DOI：10.20189/j.cnki.CN/61-1527/E.202501004

0-1背包问题上界的快速计算方法

王正元
（火箭军工程大学，陕西 西安 710025）

摘　要：为提高0-1背包问题上界求解的速度与精确度，分析了拉格朗日松弛方法构造的精确0-1背包问题上界模型，建立了该模型的快速求解算法，证明了精确0-1背包问题上界是拉格朗日乘子的凸函数。由此，提出了精确0-1背包问题最小上界的求解方法，证明了精确0-1背包问题上界是物品数的单峰函数，且0-1背包问题的上界恰好等于物品数为关键物品数（关键物品数-1）时精确0-1背包问题最小上界的最大值。结果表明：该计算方法所需计算量与背包问题物品数成比例，计算速度较快，上界相对较小。通过6 500例不同上界计算实验对比，提出的上界计算所需时间约为其他较优算法的15.1%；上界占优比例94.29%，而其他较优算法占优比例仅68.71%。进一步表明该上界算法可以快速构造较好的近似解，从而降低0-1背包问题的维数。

关键词：组合优化问题；0-1背包问题；上界；精确0-1背包问题；拉格朗日松弛
中图分类号：TJ410　　**文献标志码**：A　　**文章编号**：2097-2741(2025)01-031-10

Fast Algorithm for the Upper Bound of the 0-1 Knapsack Problem

WANG Zhengyuan
(Rocket Force University of Engineering, Xi'an 710025, China)

ABSTRACT: To improve the computational speed and accuracy of the algorithm for the upper bound of the 0-1 knapsack problem (KP01), this study analyzed the upper bound model of the exact k knapsack problem (E-kKP) constructed by Lagrangian relaxation. First, the upper bound of the (E-kKP) was proven to be a convex function of the Lagrangian multiplier, and an algorithm for the minimum upper bound of (E-kKP) was proposed. Second, the upper bound of the (E-kKP) was determined to be a unimodal function of the number of the items, and the upper bound of the (E-kKP) equaled the maximum value of the minimum upper bound of the (E-kKP) when the number of the items was the key item number (key item number-1). The experiments showed that the algorithm for the upper bound of the (E-kKP) proposed in this study requires a proportional calculation to the number of the items, with fast calculatingion speed and relatively small upper bound. From 6 500 experiments with different upper bounds, the calculatingion time of the upper bound in this study is approximately 15.1% of that of other superior algorithms, with an advantage rate of 94.29%, the others have an advantage rate of only 68.71%. In addition, the proposed upper bound algorithm can quickly construct better approximate solutions

收稿日期：2024-09-02
第一作者：王正元（1971—），男，教授，主要从事系统工程理论、组合优化方法及其应用研究。E-mail：zywang1999@163.com

引用格式：王正元. 0-1背包问题上界的快速计算方法[J]. 火箭军工程大学学报，2025，39（1）：31-40. *WANG Zhengyuan. Fast algorithm for the upper bound of the* 0-1 *knapsack problem[J]. Journal of Rocket Force University of Engineering，2025，39（1）：31-40.*

and reduce the dimensionality of KP01.

KEYWORDS: combinatorial optimization problem; 0-1 knapsack problem, upper bound; exact k knapsack problem; Lagrangian relaxation

0-1背包问题（0-1 Knapsack Problem，KP01）是一种典型的组合优化问题，在资源配置、车辆配载、工业生产等领域应用广泛，因而对（KP01）快速精确求解方法的研究具有重要理论意义和应用价值。（KP01）可用整数线性规划模型[1]表示为

$$\text{（KP01）}\quad \max \sum_{j=1}^{n} p_j x_j$$
$$\text{s.t.} \sum_{j=1}^{n} w_j x_j \leqslant C$$
$$x_j = 0,\ 1$$
$$j = 1,\ 2,\ \cdots,\ n$$

其中：p_j为物品j的价值；w_j为物品j的重量；n为背包问题的物品数；C为背包容积；决策变量$x_j \in \{0,\ 1\}$，表示是否选择物品j放入背包中。

为便于研究，假设p_j、w_j和C为正数，物品按照单位重量价值递减顺序排列，满足

$$\frac{p_1}{w_1} \geqslant \frac{p_2}{w_2} \geqslant \cdots \geqslant \frac{p_n}{w_n} \tag{1}$$

$$0 \leqslant C - (w_1 + w_2 + \cdots + w_{s-1}) < w_s \tag{2}$$

关键物品s由式（2）确定。

（KP01）是一类多项式复杂程度的非确定性（Non-deterministic Polynomial，NP）问题，研究工作主要包括（KP01）的近似求解方法和精确求解方法。近似求解方法主要表现为（KP01）求解的智能算法，如早期的遗传算法、蚁群算法、和声算法、狼群算法等，当前则倾向于将多种启发式规则混合起来解决（KP01）、调度问题等组合优化问题，包括混合变量微分进化算法[2]、自适应微分进化算法[3]、二阶段合作进化算法[4]、基于强化学习的协同水波优化算法[5]和协同多阶段超启发式算法[6]等。精确求解方法主要包括分支定界方法[7]、枚举法[8]、动态规划方法[1]等。其中，动态规划方法求解效率较高，所需计算量与背包的容积C以及背包问题的物品数n的乘积有关。若使用动态规划方法直接计算容积大、物品数量多的（KP01），计算量仍然十分巨大。例如，背包容积为123 456 789.123 456 789、物品数量为10^5时，计算量可能达到10^{23}以上，每秒百亿次的计算机需要计算约31.7万年。因此，动态规划方法往往作为（KP01）精确求解中的一个重要组成部分。Combo[1]是一种较好的混合方法，利用了（KP01）的自身特点，从关键物品开始，采用“中心开花”的扩展策略寻找（KP01）的核，即最难求解的（KP01）的子问题。此时，（KP01）子问题中物品没有确定是否放入背包中，而其余物品要么选择放入背包中，要么不会放入背包中。无论是分支定界方法，还是动态规划方法，都依赖（KP01）的上界算法来降低（KP01）的维数，以便简化问题求解。显然，（KP01）的任何一个近似解对应目标函数值都是（KP01）的下界，利用（KP01）的下界与上界，可以较好地实现（KP01）的降维[9]，而利用（KP01）的上界算法可以构造很好的近似解[10]。因此，（KP01）的上界算法是此类问题快速精确求解的关键。

采用线性规划松弛方法，Dantzig[11]构造了（KP01）目标函数值的上界U_D。Cunha等[7]改进了（KP01）的分支定界算法。Ingargiola和Korsh[12-14]提出了降维方法减少决策变量数。Martello和Toth[15]第一次提出了优于U_D的上界U_{MT2}。Fayard和Plateau[16]提出了（KP01）的一种附加约束的上界。Müller-Merbach[17]认为，为了从分数背包问题得到（KP01）的解，可以将分数背包问题解中带小数的分量变为0，或者至少改变一个分量的值。通过改变决策变量中至少2个变量的值，Martello和Toth[18]提出了一种改进的上界U_{MTM}。之后，他们进一步提出了最小紧集上界$U_{k\min}$和最大紧集上界$U_{k\max}$[19-20]，可行解中物品数量满足$k_{\min} < s \leqslant k_{\max}$时很难进一步改善上界[19]。Wang等[10]提到了一种基于精确0-1背包问题的（KP01）上界算法，但缺乏相应的理论证明和计算方法。

对此，本文从可行解中物品数量正好等于k的精确0-1背包问题（Exact k-item Knapsack Problem，E-kKP）出发，构造了一种（KP01）上界的快速计算方法。首先，分析了（E-kKP）上界的特点，采用拉格朗日松弛方法建立了（E-kKP）的上界模型，提出了给定参数（E-kKP）上界模型求解方法和（E-kKP）最小上界模型的求解方法。其次，建立（KP01）的上界模型，证明了（E-kKP）最小上界的变化规律，提出了（KP01）上界模型的快速求解方法。再次，比较了不同算

法求解相同（KP01）的上界，并进行了计算量分析。最后，简略介绍了上界算法在（KP01）降维中的应用。结果表明，与现有（KP01）上界算法相比，本文提出的上界算法具有显著优势，并且可以有效地用于（KP01）降维和近似解构造。

1 精确0-1背包问题的上界

不同（KP01）上界算法所得上界最优目标值差异较大的主要原因是，（KP01）上界问题的解不是可行解。为了获得更小的（KP01）上界，一种途径是在（E-kKP）上界的基础上得到（KP01）的上界。在（KP01）模型的约束中，增加放入背包中物品数量约束，得到（E-kKP）。（E-kKP）[21]的可行解中包含物品数正好等于k，数学规划模型可以表示为

$$\begin{aligned}(\mathrm{E}-k\mathrm{KP})\qquad &\text{maximize}\ \sum_{j=1}^{n}p_jx_j\\ &\text{s.t.}\ \sum_{j=1}^{n}w_jx_j\leqslant C\\ &\sum_{j=1}^{n}x_j=k\\ &x_j=0,\ 1,\quad j=1,\ 2,\ \cdots,\ n\end{aligned}$$

其中：放入背包物品数$k\in\{1,\ 2,\ \cdots,\ n\}$。

1.1 精确0-1背包问题上界模型

采用拉格朗日松弛方法，处理（E-kKP）模型中不等式约束，得到拉格朗日松弛问题$L(\mathrm{E}-k\mathrm{KP},\ \lambda)$[10]：

$$\begin{aligned}L(\mathrm{E}-k\mathrm{KP},\ \lambda)\ f_k(\lambda)=&\max\sum_{j=1}^{n}p_jx_j+\\ &\lambda\left(C-\sum_{j=1}^{n}w_jx_j\right)\\ \text{s.t.}\ &\sum_{j=1}^{n}x_j=k\\ &x_j=0,\ 1\\ &j=1,\ 2,\ \cdots,\ n\end{aligned}$$

其中：$f_k(\lambda)$是$L(\mathrm{E}-k\mathrm{KP})$的最优目标函数值。$\lambda(\lambda\geqslant 0)$为拉格朗日乘子。

定理1 拉格朗日松弛问题$L(\mathrm{E}-k\mathrm{KP},\ \lambda)(\lambda\geqslant 0)$的最优目标函数值$f_k(\lambda)$是$(\mathrm{E}-k\mathrm{KP})$的一个上界。

证明：设x^*是$(\mathrm{E}-k\mathrm{KP})$的最优解，$x$是$L(\mathrm{E}-k\mathrm{KP},\lambda)$的最优解。则

$$\begin{aligned}\sum_{j=1}^{n}w_jx_j^*\leqslant C\\ \sum_{j=1}^{n}p_jx_j^*\leqslant\sum_{j=1}^{n}(p_j-\lambda w_j)\ x_j^*+C\lambda\leqslant\\ \max\sum_{j=1}^{n}(p_j-\lambda w_j)\ x_j+C\lambda=\\ f_k(\lambda),\ \lambda\geqslant 0\end{aligned}$$

因此，$L(\mathrm{E}-k\mathrm{KP},\ \lambda)$的最优目标函数值是$(\mathrm{E}-k\mathrm{KP})$的一个上界。

1.2 给定参数的精确0-1背包问题上界模型求解方法

对于给定参数λ，利用算法1在$O(n)$时间内可以求解拉格朗日松弛问题$L(\mathrm{E}-k\mathrm{KP},\ \lambda)$。

算法1 计算最大k项$p_j-\lambda w_j$之和$v(\lambda,\ k)$的分割方法

Step1 输入背包中物品数n、物品重量和价值$(w_j,\ p_j),\ j=1,\ 2,\ \cdots,\ n$，设定候选集$S:=\{1,\ 2,\ \cdots,\ n\}$；

Step2 $t:=k$；

Step3 $r=\sum_{i\in S}(p_i-\lambda w_i)\ /|S|$；

Step4 $H:=\{j\in\mathrm{S}|p_j-\lambda w_j>r\}$，$E:=\{j\in\mathrm{S}|p_j-\lambda w_j=r\}$，$L:=\{j\in\mathrm{S}|p_j-\lambda w_j<r\}$；

Step5 如果$|H|>t$，则$S:=H$，返回Step3；

Step6 如果$|H|+|E|<t$，则$S:=L$，$t:=t-|H|-|E|$，返回Step3；

Step7 $v(\lambda,\ k):=\sum_{\substack{j\leqslant n\\ p_j-\lambda w_j>r}}(p_j-\lambda w_j)+kr-|\{j\leqslant n|p_j-\lambda w_j>r\}|r$，退出计算。

利用算法1，可得到$(\mathrm{E}-k\mathrm{KP})$模型的目标函数值的上界$f_k(\lambda)$：

$$f_k(\lambda)=v(\lambda,\ k)+C\lambda \tag{3}$$

例1：计算$(\mathrm{E}-6\mathrm{KP})$的上界$f_6(2)$，其中物品价值$p_i$、重量$w_i$以及背包容积$C$分别为

$$(w_i,\ p_i)=(610-10i,\ 52-i),\ i=1,\ 2,\ \cdots,\ 50,$$
$$C=5559$$

解：按照算法1，首先设定

$$S:=\{1,\ 2,\ \cdots,\ 50\},\ t:=6,\ \lambda=2$$

通过计算可得

$$r=-328.5$$
$$H:=\{j\in\mathrm{S}|p_j-\lambda w_j>r\}=\{1,\ 2,\ \cdots,\ 6\},\ E:=\varnothing,$$
$$L:=\{7,\ 8,\ \cdots,\ 50\}$$

恰好满足$|H|=t$，求得$v(2,\ 6)=-1593$，$(\mathrm{E}-6\mathrm{KP})$的上界$f_6(2)$为

$$f_6(2)=v(2,\ 6)+2C=9525$$

1.3 精确0-1背包问题上界模型及其快速求解方法

在算法1和公式（3）的基础上可计算（E-kKP）上界为$f_k(\lambda)$。为了使$f_k(\lambda)$尽可能小，（E-kKP）最小上界数学模型为

$$B_k = f_k(\lambda_k) = \min_{\lambda \geqslant 0} f_k(\lambda) \tag{4}$$

其中：B_k为最小上界。

为了快速计算公式（4）中B_k，通过定理2证明$f_k(\lambda)$是拉格朗日乘子λ的凸函数，定理3给出$f_k(\lambda)$中λ的变化范围。

定理2 $L(\mathrm{E}-k\mathrm{KP}, \lambda)$的最优目标函数值$f_k(\lambda)$是$\lambda$（$\lambda \geqslant 0$）的凸函数。

证明：对任意$\alpha(0<\alpha<1)$，$0 \leqslant \lambda_1 < \lambda_2$，$x^* = (x_1^*, x_2^*, \cdots, x_n^*)$是$L(\mathrm{E}-k\mathrm{KP}, \alpha\lambda_1 + (1-\alpha)\lambda_2)$的最优解，则

$$f_k(\alpha\lambda_1 + (1-\alpha)\lambda_2) = C[\alpha\lambda_1 + (1-\alpha)\lambda_2] + \sum_{j=1}^{n}\{p_j - [\alpha\lambda_1 + (1-\alpha)\lambda_2]w_j\}x_j^* =$$

$$\alpha\left[C + \sum_{j=1}^{n}(p_j - \lambda_1 w_j)x_j^*\right] + (1-\alpha)\left[C + \sum_{j=1}^{n}(p_j - \lambda_2 w_j)x_j^*\right] \leqslant$$

$$\alpha f_k(\lambda_1) + (1-\alpha) f_k(\lambda_2)$$

所以$L(\mathrm{E}-k\mathrm{KP}, \lambda)$的最优目标函数值是$\lambda$（$\lambda \geqslant 0$）的凸函数。

定理3 如果（E－kKP）存在可行解，则存在$f_k(\lambda)$的最小值点$\lambda_k \in [0, \lambda_0]$，使得

$$B_k = f_k(\lambda_k) = \min\{f_k(\lambda) \mid \lambda \geqslant 0\}$$

λ_0可以表示为

$$\lambda_0 = \max\left\{\frac{p_i - p_j}{w_i - w_j} \middle| w_i > w_j, p_i > p_j, i, j \leqslant n\right\} + \max\left\{\frac{p_i}{w_i} \middle| i \leqslant n\right\}$$

证明：令$x^{(1)} = (x_1^{(1)}, x_2^{(1)}, \cdots, x_n^{(1)})$表示$L(\mathrm{E}-k\mathrm{KP}, \lambda)$的最优解。显然有

$$\lambda_0 = \max\left\{\frac{p_i - p_j}{w_i - w_j} \middle| w_i > w_j, p_i > p_j, i, j \leqslant n\right\} + \max\left\{\frac{p_i}{w_i} \middle| i \leqslant n\right\} > \frac{p_i - p_j}{w_i - w_j}, w_i - w_j \neq 0$$

可得

$$p_i - \lambda_0 w_i < p_j - \lambda_0 w_j, w_i > w_j$$

由（E－kKP）存在可行解，可知

$$\min\left\{\sum_{j=1}^{n} w_j x_j \middle| \sum_{j=1}^{n} x_j = k, x_j = 0, 1\right\} \leqslant C$$

因而

$$\sum_{j=1}^{n} w_j x_j^{(1)} = \min\left\{\sum_{j=1}^{n} w_j x_j \middle| \sum_{j=1}^{n} x_j = k, x_j = 0, 1\right\} \leqslant C$$

$\lambda > \lambda_0$时

$$(p_j - \lambda w_j) - (p_i - \lambda w_i) > (p_j - \lambda_0 w_j) - (p_i - \lambda_0 w_i) > 0, w_i > w_j$$

$$f_k(\lambda) = \sum_{j=1}^{n} p_j x_j^{(1)} + \lambda\left(C - \sum_{j=1}^{n} w_j x_j^{(1)}\right) \geqslant f_k(\lambda_0)$$

由于$f_k(\lambda)$在$[0, +\infty)$上连续，其最小值点一定不超过λ_0，即

$$\lambda_k = \arg\min\{f_k(\lambda) \mid \lambda \geqslant 0\}, \lambda_k \in [0, \lambda_0]$$

从定理3可知，（E－kKP）的上界$f_k(\lambda)$的最小值点一定在有界区域内。

定理4给出了λ是否为$f_k(\lambda)$的最小值点λ_k的判别依据。

定理4 如果λ_k是$L(\mathrm{E}-k\mathrm{KP}, \lambda)$的最优解且$\lambda_k > 0$，则

$$\sum_{j=1}^{n} w_j x_j > C, 0 \leqslant \lambda < \lambda_k$$

$$\sum_{j=1}^{n} w_j x_j < C, \lambda > \lambda_k$$

其中：$k \in \{1, 2, \cdots, n\}$为放入背包中物品数；$w_j$是物品$j$的重量；$n$是物品数；$C$是背包容积；决策变量$x_j \in \{0, 1\}$表示是否选择物品$j$放入背包中；$\lambda$（$\lambda \geqslant 0$）和$\lambda_k$为拉格朗日乘子。

证明：由定理2可知，$f_k(\lambda)$是λ的凸函数，当且仅当$\sum_{j=1}^{n} w_j x_j < C$时，$f_k(\lambda)$在$[\lambda_k, +\infty)$上单调增，当且仅当$\sum_{j=1}^{n} w_j x_j > C$时，$f_k(\lambda)$在$[0, \lambda_k]$上单调减。

由定理4可知$f_k(0) \geqslant f_k(\lambda_k)$，当且仅当$\lambda_k = 0$时取等号。$\lambda_k \neq 0$时，由公式（1）和公式（2）可知：$k < s$时，$\sum_{j=1}^{k} w_j < C$，而$p_{k+1}/w_{k+1} \leqslant p_k/w_k$，从而$f_k(p_k/w_k) > f_k(p_{k+1}/w_{k+1})$。因此，$k < s$时，$\lambda_k$取值范围可变更为$[0, p_k/w_k]$。$k \geqslant s$时，$\sum_{i=1}^{k} w_i > C$，而$p_{k+1}/w_{k+1} \leqslant p_k/w_k$，从而$f_k(p_k/w_k) \leqslant f_k(p_{k+1}/w_{k+1})$。因此，$k \geqslant s$时，$\lambda_k$取值范围可变更为$[p_{k+1}/w_{k+1}, \lambda_0]$。实际计算过程中，为了避免因计算$\lambda_0$消耗太多时间，通常采用二分法获得$\lambda_k$所在区域的右端点。

为便于估计计算量，采用一维搜索方法计算

（E－kKP）的最小上界，如通过黄金分割法100次搜索后，近似解所在区间的长度变为原区间长度的$1.255\,589\times10^{-21}$倍，因而上界B_k所需计算量仍为$O(n)$。计算上界B_k的计算步骤如算法2所示。

算法2 （E－kKP）最小上界B_k的计算方法

Step1 输入物品数n、物品重量和价值$(w_j, p_j), j=1, 2, \cdots, n$，输入解中放入背包中物品数$k$，设定初始步长$d=1$，按照公式（3）计算关键物品$s$；

Step2 如果$k<s$，则给定区间$[a, b]=[0, p_{k-1}/w_{k-1}]$，执行算法3，得到最优上界$B_k$，退出计算；否则，继续执行后续步骤；

Step3 $a=p_{k+1}/w_{k+1}$，$b_0=a$，$b_1=a+d$，计算$f_k(b_0)$和$f_k(b_1)$；

Step4 $b_2=b_1+d$，计算$f_k(b_2)$；

Step5 如果$f_k(b_2)>f_k(b_1)$，则$[a, b]=[b_0, b_2]$，执行算法3，得到最优上界B_k，退出计算；否则，继续执行后续步骤；

Step6 $d=d+d, b_0=b_1, b_1=b_2$，返回Step4。

算法3 给定区间$[a, b]$时上界B_k的计算方法

Step1 输入背包中物品数、物品重量和价值$(w_j, p_j), j=1, 2, \cdots, n$，输入解中放入物品数$k$，输入区间端点$a$、$b$，$a<b$，精度$\varepsilon$；

Step2 $c=(a+b)/2$，计算$f_k(a)$、$f_k(b)$和$f_k(c)$；

Step3 计算

$$d=\frac{a+c}{2}-\frac{[f_k(c)-f_k(a)](b-a)}{2\{\frac{c-a}{b-c}[f_k(b)-f_k(c)]-f_k(c)+f_k(a)\}};$$

Step4 如果$b-a<\varepsilon$，则取$\lambda_k=\arg\min\{f_k(\lambda)\mid\lambda=a, b, c\}$，$B_k=f_k(\lambda_k)$，退出计算；

Step5 如果$|d-c|<\varepsilon$，当$b-a>2c$时，$d=0.8c+0.2b$；$b-a\leqslant 2c$时，$d=0.8c+0.2a$；

Step6 如果$d-a<\varepsilon$，$d=0.9a+0.1c$；

Step7 如果$b-d<\varepsilon$，$d=0.9b+0.1c$；

Step8 如果$c<d$，当$f_k(d)>f_k(c)$时，$b\leftarrow d$，$f_k(d)\leqslant f_k(c)$时，$a\leftarrow c$，$c\leftarrow d$，返回Step3；

Step9 $f_k(d)>f_k(c)$时，$a\leftarrow d$，$f_k(d)\leqslant f_k(c)$时，$b\leftarrow c$，$c\leftarrow d$，返回Step3；

采用算法2计算例1中（E－6KP）的最小上界B_6，参数精度为10^{-9}，计算结果如表1所示。

表1 计算最小上界B_6时最优解所在区间的左右端点

Tab. 1 Interval Terminals of the optimal solution of the minimum upper-bound B_6

迭代次数	a	b	迭代次数	a	b
1	0	0.083 636 364	2	0	0.041 818 182
3	0	0.004 181 818	4	0	0.000 418 182
5	0	4.181 82E−05	6	0	4.181 82E−06
7	0	4.181 82E−07	8	0	4.181 82E−08
9	0	4.181 82E−09	10	0	4.181 82E−10

从表1可以看出，仅仅迭代计算了9次，就将最优解所在区间从[0，0.083 636 364]压缩到了[0，4.181 82E−10]，得到满足精度要求的最小上界值$B_6=291$。

2 0-1背包问题的上界

由于（E－kKP）的最小上界随k变化而变化，（KP01）的上界为（E－kKP）最小上界的最大值，即

$$U_W=\max\{B_k\mid 1\leqslant k\leqslant n\} \tag{5}$$

2.1 精确0-1背包问题最小上界的变化规律

为了快速计算（KP01）的上界，按照公式（5）计算时所需的计算量较大，需要进一步探索（E－kKP）的最小上界B_k随k变化而变化的内在规律，用于设计（KP01）上界的计算方法。

例2：计算（E－kKP）的最小上界$B_k(k=1, 2, \cdots, 50)$和给定数据确定（KP01）的上界U_W。背包问题中物品价值、重量及背包容积与例1相同。

解：采用算法2计算（E－kKP）的最小上界$B_k(k=1, 2, \cdots, 50)$，计算结果如表2和图1所示。

表2 （E-kKP）的最小上界B_k

Tab. 2 Minimum upper-bound B_k for（E-kKP）

k	B_k	k	B_k	k	B_k
1	51	2	101.0	3	150.0
4	198.0	5	245.0	6	291.0
7	336.0	8	380.0	9	423.0
10	465.0	11	456.9	12	447.9
13	438.9	14	429.9	15	420.9
16	411.9	17	402.9	18	393.9
19	384.9	20	375.9	21	366.9
22	357.9	23	348.9	24	339.9
25	0	…	…	50	0

注：$k=1, 2, \cdots, 50$

由表 2 可知，按照公式（5），可得相应（KP01）的上界为 465。

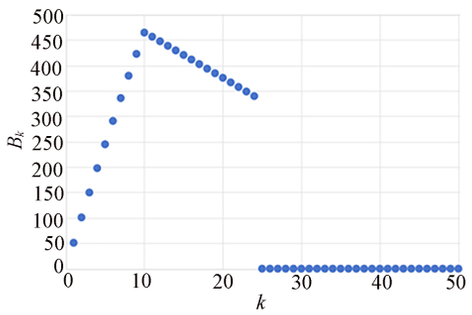

图 1 （E-kKP）的最小上界B_k的变化规律

Fig. 1 Variation law of the minimum upper bound B_k for（E-kKP）

从图 1 可以看出，（E－kKP）的最小上界 B_k 先随 k 增大而增加，达到最大值以后将随着 k 增大而减少，直至下降为 0。（E－kKP）的最小上界 B_k 在 $k=10$ 处取得最大值，而关键物品恰好是 $s=11$。通过定理 5 指出（E－kKP）的最小上界 B_k 随 k 变化的一般规律。

定理 5 （KP01）的上界 $B_{s^*}=\max\{B_k: 1\leqslant k\leqslant n\}$满足

$$B_k\leqslant B_{k+1},\ 1\leqslant k<s^*$$

$$B_k\geqslant B_{k+1},\ s^*\leqslant k<n-1$$

其中：B_k 由式（4）确定，k 为装入背包中物品数，$k=s^*$ 处 B_k 取得最大值。

证明：设 $B_k=\lambda_k C+\sum_{i=1}^{k}(p_{d_{ki}}-\lambda_k w_{d_{ki}})$，则 $d_{ki}\in\{1,2,\cdots,n\}$满足

$$p_{d_{k1}}-\lambda_k w_{d_{k1}}\geqslant p_{d_{k2}}-\lambda_k w_{d_{k2}}\geqslant\cdots\geqslant p_{d_{kn}}-\lambda_k w_{d_{kn}}$$

因为

$$B_{s^*}=\max\{B_k\mid 1\leqslant k<n\}$$

则

$$B_{s^*}\geqslant B_{s^*-1},\ B_{s^*}\geqslant B_{s^*+1}$$

由定理 3 可知

$$f_{s^*}(\lambda_{s^*-1})\geqslant f_{s^*}(\lambda_{s^*})$$

$$p_{d_{s^*-1s^*-1}}-\lambda_{s^*-1}w_{d_{s^*-1s^*-1}}\geqslant p_{d_{s^*-1s^*}}-\lambda_{s^*-1}w_{d_{s^*-1s^*}}\geqslant B_{s^*}-B_{s^*-1}>0$$

则

$$B_{s^*-1}\geqslant B_{s^*-1}-(p_{d_{s^*-1s^*-1}}-\lambda_{s^*-1}w_{d_{s^*-1s^*-1}})\geqslant B_{s^*-2}$$

同理可得，$B_k\geqslant B_{k-1},k=s^*-2,s^*-3,\cdots,2$。另一方面

$$p_{d_{s^*+1s^*+2}}-\lambda_{s^*+1}w_{d_{s^*+1s^*+2}}\leqslant p_{d_{s^*+1s^*+1}}-\lambda_{s^*+1}w_{d_{s^*+1s^*+1}}\leqslant B_{s^*+1}-B_{s^*}\leqslant 0$$

则

$$B_{s^*+2}\leqslant B_{s^*+1}+p_{d_{s^*+1s^*+2}}-\lambda_{s^*+1}w_{d_{s^*+1s^*+2}}\leqslant B_{s^*+1}$$

同理可得，$B_k\geqslant B_{k+1},k=s^*+2,\cdots,n-2$。

2.2 0-1背包问题上界模型的快速计算方法

从定理 5 可知，（KP01）上界模型的快速计算，关键在于确定 s^*。定理 6 给出（E－kKP）最小上界 B_k 何时取最大值和最小上界 B_k 最大值计算的简易方法。

定理 6 （KP01）的上界为

$$U_W=\max\{B_s,B_{s-1}\}\tag{6}$$

其中：关键物品 s 由式（2）确定。

证明：设 $d_{ki}\in\{1,\cdots,n\}$ 满足 $p_{d_{k1}}-\lambda_k w_{d_{k1}}\geqslant p_{d_{k2}}-\lambda_k w_{d_{k2}}\geqslant\cdots\geqslant p_{d_{kn}}-\lambda_k w_{d_{kn}}$。

（1）证明 $B_{s-1}=\max\{B_k\mid k=1,2,\cdots,s-1\}$。

由式（2）可知，$\sum_{i=1}^{s-1}w_i\leqslant C$。当 $\sum_{i=1}^{s-1}w_i=C$ 时

$$B_{s-1}=f_{s-1}\left(\frac{p_s}{w_s}\right)=f_{s-2}\left(\frac{p_s}{w_s}\right)+p_{s-1}-\frac{p_s}{w_s}w_{s-1}\geqslant f_{s-2}\left(\frac{p_s}{w_s}\right)\geqslant B_{s-2}$$

由式（1）可知，$\frac{p_s}{w_s}\leqslant\frac{p_{s-1}}{w_{s-1}}$，因而，当 $\sum_{i=1}^{s-1}w_i<C$ 时

$$f_{s-1}\left(\frac{p_s}{w_s}\right)\leqslant f_{s-1}\left(\frac{p_{s-1}}{w_{s-1}}\right)$$

由定理 4 可知，$\frac{p_{s-1}}{w_{s-1}}>\lambda_{s-1}$，从而

$$\lambda_{s-1}C+\sum_{i=1}^{s-1}(p_i-\lambda_{s-1}w_i)\geqslant\lambda_{s-1}C+\sum_{i=1}^{s-1}\left(p_i-\frac{p_{s-1}}{w_{s-1}}w_i\right)$$

由定理 1、定理 3 可得

$$B_{s-1}=\lambda_{s-1}C+\max\left\{\sum_{i=1}^{n}(p_i-\lambda_{s-1}w_i)\,x_i\ \middle|\ \sum_{i=1}^{n}x_i=s-1,x_i=0,1\right\}\geqslant\lambda_{s-1}C+\sum_{i=1}^{s-1}(p_i-\lambda_{s-1}w_i)$$

因此

$$p_{d_{s-1s-1}}-\lambda_{s-1}w_{d_{s-1s-1}}>\min\left\{p_i-\frac{p_{s-1}}{w_{s-1}}w_i\mid i=1,\cdots,s-1\right\}=$$

$$p_{s-1}-\frac{p_{s-1}}{w_{s-1}}w_{s-1}=0$$

$B_{s-1}>B_{s-1}-(p_{d_{s-1s-1}}-\lambda_{s-1}w_{d_{s-1s-1}})\geqslant B_{s-2}$

由定理5可得

$B_{s-1}=\max\{B_k|k=1,2,\cdots,s-1\}$

（2）证明$B_s=\max\{B_k|k=s,s+1,\cdots,n\}$。

由式（2）可知，$\frac{p_s}{w_s}\geqslant\frac{p_{s+1}}{w_{s+1}}$，$\sum_{i=1}^{s}w_i>C$，此时

$$f_s\left(\frac{p_s}{w_s}\right)\leqslant f_s\left(\frac{p_{s+1}}{w_{s+1}}\right)$$

由定理4可知，$\frac{p_s}{w_s}<\lambda_s$，从而

$$p_{d_{ss}}-\lambda_s w_{d_{ss}}=\min\{p_{d_{si}}-\lambda_s w_{d_{si}}|\ i=1,2,\cdots,s\}\leqslant$$

$$\min\left\{p_i-\frac{p_s}{w_s}w_i|\ i=1,2,\cdots,s\right\}=p_s-\frac{p_s}{w_s}w_s=0$$

$$p_{d_{ss+1}}-\lambda_s w_{d_{ss+1}}\leqslant p_{d_{ss}}-\lambda_s w_{d_{ss}}\leqslant 0$$

由定理1、定理3可知

$$B_s=\lambda_s C+\sum_{i=1}^{s+1}(p_{d_{si}}-\lambda_s w_{d_{si}})-$$

$$(p_{d_{ss+1}}-\lambda_s w_{d_{ss+1}})\geqslant B_{s+1}$$

由定理5可得$B_s=\max\{B_k|k=s,s+1,\cdots,n\}$。

综上所述，（KP01）的上界为$U_W=\max\{B_s,B_{s-1}\}$。

（KP01）的上界计算方法见算法4。

算法4 （KP01）的上界计算方法

Step1 输入背包中物品数n、背包的容积C，输入物品重量和价值(w_j,p_j)，$j=1,2,\cdots,n$，按照公式计算关键物品s；

Step2 采用算法2，计算精确0-1背包问题的最小上界B_{s-1}、B_s；

Step3 按照公式（6），得到（KP01）的上界$U_W=\max\{B_{s-1},B_s\}$。输出结果，退出计算。

例2中，为了得到（KP01）的上界，利用算法4，得到$s=11$，$B_{10}=465$，$B_{11}=456.9$，从而得到（KP01）的上界$U_W=\max\{B_{10},B_{11}\}=465$。可以看出，利用算法4，计算（KP01）上界的计算量仍然是$O(n)$。

3 实验结果比较

3.1 0-1背包问题不同类型上界算法计算结果对比

例3：采用不同上界算法计算表3中（KP01）的上界，物品价值和重量随机生成，每一个类型的（KP01）都有100个（KP01），背包容积按照下式确定：

$$C=[h\sum_{i=1}^{n}(w_i/101)],\ h=1,2,\cdots,100,\ n=10^4$$

采用不同上界算法计算表3中6 500个不同（KP01），表4列举了不同上界占优的比例。可以看出，6 500个问题中上界U_W有6 129次占优，比例高达94.29%，而其他最好的上界$U_{k\max}$占优次数为4 466（比例为68.71%）。结果表明，上界U_W总是优于上界$U_{k\max}$和U_D，且大多数情况下上界U_W优于上界U_{MT2}和U_{MTM}。

3.2 计算量分析

本文研究（KP01）上界的快速计算方法，算法4的关键是调用算法2计算两个（E－kKP）的最小上界$B_k(k=s-1,s)$；算法2针对（E－kKP）中k

表3 0-1背包问题的物品重量与价值[22]

Tab. 3 Weights and values of 0-1 knapsack problems[22]

序号	$(w_i,p_i),\ i=1,2,\cdots,10^4$	序号	$(w_i,p_i),\ i=1,2,\cdots,10^4$
1	$([R+100u_i],[1000v_i])$	2	$([Ru_i],[Ru_i])$
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	$([Ru_i],[R(u_i+0.098+0.004v_i)])$
7	$([Ru_i],[Ru_i])$	8	$([Ru_i],\lfloor 2R/3\sqrt{u_i(4-u_i)}\rfloor)$
9	$([Ru_i],\lceil Ru_i/3\rceil)$	10	$([Ru_i],[Ru_i+0.3R]),\ Ru_i\%6=0$； $([Ru_i],[Ru_i+0.2R]),\ Ru_i\%6\neq 0$
11	Uncorrelated span(2;10)	12	Weakly correlated span(2;10)
13	Strongly correlated span(2;10)		

*注：u_i、v_i在[0，1]上均匀分布。$R=10^3,10^4,10^5,10^6,10^7$。

表 4　6 500 个例子中达到最优上界的例子数

Tab. 4　Sum of the optimal upper bounds in 6 500 instances

上界	上界占优次数[10]	占优上界等于 U_W 的比例/%
U_D	2 957	100
U_{MT2}	3 029	98.94
U_{MTM}	3 409	89.12
$U_{k\max}$	4 466	100
U_W	6 129	100

与关键物品 s 的大小关系确定最小上界 B_k 对应拉格朗日乘子 λ_k 所在区间（$k<s$ 时所在区间为 $[0, P_{k-1}/w_{k-1}]$，$k \geqslant s$ 时所在区间采用二分法搜索得到），然后调用算法 3 计算最小上界 B_k；算法 3 搜索最小上界 B_k，迭代计算过程中调用算法 1 的次数不超过 30 次；算法 1 计算给定拉格朗日乘子 λ 时（E－kKP）的上界，计算量为 $O(n)$。因此，本文研究的（KP01）上界的快速计算方法所需计算量为 $O(n)$。

表 5 列举了不同上界算法分别求解例 3 中不同（KP01）所需平均计算时间，可以看出，本文上界 U_W 的平均计算时间不到上界 $U_{k\max}$ 平均计算时间的一半，全部 6 500 例上界 U_W 的计算时间仅为上界 $U_{k\max}$ 计算时间的 15.1%，其主要原因是给定初始解较好，算法 2、算法 3 的迭代求解效率很高。

表 5　不同方法计算上界平均消耗时间

Tab. 5　Mean time consumption of different upper bound algorithms

(No, R)	平均消耗时间/s				(No, R)	平均消耗时间/s			
	U_{MT2}	U_{MTM}	$U_{k\max}$	U_W		U_{MT2}	U_{MTM}	$U_{k\max}$	U_W
$(1, 10^3)$	0.000 9	0.000 9	0.051 2	0.021 9	$(2, 10^3)$	0.002 0	0.000 9	0.052 2	0.013 3
$(1, 10^4)$	0.000 2	0.000 9	0.057 8	0.020 6	$(2, 10^4)$	0.000 9	0.000 9	0.056 7	0.013 4
$(1, 10^5)$	0.000 2	0.001 9	0.076 7	0.035 8	$(2, 10^5)$	0.000 9	0.001 7	0.058 6	0.012 8
$(1, 10^6)$	0.000 2	0.000 9	0.133 1	0.002 5	$(2, 10^6)$	0.000 9	0.001 9	0.065 9	0.012 7
$(1, 10^7)$	0.000 9	0.000 2	0.132 5	0.002 5	$(2, 10^7)$	0.001 9	0.000 9	0.063 8	0.016 9
$(3, 10^3)$	0.000 9	0.001 9	0.051 4	0.013 6	$(4, 10^3)$	0.000 9	0.001 7	0.149 2	0.003 7
$(3, 10^4)$	0.000 9	0.000 9	0.045 6	0.010 6	$(4, 10^4)$	0.001 9	0.000 2	0.159 4	0.003 9
$(3, 10^5)$	0.000 9	0.000 2	0.050 8	0.011 7	$(4, 10^5)$	0.000 9	0.000 9	0.149 5	0.003 7
$(3, 10^6)$	0.000 9	0.000 9	0.056 9	0.013 9	$(4, 10^6)$	0.000 9	0.001 9	0.159 7	0.003 0
$(3, 10^7)$	0.000 9	0.000 9	0.058 3	0.015 6	$(4, 10^7)$	0.000 2	0.000 9	0.167 0	0.002 8
$(5, 10^3)$	0.002 0	0.000 9	0.050 6	0.004 4	$(6, 10^3)$	0.001 9	0.001 9	0.146 1	0.020 2
$(5, 10^4)$	0.000 9	0.001 9	0.058 6	0.005 3	$(6, 10^4)$	0.000 9	0.000 9	0.142 5	0.015 6
$(5, 10^5)$	0.000 9	0.000 9	0.044 8	0.004 7	$(6, 10^5)$	0.000 9	0.001 6	0.177 7	0.015 6
$(5, 10^6)$	0.000 2	0.000 9	0.050 6	0.003 9	$(6, 10^6)$	0.000 9	0.000 9	0.164 1	0.012 8
$(5, 10^7)$	0.000 9	0.001 1	0.050 1	0.004 7	$(6, 10^7)$	0.001 9	0.000 9	0.173 4	0.013 9
$(7, 10^3)$	0.000 9	0.001 7	0.053 4	0.003 9	$(8, 10^3)$	0.000 9	0.001 9	0.157 7	0.006 4
$(7, 10^4)$	0.000 3	0.001 9	0.053 4	0.003 1	$(8, 10^4)$	0.001 9	0.000 9	0.169 4	0.008 6
$(7, 10^5)$	0.001 9	0.001 9	0.049 4	0.003 9	$(8, 10^5)$	0.000 9	0.000 9	0.155 6	0.005 3
$(7, 10^6)$	0.001 1	0.000 9	0.051 2	0.003 9	$(8, 10^6)$	0.000 9	0.000 9	0.163 3	0.007 5
$(7, 10^7)$	0.000 9	0.001 1	0.057 2	0.003 1	$(8, 10^7)$	0.000 9	0.001 1	0.173 6	0.006 7
$(9, 10^3)$	0.001 1	0.000 2	0.080 0	0.013 0	$(10, 10^3)$	0.001 9	0.000 9	0.068 3	0.010 6

续表

(No, R)	平均消耗时间/s				(No, R)	平均消耗时间/s			
	U_{MT2}	U_{MTM}	$U_{k\max}$	U_W		U_{MT2}	U_{MTM}	$U_{k\max}$	U_W
$(9, 10^4)$	0.000 9	0.001 1	0.088 7	0.009 4	$(10, 10^4)$	0.000 9	0.001 9	0.080 2	0.012 8
$(9, 10^5)$	0.000 9	0.001 9	0.086 6	0.014 8	$(10, 10^5)$	0.000 9	0.000 9	0.078 4	0.016 3
$(9, 10^6)$	0.001 9	0.000 2	0.090 9	0.013 1	$(10, 10^6)$	0.000 9	0.001 9	0.079 8	0.013 9
$(9, 10^7)$	0.001 9	0.000 9	0.093 8	0.014 7	$(10, 10^7)$	0.000 9	0.000 9	0.079 7	0.016 6
$(11, 10^3)$	0.001 1	0.000 2	0.070 0	0.016 4	$(12, 10^3)$	0.001 7	0.000 9	0.072 5	0.012 5
$(11, 10^4)$	0.001 9	0.001 1	0.087 5	0.019 8	$(12, 10^4)$	0.000 9	0.001 9	0.077 3	0.013 6
$(11, 10^5)$	0.000 2	0.000 9	0.069 4	0.012 7	$(12, 10^5)$	0.001 9	0.000 9	0.068 1	0.011 4
$(11, 10^6)$	0.000 9	0.000 9	0.063 4	0.014 4	$(12, 10^6)$	0.000 9	0.001 9	0.073 4	0.010 9
$(11, 10^7)$	0.000 2	0.001 9	0.078 0	0.014 2	$(12, 10^7)$	0.000 9	0.001 6	0.085 0	0.017 5
$(13, 10^3)$	0.000 9	0.001 9	0.066 6	0.013 9	$(13, 10^6)$	0.000 9	0.000 9	0.073 4	0.013 4
$(13, 10^4)$	0.000 9	0.001 9	0.078 0	0.012 2	$(13, 10^7)$	0.000 2	0.000 9	0.077 0	0.015 3
$(13, 10^5)$	0.000 9	0.000 9	0.076 2	0.015 0					

4 0-1背包问题上界的应用

（KP01）的上界可用于构造（KP01）的近似解[10]，从而实现（KP01）的降维。

例4：背包问题中物品价值、重量以及背包容积与例1相同，采用上界算法降低其维数。

解：由公式（2）可知，关键物品$s=11$。采用文献[10]中的方法，得到（KP01）的最优近似解为$x^{(0)}=(1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, \cdots, 0)$，对应的目标函数值为$f_0=465$。从表2可知，精确0-1背包问题的上界$B_9=423$，$B_{11}=456.9<f_0$，因而放入背包的物品数量不等于10时，可行解对应目标函数值都小于465。

放入背包的物品数量等于10时，$\lambda_{10}=0$，后40件物品的任一物品价值都小于前10件物品中任一物品的价值，说明近似解为$x^{(0)}$对应放入背包的10件物品价值之和f_0为最大值。

综上，可确定物品11、物品12、…、物品50不放入背包中，物品1、物品2、…、物品10放入背包中，从而将原（KP01）的变量维数降低到0。

一般地，按照文献[9]中（KP01）的降维方法，比较不同情况下（KP01）的上界与近似解的目标函数值（下界），可以确定一件物品是否应该选择放入背包中，达到（KP01）降维的目的。

5 结论

0-1背包问题的上界是快速精确求解（KP01）的基础。利用上界算法可以构造（KP01）近似解，也可以降维。本文证明了（E－kKP）的上界$f_k(\lambda)$是λ的凸函数，（E－kKP）的最小上界B_k是非负整数型变量k的单峰函数，在此基础上构造了（KP01）上界的一种快速计算方法，计算复杂性为$O(n)$。后续，可以采用文献[23]中线搜索方法代替算法3进一步减少计算量。如何利用本文上界算法快速高效降维，值得进一步深入研究。

参考文献：

[1] MARTELLO S，PISINGER D，TOTH P. Dynamic programming and strong bounds for the 0-1 knapsack problem[J]. Management Science，1999，45(3)：414-424.

[2] LIU W L，GONG Y J，CHEN W N，et al. Coordinated charging scheduling of electric vehicles：A mixed-variable differential evolution approach[J]. IEEE Transactions on Intelligent Transportation Systems，2020 21(12)：5094-5109.

[3] ZHOU S C，XING L N，ZHENG X，et al. A self-adaptive differential evolution algorithm for scheduling a single batch-processing machine with arbitrary job sizes and release times[J]. IEEE

Transactions on Cybernetics, 2021, 51(3): 1430-1442.

[4] ZHAO F Q, HE X, WANG L. A two-stage cooperative evolutionary algorithm with problem-specific knowledge for energy-efficient scheduling of no-wait flowshop problem[J]. IEEE Transactions on Cybernetics, 2021, 51(11): 5291-5303.

[5] ZHAO F Q, ZHANG L X, CAO J, et al. A cooperative water wave optimization algorithm with reinforcement learning for the distributed assembly no-idle flowshop scheduling problem[J]. Computers & Industrial Engineering, 2021, 153(10): 107082.

[6] ZHAO F Q, DI S L, CAO J, et al. A novel cooperative multi-stage hyper-heuristic for combination optimization problems[J]. Complex System Modeling and Simulation, 2021(2): 91-108.

[7] CUNHA A S D, BAHIENSE L, LUCENA A, et al. A new Lagrangian based branch and bound algorithm for the 0-1 knapsack problem[J]. Electronic Notes in Discrete Mathematics, 2010, 36: 623-630.

[8] PISINGER D. A minimal algorithm for the 0-1 knapsack problem[J]. Operations Research, 1997, 45(5): 758-767.

[9] 王正元. 基于状态转移的组合优化方法[M]. 西安: 西安交通大学出版社, 2010: 174-179.

[10] WANG Z Y, ZHANG H, LI Y L. Fast polynomial time approximate solution for the 0-1 knapsack problem[J]. Computational Intelligence and Neuroscience, 2022: 1266529.

[11] DANTZIG G B. Discrete-variable extremum problems[J]. Operations Research, 1957, 5(2): 266-288.

[12] INGARGIOLA G P, KORSH J F. Reduction algorithm for zero-one single knapsack problems[J]. Management Science, 1973, 20(4-part-i): 460-463.

[13] INGARGIOLA G P, KORSH J F. An algorithm for the solution of 0-1 loading problems[J]. Operations Research, 1975, 23(6): 1110-1119.

[14] INGARGIOLA G P, KORSH J F. A general algorithm for one-dimensional knapsack problems[J]. Operations Research, 1977, 25(5): 752-759.

[15] MARTELLO S, TOTH P. An upper bound for the zero-one knapsack problem and a branch and bound algorithm[J]. European Journal of Operational Research, 1977, 1(3): 169-175.

[16] FAYARD D, PLATEAU G. An algorithm for the solution of the 0-1 knapsack problem[J]. Computing, 1982, 28(3): 269-287.

[17] MÜLLER-MERBACH H. An improved upper bound for the zero-one knapsack problem[J]. European Journal of Operational Research, 1978, 2(3): 212-213.

[18] MARTELLO S, TOTH P. A new algorithm for the 0-1 knapsack problem[J]. Management Science, 1988, 34(5): 633-644.

[19] MARTELLO S, TOTH P. Upper bounds and algorithms for hard 0-1 knapsack problems[J]. Operations Research, 1997, 45(5): 768-778.

[20] MARTELLO S, PISINGER D, TOTH P. New trends in exact algorithms for the 0-1 knapsack problem[J]. European Journal of Operational Research, 2000, 123(2): 325-332.

[21] LI W X, LEE J. A faster FPTAS for knapsack problem with cardinality constraint[EB/OL].(2020-12-12)[2024-08-12]. http://arxiv.org/abs/1902.00919.

[22] PISINGER D. Where are the hard knapsack problems[J]. Computers & Operations Research, 2005, 32(9): 2271-2284.

[23] WANG Z Y, GAO L, WANG H Z. A hybrid one dimensional optimization[M]//Electronics, Communications and Networks IV. Boca Raton: CRC Press, 2015: 409-414.

（编辑：于福兴）