A Faster FPTAS for Knapsack Problem with Cardinality Constraint

Wenxin Li[1(✉)], Joohyun Lee[2], and Ness Shroff[3]

[1] Department of ECE, The Ohio State University, Columbus, USA
li.7328@osu.edu
[2] Division of Electrical Engineering, Hanyang University, Seoul, South Korea
joohyunlee@hanyang.ac.kr
[3] Department of ECE and CSE, The Ohio State University, Columbus, USA
shroff.11@osu.edu

Abstract. We study the K-item knapsack problem (*i.e.*, 1.5-dimensional knapsack problem), a generalization of the famous 0–1 knapsack problem (*i.e.*, 1-dimensional knapsack problem) in which an upper bound K is imposed on the number of items selected. This problem is of fundamental importance and is known to have a broad range of applications in various fields. It is well known that, there is no *fully polynomial time approximation scheme* (FPTAS) for the d-dimensional knapsack problem when $d \geq 2$, unless P = NP. While the K-item knapsack problem is known to admit an FPTAS, the complexity of all existing FPTASs have a high dependency on the cardinality bound K and approximation error ε, which could result in inefficiencies especially when K and ε^{-1} increase. The current best results are due to [Mastrolilli and Hutter, 2006], in which two schemes are presented exhibiting a space-time tradeoff–one scheme with time complexity $O(n + Kz^2/\varepsilon^2)$ and space complexity $O(n + z^3/\varepsilon)$, and another scheme that requires a run-time of $O(n + (Kz^2 + z^4)/\varepsilon^2)$ but only needs $O(n + z^2/\varepsilon)$ space, where $z = \min\{K, 1/\varepsilon\}$.

In this paper we close the space-time tradeoff exhibited in [Mastrolilli and Hutter, 2006] by designing a new FPTAS with a running time of $\widetilde{O}(n + z^2/\varepsilon^2)$, while simultaneously reaching a space complexity ($\widetilde{O}$ notation hides terms poly-logarithmic in n and $1/\varepsilon$) of $O(n + z^2/\varepsilon)$. Our scheme provides $\widetilde{O}(K)$ and $O(z)$ improvements on the state-of-the-art algorithms in time and space complexity respectively, and is the *first* scheme that achieves a running time that is *independent* of the cardinality bound K (up to logarithmic factors) under fixed ε. Another salient feature of our algorithm is that it is the *first* FPTAS that achieves better time and space complexity bounds than the very first standard FPTAS *over all parameter regimes*.

1 Introduction

The famous *0–1 knapsack problem* (0–1 KP), also known as the *binary knapsack problem* (BKP), is a classical combinatorial optimization problem which often

C. Kaklamanis and A. Levin (Eds.): WAOA 2020, LNCS 12806, pp. 16–29, 2021.
https://doi.org/10.1007/978-3-030-80879-2_2

arises when there are resources to be allocated within a budget. In addition, the 0–1 knapsack problem can be also viewed as the most fundamental non-trivial *integer linear programming* (ILP) problem, and can be formally formulated as follows:

$$\max \sum_{i \in E} p_i x_i, \tag{1}$$

$$s.t. \sum_{i \in E} w_i x_i \leq W \text{ and } x_i \in \{0, 1\}. \tag{2}$$

The value and size of each item i is called *profit* (p_i) and *weight* (w_i) respectively. For any positive integer m, let $[m] = \{1, 2, \ldots, m\}$, we use set $E = [n]$ to denote the ground set, which includes all possible items. Our goal is to make a binary choice for each item i to maximize the overall profit subject to a budget constraint W. Beyond this basic model, there are several interesting practical extensions and variations of 0–1 KP, readers are referred to [9] for details.

In this paper, we study the *K-item knapsack problem* (KKP), a well known generalization of the famous 0–1 KP that can be formulated as (1)–(2) with the additional constraint $\sum_{i \in E} x_i \leq K$, which means that the number of items in any feasible solutions is upper bounded by K. The KKP can be cast as a special case of the *two-dimensional knapsack problem*, which is a knapsack problem with two different packing constraints. Hence KKP problem can also be interpreted as 1.5-*dimensional knapsack problem* (1.5-KP) [9, p. 269]. Another closely related problem is the *exact K-item knapsack problem* (E-KKP), for which the results in this paper still hold and discussions are included in [18].

The KKP (and E-KKP) represents many practical applications in various fields ranging from *assortment planning* [5] to *multiprocessor task scheduling* [3], and *crowdsourcing* [19]. For example, the worker selection problem in crowdsourcing systems [7], *i.e.*, maximizing opinion diversity in constructing a wise crowd, can be reduced to E-KKP. On the other hand, KKP also appears as a key subproblem in the solutions of several more complicated problems [1,2,6,8,12]. For example, in the *bin packing* problem [6], to apply the ellipsoid algorithm to approximately solve the linear program, the (approximation) algorithm to the KKP is utilized to construct a polynomial time (approximate) separation oracle. In many such practical and theoretical applications, e.g., the *single-sink capacitated K-facility location problem* [1] and the *resource constrained scheduling problem* [8], the subroutine utilized to solve KKP frequently appears to be one of the main complexity bottleneck. These observations and facts motivate our study of designing a faster algorithm for KKP.

Complexity of Knapsack Problems. An FPTAS is highly desirable for NP-hard problems. Unfortunately, it has been shown that there exists no FPTAS for d-dimensional knapsack problem for $d \geq 2$, unless P =NP [11].

1.1 Theoretical Motivations and Contributions

Known Results of KKP. In this paper we focus on FPTAS for KKP (and E-KKP). The first FPTAS for KKP was proposed in [3], by utilizing standard dynamic programming and profit scaling techniques, which runs in $O(nK^2/\varepsilon)$ time and requires $O(n+K^3/\varepsilon)$ space. This algorithm was later improved by [13]. Based on the *hybrid rounding* technique, two alternative FPTASs (denoted by Scheme A and Scheme B) were presented, which significantly accelerate the dynamic programming procedure while exhibiting a space-time tradeoff. More specifically, Scheme A achieves a time complexity of $O(n + Kz^2/\varepsilon^2)$ and space complexity of $O(n + z^3/\varepsilon)$, Scheme B needs $O(n + z^2/\varepsilon)$ space but requires a run-time of $O(n + Kz^2 + z^4/\varepsilon^2)$. We remark that [10] also investigated this problem, under an additional assumption that item profits follow an underlying distribution. This assumption enables the design of a fast algorithm via rounding the item profits adaptively according to the profit distribution.

The current fastest FPTAS (Scheme A) sacrifices its space complexity in order to improve run-time performance. This may not be desirable as the space requirement is often a more serious bottleneck for practical applications than running time [9, p. 168]. Despite the recent widespread applications of the KKP problem [2,5,6,16,17,19], the state-of-the-art complexity results established in [13] have not been improved since then. This lack of progress brings us to our first question: *Is it possible to design a more efficient FPTAS with lower time and/or space complexity to enhance practicality?*

Moreover, while the two schemes in [13] achieve substantial improvements compared with [3], it is worth noting that there exists a hard parameter regime $\mathcal{H} = \{(n, K, \varepsilon) | K = \Theta(n), \varepsilon^{-1} = \Omega(n)\}$, in which existing FPTASs in the literature fail to surpass both the time and space complexity barriers guaranteed by the standard scheme in [3]. For example, the run-time of Scheme B is higher than that of [3]. Hence from a theoretical point of view, it is natural to ask: *Can we design a new FPTAS that has lower time complexity or space complexity than the standard FPTAS* [3] *over all parameter regimes?*

Table 1. Comparisons between different FPTASs

Reference	Time complexity	Space complexity
[3]	$O(\frac{nK^2}{\varepsilon})$	$O(n + \frac{K^3}{\varepsilon})$
[13] (Scheme A)	$O(n + \frac{Kz^2}{\varepsilon^2})$	$O(n + \frac{z^3}{\varepsilon})$
[13] (Scheme B)	$O(n + \frac{Kz^2+z^4}{\varepsilon^2})$	$O(n + \frac{z^2}{\varepsilon})$
This paper [2]	$\widetilde{O}(n + \frac{z^2}{\varepsilon^2})$	$O(n + \frac{z^2}{\varepsilon})$

[1] $z = \min\{K, \varepsilon^{-1}\}$;
[2] [2] As shown in Theorem 2, our time complexity can be refined to $\widetilde{O}(n + z^4 + \frac{z^2}{\varepsilon} \cdot \min\{n, \varepsilon^{-1}\})$.

Our Contributions. As summarized in Table 1, we break the longstanding barrier and answer the aforementioned questions in the affirmative. In particular,

we present a new FPTAS with $\widetilde{O}(n + z^2/\varepsilon^2)$ running time and $O(n + z^2/\varepsilon)$ space requirement, which offers $\widetilde{O}(K)$ and $O(z)$ improvements in time and space complexity respectively. Our FPTAS is the **first** to achieve time complexity that is independent of K (up to logarithmic factors, for a given ε). According to Theorem 2, the time complexity of our algorithm can be indeed refined to $\widetilde{O}(n + z^4 + \frac{z^2}{\varepsilon} \cdot \min\{n, \varepsilon^{-1}\})$. From this refined bound, it can be seen that even in the hard regime $\mathcal{H}$, our algorithm has the same time complexity (up to log factors) as the standard FPTAS [3], while improving its space complexity by a factor of n. This implies that our algorithm is also the **first** FPTAS that outperforms the standard FPTAS [3] over all parameter regimes, thus answering the second question in the affirmative.

Our new scheme also helps to improve the state-of-the-art complexity results of several problems in other fields, owing to the widespread applications of KKP. In [18], we take the resource constrained scheduling problem [8] as an illustrative example.

1.2 Technique Overview

Different from the *hybrid rounding* technique proposed in [13], which simplifies the structure of the input instance and approximately guarantees the objective value, we show that it is possible to achieve a better complexity result solely via the *geometric rounding* in the preprocessing phase. We divide items into two classes according to their profits and present distinct methods for each class of items. To solve the subproblem for items with low profit, we present a continuous relaxation function, using the natural linear programming relaxation and other alternatives based on structured weights and scaled budget constraint. The carefully designed relaxation function well approximates the optimal objective value of the subproblem and allows us to exploit the redundancy among various input. For every new input parameters, the relaxation can be computed in $O(z/\varepsilon)$ time on average. As for items with large profit, our treatment mainly follows from the novel "functional" approximation approach and point of view, which was recently proposed in [4]. As a straightforward generalization of the 0–1 KP, a two dimensional convolution operator is defined. We perform the convolution procedure in parallel planes to reduce the running time. The fact that there are at most z elements with large profits helps us to bound the discretization precision via parameter z, instead of the number of profit functions. Here we adopt a slightly different but rather (unnecessary) sophisticated and tedious presentation via the lens of numerical discretization. We hope that this presentation helps to make the approach more clear (in the context of KKP). Finally, an approximate solution is obtained by appropriately putting these two modules together.

2 Item Preprocessing

Definition 1 (Item Partition). *Let $\mathcal{L}$ and $\mathcal{S}$ denote the set of large and small items, respectively. Item $e \in E$ is called a* small *item if its profit is no more than*

$\varepsilon\mathrm{OPT}$, *otherwise it is called a* large *item*[1], i.e., $\mathcal{S} = \{e \in E | \varepsilon\mathrm{OPT}/K \leq p_e \leq \varepsilon\mathrm{OPT}\}$ *and* $\mathcal{L} = \{e \in E | p_e \in \Xi\}$, *where* $\Xi = [\varepsilon\mathrm{OPT}, \mathrm{OPT}]$. *We further divide* $\mathcal{L}$ *and* $\mathcal{S}$ *into different classes,* $\{\mathcal{L}_i^\dagger\}_{i\in[r_\mathcal{L}]}$ *and* $\{\mathcal{S}_i^\dagger\}_{i\in[r_\mathcal{S}]}$, *where* $\mathcal{L}_i^\dagger = \{e \in \mathcal{L} | p_e \in (\varepsilon(1+\varepsilon)^{i-1}\mathrm{OPT},\ \varepsilon(1+\varepsilon)^i\mathrm{OPT}]\}$ $(i \in [r_\mathcal{L}])$ *and* $\mathcal{S}_i^\dagger = \{e \in \mathcal{S} | p_e \in (\varepsilon(1+\varepsilon)^{-i}\mathrm{OPT},\ \varepsilon(1+\varepsilon)^{-i+1}\mathrm{OPT}]\}$ $(i \in [r_\mathcal{S}])$. *Let* r *denote the number of non-empty classes in* E, *as shown in [18], we have*

$$r = O(\min\{r_\mathcal{L} + r_\mathcal{S}, n\}) = O(\min\{\log(K/\varepsilon)/\varepsilon, n\}) = \widetilde{O}(\min\{1/\varepsilon, n\}). \quad (3)$$

Definition 2 (Geometric Rounding). *Without loss of generality, we can assume that elements in the same class have the same profit value. More specifically, we let* $p_e = p_i^\dagger = \varepsilon(1+\varepsilon)^i\mathrm{OPT}$ $(\forall e \in \mathcal{L}_i)$ *and* $p_e = p_i^\ddagger = \varepsilon(1+\varepsilon)^{-i}\mathrm{OPT}$ $(\forall e \in \mathcal{S}_i)$.

The simplification in Definition 2 does not hurt the solution since it will incur a loss of $O(\varepsilon\mathrm{OPT})$ in the objective value. Let O^* denote the optimal solution, exploiting the simple structure of item profits after item partition and profit rounding, we are able to derive the following more fine-grained bound on the size of $|O^* \cap \mathcal{L}|$ and $\mathcal{S}$. Its proof is deferred to [18].

Proposition 1. *There are no more than* $|O^* \cap \mathcal{L}| \leq z$ *large items in the optimal solution set* O^*. *Without loss of generality, we can assume that the number of small items* $|\mathcal{S}| = O(\min\{K \cdot \log(K/\varepsilon)/\varepsilon, n\}) = \widetilde{O}(\min\{K/\varepsilon, n\})$.

3 Algorithm for Large Items

To approximately solve the K-item knapsack problem on ground set E, the first step of our approach is to divide this problem into two smaller KKP problems, which are defined on the large item set $\mathcal{L}$ and small item set $\mathcal{S}$ respectively. In this section we study the subproblem on $\mathcal{L}$, which is the same as the original problem, except that the ground set is substituted by $\mathcal{L}$ and the cardinality upper bound k must be no less than z.

3.1 An Abstract Algorithm Based on Convolution

We first define the profit function $\varphi_{(\cdot)}(\cdot,\cdot) : 2^\mathcal{L} \times \mathbb{R}^+ \times [z] \to \mathbb{R}^+$. From the definition we can see that $\varphi_\mathcal{L}(\omega, k)$ is equal to the optimal objective value of the subproblem considered in this section.

Definition 3 (Profit function [4]). *For any given set* $T \subseteq E$, *real number* ω, *and integer* k, $\varphi_T(p,k)$ *is given by* $\varphi_T(\omega,k) = \max\{\sum_{e\in T'} p_e | \sum_{e\in T'} w_e \leq \omega, |T'| \leq k, T' \subseteq T \subseteq E\}$, *which denotes the optimal objective value of the* K*-item knapsack problem that is defined on set* T, *while the budget and cardinality are* ω, k *respectively.*

[1] We discuss the method of obtaining OPT in [18].

Our objective is to approximately compute matrix $\mathbf{A}_{\mathcal{L}} = \{\varphi_{\mathcal{L}}(\omega, k)\}_{\omega \in X, k \in [z]}$, in which the value of X will be specified in Sect. 3.3. This matrix plays an important role in our final item combination procedure, as we will show later in Sect. 5. To compute the profit function efficiently, we introduce the following *inverse weight function* $\phi_{(\cdot)}(\cdot,\cdot) : 2^{\mathcal{L}} \times \Xi \times [z] \to \mathbb{R}^+$, which is one of the key ingredients in computing the profit function.

Definition 4 (Inverse weight function). *For any given set $T \subseteq E$, real number p and integer k, $\phi_T(p,k)$ is given by $\phi_T(p,k) = \min\{\sum_{e\in T'} w_e | \sum_{e\in T'} p_e \geq p, |T'| \leq k, T' \subseteq T\}$, which characterizes the minimum possible total weights under which there exists a subset of T with total profit being no less than p and cardinality no more than k.*

An immediate consequence of Definitions 3 and 4 is that we can easily obtain the value of $\varphi_{\mathcal{L}}(\omega, k)$ based on ϕ, *i.e.*, via equation $\varphi_{\mathcal{L}}(\omega, k) = \sup\{p \in \mathbb{R}^+ | \phi_{\mathcal{L}}(p,k) \leq \omega\}$. Therefore it suffices to derive the inverse weight function $\phi_{\mathcal{L}}(\cdot,\cdot)$ to compute $\mathbf{A}_{\mathcal{L}}$.

Algorithm for Computing $\phi_{\mathcal{L}}$. If we partition the large item set $\mathcal{L}$ into ℓ disjoint subsets as $\mathcal{L} = \cup_{i=1}^{\ell} \mathcal{L}^{(i)}$, then $\phi_{\mathcal{L}}$ can be computed by performing convolution operations sequentially. We specify the details of the algorithm in [18]. The *convolution operation* $\otimes$ is defined as follows.

Definition 5 (Two Dimensional Convolution Operator $\otimes$). *For any two disjoint sets $S_1, S_2 \subseteq E$, we use $(\phi_{S_1} \otimes \phi_{S_2})(\cdot,\cdot)$ to denote the convolution of functions $\phi_{S_1}(\cdot,\cdot)$ and $\phi_{S_2}(\cdot,\cdot)$, then it can be represented as,*

$$\begin{aligned}(\phi_{S_1} \otimes \phi_{S_2})(p,k) &= \min\left\{\phi_{S_1}(p_1,k_1) + \phi_{S_2}(p_2,k_2) \middle| k_1 + k_2 \leq k, p_1 + p_2 \geq p\right\} \\ &\equiv \phi_{S_1 \cup S_2}(p,k).\end{aligned}$$

Under this notation, function $\phi_{\mathcal{L}}(\cdot,\cdot)$ defined on $\mathcal{L}$ can be represented as $\phi_{\mathcal{L}}(p,k) = (\otimes_{i=1}^{\ell} \phi_{\mathcal{L}^{(i)}})(p,k)$. It is important to remark that the algorithm is a rather general description of the convolution procedure, and the partition scheme should be further specified. Generally speaking, different partition schemes will induce different complexity results. For example, if we partition $\mathcal{L}$ into singletons, *i.e.*, $\mathcal{L}^{(i)} = \{e_i\}$ and $\ell = |\mathcal{L}|$, then $\phi_{\mathcal{L}}(p,K) = (\otimes_{i=1}^{|\mathcal{L}|} \phi_{\{e_i\}})(p,K)$. In this case, the algorithm is equivalent to the standard dynamic programming paradigm. In each stage we are in charge of making the decision of whether to include item e_i or not. Here we divide $\mathcal{L}$ in the same way as Definition 1, *i.e.*, $\mathcal{L}^{(i)} = \mathcal{L}_i^{\dagger}, \forall i \in [r_{\mathcal{L}}]$.

3.2 Discretizing the Function Domain

At the current stage, it is worth pointing out that in the convolution operation between inverse weight functions, the profit variable p appears as a decision variable that varies continuously in Ξ. In addition, we are not able to obtain the closed form solution of the convolution operation analytically. The solution is to

transform the problem into a computationally tractable one via discretization, then compute an (approximate) solution utilizing the computable version.

Discretizing the Profit Space. To implement the convolution in polynomial time, we discretize the interval Ξ with the points $\{x_i\}_{i\in[m]}$ as $X = \{x_i : \varepsilon\mathrm{OPT} = x_1 < x_2 < \ldots < x_{m-1} < x_m = \mathrm{OPT}\} \subseteq \Xi$. We denote the *discretization parameter* of X by *discretization parameter* $\delta_X = \max_{1\le i\le m-1}\{x_{i+1} - x_i\}$. To tackle the computational challenge induced by the continuity of profit p, we execute the convolution operation over the discrete functions that are defined on $X \times [z]$, *i.e.*,

$$(\phi_{S_1} \otimes \phi_{S_2})^X(p,k) = \min_{p_1,p_2\in X}\left\{\phi^X_{S_1}(p_1,k_1) + \phi^X_{S_2}(p_2,k_2)\middle| k_1+k_2\le k, p_1+p_2\ge p\right\}.$$

More specifically, we start with functions $\phi^X_{\mathcal{L}^{(i)}}$, and compute $\phi^X_{\bigcup_{j=1}^{i}\mathcal{L}^{(j)}}$ iteratively until $\phi^X_{\mathcal{L}}$ is obtained. In general, function $\phi^X_{\bigcup_{i\in I}\mathcal{L}^{(i)}}(\cdot,\cdot) \equiv (\otimes_{i\in I}\phi^X_{\mathcal{L}^{(i)}})(\cdot,\cdot)$ for any $I \subseteq [\ell]$. The discrete profit function $\varphi^X_S(\cdot,\cdot)$ can also be recovered by its relation with the inverse weight function, *i.e.*, $\varphi^X_S(\omega,k) = \max\{p \in X : \phi^X_S(p,k) \le \omega\}, \forall S \subseteq E$.

Convergence Behaviour of $\varphi^X(\cdot,\cdot)$. We first show point-wise convergence of $\{\varphi^X_{(\cdot)}(\cdot,\cdot)\}_X$ towards $\varphi_{(\cdot)}(\cdot,\cdot)$ when δ_X goes to zero. The proof of Lemma 1 is deferred to [18]. It is worth pointing out that the straightforward intuition that convergence occurs if discretization is small, may not always hold. Indeed we can verify that the weight function ϕ^X may not converge to ϕ through the example in [18].

Lemma 1. *For any finite index set I and ω, k, we have $\lim_{\delta_X\to 0}\varphi^X_{\bigcup_{i\in I}\mathcal{L}^{(i)}}(\omega,k) = \varphi_{\bigcup_{i\in I}\mathcal{L}^{(i)}}(\omega,k)$ for fixed ω, k.*

The theoretical convergence of $\varphi^X(\cdot,\cdot)$ ensures the near-optimality of the solution obtained by discretization, as long as X is dense enough in Ξ. However, what matters greatly is the *order of the accuracy*, which refers to how rapidly the error decreases in the limit as the discretization parameter tends to zero. The formal definition of the convergence speed of discretization methods is given in [18]. This speed is directly related to the complexity of our algorithm. From the following lemma, we can conclude that the method of discretizing X by a uniform grid set converges with order 1, as $\delta_X = O(1/|X|)$ for uniform grid set X. The proof of Lemma 2 is deferred to [18].

Lemma 2. *Let $\phi^X_{\mathcal{L}}$ be the weight function, then for any given budget $\omega \le W$, cardinality upper bound $k \le z$, and discretization set X, we have $|\varphi^X_{\mathcal{L}}(\omega,k) - \varphi_{\mathcal{L}}(\omega,k)| \le C\delta_X$, where the coefficient $C = z+1$. As a consequence, $|X|$ must be of order $\Omega(z/\varepsilon)$ to ensure an error of order $O(\varepsilon\mathrm{OPT})$.*

3.3 Fast Convolution Algorithm

In this subsection we settle the problem of designing a fast convolution algorithm, which is the last remaining issue that has a critical impact on the efficiency of the

algorithm for large items. To this end, we show an inherent connection between convolution results under different inputs p and k, which is formally described in Lemma 3. Owing to this observation, we are able to remove a large amount of redundant calculations when facing new input parameters. To start with, we first sort items in each $\mathcal{L}_i^\dagger$ in non-increasing order of weights, which takes $O(z \log z)$ time. We define the optimum index function as follows.

Definition 6 (Optimum index function). *$\psi : X \times [K] \to [K]$ is defined as,*

$$\psi(p,k) = \operatorname{argmin}\Big\{\theta \in [k] \Big| \phi^X_{\mathcal{L}_a^\dagger}(\max\{x \in X : x \le \theta \cdot p_a^\dagger\}, \theta) + \phi^X_S(\max\{x \in X : x \le p - \theta \cdot p_a^\dagger\}, k-\theta)\Big\} \ (p \in X). \tag{4}$$

Here (4) benefits from the partition in which all items in the same set $\mathcal{L}_i^\dagger$ have equal profit value. Specifically, when we derive the result of $(\phi^X_{\mathcal{L}_a^\dagger} \otimes \phi^X_S)(p,k)$, there is indeed only one decision variable θ, *i.e.*, the number of elements selected from $\mathcal{L}_i^\dagger$, that should be figured out. Hence, we denote the optimal value of θ by the index function ψ. Our primary objective is then reduced to figure out all the indices $\{\psi(p,k)\}_{p \in X, k \in [z]}$, for which we give a graphic illustration in Fig. 1(a). It can be regarded as finding *column minimums* in the cube, here column minimum refers to the optimal indices defined in Definition 6.

Consider the Problem in Parallel Slices. We divide the cube into parallel slices. Consider slice

$$H = \Big\{(p,k) \Big| p = p_0 + \zeta\lambda_a, k = k_0 + \zeta\Big\} \bigcap \Big(\Xi \times [0,z]\Big), \tag{5}$$

as shown in Fig. 1(b), where (p_0, k_0) denotes the boundary point of slice H, hence $p_0 k_0 = 0$, and ζ represents the drift of point (p,k) from boundary. It can be seen that the angle between slice H and the frontal plane is equal to $\arctan \lambda_a^{-1}$, and there are $O(|X|) = O(z/\varepsilon)$ such parallel slices in the cube. On the other hand, plugging (5) into (4), the index function can be simplified to

$$\chi_H(\zeta) = \operatorname{argmin}\Big\{\theta \in [z] \Big| \phi^X_{\mathcal{L}_a^\dagger}(\lambda_a\theta, \theta) + \phi^X_S(p_0 + \lambda_a[\zeta - \theta], k_0 + [\zeta - \theta])\Big\}.$$

Bounded "Gradient" of χ_H. Without loss of generality we could assume that there exists an integer $\tau_a \in \mathbb{Z}^+$ such that $p_a^\dagger = \tau_a \cdot \frac{\varepsilon \mathrm{OPT}}{z}$, otherwise we can always modify $p_a^\dagger$ by an $O(\frac{\varepsilon \mathrm{OPT}}{z})$ additive factor to meet this criteria while inducing a $O(\varepsilon \mathrm{OPT})$ loss in the objective function. Consequently we have $\lambda_a = \tau_a \varepsilon \mathrm{OPT}$. We consider the case when Ξ is discretized by the uniform grid set $X = \{i \cdot \frac{\varepsilon \mathrm{OPT}}{z} | i \in [z/\varepsilon]\}$. Then the following key observation about the distribution of column minima in slice H holds. The proof of this lemma is deferred to [18].

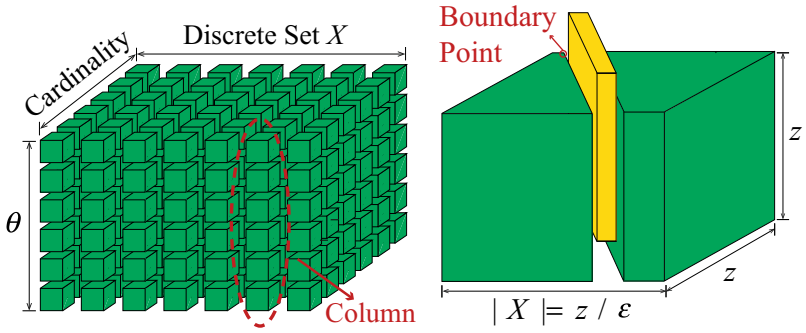
Lemma 3. *Consider two columns in H that are indexed by ζ_1 and ζ_2. We have $\frac{\chi_H(\zeta_2) - \chi_H(\zeta_1)}{\zeta_2 - \zeta_1} \le 1$.*

Divide-and-Conquer on Slice H. In Lemma 3, we establish an upper bound on the growth rate of the index function. Taking advantage of this lemma, we are able to reduce the size of the searching space in one column, given that we have figured out the optimum indices at some other columns in the slice H. More specifically, consider columns indexed by $\zeta_1 \le \zeta_2 \le \zeta_3$, the information of $\chi_H(\zeta_1)$ and $\chi_H(\zeta_3)$ indeed provide two cutting planes to help us locate $\chi_H(\zeta_2)$ in a smaller interval $[\chi_H(\zeta_3) + \zeta_2 - \zeta_3, \chi_H(\zeta_1) + \zeta_2 - \zeta_1]$.

Inspired by this observation, for any slice in the form of (5), we design a *divide-and-conquer* procedure to compute the optimum indices efficiently. We start with a recursive call to determine the optimum indices of all the even-indexed columns. Here a column is called even (odd) column if and only if its corresponding ζ value in (5) is even (odd). Then for each odd column $\chi_H(2i)$, it can be computed by enumerating the interval $[\chi_H(2i+1) - 1, \chi_H(2i-1) + 1]$. The details are specified in [18].

The time complexity of computing the index function for a single slice is summarized in the following proposition, whose proof is presented in [18].

Proposition 2. *It takes* $O(z \log z) = \widetilde{O}(z)$ *time to compute* $\chi_H(\cdot)$.



(a) Searching Space of $\phi^X_{\mathcal{L}^\dagger_a} \otimes \phi^X_S$ (b) Divide-and-Conquer on H

Fig. 1. Graphic illustrations of the convolution operation

Fast Convolution Operation. The details of the convolution operation are specified in [18]. The following lemma summarizes the complexity of our algorithm, the proof is presented in [18].

Lemma 4. *It takes* $O(n)$ *space and*

$$O(n + (z^2 \log z/\varepsilon) \cdot \min\{\log(1/\varepsilon)/\varepsilon, n\}) = O(n) + \widetilde{O}(\min\{z^2/\varepsilon^2, nz^2/\varepsilon\})$$

time to complete the convolution operation.

Generally speaking, for given $p \in X$ and $k \in \mathbb{Z}^+$, it requires $O(z \cdot |X|)$ arithmetic operations to compute $(\phi_{S_1} \otimes \phi_{S_2})(p,k)$ if we enumerate all possible pairs of (p_1, k_1), which further results in a total complexity of $O(z^2|X|^2)$ for operator $\otimes$. Compared with our Algorithm, this is unnecessarily inefficient, since it restarts all the arithmetic operations when the input parameters varies.

4 Continuous Relaxation for Small Items

In Sect. 3 we have shown how to approximately select the most profitable large items under any given budget and cardinality constraints. One important task left is to solve the subproblem with only small items involved. In this section we show how to approximately solve this subproblem efficiently. Similar to Definition 4, the profit function of small items, $\varphi_{\mathcal{S}}(\cdot,\cdot) : \mathbb{R}^+ \times [K] \to \mathbb{R}^+$, is given by $\varphi_{\mathcal{S}}(\omega,k) = \max\{\sum_{e\in\mathcal{S}} p_e x_e | \sum_{e\in\mathcal{S}} x_e \le k, \sum_{e\in\mathcal{S}} w_e x_e \le \omega, x_e \in \{0,1\}\}$. The main spirit of our approach for small items is similar to that of Sect. 3, *i.e.*, find a new function $\widetilde{\varphi}_{\mathcal{S}}$, which is a good approximation of $\varphi_{\mathcal{S}}$ and is economical in computations. To this end, our main result in this subsection is formally stated in the following lemma. We leave the proof of this lemma in [18], which relies on our analysis in the following two subsections.

Lemma 5. *There exists a relaxation* $\widetilde{\varphi}_{\mathcal{S}}(\cdot,\cdot) : \mathbb{R}^+ \times [K] \to \mathbb{R}^+$ *that satisfies* $|\widetilde{\varphi}_{\mathcal{S}} - \varphi_{\mathcal{S}}| = O(\varepsilon \mathrm{OPT})$*, and the corresponding matrix* $\mathbf{A}_{\mathcal{S}} = \{\widetilde{\varphi}_{\mathcal{S}}(\omega,k) | \omega \in \mathcal{W}, k \in \mathcal{K}\}$ *can be computed within* $\widetilde{O}(n + z^4 + \min\{\frac{z^2}{\varepsilon^2}, \frac{nz}{\varepsilon}\})$ *time when* $|\mathcal{W}| = O(\varepsilon^{-1})$ *and* $|\mathcal{K}| = O(z)$*, while requiring* $O(z/\varepsilon)$ *space.*

One question that may arise is the following: can the methods in Sect. 3 still work for the small item set $\mathcal{S}$, *i.e.*, can we apply the algorithm for large items over $\mathcal{S}$ and use the output discrete function as an approximation of $\varphi_{\mathcal{S}}$? Unfortunately it can be verified that $O(n) + \widetilde{O}(K^2/\varepsilon^2)$ time is required, which is significantly high especially when K is large, and fails to provide the desired complexity result. This is because there could be many more small items than large items, which will result in a larger searching space.

To construct the new function $\widetilde{\varphi}_{\mathcal{S}}$, we turn to the continuous relaxation of the subproblem, as the continuous problem is much easier to deal with. More importantly, the boundness of small item profits will ensure that the gap between the optimal values of the two problems is small.

4.1 Continuous Relaxation Design and Error Analysis

Designing $\widetilde{\varphi}_{\mathcal{S}}$. In our algorithm, we let

$$\widetilde{\varphi}_{\mathcal{S}}(\omega,k) = \widetilde{\varphi}^{(1)}_{\mathcal{S}}(\omega,k) \cdot \mathbb{1}_{\{K \le \varepsilon^{-1}\}} + \widetilde{\varphi}^{(2)}_{\mathcal{S}}(\omega,k) \cdot \mathbb{1}_{\{K > \varepsilon^{-1}\}},$$

in which the two building block functions $\widetilde{\varphi}^{(i)}_{\mathcal{S}}$ $(i = 1, 2)$ are specified in the following definition. The first function $\widetilde{\varphi}^{(1)}_{\mathcal{S}}$ is the most natural linear programming relaxation of $\widetilde{\varphi}_{\mathcal{S}}$, in which all the integer variables are relaxed to real numbers

in $[0,1]$. In the second function $\widetilde{\varphi}_{\mathcal{S}}^{(2)}$, we only relax variables corresponding to elements in $\mathcal{S}_\omega$, while the element weights are rounded to an integer power of $(1+\varepsilon)$, and the budget is given by $(1-\varepsilon)\omega$ instead of ω.

Definition 7 (Definition of $\widetilde{\varphi}_{\mathcal{S}}^{(1)}, \widetilde{\varphi}_{\mathcal{S}}^{(2)}$). *Functions $\widetilde{\varphi}_{\mathcal{S}}^{(i)}(\cdot,\cdot) : \mathbb{R}^+ \times [K] \to \mathbb{R}^+$ $(i=1,2)$ are constructed as*

$$\widetilde{\varphi}_{\mathcal{S}}^{(1)}(\omega,k) = \max\Big\{ \sum_{e\in\mathcal{S}} p_e x_e \Big| \sum_{e\in\mathcal{S}} x_e \le k, \sum_{e\in\mathcal{S}} w_e x_e \le \omega, x_e \in [0,1] \Big\},$$

and

$$\widetilde{\varphi}_{\mathcal{S}}^{(2)}(\omega,k) = \max_{0\le t\le k} \Big\{ \widetilde{\varphi}_{\mathcal{S}_\omega}^{(2)}(\omega, k-t) + \widetilde{\varphi}_{\bar{\mathcal{S}}_\omega}^{(2)}(t) \Big\}.$$

Here set $\bar{\mathcal{S}}_\omega = \{e \in \mathcal{S} | w_e \le \varepsilon\omega/K\}$ represents the set of elements in $\mathcal{S}$ with weight less than a threshold $\varepsilon\omega/K$, and $\mathcal{S}_\omega = (\mathcal{S}\backslash\bar{\mathcal{S}}_\omega)\backslash\{e \in \mathcal{S} | w_e > \omega\}$. Function $\widetilde{\varphi}_{\bar{\mathcal{S}}_\omega}^{(2)}(t) = \max\{\sum_{e\in T} p_e | T \subseteq \bar{\mathcal{S}}_\omega, |T| \le t\}$ denotes the total profits of the top t elements in $\bar{\mathcal{S}}_\omega$. In addition, $\widetilde{\varphi}_{\mathcal{S}_\omega}^{(2)}(\omega,t)$ is given by

$$\widetilde{\varphi}_{\mathcal{S}_\omega}^{(2)}(\omega,t) = \max_{x_e\in[0,1]} \Big\{ \sum_{e\in\mathcal{S}_\omega} p_e x_e \Big| \sum_{e\in\mathcal{S}_\omega} x_e \le t, \sum_{e\in\mathcal{S}_\omega} w'_e x_e \le (1-\varepsilon)\omega \Big\} \tag{6}$$

where $w'_e = (\omega(1+\varepsilon)^{\lceil \log_{(1+\varepsilon)}(\frac{\varepsilon K w_e}{\omega})\rceil})/(K\varepsilon)$ and $\lceil\cdot\rceil$ refers to the ceiling function.

Error of Approximation. We show that $\widetilde{\varphi}_{\mathcal{S}}$ provides a good approximation of $\varphi_{\mathcal{S}}$ in the following lemma. The proof of Lemma 6 is presented in [18].

Lemma 6. *The differences between functions $\widetilde{\varphi}_{\mathcal{S}}$ and $\varphi_{\mathcal{S}}$ is bounded as $|\widetilde{\varphi}_{\mathcal{S}}(\omega,k) - \varphi_{\mathcal{S}}(\omega,k)| \le 4\varepsilon\mathrm{OPT}$.*

Obtain the Final Solution Set. Recall that our ultimate objective is to retrieve an solution set that has near optimal objective function value. To this end, for the subproblem of small items, we can solve the continuous problem and return the corresponding integer components $S = \{e | x_e^* = 1\}$ as an approximate solution, where $\mathbf{x}^*$ denotes the optimal fractional solution.

4.2 Computing $\widetilde{\varphi}_{\mathcal{S}}$ Efficiently

In this subsection, we consider how to efficiently compute the function $\widetilde{\varphi}_{\mathcal{S}}(\cdot,\cdot)$. More specifically, our objective is to compute set $\{\widetilde{\varphi}_{\mathcal{S}}(\omega,k) | \omega \in \mathcal{W}, k \in \mathcal{K}\}$, for given $\mathcal{K} \in \mathbb{Z}^{|\mathcal{K}|}$ and $\mathcal{W} \in \mathbb{R}^{|\mathcal{W}|}$.

Computing Relaxation $\widetilde{\varphi}_{\mathcal{S}}^{(1)}(\cdot,\cdot)$. One straightforward approach is to utilize the linear time algorithm [3,14,15] to solve $\widetilde{\varphi}_{\mathcal{S}}^{(1)}$ under distinct parameters in $\mathcal{W},\mathcal{K}$ separately, which will result in a total complexity of $O(|\mathcal{S}| \cdot |\mathcal{K}| \cdot |\mathcal{W}|) =$

$O(\frac{z}{\varepsilon} \cdot \min\{K/\varepsilon, n\})$. Note that under this approach, the complexity has a high dependence on the parameter K.

Computing Relaxation $\widetilde{\varphi}_{\mathcal{S}}^{(2)}(\cdot,\cdot)$. Let $f_t(\omega,k) = \widetilde{\varphi}_{\mathcal{S}_\omega}^{(2)}(\omega,k-t) + \widetilde{\varphi}_{\bar{\mathcal{S}}_\omega}^{(2)}(t)$, then $\widetilde{\varphi}_{\mathcal{S}}^{(2)}(\omega,k) = \max_{0\le t\le k} f_t(\omega,k)$ according to Definition 7. We first claim the following key observation with regard to $\{f_t(\omega,k)\}_{0\le t\le k}$. Basically this concavity property enables us to compute $\widetilde{\varphi}_{\mathcal{S}}^{(2)}$ using $O(\log k)$ calls to the subroutine of computing $f_t(\omega,k)$. The proof is presented in [18].

Theorem 1 (Concavity of f_t). *The sequence $\{f_t(\omega,k)\}_{t\in[k]}$ is a concave sequence with respect to t. As a result, $\widetilde{\varphi}_{\mathcal{S}}^{(2)}(\omega,k)$ can be computed in $O(\mathcal{T}_f \log k) = \widetilde{O}(\mathcal{T}_f)$ time, where $\mathcal{T}_f$ represents the worst case time complexity for computing $f_t(\omega,k)$ under fixed values of t,ω,k.*

At the current stage, the problem of computing $\widetilde{\varphi}_{\mathcal{S}}^{(2)}(\omega,k)$ has been shown to have the same time complexity (up to a factor of $O(\log k)$) as computing $f_t(\omega,k)$, which is further determined by the following two subroutines– calculating $\widetilde{\varphi}_{\mathcal{S}_\omega}^{(2)}(\omega,t)$ and $\widetilde{\varphi}_{\bar{\mathcal{S}}_\omega}^{(2)}(t)$. We dualize the budget constraint as

$$L(\mu,\omega,t) = \max_{x_e\in[0,1]} \Big\{ \sum_{e\in\mathcal{S}_\omega} (p_e - \mu w_e)x_e + \mu\omega \Big| \sum_{e\in\mathcal{S}_\omega} x_e \le t \Big\},$$

which helps to figure out the first function under multiple input parameters. Indeed we can always apply binary search on set

$$\mathcal{B}' = \Big\{ (1+\varepsilon)^b \cdot \frac{(1+\varepsilon)^c - 1}{(1+\varepsilon)^d - 1} \Big| |b|,|c|,|d| \le \log(K/\varepsilon)/\varepsilon, \text{ and } b,c,d\in\mathbb{Z} \Big\}$$

to figure out the optimal multiplier μ^*, owing to the convexity of the Lagrange function. As for function $\widetilde{\varphi}_{\bar{\mathcal{S}}_\omega}^{(2)}(t)$, we take advantage of the fact that $\{\widetilde{\varphi}_{\bar{\mathcal{S}}_\omega}^{(2)}(t)\}_{t\in[|\mathcal{S}|]}$ can be computed together to reduce running time, under the same budget ω. We present the details and complexity analysis in [18].

5 Putting the Pieces Together–Combining Small and Large Items

In our main algorithm, we utilize our two algorithms established in Sect. 3 and 4 as two basic building blocks, to approximately enumerate all the possible profit allocations among $\mathcal{L}$ and $\mathcal{S}$. The details are specified in [18] and performance guarantee is given by Theorem 2. We remark that set X' in the algorithm is not equal to X but a subset of X, and is given by $X' = \{i\varepsilon\mathrm{OPT} | i \in [1/\varepsilon]\}$. The proof of theorem 2 is deferred to [18].

Theorem 2. *The total profits of items in set S_o given in the main algorithm is no less than $p(S_o) \ge (1-O(\varepsilon))\mathrm{OPT}$, while requires $\widetilde{O}(n+z^4+\frac{z^2}{\varepsilon}\cdot\min\{n,\varepsilon^{-1}\})$ time, which is within the order of $\widetilde{O}(n+\frac{z^2}{\varepsilon^2})$.*

6 Conclusion

In this paper we proposed a new FPTAS for the *K-item knapsack problem* (and *Exactly K-item knapsack problem*) that exhibits $\widetilde{O}(K)$ and $O(z)$ improvements in time and space complexity respectively, compared with the state-of-the-art [13]. More importantly, our result suggests that for a fixed value of ε, an $(1-\varepsilon)$-approximation solution of KKP can be computed in time asymptotically independent of cardinality bound K. Our scheme is also the first FPTAS that achieves better time and space complexity (up to logarithmic factors) than the standard dynamic programming scheme in [3] over all parameter regimes.

References

1. Aardal, K., van den Berg, P.L., Gijswijt, D., Li, S.: Approximation algorithms for hard capacitated k-facility location problems. Eur. J. Oper. Res. **242**(2), 358–368 (2015)
2. Ahuja, R.K., Orlin, J.B., Pallottino, S., Scaparra, M.P., Scutellà, M.G.: A multi-exchange heuristic for the single-source capacitated facility location problem. Manage. Sci. **50**(6), 749–760 (2004)
3. Caprara, A., Kellerer, H., Pferschy, U., Pisinger, D.: Approximation algorithms for knapsack problems with cardinality constraints. Eur. J. Oper. Res. **123**(2), 333–345 (2000)
4. Chan, T.M.: Approximation schemes for 0–1 knapsack. In: SOSA (2018)
5. Désir, A., Goyal, V., Segev, D.: Assortment optimization under a random swap based distribution over permutations model. In: EC, pp. 341–342 (2016)
6. Epstein, L., Levin, A.: Bin packing with general cost structures. Math. Program. **132**(1–2), 355–391 (2012)
7. Gong, X., Shroff, N.: Incentivizing truthful data quality for quality-aware mobile data crowdsourcing. In: MobiHoc (2018)
8. Jansen, K., Porkolab, L.: On preemptiveresource constrained scheduling: polynomial-time approximation schemes. SIAM J. Discrete Math. **20**(3), 545–563 (2006)
9. Kellerer, H., Pferschy, U., Pisinger, D.: Knapsack Problems. Springer, Berlin (2004). http://dx.doi.org/10.1007/978-3-540-24777-7
10. Krishnan, B.K.: A multiscale approximation algorithm for the cardinality constrained knapsack problem. Ph.D. thesis, Massachusetts Institute of Technology (2006)
11. Magazine, M.J., Chern, M.S.: A note on approximation schemes for multidimensional knapsack problems. Math. Oper. Res. **9**(2), 244–247 (1984)
12. Martello, S., Pisinger, D., Toth, P.: Dynamic programming and strong bounds for the 0–1 knapsack problem. Manage. Sci. **45**(3), 414–424 (1999)
13. Mastrolilli, M., Hutter, M.: Hybrid rounding techniques for knapsack problems. Discrete Appl. Math. **154**(4), 640–649 (2006)
14. Megiddo, N.: Linear programming in linear time when the dimension is fixed. J. ACM **31**(1), 114–127 (1984)
15. Megiddo, N., Tamir, A.: Linear time algorithms for some separable quadratic programming problems. Oper. Res. Lett. **13**(4), 203–211 (1993)

16. Nobibon, F.T., Leus, R., Spieksma, F.C.: Optimization models for targeted offers in direct marketing: exact and heuristic algorithms. Eur. J. Oper. Res. **210**(3), 670–683 (2011)
17. Soldo, F., Argyraki, K., Markopoulou, A.: Optimal source-based filtering of malicious traffic. IEEE/ACM Trans. Networking **20**(2), 381–395 (2012)
18. Wenxin, L., Joohyun Lee, N.S.: A faster fptas for knapsack problem with cardinality constraint. arXiv:1902.00919
19. Wu, T., Chen, L., Hui, P., Zhang, C.J., Li, W.: Hear the whole story: towards the diversity of opinion in crowdsourcing markets. Proc. VLDB Endowment **8**(5), 485–496 (2015)