as follows.

$$irp_i(p) = \frac{p_i}{\sum_{k=1}^{i} p_k}, \tag{16}$$

where $i \in [m + 1, Q + 1]$.

The usage of (16) to determine which items are loaded into knapsack from the remaining $(Q-m+1)$ items is described as follows. Firstly, obtain the increasing rate of profit of each item from $(m+1)$ to $(Q+1)$ by performing (16), denoted by $irp_{m+1}, irp_{m+2}, \ldots, irp_{Q+1}$. Secondly, first put the item with the maximal increasing rate of profit value into knapsack. Then, check whether the total weight of items in knapsack is smaller than M: if yes, continually put the item with the secondary maximal increasing rate of profit value into knapsack; otherwise, the remaining items must not be loaded into knapsack. Finally, obtain the total profit of items in knapsack, denoted by *irpoptp*, and it is calculated by Algorithm 4.

Algorithm 4: Increasing rate of profit algorithm

Input: **W**, **P**, n, m and M
Initially: $irpoptp = \sum_{k=1}^{m} p_k$;
for $i = m+1$ to $Q+1$, **do**
Perform (16);
Obtain irp_i;
end for
for $i = m+1$ to $Q+1$, **do**
while the total weight of items in knapsack $< M$, **do**
$irp_\tau = \max\{irp_{m+1}, irp_{m+2}, \ldots, irp_{Q+1}\}$ $//m+1 \le \tau \le Q+1$
Put item τ into knapsack
$irp_\tau = 0$ $//irp_\tau$ has no chance in the next iteration
$irpoptp = irpoptp + p_\tau$
end while
end for
Output: *irpoptp*

Algorithms 3 and 4 are both used to determine which items are loaded into knapsack from the remaining $(Q-m+1)$ items, where Algorithm 3 obtains one objective function value *foptp* based on expectation efficiency model and Algorithm 4 obtains one objective function value *irpoptp* based on increasing rate of profit model. Two examples, i.e., EX9 and EX10, are given to illustrate Algorithms 3 and 4, as shown in Table 4, here $\alpha = 0.5$ and $\beta = 0.7$. In Table 4, $foptp = 1284$, $irpoptp = 1263$ and $optp = 1284$ in EX9, in this case, *foptp* is regarded as the best profit; $foptp = 1322$, $irpoptp = 1420$ and $optp = 1420$ in EX10, in this case, *irpoptp* is regarded as the best profit.

Table 4. EX9 and EX10 examples

Example	M	m	n	**W**	**P**	*foptp*	*irpoptp*	*optp*
EX9	700	11	20	(24, 39, 26, 10, 25, 75, 8, 31, 37, 48, 46, 18, 49, 12, 50, 92, 200, 145, 95, 65)	(100, 120, 70, 26, 60, 150, 16, 60, 70, 90, 78, 30, 80, 19, 75, 120, 240, 160, 100, 25)	**1284**	1263	1284
EX10	950	10	20	(9, 36, 67, 29, 58, 19, 98, 205, 39, 96, 45, 79, 49, 88, 124, 105, 38, 113, 300, 160)	(28, 70, 130, 50, 93, 29, 145, 300, 56, 137, 62, 108, 65, 116, 160, 120, 40, 100, 200, 82)	1322	**1420**	1420

3.5. **Size expansion.** However, the combination of Algorithms 3 and 4 cannot obtain the best profit for all cases. If solving EX6, the limitation size is 12. In this case, $optp(11) = 763$, $optp(12) = 868$, $foptp = 763$ and $irpoptp = 763$. However, for EX6, the optimal solution is $(1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0)$ and the best profit is 843. It is obvious that the optimal solution includes items 14 and 15, which suggests that the limitation size (i.e., 12) is unacceptable.

Thus, the limitation size has to be expanded to an appropriate length so that the items included by the optimal solution are not ignored. In this way, item $(Q + 1)$ must not be loaded into knapsack, mathematically,

$$x_{Q+1} = 0. \tag{17}$$

Let $xoptp = \max\{foptp, irpoptp\}$ and XM denote the total weight which $xoptp$ corresponds to. For any item from $(Q + 2)$ to n, it is likely to be loaded into knapsack when the sum of XM and its weight is smaller than or equal to M. Let the size be expanded to S, here $S > Q + 1$, and CC4 is presented to determine S as follows.

$$CC4 : \begin{cases} XM + w_k \leq M \\ Q + 2 \leq k \leq n, \end{cases} \tag{18}$$

$$S = k. \tag{19}$$

When the size is expanded to S, we need to calculate the additional $(S - Q - 1)$ expectation efficiency values and $(S - Q - 1)$ increasing rate of profit values which are obtained with (15) and (16) respectively. However, in this case, the domains of (15) and (16) are $[Q + 2, S]$ rather than $[m + 1, Q + 1]$. The size expansion algorithm is described in Algorithm 5.

Algorithm 5: Size expansion algorithm

Input: **W**, **P**, n, M, and Q
Initially: $x_{Q+1} = 0$;
Perform Algorithm 3;
Perform Algorithm 4;
for $k = n$ to $Q + 2$, **do**
 if $XM + w_k \leq M$, **then**
 break;
 end if
end for
$S = k$;
Output: S

Meanwhile, we obtain one objective function value by combining Algorithm 5 with Algorithm 3, and another objective function value by combining Algorithm 5 with Algorithm 4, denoted by *efoptp* and *eirpoptp* respectively. If the combination of Algorithms 3-5 is used to solve EX6, the size is expanded to 20 and $x_{12} = 0$, $efoptp = 843$, and $eirpoptp = 673$. Here, *efoptp* is regarded as the best profit.

The purpose of this paper is to select one objective function value from $optp(Q)$, *foptp*, *irpoptp*, *efoptp* and *eirpoptp* regardless of $optp(Q + 1)$ due to the inequality to *optp* by Theorem 3.1, and to regard it as the best profit. Namely,

$$optp = \max\big\{optp(Q), foptp, irpoptp, efoptp, eirpoptp\big\}. \tag{20}$$

We propose an algorithm based on Greedy, Expectation efficiency and Increasing rate of profit (GEI), and it is described in Algorithm 6.

Algorithm 6: GEI algorithm

Input: **W**, **P**, n, M, and Q
Initially: $Maxoptp = 0$;
Perform Algorithm 2;
Perform Algorithm 3 from items $m+1$ to $Q+1$;
Perform Algorithm 4 from items $m+1$ to $Q+1$;
Perform Algorithm 5;
Perform Algorithm 3 from items $Q+2$ to S;
Perform Algorithm 4 from items $Q+2$ to S;
Calculate $Maxoptp$ with (20);
Output: $Maxoptp$

Theorem 3.2. *Algorithm 6 runs in $O(n)$. The related proof is found in Appendix B.*

3.6. **Improved greedy strategy.** The data discrete phenomenon can be used to reflect the separation degree of data. In this paper, we pay attention to the cost-performance ratio of each item, i.e., elements of **R**, and use variation coefficient to describe the separation degree of **R**, defined as follows.

$$rr = \frac{\sqrt{\sum_{i=1}^{n}\left((r_i - r_{ave})^2/n\right)}}{r_{ave}}, \tag{21}$$

$$r_{ave} = \frac{1}{n}\sum_{i=1}^{n} r_i, \tag{22}$$

where rr is the variation coefficient of **R**. If rr is relatively small, KP01 is difficult to be solved. Nevertheless, KP01 is easily solved by the combination of expectation efficiency model and increasing rate of profit model. For example, assume that the cost-performance ratio of apple is 1 dollar per kilogram, the cost-performance ratio of orange is 0.3 dollar per kilogram, and the cost-performance ratio of banana is 0.7 dollar per kilogram. In this case, one tends to select more apples to be loaded into his knapsack rather than oranges. In fact, this special KP01 can be solved with an improved greedy strategy, which is described as follows.

If rr has reached a certain level, put items into knapsack one by one until their total weight is larger than or equal to M. Let ρ denote the threshold of the certain level, and consider $rr \geq \rho$, CC5 is presented as follows.

$$CC5: \begin{cases} \sum_{i=1}^{\Omega} w_i < M \\ \sum_{i=1}^{\Omega} w_i + w_{\Omega+k} > M \\ \sum_{i=1}^{\Omega} w_i + \sum_{t=1}^{l} w_{\Omega+j+t} \leq M \\ \sum_{i=1}^{\Omega} w_i + \sum_{t=1}^{l+1} w_{\Omega+j+t} > M, \end{cases} \tag{23}$$

where $k = 1, 2, \ldots, j$. In (23), n items are divided into $[1, \Omega]$, $[\Omega+1, \Omega+j]$, $[\Omega+j+1, \Omega+j+l]$ and $[\Omega+j+l+1, n]$. Among them, the first Ω items can be loaded into knapsack, the items from $(\Omega+1)$ to $(\Omega+j)$ cannot be loaded into knapsack, the items from $(\Omega+j+1)$

to $(\Omega + j + l + 1)$ can be loaded into knapsack, and the last $(n - \Omega - j - l)$ items cannot be loaded into knapsack. If CC5 is met, one objective function value is obtained, denoted by *goptp*, as follows.

$$goptp = \sum_{i=1}^{\Omega} p_i + \sum_{t=1}^{l} p_{\Omega+j+t}. \tag{24}$$

According to the above statements, the improved greedy algorithm is described in Algorithm 7.

Algorithm 7: Improved greedy algorithm

Input: **W**, **P**, n and M
if CC5 met, **then**
Obtain *goptp* with (24);
end if
Output: *goptp*

In this paper, Algorithm 7 is performed when $rr \geq \rho$, and Algorithm 6 is performed when $rr < \rho$, which is called EEII, and it is described in Algorithm 8.

Algorithm 8: EEII algorithm

Input: **W**, **P**, **R** and n
Calculate rr with (21);
if $rr \geq \rho$, **then**
Perform Algorithm 7;
else
Perform Algorithm 6;
end if
if $Maxoptp < goptp$, **then**
$Max = goptp$;
else
$Max = Maxoptp$;
end if
Output: *Max*

Theorem 3.3. *Let ITn represent the number of iterations of Algorithm 8, and* $Q \leq ITn < 3n$*, that is, Algorithm 8 runs in* $O(n)$*. The related proof is found in Appendix C.*

In this section, six sub-algorithms, i.e., threshold determination, size limitation, expectation efficiency, increasing rate of profit, size expansion and improved greedy strategies are designed. Among them, Algorithm 1 is to select and put the first few items into knapsack early, Algorithms 3 and 4 are to determine which items are loaded into knapsack, and Algorithms 2, 5 and 7 are to improve algorithm performance. Meanwhile, $optp(Q)$ and $optp(Q+1)$ are obtained by Algorithm 2; *foptp* is obtained by the combination of Algorithms 2 and 3; *irpoptp* is obtained by the combination of Algorithms 2 and 4; *efoptp* is obtained by the combination of Algorithms 3 and 5; *eirpoptp* is obtained by the combination of Algorithms 4 and 5; and *goptp* is obtained by Algorithm 7. The seven objective function values obtained are shown in Table 5, where "$\surd$" means that the corresponding value can be regarded as the best profit, and "$\times$" means cannot. In fact, EEII is to find a maximum value from them, and the value is the best profit of KP01.

TABLE 5. Seven objective function values in EEII algorithm

Objective function value	$optp(Q)$	$optp(Q+1)$	*foptp*	*irpoptp*	*efoptp*	*eirpoptp*	*goptp*
Algorithm	2	2	2 and 3	2 and 4	3 and 5	4 and 5	7
The best profit	$\surd$	$\times$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$
rr	$< \rho$	$< \rho$	$< \rho$	$< \rho$	$< \rho$	$< \rho$	$\geq \rho$

4. **Simulation Results.** In this section, a large number of experimental studies on KP01 are extensively investigated. Among them, we use 2 examples to illustrate the correctness of EEII algorithm; we use 17 examples (EX13-EX29) from standard test cases libraries [31,32] to evaluate the effectiveness of EEII algorithm, and its performance is compared with Greedy Approximate Algorithm (GAA) and the solution based on Greedy Degree and Expectation Efficiency (GDEE) in [29] with $\alpha = 0.5$, $\beta = 0.7$ and $\rho = 1$. All computational experiments are conducted on a personal computer with the Intel Q8400, 2.66 GHZ CPU, 4G RAM and running on Windows 7.

4.1. EEII algorithm test.

4.1.1. *Example test on condition of* $0 \leq rr < 1$. EX11 is given as follows: $\mathbf{W} = (70, 73, 77, 80, 82, 87, 90, 94, 98, 106, 110, 113, 115, 118, 120)$, $\mathbf{P} = (135, 139, 149, 150, 156, 163, 173, 184, 192, 201, 210, 214, 221, 229, 240)$, $M = 750$, and $n = 15$. In EX11, $rr = 0.0171 < 1$; thus, only Algorithm 6 needs to be performed. Furthermore, $m = 5$ with Algorithm 1, $Q + 1 = 8$ and $optp(Q) = optp(7) = 1302$ with Algorithm 2 (the size is limited to 8), $S = 14$ with Algorithm 5 (the size is expanded to 14). The corresponding values of expectation efficiency and increasing rate of profit are obtained, as shown in Table 6.

TABLE 6. The obtained values on f, *irp*, *ef* and *eirp* for EX11

Item	f	*ef*	*irp*	*eirp*
6	1.9187(2)	1.9187(1)	0.1196	0.1196
7	1.9142	1.9142(2)	0.1329(2)	0.1329(2)
8	1.9212(1)	0	0.1451(1)	0
9	–	1.8950	–	0.1389(1)
10	–	1.8980	–	0.0842
11	–	1.9007(3)	–	0.0863
12	–	1.8894	–	0.1001
13	–	1.8911	–	0.0963
14	–	1.8519	–	0.0632

In Table 6, the values of expectation efficiency and increasing rate of profit of the 8th item are both zero since the 8th item must not be loaded into knapsack when the size is expanded. If the size is limited to 8, *foptp* is obtained by putting the first 5 items, the 8th item, and the 6th item into knapsack in turn (see the second column of Table 6); *irpoptp* is obtained by putting the first 5 items, the 8th item, and the 7th item into knapsack in turn (see the fourth column of Table 6). If the size is expanded to 14, *efoptp* is obtained by putting the first 5 items, the 6th item, the 7th item, and the 11th item into knapsack in turn (see the third column of Table 6); *eirpoptp* is obtained by putting the first 5 items, the 9th item, and the 7th item into knapsack in turn (see the fifth column of Table 6).

TABLE 7. Experimental results of EX11

Objective function value	*optp*(Q)	*foptp*	*irpoptp*	*efoptp*	*eirpoptp*
The total weight	667	692	712	749	707
The total profit	1302	1350	1388	1458	1377
Solution	(1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0)	(1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0)	(1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0)	(1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0)	(1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0)

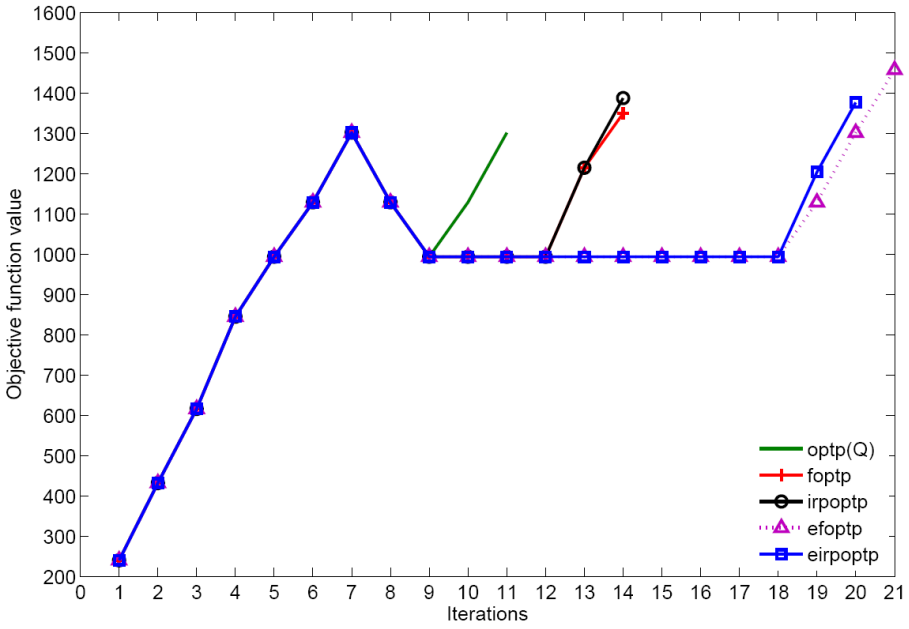

FIGURE 2. The changing process of objective function value for EX11

The total weight, the total profit and the solution which *optp*(Q), *foptp*, *irpoptp*, *efoptp* and *eirpoptp* correspond to are shown in Table 7, and the corresponding changing process of objective function value is shown in Figure 2.

In Table 7, *efoptp* is considered as the best profit of EX11 because it is the maximum. In Figure 2, the first 7 iterations represent that the first 7 items ($n/2 = 7$) are loaded into knapsack; the 8th iteration and the 9th iteration represent that items 7 and 6 are removed from knapsack so that $0.5 \leq \varphi = 0.676 \leq 0.7$; the iterations from the 10th to the 18th represent that expectation efficiency values are calculated from items 6 to 14; the 19th iteration represents that item 6 is loaded into knapsack; the 20th iteration represents that item 7 is loaded into knapsack; the 21st iteration represents that item 11 is loaded into knapsack. In conclusion, the best profit of EX11 is obtained with 21 iterations.

4.1.2. *Example test on condition of* $rr \geq 1$. The EX12 is given as follows: $\mathbf{W} = (36, 27, 19, 30, 24, 32, 17, 35, 9, 47, 90, 99, 78, 75, 52, 89, 12, 42, 80, 10)$, $\mathbf{P} = (90, 57, 38, 54, 37, 48, 24, 49, 12, 56, 100, 95, 68, 63, 43, 72, 9, 30, 56, 1)$, $M = 660$ and $n = 20$. In EX12, $rr = 1.4796 > 1$; thus, only Algorithm 7 needs to be performed. Furthermore, $\Omega = 14$, $\Omega + j + l = 18$, *goptp* $= 821$, and $\mathbf{X}^* = (1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0)$. The corresponding changing process of objective function value is shown in Figure 3.

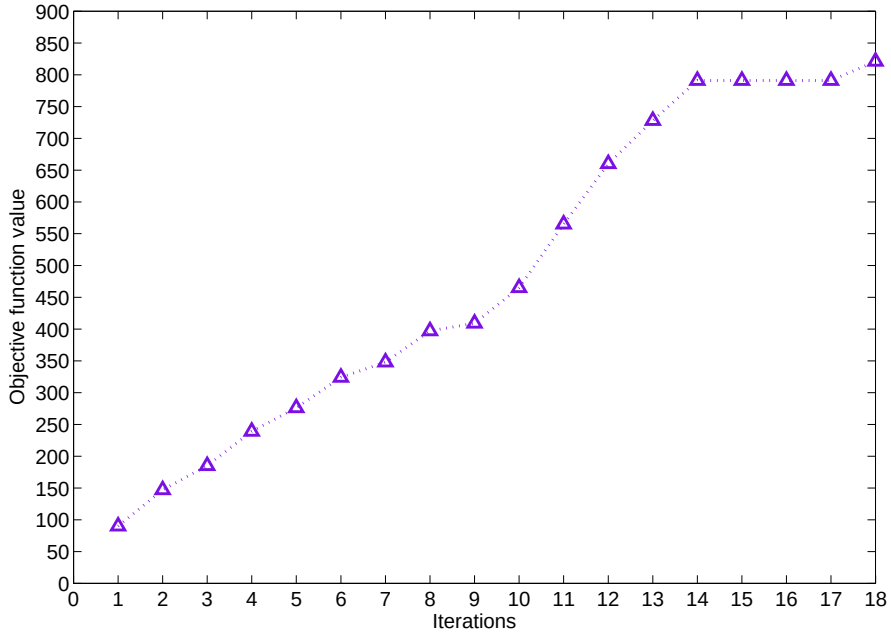

FIGURE 3. The changing process of objective function value for EX12

In Figure 3, the first 14 iterations represent that items 1 to 14 are loaded into knapsack in turn; the following 3 iterations represent that items 15, 16 and 17 cannot be loaded into knapsack; the last iteration represents that item 18 is loaded into knapsack. In conclusion, the best profit of EX12 is obtained with 18 iterations.

4.2. **Comparison analysis.** The comparison results among GAA, GDEE and EEII based on 17 benchmark knapsack examples are shown in Table 8. Some conclusions are described as follows.

- (i) EEII, GDEE and GAA can obtain the optimal solution. Both EEII and GDEE use threshold determination algorithm and expectation efficiency model, while GAA is a classical approximate algorithm and the quality of obtained solution is guaranteed. In particular, EEII always has the considerably worst profit, and this is because it generates the smallest solution space.
- (ii) EEII has the smallest standard deviation, followed by GDEE and GAA, which indicates that EEII has the best stability. GAA aims at finding a relatively best approximate solution irrespective of generating the other solutions; thus, the generated solutions have the most obvious differences, which causes the largest standard deviation. Although GDEE and EEII have the same idea, GDEE generates more redundant solutions due to the parallel computation. Thus, GDEE has larger standard deviation than EEII.
- (iii) EEII has the smallest running time, followed by GDEE and GAA except in the example where $n = 4$. Especially, the running time of EEII regarding n has the steadiest change, which indicates EEII has the best adaptability. In fact, GDEE and EEII have the same time complexity, that is, $O(n)$, while the time complexity of GAA is $O(n \log n)$.

5. **Conclusions.** KP01 is a classical combination optimization problem, and it has a wide range of applications. In this paper, we present a hybrid EEII algorithm to solve KP01 by considering expectation efficiency, increasing rate of profit and improved greedy strategy comprehensively. At first, we design a threshold determination algorithm to put the first

TABLE 8. Comparison results among GAA, GDEE and EEII

Example	M	n	Algorithm	Best profit	Worst profit	Average profit	Standard deviation	Running time (ms)
EX13	11	4	GAA	23	23	23	0.00	8.5537
			GDEE	23	23	23	0.00	25.8312
			EEII	23	23	23	0.00	11.0805
EX14	20	4	GAA	35	35	35	0.00	8.6203
			GDEE	35	35	35	0.00	25.4112
			EEII	35	35	35	0.00	11.1637
EX15	26	5	GAA	51	48	49.36	1.47	12.6495
			GDEE	51	51	51	0.00	26.1841
			EEII	51	51	51	0.00	11.2038
EX16	80	5	GAA	130	125	127.21	2.35	13.2205
			GDEE	130	114	120.49	8.62	26.8329
			EEII	130	128	128.52	1.23	11.6073
EX17	190	6	GAA	150	136	141.08	10.57	16.8546
			GDEE	150	142	145.74	7.64	26.6377
			EEII	150	143	146.22	5.38	11.0842
EX18	50	7	GAA	107	93	96.91	12.56	19.0349
			GDEE	107	98	103.23	9.42	27.7560
			EEII	107	107	107	0.00	11.7421
EX19	170	7	GAA	1735	1680	1696.54	41.53	18.0442
			GDEE	1735	1711	1718.69	15.61	26.7943
			EEII	1735	1726	1730.81	6.49	11.5137
EX20	104	8	GAA	900	857	873.62	23.86	28.0242
			GDEE	900	900	900	0.00	27.0519
			EEII	900	900	900	0.00	11.2681
EX21	60	10	GAA	52	48	48.51	1.74	32.2917
			GDEE	52	52	52	0.00	27.3208
			EEII	52	52	52	0.00	11.3476
EX22	165	10	GAA	309	285	297.44	19.57	33.5074
			GDEE	309	297	301.83	12.71	28.2802
			EEII	309	306	307.16	2.08	11.3819
EX23	269	10	GAA	295	283	286.59	8.11	32.8061
			GDEE	295	294	294.27	0.83	27.5272
			EEII	295	294	294.27	0.83	11.4904
EX24	375	15	GAA	481.0694	426.3413	439.6451	28.13	58.0563
			GDEE	481.0694	462.8481	472.9024	7.6817	28.7538
			EEII	481.0694	473.0285	476.3852	3.3076	11.7225
EX25	750	15	GAA	1458	1416	1437.17	16.46	61.3705
			GDEE	1458	1458	1458	0.00	29.6361
			EEII	1458	1458	1458	0.00	11.6902
EX26	878	20	GAA	1024	1006	1015.74	9.07	85.3673
			GDEE	1024	1018	1020.58	4.92	29.4753
			EEII	1024	1018	1020.58	4.92	12.1028
EX27	879	20	GAA	1025	1025	1025	0.00	87.1573
			GDEE	1025	1025	1025	0.00	29.5038
			EEII	1025	1025	1025	0.00	12.2981
EX28	10000	23	GAA	9767	9721	9743.29	27.82	102.0214
			GDEE	9767	9767	9767	0.00	29.4961
			EEII	9767	9767	9767	0.00	11.8526
EX29	6404180	24	GAA	13549094	13548136	13548625.67	463.18	110.4168
			GDEE	13549094	13548495	13548684.22	397.54	29.5973
			EEII	13549094	13548495	13548684.22	397.54	11.9475

few items into knapsack. Then, we propose an expectation efficiency algorithm and an increasing rate of profit algorithm to put some other items into knapsack; meanwhile, we prove the upper and lower bounds of the objective function value by limiting the size of items. Apart that, we propose an improved greedy strategy to solve the special KP01. The experimental results regarding a large number of examples show that EEII algorithm can solve KP01.

However, as an improved algorithm, EEII also has some limitations which need to be enhanced further. Firstly, more effective and efficient threshold determination algorithm should be devised in order to guarantee that the items put into knapsack in advance are the accurate part of the optimal solution. Secondly, expectation efficiency model as well as increasing rate of profit model should be devised by considering more factors, for example, the relationship among multiple items. Finally, the core idea of EEII is expected to solve multi-objective and multi-dimensional KP.

REFERENCES

[1] H. Buhrman, B. Loff and L. Torenvliet, Hardness of approximation for knapsack problems, *Theory of Computing Systems*, vol.56, pp.372-393, 2015.

[2] G. Dantzig, Discrete variable extremum problem, *Operations Research*, vol.5, pp.266-277, 1957.

[3] S. Sahni, Approximate algorithms for the 0/1 knapsack problem, *Journal of ACM*, vol.22, pp.115-124, 1975.

[4] P. Toth, Dynamic programming algorithms for the zero-one knapsack problem, *Computing*, vol.25, pp.29-45, 1980.

[5] C. Bazgan, H. Hugot and D. Vanderpooten, Implementing an efficient fptas for the 0-1 multi-objective knapsack problem, *European Journal of Operational Research*, vol.198, pp.47-56, 2009.

[6] A. Rong and J. R. Figueira, A reduction dynamic programming algorithm for the bi-objective integer knapsack problem, *European Journal of Operational Research*, vol.231, pp.299-313, 2013.

[7] A. Rong and J. R. Figueira, Dynamic programming algorithms for the bi-objective integer knapsack problem, *European Journal of Operational Research*, vol.236, pp.85-99, 2014.

[8] P. J. Kolesar, A branch and bound algorithm for the knapsack problem, *Management Science*, vol.13, pp.723-735, 1967.

[9] E. Horowitz and S. Sahni, Computing partitions with applications to the knapsack problem, *Journal of the Association for Computing Machinery*, vol.21, pp.277-292, 1974.

[10] S. Martello and P. Toth, A bound and branch algorithm for the zero-one multiple knapsack problem, *Discrete Applied Mathematics*, vol.3, pp.275-288, 1981.

[11] D. Pisinger, An expanding-core algorithm for the exact 0-1 knapsack problem, *European Journal of Operational Research*, vol.87, pp.175-187, 1995.

[12] M. Visee, J. Teghem, M. Pirlot and E. L. Ulungu, Two-phases method and branch and bound procedures to solve the bi-objective knapsack problem, *Journal of Global Optimization*, vol.12, pp.139-155, 1998.

[13] S. Martello and P. Toth, A mixture of dynamic programming and branch-and-bound for the subset-sum problem, *Management Science*, vol.30, pp.765-771, 1984.

[14] S. Martello, D. Pisinger and P. Toth, Dynamic programming and strong bounds for the 0-1 knapsack problem, *Management Science*, vol.45, pp.414-424, 1999.

[15] S. C. Neoh, N. Morad, C. P. Lim and Z. A. Aziz, A GA-PSO layered encording evolutionary approach to 0/1 knapsack optimization, *International Journal of Innovation Computing, Information and Control*, vol.6, no.8, pp.3489-3505, 2010.

[16] H.-H. Yang, M.-T. Wang and C.-H. Yang, A hybrid of rough sets and genetic algorithms for solving the 0-1 multidimensional knapsack problem, *International Journal of Innovation Computing, Information and Control*, vol.9, no.9, pp.3537-3548, 2013.

[17] T. K. Truong, K. Li and Y. Xu, Chemical reaction optimization with greedy strategy for the 0-1 knapsack problem, *Applied Soft Computing*, vol.13, pp.1774-1780, 2013.

[18] 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, pp.9959-9970, 2013.

[19] Z. Yang, G. Wang and F. Chu, An effective GRASP and tabu search for the 0-1 quadratic knapsack problem, *Computers & Operations Research*, vol.40, pp.1176-1185, 2013.
[20] M. Azad, A. Rocha and E. Fernandes, A simplified binary artificial fish swarm algorithm for 0-1 quadratic knapsack problems, *Journal of Computational and Applied Mathematics*, vol.259, pp.897-904, 2014.
[21] P. H. Nguyen, D. Wang and T. K. Truong, A novel binary social spider algorithm for 0-1 knapsack problem, *International Journal of Innovation Computing, Information and Control*, vol.13, no.6, pp.2039-2049, 2017.
[22] T. C. Lu and G. R. Yu, An adaptive population multi-objective quantum-inspired evolutionary algorithm for multi-objective 0/1 knapsack problems, *Information Sciences*, vol.243, pp.39-56, 2013.
[23] Z. Cheng, Y. Huang, Z. Chen, X. Zhang and J. Xu, Solving the 0-1 multi-objective knapsack problem using self-assembly of DNA tiles, *Proc. of the 4th International Conference on Bio-Inspired Computing*, Beijing, China, pp.137-145, 2009.
[24] V. Boyer, M. Elkihel and D. E. Baz, Heuristics for the 0-1 multidimensional knapsack problem, *European Journal of Operational Research*, vol.199, pp.658-664, 2009.
[25] C. Wilbaut, S. Salhi and S. Hanafi, An iterative variable-based fixation heuristic for the 0-1 multidimensional knapsack problem, *European Journal of Operational Research*, vol.199, pp.339-348, 2009.
[26] C. Archetti, L. Bertazzi and M. G. Speranza, Reoptimizing the 0-1 knapsack problem, *Discrete Applied Mathematics*, vol.158, pp.1879-1887, 2010.
[27] R. R. Hill, Y. K. Cho and J. T. Moore, Problem reduction heuristic for the 0-1 multidimensional knapsack problem, *Computers & Operations Research*, vol.39, pp.19-26, 2012.
[28] J. Zhou, D. Chen, Z. Wang and W. Xing, A conic appoximation method for the 0-1 qudratic knapsack problem, *Journal of Industrial and Management Optimization*, vol.9, pp.531-547, 2013.
[29] J. Lv, X. Wang, M. Huang, H. Cheng and F. Li, Solving 0-1 knapsack problem by greedy degree and expectation efficiency, *Applied Soft Computing*, vol.41, pp.94-103, 2016.
[30] D. Kahneman and A. Tversky, Prospect theory: An analysis of decision under risk, *Econometrica*, vol.47, pp.263-291, 1979.
[31] *KP01 Cases*, https://people.sc.fsu.edu/jburkardt/datasets/knapsack_01/knapsack_01.html.
[32] K. K. Bhattacharjee and S. P. Sarmah, Shuffled frog leaping algorithm and its application to 0/1 knapsack problem, *Applied Soft Computing*, vol.19, pp.252-263, 2014.

Appendix A: The Proof of Theorem 3.1.

Proof: (i) The total weight of the first Q items is smaller than M with (7), and the corresponding $optp(Q)$ is smaller than $optp$ with (8). Furthermore, we know $optp(Q) = optp$ from (10). In this case, $optp(Q) \leq optp$.

(ii) The first m items are loaded into knapsack, and $optp$ is

$$optp = \sum_{i=1}^{m} p_i + \sum_{i=m+1}^{Q+1} p_i x_i + \sum_{i=Q+2}^{U} p_i x_i. \tag{25}$$

The corresponding constraint condition is

$$\sum_{i=1}^{m} w_i + \sum_{i=m+1}^{Q+1} w_i x_i + \sum_{i=Q+2}^{U} w_i x_i \leq M. \tag{26}$$

If item $(Q+1)$ is in knapsack, we have

$$\begin{cases} optp = \sum_{i=1}^{m} p_i + \sum_{i=m+1}^{Q} p_i \\ \text{subject to } \sum_{i=1}^{m} w_i + \sum_{i=m+1}^{Q} w_i \leq M. \end{cases} \tag{27}$$

$$optp(Q+1) = \sum_{i=1}^{m} p_i + \sum_{i=m+1}^{Q} p_i + p_Q > \sum_{i=1}^{m} p_i + \sum_{i=m+1}^{Q} p_i = optp. \tag{28}$$

If item $(Q+1)$ is not in knapsack, we have

$$\sum_{i=m+1}^{Q} w_i x_i + \sum_{i=Q+2}^{U} w_i x_i < w_{Q+1} + \sum_{i=m+1}^{Q} w_i. \tag{29}$$

Let $x_{\tau 1}, x_{\tau 2}, \ldots, x_{\tau i}, \ldots, x_{\tau l} = 0$, here $m+1 \le \tau i \le Q$ and $1 \le i \le l$, and we have

$$\sum_{i=Q+2}^{U} w_i x_i \le w_{Q+1} + w_{\tau 1} + w_{\tau 2} + \cdots + w_{\tau l}. \tag{30}$$

Both sides of (30) are multiplied by r_{Q+2}, and we have

$$r_{Q+2} \sum_{i=Q+2}^{U} w_i x_i < r_{Q+2} \left(w_{Q+1} + w_{\tau 1} + w_{\tau 2} + \cdots + w_{\tau l} \right). \tag{31}$$

By magnifying or reducing method, (31) is changed to

$$\begin{cases} r_{Q+2} \sum_{i=Q+2}^{U} w_i x_i \ge (r_{Q+2} w_{Q+2} x_{Q+2} + r_{Q+3} w_{Q+3} x_{Q+3} + \cdots + r_U w_U x_U) \\ \qquad\qquad = \sum_{i=Q+2}^{U} p_i x_i \\ r_{Q+2}(w_{Q+1} + w_{\tau 1} + w_{\tau 2} + \cdots + w_{\tau l}) \le p_{Q+1} + p_{\tau 1} + p_{\tau 2} + \cdots + p_{\tau l}. \end{cases} \tag{32}$$

By simplifying (32), we have

$$\sum_{i=Q+2}^{U} p_i x_i < p_{Q+1} + p_{\tau 1} + p_{\tau 2} + \cdots + p_{\tau l}. \tag{33}$$

After conducting transposition for (33), we have

$$\sum_{i=m+1}^{Q} p_i x_i + \sum_{i=Q+2}^{U} p_i x_i < p_{Q+1} + \sum_{i=m+1}^{Q} p_i. \tag{34}$$

$$\begin{aligned} optp &= \sum_{i=1}^{m} p_i + \sum_{i=m+1}^{Q} p_i x_i + \sum_{i=Q+2}^{U} p_i x_i \\ &< p_{Q+1} + \sum_{i=m+1}^{Q} p_i + \sum_{i=1}^{m} p_i = optp(Q+1). \end{aligned} \tag{35}$$

To sum up (i) and (ii), Theorem 3.1 is proved. □

Appendix B: The Proof of Theorem 3.2.

Proof: Algorithm 6 consists of Algorithms 1-5, and they are performed in tandem. In Algorithm 1, it needs to traverse items from 1 to $n/2$, and runs in $O(n)$. In Algorithm 2, it needs to traverse items from 1 to $(Q+1)$, and runs in $O(n)$. In Algorithm 3, (15) is performed from items $(m+1)$ to $(Q+1)$, and runs in $O(n)$. In Algorithm 4, (16) is performed from items $(m+1)$ to $(Q+1)$, and runs in $O(n)$. In Algorithm 5, it needs to traverse items from 1 to S, and runs in $O(n)$. In conclusion, Algorithm 6 runs in $O(n)$. □

Appendix C: The Proof of Theorem 3.3.

Proof: Algorithm 8 consists of Algorithms 6 and 7.

(i) Consider $rr \geq \rho$ and Algorithm 7 is performed. $ITn = \Omega + j + l < n < 3n$. Furthermore, $egoptp > optp(Q)$ and $goptp \geq optp(Q)$, and $ITn \geq Q$. Therefore, $Q \leq ITn < 3n$.

(ii) Consider $rr < \rho$ and Algorithm 6 is performed. ITn got with $optp(Q)$ is the smallest, and ITn got with *efoptp* or *eirpoptp* is the largest.

On condition of $\varphi > \beta$, suppose that x items are removed from knapsack until CC1 is met, and z items from $(m+1)$ to S are loaded into knapsack later. ITn got with $optp(Q)$ is shown as

$$\begin{aligned} ITn &= \frac{n}{2} + x + (Q - m) \\ &= Q + \left(\frac{n}{2} - m\right) + x, \ //\frac{n}{2} - m = x \\ &= Q + 2x \geq Q, \ //x \geq 0 \end{aligned} \tag{36}$$

ITn got with *efoptp* or *eirpoptp* is shown as

$$\begin{aligned} ITn &= \frac{n}{2} + x + (S - m) + z \\ &= S + \left(\frac{n}{2} - m\right) + x + z \\ &\leq S + 2x + z, \ //\frac{n}{2} - m = x \\ &\leq 2S + 2x - m, \ //z \leq S - m \\ &\leq 2n + 2(x + m) - 3m, \ //S \leq n \\ &= 3n - 3m, \ //x + m = \frac{n}{2} \\ &< 3n, \ //m > 0 \end{aligned} \tag{37}$$

On condition of $\varphi < \alpha$, suppose that y items are put into knapsack until CC2 is met. ITn got with $optp(Q)$ is shown as

$$\begin{aligned} ITn &= \frac{n}{2} + y + (Q - m) \\ &= Q + \left(\frac{n}{2} + y\right) - m \\ &= Q, \ //\frac{n}{2} + y = m \end{aligned} \tag{38}$$

ITn got with *efoptp* or *eirpoptp* is shown as

$$\begin{aligned} ITn &= \frac{n}{2} + y + (S - m) + z \\ &= S + z, \ //\frac{n}{2} + y = m \\ &\leq 2S - m, \ //z \leq S - m \\ &\leq 2n - m, \ //S \leq n \\ &< 2n, \ //m > 0 \end{aligned} \tag{39}$$

To sum up (i) and (ii), Theorem 3.3 is proved. □