


Weakly Approximating Knapsack in Subquadratic Time

Lin Chen ✉ 🏠 


Zhejiang University, Hangzhou, China

Jiayi Lian ✉ 🏠 

Zhejiang University, Hangzhou, China

Yuchen Mao ✉ 🏠 

Zhejiang University, Hangzhou, China

Guochuan Zhang ✉ 🏠 

Zhejiang University, Hangzhou, China

Abstract

We consider the classic Knapsack problem. Let t and OPT be the capacity and the optimal value, respectively. If one seeks a solution with total profit at least $\text{OPT}/(1+\varepsilon)$ and total weight at most t , then Knapsack can be solved in $\tilde{O}(n + (\frac{1}{\varepsilon})^2)$ time [Chen, Lian, Mao, and Zhang '24][Mao '24]. This running time is the best possible (up to a logarithmic factor), assuming that $(\min, +)$ -convolution cannot be solved in truly subquadratic time [Künnemann, Paturi, and Schneider '17][Cygan, Mucha, Węgrzycki, and Włodarczyk '19]. The same upper and lower bounds hold if one seeks a solution with total profit at least OPT and total weight at most $(1+\varepsilon)t$. Therefore, it is natural to ask the following question.

If one seeks a solution with total profit at least $\text{OPT}/(1+\varepsilon)$ and total weight at most $(1+\varepsilon)t$, can Knapsack be solved in $\tilde{O}(n + (\frac{1}{\varepsilon})^{2-\delta})$ time for some constant $\delta > 0$?

We answer this open question affirmatively by proposing an $\tilde{O}(n + (\frac{1}{\varepsilon})^{7/4})$ -time algorithm.

2012 ACM Subject Classification Theory of computation \rightarrow Approximation algorithms analysis

Keywords and phrases Knapsack, FPTAS

Funding *Yuchen Mao*: Supported by National Natural Science Foundation of China [Project No. 62402436].

Guochuan Zhang: Supported by National Natural Science Foundation of China [Project No. 12271477].

1 Introduction

In the Knapsack problem, one is given a knapsack of capacity t and a set of n items. Each item i has a weight w_i and a profit p_i . The goal is to select a subset of items such that their total weight does not exceed t , and their total profit is maximized.

Knapsack is one of Karp's 21 NP-complete problems [27], which motivates the study of approximation algorithms. In particular, it is well known that Knapsack admits fully polynomial-time approximation schemes (FPTASes), which can return a feasible solution with a total profit of at least $\text{OPT}/(1+\varepsilon)$ in $\text{poly}(n, \frac{1}{\varepsilon})$ time. (We use OPT to denote the maximum total profit one can achieve using capacity t .) There has been a long line of research on designing faster FPTASes for Knapsack [22, 34, 30, 38, 10, 25, 19, 16, 35]. Very recently, Chen, Lian, Mao and Zhang [16] and Mao [35] independently obtained $\tilde{O}(n + (\frac{1}{\varepsilon})^2)$ -time FPTASes¹. On the negative side, there is no $O((n + \frac{1}{\varepsilon})^{2-\delta})$ -time FPTAS

¹ Throughout, we use an $\tilde{O}(\cdot)$ notation to hide polylogarithmic factors in n and $\frac{1}{\varepsilon}$.

for any constant $\delta > 0$ assuming $(\min, +)$ -convolution cannot be solved in subquadratic time [18, 33]. Hence, further improvement on FPTASs for Knapsack seems hopeless unless the $(\min, +)$ -convolution conjecture is overturned. Due to the symmetry of weight and profit, the same upper and lower bounds also hold if we seek a solution of total profit at least OPT , but relax the capacity to $(1 + \varepsilon)t$. In this paper, we consider *weak approximation schemes*, which relax both the total profit and the capacity. More precisely, a weak approximation scheme seeks a subset of items with total weight at most $(1 + \varepsilon)t$ and total profit at least $\text{OPT}/(1 + \varepsilon)$.

Weak approximation algorithms are commonly studied in the context of resource augmentation (see e.g., [21, 23]). One often resorts to resource augmentation when there is a barrier to further improvement on standard (i.e., non-weak) approximation algorithms. The two most relevant examples are Subset Sum and Unbounded Knapsack. Both problems admit an $\tilde{O}(n + (\frac{1}{\varepsilon})^2)$ -time FPTAS [28, 9, 24], while an $O((n + \frac{1}{\varepsilon})^{2-\delta})$ -time FPTAS has been ruled out assuming the $(\min, +)$ -convolution conjecture [33, 18, 36, 9]. Their weak approximation, however, can be done in truly subquadratic time. For Subset Sum, Mucha, Węgrzycki and Włodarczyk [36] showed a $O(n + (\frac{1}{\varepsilon})^{5/3})$ -time weak approximation scheme, and very recently, Chen et al. [13] further improved the running time to $\tilde{O}(n + \frac{1}{\varepsilon})$ -time. For Unbounded Knapsack, Bringmann and Cassis [6] established an $\tilde{O}(n + (\frac{1}{\varepsilon})^{3/2})$ -time weak approximation scheme. Given that the standard approximation of Knapsack has been in a similar situation, it is natural to ask whether the weak approximation of Knapsack can be done in truly subquadratic time.

It is worth mentioning that weak approximation schemes for Subset Sum and Unbounded Knapsack are not straightforward extensions of the existing standard (non-weak) approximation schemes. The first truly subquadratic-time weak approximation scheme for Subset Sum [36] as well as its recent improvement [13] both rely on a novel application of additive combinatorics, and the first truly subquadratic-time weak approximation scheme for Unbounded Knapsack [6] relies on a breakthrough on bounded monotone $(\min, +)$ -convolution [11, 17]. Unfortunately, techniques developed in these prior papers are not sufficient to obtain a truly subquadratic-time weak approximation scheme for Knapsack.

Our contribution.

We propose the first truly subquadratic-time weak approximation scheme for Knapsack.

► **Theorem 1.** *Knapsack admits an $\tilde{O}(n + (\frac{1}{\varepsilon})^{7/4})$ -time weak approximation scheme, which is randomized and succeeds with high probability.²*

Our algorithm can be easily generalized to Bounded Knapsack, in which, every item i can be selected up to u_i times. When $u_i = O(1)$ for all i , Bounded Knapsack is equivalent to Knapsack (up to a constant factor), and therefore can be handled by our algorithm. For the general case, there is a standard approach of reducing Bounded Knapsack to instances where $u_i = O(1)$ (see e.g., [36, Lemma 4.1][32, Lemma 2.2]). Note that such a reduction will blow up item weights and profits by a factor $O(u_i)$, but this does not make much difference in (weak) approximation schemes due to scaling.

1.1 Technical Overview

We give a brief overview of the techniques that are used by our algorithm.

² Throughout, “with high probability” stands for “with probability at least $1 - (n + \frac{1}{\varepsilon})^{-\Omega(1)}$ ”.

Bounded Monotone (min, +)-Convolution

(min, +)-convolution (more precisely, its counterpart, (max, +)-convolution) has been used in almost all the recent progress on Knapsack [10, 25, 14, 26, 5, 16, 35]. It allows one to cut the input instance into sub-instances, deal with each sub-instance independently, and then merge the results using (min, +)-convolution. In general, it takes $O(n^2)$ time to compute the (min, +)-convolution of two length- n sequences³, and it is conjectured that there is no algorithm with time $O(n^{2-\delta})$ for any constant $\delta > 0$. Nevertheless, some special cases of (min, +)-convolution can be computed in subquadratic time (See [1, 2, 37, 11, 17] for example), and among them is bounded monotone (min, +)-convolution where the two sequences are monotone and their entries are integers bounded by $O(n)$. In a celebrated work, Chan and Lewenstein [11] showed that bounded monotone (min, +)-convolution can be computed in $O(n^{1.859})$ time. This was later improved to $\tilde{O}(n^{1.5})$ by Chi, Duan, Xie, and Zhang [17], and further generalized to rectangular monotone (min, +)-convolution by Bringmann, Dürr and Polak [8].

Bounded monotone (min, +)-convolution is closely related to Unbounded Knapsack. Indeed, Bringmann and Cassis presented a $\tilde{O}(n + (\frac{1}{\epsilon})^{3/2})$ -time weak approximation scheme [6] that builds upon bounded monotone (min, +)-convolution, and they also showed an equivalence result between the two problems in the sense that a truly subquadratic-time weak approximation scheme for Unbounded Knapsack implies a truly subquadratic algorithm for bounded monotone (min, +)-convolution, and vice versa.

For our problem, we follow the general framework mentioned above: we cut the input instance into sub-instances (called the reduced problem), deal with each sub-instance independently, and then merge the results using bounded monotone (min, +)-convolution. Note that following this framework, for each sub-instance we need to compute the near-optimal objective value for all (approximate) knapsack capacities.

One might anticipate that a subquadratic algorithm for bounded monotone (min, +)-convolution leads to a subquadratic weak approximation scheme for Knapsack straightforwardly, however, this is not the case. Bounded monotone (min, +)-convolution is only used to merge results; the main challenge lies in dealing with each group, where we need the following two techniques.

Knapsack with Items of Similar Efficiencies

When all items have the same efficiency, Knapsack becomes Subset Sum, and it can be weakly approximated in $\tilde{O}(n + \frac{1}{\epsilon})$ time [13]. We extend the result to the case where all the items have their efficiencies (that is, profit-to-weight ratio) in the range $[\rho, \rho + \Delta]$ for constant ρ , and show that in such a case, Knapsack can be weakly approximated in $\tilde{O}(n + \frac{1}{\epsilon} + (\frac{1}{\epsilon})^2 \Delta)$ time. This result may be of independent interest. The following is a rough idea. By scaling, we can assume that there are only $O(\frac{\Delta}{\epsilon})$ distinct item efficiencies. Then we divide the items into $O(\frac{\Delta}{\epsilon})$ groups so that the items in the same group have the same efficiency, and therefore, each group can be treated as a Subset Sum problem. For each group, we solve it by invoking Chen et al.'s $\tilde{O}(n + \frac{1}{\epsilon})$ -time weak approximation scheme for Subset Sum [13]. After solving all groups, we merge their results using 2D-FFT and divide-and-conquer, which takes $\tilde{O}((\frac{1}{\epsilon})^2 \Delta)$ time.

³ A slightly faster $O(n^2/2^{\Omega(\sqrt{\log n})})$ -time algorithm is known by Bremner et.al.'s reduction to (min, +)-matrix multiplication [3] and William's algorithm for the latter problem [39] (which was derandomized by Chan and Williams [12]).

We note that the general idea of using Subset Sum to solve Knapsack when items have similar efficiencies was used by Jin [25], and the idea of merging Subset Sum results using 2D-FFT and divide-and-conquer was used by Deng, Jin and Mao [19].

Proximity Bounds

Proximity is another tool that is crucial to the recent progress of Knapsack [37, 14, 26, 5, 16]. Proximity bounds capture the intuition that the optimal solution should contain a lot of high-efficiency items and only a few low-efficiency items. (The efficiency of an item is defined to be its profit-to-weight ratio.) It was pioneered by Eisenbrand and Weismantel [20]. Their proximity bound works for a more general problem of integer programming and is based on Steinitz's lemma. The proof can be greatly simplified if one only considers Knapsack [37]. This bound was subsequently improved using tools from additive combinatorics [14, 5, 26]. These bounds depend on the maximum item weight w_{\max} or the maximum item profits p_{\max} , and leads to $\tilde{O}(n + w_{\max}^2)$ -time and $\tilde{O}(n + p_{\max}^2)$ -time algorithm for Knapsack [26, 5]. (We remark that all the proximity bounds of this type assume that the item weights and profits are integers.) These results, however, seem not helpful in breaking the quadratic barrier: even if we can scale the w_{\max} and p_{\max} to integers bounded by $O(\frac{1}{\varepsilon})$, it still takes $\tilde{O}(n + (\frac{1}{\varepsilon})^2)$ time to solve the problem.

We shall use another type of proximity bounds, which depends on the item efficiencies. Such a proximity bound was used to design FPTASes for Knapsack [25, 19] before. This proximity states that the optimal solution selects almost all the high-efficiency items, and exchanges only a few of them for the low-efficiency items. Moreover, the number of exchanged items is inversely proportional to the efficiency gap between high- and low-efficiency items. Fewer exchanged items allow a larger additive error per exchanged item, and therefore, a faster approximation.

Our Main Technical Contribution

While each of the above-mentioned techniques appeared in previous papers before, it seems that none of them alone can break the quadratic barrier for weakly approximating Knapsack. Our main contribution is to modify and then combine these techniques to obtain a truly subquadratic-time algorithm.

Note that if the item efficiencies differ significantly, then the efficiency-based proximity bound becomes sharp; If item efficiencies stay similar, then the Subset Sum based approach becomes efficient. To combine both ideas, the main difficulty is that we need to approximate the optimal objective value for all (approximate) capacities, and the efficiency-based proximity bound may be sharp for certain capacities, and weak for other capacities. Towards this, the main idea is to use different rounding precision for different items. We roughly explain it below. For simplicity, when we say we approximately solve a group of items, we mean to approximately compute the optimal objective value for all knapsack capacities when taking items from this group. We partition all items into multiple groups of fixed size. If the items in a group have similar efficiencies, then we approximately solve this group with high precision using the Subset Sum based approach; Otherwise, we approximately solve this group with low precision. Low precision means a larger error per item, but if proximity guarantees that only a few items from this group will contribute to the optimal solution (or only a few items will not contribute), then the total error may still be bounded. So the challenge is to prove that a sharp proximity holds for such a group regardless of the knapsack capacity.

We remark that a simpler implementation of the above idea leads to an algorithm with

running time $\tilde{O}(n + (\frac{1}{\varepsilon})^{11/6})$. By grouping the items in a more sophisticated way, the running time can be improved to $\tilde{O}(n + (\frac{1}{\varepsilon})^{7/4})$.

1.2 Further Related Work

FPTASes with Running Time in Multiplicative Form

So far, we have discussed FPTASes for Knapsack whose running times are $n + f(\frac{1}{\varepsilon})$. There is also a line of research focusing on FPTASes with a running time of the form $n^{O(1)} \cdot f(\frac{1}{\varepsilon})$ (see, e.g., [34, 29, 10]). The best known running time of this form is $\tilde{O}(\frac{n^{3/2}}{\varepsilon})$ due to Chan [10]. It remains open whether there exists an FPTAS of running time $O(\frac{n}{\varepsilon})$. It is not even clear whether Knapsack admits a weak approximation scheme in $O(\frac{n}{\varepsilon})$ time.

Pseudo-polynomial Time Algorithm

Recent years have witnessed important progress in the study of pseudo-polynomial time algorithms for Knapsack and Unbounded Knapsack [6, 7, 14, 5, 26]. In particular, the best-known algorithm for Knapsack that is parametrized by the largest weight w_{\max} (or the largest profit p_{\max}) runs in $\tilde{O}(n + w_{\max}^2)$ -time (or $\tilde{O}(n + p_{\max}^2)$ -time) [5, 26], and is the best possible assuming the (min, +)-convolution conjecture [18, 33]. It is, however, a major open problem whether the quadratic barrier can be broken by combining both parameters, that is, whether an $\tilde{O}(n + (w_{\max} + p_{\max})^{2-\delta})$ -time algorithm exists for Knapsack. Notably, for Unbounded Knapsack, one can find an algorithm that runs in $\tilde{O}(n + (w_{\max} + p_{\max})^{3/2})$ time [6].

1.3 Paper Outline

In Section 2, we introduce all the necessary notation and terminology. In Section 3, we reduce the problem so that it suffices to consider instances with nice properties. In Section 4, we present all the ingredients that are needed by our algorithm. In Section 5, we present an $\tilde{O}(n + (\frac{1}{\varepsilon})^{11/6})$ -time algorithm to illustrate our main idea. In Section 6, we show how to improve the running time to $\tilde{O}(n + (\frac{1}{\varepsilon})^{7/4})$. All the missing proofs can be found in the appendix.

2 Preliminaries

We assume that $\varepsilon \leq 1$, since otherwise we can approximate using a linear-time 2-approximation algorithm for Knapsack [31, Section 2.5]. We denote the set of real numbers between a and b as $[a, b]$. All logarithms are based 2.

To ease the presentation of our algorithm, we interpret Knapsack from the perspective of tuples. An item can be regarded as a tuple (w, p) , where w and p represent the weight and profit, respectively. Throughout, we do not distinguish an item and a tuple, and always refer to the first entry as weight and the second entry as profit. The efficiency of a tuple (w, p) is defined to be its profit-to-weight ratio $\frac{p}{w}$. The efficiency of $(0, 0)$ is defined to be 0.

Let I be a multiset of tuples. We use $w(I)$ to denote the total weight of tuples in I . That is, $w(I) = \sum_{(w,p) \in I} w$. Similarly, we define $p(I) = \sum_{(w,p) \in I} p$. Define the set of all (2-dimensional) subset sums of I as follows:

$$\mathcal{S}(I) = \{(w(I'), p(I')) : I' \subseteq I\}.$$

Every tuple $(w, p) \in \mathcal{S}(I)$ corresponds to a Knapsack solution with total weight w and total profit p . We say a tuple (w, p) dominates another tuple (w', p') if $w \leq w'$, $p \geq p'$, and at least one of the two inequalities holds strictly. Let $\mathcal{S}^+(I)$ be the set obtained from $\mathcal{S}(I)$ by removing all dominated tuples. Every tuple in $\mathcal{S}^+(I)$ corresponds to a Pareto optimal solution for Knapsack, and vice versa. Let t be the given capacity. Knapsack can be formulated as follows:

$$\max\{p : (w, p) \in \mathcal{S}^+(I) \text{ and } w \leq t\}.$$

We consider the more general problem of (weakly) approximating $\mathcal{S}^+(I)$.

► **Definition 2.** Let S and S' be two sets of tuples.

- (i) S' approximates S with factor $1 + \varepsilon$ if for any $(w, p) \in S$, there exists $(w', p') \in S'$ such that $w' \leq (1 + \varepsilon)w$ and $p' \geq p/(1 + \varepsilon)$;
- (ii) S' approximates S with additive error (δ_w, δ_p) if for any $(w, p) \in S$, there exists $(w', p') \in S'$ such that $w' \leq w + \delta_w$ and $p' \geq p - \delta_p$;
- (iii) S dominates S' if for any $(w', p') \in S'$, there exists $(w, p) \in S$ such that $w \leq w'$ and $p \geq p'$.

When S' approximates a set S that contains only one tuple (w, p) , we also say that S' approximates the tuple (w, p) .

Let (I, t) be a Knapsack instance. Let OPT be the optimal value. Our goal is to compute a set S' such that

- S' approximates $\mathcal{S}^+(I) \cap ([0, t] \times [0, \text{OPT}])$ with additive error $(\varepsilon t, \varepsilon \cdot \text{OPT})$ and
- S' is dominated by $\mathcal{S}(I)$.

The former ensures that S' provides a good (weak) approximation of the optimal solution, and the latter ensures that every tuple in S' is not better than a true solution. Our definition is essentially the same as that by Bringmann and Cassis [6], albeit their definition is for sequences.

Let $I_1 \cup I_2$ be a partition of I . It is easy to observe the following.

- $\mathcal{S}(I) = \mathcal{S}(I_1) + \mathcal{S}(I_2)$, where the sumset $X + Y$ is defined by the following formula.

$$X + Y = \{(w_1 + w_2, p_1 + p_2) : (w_1, p_1) \in X \text{ and } (w_2, p_2) \in Y\};$$

- $\mathcal{S}^+(I) = \mathcal{S}^+(I_1) \oplus \mathcal{S}^+(I_2)$, where \oplus represents the $(\max, +)$ -convolution defined by the following formula.

$$X \oplus Y = \{(w, p) \in X + Y : (w, p) \text{ is not dominated in } X + Y\}.$$

Using Chi et al.'s $\tilde{O}(n^{3/2})$ -time algorithm for bounded monotone $(\min, +)$ -convolution [17] and Bringmann and Cassis's algorithm for approximating the $(\max, +)$ -convolution [6, Lemma 28 in the full version], we can approximate the $(\max, +)$ -convolution of two sets in $\tilde{O}((\frac{1}{\varepsilon})^{3/2})$ time. Using the divide-and-conquer approach from [10], we can generalize it to multiple sets. We defer the proof of the following lemma to Appendix A.

► **Lemma 3.** Let $S_1, S_2, \dots, S_m \subseteq (\{0\} \cup [A, B]) \times (\{0\} \cup [C, D])$ be sets of tuples with total size ℓ . In $\tilde{O}(\ell + (\frac{1}{\varepsilon})^{3/2}m)$ time⁴ and with high probability, we can compute a set \tilde{S} of size $\tilde{O}(\frac{1}{\varepsilon})$ that approximates $S_1 \oplus S_2 \oplus \dots \oplus S_m$ with factor $1 + O(\varepsilon)$. Moreover, \tilde{S} is dominated by $S_1 + S_2 + \dots + S_m$.

⁴ Here, the $\tilde{O}(\cdot)$ notation hides polylogarithmic factor not only in n and $\frac{1}{\varepsilon}$, but also in other parameters such as $\frac{B}{A}$ and $\frac{D}{C}$. These parameters will eventually be set to be polynomial in n and $\frac{1}{\varepsilon}$.

With Lemma 3, we can approximate $\mathcal{S}^+(I) \cap ([0, t] \times [0, OPT])$ as follows. Partition I into several groups, approximate each group independently, and merge their results using Lemma 3. As long as there are not too many groups, the merging time will be subquadratic. The following lemmas explain how the multiplicative factor and additive error accumulate during the computation. Their proofs are deferred to Appendix B.

► **Lemma 4.** *Let S , S_1 , and S_2 be sets of tuples.*

- (i) *If S_1 approximates S with factor $1 + \varepsilon_1$ and S_2 approximates S_1 with factor $1 + \varepsilon_2$, then S_2 approximates S with factor $(1 + \varepsilon_1)(1 + \varepsilon_2)$.*
- (ii) *If S_1 approximates S with additive error $(\delta_{w1}, \delta_{p1})$ and S_2 approximates S_1 with factor $(\delta_{w2}, \delta_{p2})$, then S_2 approximates S with additive error $(\delta_{w1} + \delta_{w2}, \delta_{p1} + \delta_{p2})$.*

► **Lemma 5.** *Let S_1 , S_2 , S'_1 , S'_2 be sets of tuples.*

- (i) *If S'_1 and S'_2 approximates S_1 and S_2 respectively with factor $1 + \varepsilon$, then $S'_1 \oplus S'_2$ and $S'_1 + S'_2$ approximate $S_1 \oplus S_2$ and $S_1 + S_2$ respectively with factor $1 + \varepsilon$.*
- (ii) *If S'_1 and S'_2 approximates S_1 and S_2 with additive error $(\delta_{w1}, \delta_{p1})$ and $(\delta_{w2}, \delta_{p2})$, respectively, then $S'_1 \oplus S'_2$ and $S'_1 + S'_2$ approximate $S_1 \oplus S_2$ and $S_1 + S_2$ respectively with additive error $(\delta_{w1} + \delta_{w2}, \delta_{p1} + \delta_{p2})$.*

In the rest of the paper, we may switch between the multiplicative factor and the additive error, although our goal is to bound the latter. It is helpful to keep the following observation in mind.

► **Observation 6.** *If S' approximates S with factor $1 + \varepsilon$, then S' approximates each tuple $(w, p) \in S$ with additive error $(\varepsilon w, \varepsilon p)$.*

3 Reducing the Problem

With Lemma 3, we can reduce the problem of weakly approximating Knapsack to the following problem.

The reduced problem $RP(\alpha)$: given a set I of n items from $[1, 2] \times [1, 2]$ and a real number $\alpha \in [\frac{1}{2}, \frac{1}{4\varepsilon}]$, compute a set S of size $\tilde{O}(\frac{1}{\varepsilon})$ that approximates $\mathcal{S}^+(I) \cap ([0, \frac{1}{\alpha\varepsilon}] \times [0, \frac{1}{\alpha\varepsilon}])$ with additive error $(O(\frac{1}{\alpha}), O(\frac{1}{\alpha}))$. Moreover, S should be dominated by $\mathcal{S}(I)$.

► **Lemma 7.** *An $\tilde{O}(n + T(\frac{1}{\varepsilon}))$ -time algorithm for the reduced problem $RP(\alpha)$ implies an $\tilde{O}(n + (\frac{1}{\varepsilon})^{3/2} + T(\frac{1}{\varepsilon}))$ -time weak approximation scheme for Knapsack.*

The approach to reducing the problem is similar to that in [16]. The details of the proof are deferred to Appendix C.

Throughout, we write $\mathcal{S}^+(I) \cap ([0, \frac{1}{\alpha\varepsilon}] \times [0, \frac{1}{\alpha\varepsilon}])$ as $\mathcal{S}^+(I; \frac{1}{\alpha\varepsilon})$ for short, and when the weight and profit both have an additive error of δ , we simply say that additive error is δ .

We also remark that when presenting algorithms for the reduced problem $RP(\alpha)$, we will omit the requirement that the resulting set S should be dominated by $\mathcal{S}(I)$ since it is guaranteed by the algorithms in a quite straightforward way. More specifically, in those algorithms (including Lemma 3), rounding is the only place that may incur approximation errors, and whenever we round a tuple (w, p) , we always round w up and round p down. As a consequence, a set after rounding is always dominated by the original set.

4 Algorithmic Ingredients

We present a few results that will be used by our algorithm as subroutines.

4.1 An Algorithm for Small Solution Size

Consider the reduced problem $\text{RP}(\alpha)$. Since items are from $[1, 2] \times [1, 2]$, it is easy to see that $|I'| \leq \frac{1}{\alpha\varepsilon}$ for any $I' \subseteq I$ with $(w(I'), p(I')) \in \mathcal{S}^+(I; \frac{1}{\alpha\varepsilon})$. When α is large, $|I'|$ is small, and such a case can be tackled using Bringmann's two-layer color-coding technique [4]. Roughly speaking, we can divide I into $k = \tilde{O}(\frac{1}{\alpha\varepsilon})$ subsets I_1, \dots, I_k so that with high probability, $|I_j \cap I'| \leq 1$ for every j . Then it suffices to (approximately) compute $I_1 \oplus \dots \oplus I_k$, and the resulting set will provide a good approximation for the tuple $(w(I'), p(I'))$. Since this technique becomes quite standard nowadays, we defer the proof to Appendix D.

► **Lemma 8.** *Let I be a set of items from $[1, 2] \times [1, 2]$. In $\tilde{O}(n + (\frac{1}{\varepsilon})^{5/2} \frac{1}{\alpha})$ time and with high probability, we can compute a set S of size $\tilde{O}(\frac{1}{\varepsilon})$ that approximates $\mathcal{S}^+(I; \frac{1}{\alpha\varepsilon})$ with factor $1 + O(\varepsilon)$.*

4.2 An Efficiency-Based Proximity Bound

A proximity bound basically states that the optimal solution of Knapsack must contain lots of high-efficiency items and very few low-efficiency items. Let $I = \{(w_1, p_1), \dots, (w_n, p_n)\}$ be a set of items. Assume that the items are labeled in a non-increasing order of efficiencies. That is, $\frac{p_1}{w_1} \geq \dots \geq \frac{p_n}{w_n}$. (The labeling may not be unique. We fix an arbitrary one in that case.) Given a capacity w , let b be the minimum integer such that $w_1 + \dots + w_b > w$. We say that the item (w_b, p_b) is the breaking item with respect to w . (When no such b exists, we simply let $b = n + 1$ and define the breaking item to be $(+\infty, 0)$.)

The following proximity bound is essentially the same as the one used in [25, 19]. For completeness, we provide a proof in Appendix E.

► **Lemma 9.** *Let I be a set of items from $[1, 2] \times [1, 2]$. Let $(w^*, p^*) \in \mathcal{S}^+(I)$. Let $I^* \subseteq I$ be a subset of items with $w(I^*) = w^*$ and $p(I^*) = p^*$. Let ρ_b be the efficiency of the breaking item with respect to w^* . For any $I' \subseteq I$,*

(i) *if every item in I' has efficiency at most $\rho_b - \Delta$ for some $\Delta > 0$, then*

$$w(I' \cap I^*) \leq \frac{2}{\Delta};$$

(ii) *if every item in I' has efficiency at least $\rho_b + \Delta$ for some $\Delta > 0$, then*

$$w(I' \setminus I^*) \leq \frac{2}{\Delta}.$$

4.3 An Algorithm for Items with Similar Efficiencies

When all the items have the same efficiency, Knapsack becomes Subset Sum, and there is an $\tilde{O}(n + \frac{1}{\varepsilon})$ -time weak approximation scheme for Subset Sum [13]. The following lemma extends this result to the case where the item efficiencies differ by only a small amount.

► **Lemma 10.** *Let I be a set of n items from $[1, 2] \times [0, \infty]$ whose efficiencies are in the range $[\rho, \rho + \Delta]$. In $\tilde{O}(n + \frac{1}{\varepsilon} + (\frac{1}{\varepsilon})^2 \frac{\Delta}{\rho})$ time and with high probability, we can compute a set of size $\tilde{O}(\frac{1}{\varepsilon})$ that approximates $\mathcal{S}^+(I)$ with factor $1 + O(\varepsilon)$.*

We first consider the $(\max, +)$ -convolution of two sets of tuples.

► **Lemma 11.** *Let S_1, S_2 be two sets of tuples from $(\{0\} \cup [A, B]) \times [0, +\infty]$ with total size ℓ . Suppose that the efficiencies of the tuples in $S_1 \cup S_2$ are in the range $\{0\} \cup [\rho, \rho + \Delta]$. In*

$\tilde{O}(\ell + \frac{1}{\varepsilon} + (\frac{1}{\varepsilon})^2 \frac{\Delta}{\rho})$ time⁵, we can compute a set S of size $\tilde{O}(\frac{1}{\varepsilon})$ that approximates $S_1 \oplus S_2$ with factor $1 + \tilde{O}(\varepsilon)$. Moreover, S is dominated by $S_1 + S_2$.

Proof. TOPROVE 0 ◀

Now we are ready to prove Lemma 10.

Proof. TOPROVE 1 ◀

The Lemma 10 immediately implies the following.

► **Corollary 12.** *Let I be a set of n items from $[1, 2] \times [0, +\infty]$ whose efficiencies are in the range $[\rho, \rho + \Delta]$. In $\tilde{O}(n + \frac{1}{\varepsilon} + (\frac{1}{\varepsilon})^2 \frac{\Delta}{\rho})$ time and with high probability, we can compute a set of size $\tilde{O}(\frac{1}{\varepsilon})$ that approximates each tuple $(w, p) \in \mathcal{S}^+(I)$ with additive error $(\varepsilon w, \varepsilon p)$.*

In the above corollary, the additive error is proportional to the value of the entry of the tuple: a large entry means a large additive error. We can also achieve the contrary and approximate in a way so that a large entry means a small additive error. Basically, this can be done by (approximately) solving a problem symmetric to Knapsack: instead of determining which items should be selected, we determine which items should not be selected.

► **Corollary 13.** *Let I be a set of n items from $[1, 2] \times [0, +\infty]$ whose efficiencies are in the range $[\rho, \rho + \Delta]$. In $\tilde{O}(n + \frac{1}{\varepsilon} + (\frac{1}{\varepsilon})^2 \frac{\Delta}{\rho})$ time and with high probability, we can compute a set of size $\tilde{O}(\frac{1}{\varepsilon})$ that approximates $\mathcal{S}^+(I)$ with additive error $(\varepsilon(w(I) - w), \varepsilon(p(I) - p))$.*

Proof. TOPROVE 2 ◀

5 An Algorithm with Exponent 11/6

To illustrate our main idea, we first present a simpler algorithm that runs in $\tilde{O}(n + (\frac{1}{\varepsilon})^{11/6})$ time. When $\alpha \geq (\frac{1}{\varepsilon})^{2/3}$, the algorithm in Lemma 8 already runs in $\tilde{O}(n + (\frac{1}{\varepsilon})^{11/6})$ time. It remains to consider the case where $\frac{1}{2} \leq \alpha \leq (\frac{1}{\varepsilon})^{2/3}$. In the rest of this section, we assume that $I = \{(w_1, p_1), \dots, (w_n, p_n)\}$ and that $\frac{p_1}{w_1} \geq \dots \geq \frac{p_n}{w_n}$.

Let τ be a parameter that will be specified later. We first partition I into groups as follows. For the items $(w_1, p_1), \dots, (w_n, p_n)$ in I , we bundle every τ items as a group until we obtain $\lceil \frac{1}{\alpha \varepsilon \tau} \rceil + 1$ groups. For the remaining items, say $\{(w_i, p_i), \dots, (w_n, p_n)\}$, we further divide them into two groups $\{(w_i, p_i), \dots, (w_{i'}, p_{i'})\}$ and $\{(w_{i'+1}, p_{i'+1}), \dots, (w_n, p_n)\}$, where i' is the maximum index such that $\frac{p_i}{w_i} - \frac{p_{i'}}{w_{i'}} \leq \frac{1}{\tau}$. At the end, we obtain $m = \lceil \frac{1}{\alpha \varepsilon \tau} \rceil + 3$ groups I_1, I_2, \dots, I_m . For simplicity, we assume that none of these groups is empty. This assumption is without loss of generality, since empty groups only make things easier.

For each group I_j , we define Δ_j to be the maximum difference between the efficiencies of the items in I_j . That is,

$$\Delta_j = \max \left\{ \frac{p}{w} - \frac{p'}{w'} : (w, p), (w', p') \in I_j \right\}.$$

By the way we construct the groups, we have that $\sum_j \Delta_j \leq \frac{3}{2}$.

We say a group I_j is good if $\Delta_j \leq \frac{1}{\tau}$, and bad otherwise.

⁵ Here, the $\tilde{O}(\cdot)$ notation hides polylogarithmic factor not only in n and $\frac{1}{\varepsilon}$, but also in $\frac{B}{A}$. These parameters will eventually be set to be polynomial in n and $\frac{1}{\varepsilon}$.

Note that the efficiency of any item in I is within the range $[\frac{1}{2}, 2]$. For each good group I_j , its item efficiencies are in the range $[\rho, \rho + \frac{1}{\tau}]$ for some constant $\rho \in [\frac{1}{2}, 2]$, so we can compute a set S_j of $\tilde{O}(\frac{1}{\varepsilon})$ that approximates each tuple $(w, p) \in \mathcal{S}^+(I_j)$ with additive error $(\varepsilon w, \varepsilon p)$ via Corollary 12 in $O(|I_j| + \frac{1}{\varepsilon} + (\frac{1}{\varepsilon})^2 \frac{1}{\tau})$ time. Since the items are from $[1, 2] \times [1, 2]$, for any tuple in $\mathcal{S}^+(I_j)$, its weight and profit are of the same magnitude, so are the corresponding additive errors. We can say that S_j approximates each tuple $(w, p) \in \mathcal{S}^+(I_j)$ with additive error $O(\varepsilon w)$.

For bad groups, we shall approximate them less accurately: we shall use $\frac{1}{\alpha\tau}$ instead of ε as the accuracy parameter. (Later we will show that we can still obtain a good bound on the total additive error using the proximity bound in Lemma 9.) We compute a set S'_j of size $\tilde{O}(\alpha\tau)$ that approximates each tuple $(w, p) \in \mathcal{S}^+(I_j)$ with additive error $(\frac{w}{\alpha\tau}, \frac{p}{\alpha\tau})$ via Corollary 12 in $O(|I_j| + \alpha\tau + \alpha^2\tau^2\Delta_j)$ time. We also compute a set S''_j of size $\tilde{O}(\alpha\tau)$ that approximates each $(w, p) \in \mathcal{S}^+(I_j)$ with additive error $(\frac{w(I)-w}{\alpha\tau}, \frac{p(I)-p}{\alpha\tau})$ via Corollary 13 in $O(|I_j| + \alpha\tau + \alpha^2\tau^2\Delta_j)$ time. Again, since the weight and the profit are of the same magnitude, we can say that S'_j approximates each tuple $(w, p) \in \mathcal{S}^+(I_j)$ with additive error $O(\frac{1}{\alpha\tau} \cdot w)$, and that S''_j approximates each tuple $(w, p) \in \mathcal{S}^+(I_j)$ with additive error $O(\frac{1}{\alpha\tau} \cdot (w(I) - w))$. Now consider the set $S_j = S'_j \cup S''_j$. One can verify that S_j approximates each tuple $(w, p) \in \mathcal{S}^+(I_j)$ with additive error $\tilde{O}(\frac{1}{\alpha\tau} \cdot \min\{w, (w(I) - w)\})$.

The last step is to compute a set S of size $\tilde{O}(\frac{1}{\varepsilon})$ that approximates $S_1 \oplus \dots \oplus S_m$ with factor $1 + O(\varepsilon)$ via Lemma 3. Note that each set S_j is of size $\tilde{O}(\frac{1}{\varepsilon})$ or $\tilde{O}(\alpha\tau)$. This step takes time $\tilde{O}(\frac{m}{\varepsilon} + m\alpha\tau + (\frac{1}{\varepsilon})^{3/2}m)$.

Bounding Total Time Cost

Partitioning the item into I_1, \dots, I_m can be done by sorting and scanning, and therefore, it takes only $O(n \log n + m)$ time. Each good group costs $O(|I_j| + \frac{1}{\varepsilon} + (\frac{1}{\varepsilon})^2 \frac{1}{\tau})$ time. Since there are at most m good groups, the good groups cost total time

$$\tilde{O}\left(n + \frac{m}{\varepsilon} + (\frac{1}{\varepsilon})^2 \frac{m}{\tau}\right).$$

Each bad group I_j costs $O(|I_j| + \alpha\tau + \alpha^2\tau^2\Delta_j)$ time, so the total time cost for bad groups is

$$\tilde{O}(n + m\alpha\tau + \alpha^2\tau^2 \sum_j \Delta_j) \leq \tilde{O}(n + m\alpha\tau + \alpha^2\tau^2).$$

The inequality is due to that $\sum_j \Delta_j \leq \frac{3}{2}$. Merging all the sets S_j via Lemma 3 costs $\tilde{O}(\frac{m}{\varepsilon} + m\alpha\tau + (\frac{1}{\varepsilon})^{3/2}m)$ time. Taking the sum of all these time costs, we have that the running time of the algorithm is

$$\tilde{O}\left(n + \frac{m}{\varepsilon} + m\alpha\tau + (\frac{1}{\varepsilon})^2 \frac{m}{\tau} + \alpha^2\tau^2 + (\frac{1}{\varepsilon})^{3/2}m\right).$$

Recall that $m = \lceil \frac{1}{\alpha\varepsilon\tau} \rceil$ and that $\frac{1}{2} \leq \alpha \leq (\frac{1}{\varepsilon})^{2/3}$. Set $\tau = \lceil (\frac{1}{\varepsilon})^{\frac{11}{12}} \frac{1}{\alpha} \rceil$. One can verify that the running time is $\tilde{O}(n + (\frac{1}{\varepsilon})^{11/6})$.

Bounding Additive Error

To prove the correctness of the algorithm, it suffices to show that the set S returned by the algorithm approximates every tuple in $\mathcal{S}^+(I; \frac{1}{\alpha\varepsilon})$ with additive error $O(\frac{1}{\alpha})$.

Let (w^*, p^*) be an arbitrary tuple in $\mathcal{S}^+(I; \frac{1}{\alpha\varepsilon})$. Let (w_b, p_b) be the breaking item with respect to w^* . Let $\rho_b = \frac{p_b}{w_b}$ be the efficiency of the breaking item. We first show that the proximity bound can be applied to all but four bad groups.

► **Lemma 14.** *At most four bad groups may contain an item whose efficiency is within the range $[\rho_b - \frac{1}{\tau}, \rho_b + \frac{1}{\tau}]$. Moreover, each of these groups is of size τ .*

Proof. TOPROVE 3 ◀

Now we are ready to bound the total additive error.

► **Lemma 15.** *$S_1 \oplus S_2 \oplus \dots \oplus S_m$ approximates (w^*, p^*) with additive error $O(\frac{1}{\alpha})$.*

Proof. TOPROVE 4 ◀

The above lemma immediately implies the following since S approximates $S_0 \oplus S_1 \oplus \dots \oplus S_m$ with factor $1 + O(\varepsilon)$.

► **Corollary 16.** *S approximates (w^*, p^*) with additive error $O(\frac{1}{\alpha})$.*

We summarize this section by the following lemma.

► **Lemma 17.** *There is an $\tilde{O}(n + (\frac{1}{\varepsilon})^{11/6})$ -time algorithm for the reduced problem, which is randomized and succeeds with high probability.*

6 An Improved Algorithm with Exponent 7/4

Due to the space limit, we present only the most crucial part of the algorithm and the analysis. A full version of this section can be found in Appendix F.

In our $\tilde{O}(n + (\frac{1}{\varepsilon})^{11/6})$ -time algorithm, only the bad groups benefit from the proximity bound, which allows us to approximate with larger factor $1 + \frac{1}{\alpha\tau}$, while for the good groups, we simply approximate with factor $1 + \varepsilon$. To further improve the running time, we shall partition the items in a way that all the groups can benefit from the proximity bound.

We assume that $\frac{1}{2} \leq \alpha \leq (\frac{1}{\varepsilon})^{3/4}$ since the algorithm in Lemma 8 already runs in $\tilde{O}(n + (\frac{1}{\varepsilon})^{7/4})$ time for all $\alpha \geq (\frac{1}{\varepsilon})^{3/4}$. We assume that $I = \{(w_1, p_1), \dots, (w_n, p_n)\}$ and that $\frac{p_1}{w_1} \geq \dots \geq \frac{p_n}{w_n}$.

We first partition I into the head group I_{head} and the tail group I_{tail} where I_{head} contains the first $\lceil \frac{1}{\alpha\varepsilon} \rceil + 1$ items and I_{tail} contains the rest of the items. Below we only give an algorithm for I_{head} , and analyze its additive error. The remaining analysis and the algorithm for I_{tail} can be found in the full version (Appendix F).

6.1 Approximating Head Group

Let $n' = \lceil \frac{1}{\alpha\varepsilon} \rceil + 1$. We shall show that in $\tilde{O}((\frac{1}{\varepsilon})^{7/4})$ time, we can compute a set S of size $\tilde{O}(\frac{1}{\varepsilon})$ that approximate $\mathcal{S}^+(I_{\text{head}})$ with additive error $O(\frac{1}{\alpha})$.

Let τ be a parameter that will be specified later. Roughly speaking, we will further partition I_{head} into good groups and bad groups. The bad groups are the same as before: a bad group is a group of τ items whose efficiencies differ by at least $\frac{1}{\tau}$. For good groups, we will make them large than before: when we obtain a good group I' of size τ , we will keep adding items to I' until the difference between the maximum and the minimum item efficiencies exceeds $\frac{1}{|I'|}$ with the next item. A more precise description of the partition process is given below.

We partition I_{head} into I_1, \dots, I_m as follows. Initially, $j = 1$ and $k = 1$, and the remaining items are $\{(w_k, p_k), \dots, (w_{n'}, p_{n'})\}$. Let k' be the minimum integer such that

$$(k' - k) \cdot \left(\frac{p_k}{w_k} - \frac{p_{k'}}{w_{k'}} \right) > 1.$$

If such k' does not exist, we set $I_j = \{(w_k, p_k), \dots, (w_{n'}, p_{n'})\}$, set $\Delta_j = \Delta'_j = \frac{p_k}{w_k} - \frac{p_{n'}}{w_{n'}}$, set $m := j$, and finish. Assume that k' exists.

- If $k' - k \geq \tau$, we set $I_j = \{(w_k, p_k), \dots, (w_{k'-1}, p_{k'-1})\}$, set $\Delta_j = \frac{p_k}{w_k} - \frac{p_{k'-1}}{w_{k'-1}}$, set $\Delta'_j = \frac{p_k}{w_k} - \frac{p_{k'}}{w_{k'}}$, and proceed with $j := j + 1$ and $k := k'$. In this case, we say I_j is a good group.
- Otherwise, we let $I_j = \{(w_k, p_k), \dots, (w_{k+\tau-1}, p_{k+\tau-1})\}$, set $\Delta_j = \frac{p_k}{w_k} - \frac{p_{k+\tau-1}}{w_{k+\tau-1}}$, set $\Delta'_j = \frac{p_k}{w_k} - \frac{p_{k+\tau}}{w_{k+\tau}}$, and proceed with $j := j + 1$ and $k := k + \tau$. In this case, we say that I_j is a bad group.

(We remark that Δ_j and Δ'_j are not required by the algorithm. They are maintained only for the purpose of analysis. Δ_j is the actual difference between the maximum and minimum item efficiencies in I_j . For technical reasons, we also need Δ'_j . Basically, compared to Δ_j , the gap Δ'_j also includes the efficiency gap between the minimum efficiency in I_j and the maximum efficiency in I_{j+1} . We will use Δ_j to bound the time cost of I_j , and use Δ'_j to create an efficiency gap for other groups.)

We have the following observations.

► **Observation 18.** *The groups I_1, \dots, I_m satisfy the following properties.*

- (i) $m \leq \frac{1}{\alpha\epsilon\tau} + 1$
- (ii) $\sum_{j=1}^m |I_j| = |I_{\text{head}}| = \frac{1}{\alpha\epsilon} + 1$ and $\sum_{j=1}^m \Delta_j \leq \sum_{j=1}^m \Delta'_j \leq \frac{3}{2}$.
- (iii) For every group I_j , the efficiencies of the items in I_j differ by at most Δ_j .
- (iv) For every good group I_j , we have $|I_j|\Delta_j \leq 1$ and $|I_j|\Delta'_j \geq \frac{1}{2}$.
- (v) For every bad group I_j , we have $|I_j|\Delta'_j \geq |I_j|\Delta_j > 1$.
- (vi) For the last group I_m , we have $|I_m|\Delta_m \leq 1$.

For each group, we shall approximate it in exactly the same way as we did for the bad groups in the $\tilde{O}(n + (\frac{1}{\epsilon})^{11/6})$ -time algorithm, except that we shall use $\frac{1}{\alpha|I_j|}$ as the accuracy parameter. More specifically, we use Corollary 12 and Corollary 13 to compute a set S_j of size $\tilde{O}(\alpha|I_j|)$ that approximate each tuple $(w, p) \in \mathcal{S}^+(I_j)$ with additive error $\frac{1}{\alpha|I_j|} \cdot \min\{w, w(I) - w\}$. The time cost for I_j is $\tilde{O}(|I_j| + \alpha|I_j| + \alpha^2|I_j|^2\Delta_j)$.

Then we compute a set S_{head} of size $\tilde{O}(\frac{1}{\epsilon})$ that approximates $S_1 \oplus \dots \oplus S_m$ with factor $1 + O(\epsilon)$ via Lemma 3.

Bound Additive Error

We shall show that S_{head} approximates every tuple $(w^*, p^*) \in \mathcal{S}^+(I_{\text{head}})$ with additive error $\tilde{O}(\frac{1}{\alpha})$.

Let (w^*, p^*) be an arbitrary tuple in $\mathcal{S}^+(I_{\text{head}})$. Let (w_b, p_b) be the breaking item with respect to w^* . Let $\rho_b = \frac{p_b}{w_b}$ be the efficiency of the breaking item.

Let $I_{j'}$ be the group containing the breaking item (w_b, p_b) . To apply the proximity bound, we will show that the groups can be divided into $O(\log \frac{1}{\epsilon})$ collections so that for each collection $\{I_{j_k+1}, I_{j_k+2}, \dots, I_{j_{k+1}-1}\}$, the efficiency gap between ρ_b and the groups in this collection is at least the maximum Δ'_j of these groups. To do this, we need the following auxiliary lemma. Its proof is deferred to the full version.

► **Lemma 19.** *Let $\Delta_1, \dots, \Delta_n$ be a sequence of positive real numbers. Let Δ_{\min} and Δ_{\max} be the minimum and maximum numbers in the sequence. There exists $h = O(\log \frac{\Delta_{\max}}{\Delta_{\min}})$ indices $1 = j_1 < j_2 < \dots < j_h = n$ such that for any $k \in \{1, \dots, h-1\}$, we have that*

$$\max\{\Delta_j : j_k < j < j_{k+1}\} \leq \sum\{\Delta_j : j_{k+1} \leq j \leq n\}, \quad (1)$$

where the maximum of an empty set is defined to be 0.

► **Lemma 20.** $S_1 \oplus \cdots \oplus S_m$ approximates (w^*, p^*) with additive error $\tilde{O}(\frac{1}{\alpha})$.

Proof. TOPROVE 5 ◀

The above lemma immediately implies the following since S_{head} approximates $S_1 \oplus \cdots \oplus S_m$ with factor $1 + O(\varepsilon)$.

► **Corollary 21.** S_{head} approximates (w^*, p^*) with additive error $\tilde{O}(\frac{1}{\alpha})$.

The remaining analysis and the algorithm for I_{tail} can be found in the full version (Appendix F). We summarize this with the following lemma.

► **Lemma 22.** *There is an $\tilde{O}(n + (\frac{1}{\varepsilon})^{7/4})$ -time algorithm for the reduced problem, which is randomized and succeeds with high probability.*

References

- 1 Alok Aggarwal, Maria M. Klawe, Shlomo Moran, Peter Shor, and Robert Wilber. Geometric applications of a matrix-searching algorithm. *Algorithmica*, 2(1):195–208, November 1987. doi:10.1007/BF01840359.
- 2 Kyriakos Axiotis and Christos Tzamos. Capacitated Dynamic Programming: Faster Knapsack and Graph Algorithms. In *46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)*, pages 19:1–19:13. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.ICALP.2019.19.
- 3 David Bremner, Timothy M Chan, Erik D Demaine, Jeff Erickson, Ferran Hurtado, John Iacono, Stefan Langerman, Mihai Pătraşcu, and Perouz Taslakian. Necklaces, convolutions, and $x+y$. *Algorithmica*, 69(2):294–314, 2014.
- 4 Karl Bringmann. A Near-Linear Pseudopolynomial Time Algorithm for Subset Sum. In *Proceedings of the 2017 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2017)*, pages 1073–1084. Society for Industrial and Applied Mathematics, 2017. doi:10.1137/1.9781611974782.69.
- 5 Karl Bringmann. Knapsack with small items in near-quadratic time. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing (STOC 2024)*, pages 259–270. Association for Computing Machinery, 2024. doi:10.1145/3618260.3649719.
- 6 Karl Bringmann and Alejandro Cassis. Faster Knapsack Algorithms via Bounded Monotone Min-Plus-Convolution. In *49th International Colloquium on Automata, Languages, and Programming (ICALP 2022)*, pages 31:1–31:21. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.ICALP.2022.31.
- 7 Karl Bringmann and Alejandro Cassis. Faster 0-1-Knapsack via Near-Convex Min-Plus-Convolution. In *31st Annual European Symposium on Algorithms (ESA 2023)*, pages 24:1–24:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.ESA.2023.24.
- 8 Karl Bringmann, Anita Dür, and Adam Polak. Even faster knapsack via rectangular monotone min-plus convolution and balancing. In Timothy M. Chan, Johannes Fischer, John Iacono, and Grzegorz Herman, editors, *32nd Annual European Symposium on Algorithms, ESA 2024, September 2-4, 2024, Royal Holloway, London, United Kingdom*, volume 308 of *LIPIcs*, pages 33:1–33:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. URL: <https://doi.org/10.4230/LIPIcs.ESA.2024.33>, doi:10.4230/LIPIcs.ESA.2024.33.
- 9 Karl Bringmann and Vasileios Nakos. A Fine-Grained Perspective on Approximating Subset Sum and Partition. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA 2021)*, pages 1797–1815. Society for Industrial and Applied Mathematics, 2021. doi:10.1137/1.9781611976465.108.
- 10 Timothy M. Chan. Approximation Schemes for 0-1 Knapsack. In *1st Symposium on Simplicity in Algorithms (SOSA 2018)*, pages 5:1–5:12. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. doi:10.4230/OASIcs.SOSA.2018.5.

- 11 Timothy M. Chan and Moshe Lewenstein. Clustered Integer 3SUM via Additive Combinatorics. In *Proceedings of the Forty-Seventh Annual ACM Symposium on Theory of Computing (STOC 2015)*, pages 31–40. Association for Computing Machinery, 2015. doi:10.1145/2746539.2746568.
- 12 Timothy M. Chan and Ryan Williams. Deterministic apsp, orthogonal vectors, and more: Quickly derandomizing razborov-smolensky. In *Proceedings of the 2016 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2016)*, pages 1246–1255, 2016. doi:10.1137/1.9781611974331.ch87.
- 13 Lin Chen, Jiayi Lian, Yuchen Mao, and Guochuan Zhang. Approximating partition in near-linear time. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing (STOC 2024)*, pages 307–318. Association for Computing Machinery, 2024. doi:10.1145/3618260.3649727.
- 14 Lin Chen, Jiayi Lian, Yuchen Mao, and Guochuan Zhang. Faster algorithms for bounded knapsack and bounded subset sum via fine-grained proximity results. In *Proceedings of the 2024 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2024)*, pages 4828–4848. Society for Industrial and Applied Mathematics, 2024. doi:10.1137/1.9781611977912.171.
- 15 Lin Chen, Jiayi Lian, Yuchen Mao, and Guochuan Zhang. An improved pseudopolynomial time algorithm for subset sum. In *2024 IEEE 65th Annual Symposium on Foundations of Computer Science (FOCS 2024)*, pages 2202–2216. IEEE, 2024. doi:10.1109/FOCS61266.2024.00129.
- 16 Lin Chen, Jiayi Lian, Yuchen Mao, and Guochuan Zhang. A nearly quadratic-time fptas for knapsack. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing (STOC 2024)*, pages 283–294. Association for Computing Machinery, 2024. doi:10.1145/3618260.3649730.
- 17 Shucheng Chi, Ran Duan, Tianle Xie, and Tianyi Zhang. Faster min-plus product for monotone instances. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing (STOC 2022)*, pages 1529–1542. Association for Computing Machinery, 2022. doi:10.1145/3519935.3520057.
- 18 Marek Cygan, Marcin Mucha, Karol Węgrzycki, and Michał Włodarczyk. On problems equivalent to $(\min, +)$ -convolution. *ACM Transactions on Algorithms*, 15(1):1–25, January 2019. doi:10.1145/3293465.
- 19 Mingyang Deng, Ce Jin, and Xiao Mao. Approximating Knapsack and Partition via Dense Subset Sums. In *Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2023)*, pages 2961–2979. Society for Industrial and Applied Mathematics, 2023. doi:10.1137/1.9781611977554.ch113.
- 20 Friedrich Eisenbrand and Robert Weismantel. Proximity Results and Faster Algorithms for Integer Programming Using the Steinitz Lemma. *ACM Transactions on Algorithms*, 16(1):5:1–5:14, November 2019. doi:10.1145/3340322.
- 21 Aleksei V. Fishkin, Olga Gerber, Klaus Jansen, and Roberto Solis-Oba. On packing squares with resource augmentation: maximizing the profit. In *Proceedings of the 2005 Australasian Symposium on Theory of Computing*, CATS '05, page 61–67, AUS, 2005. Australian Computer Society, Inc. URL: <https://dl.acm.org/doi/10.5555/1082260.1082268>.
- 22 Oscar H. Ibarra and Chul E. Kim. Fast Approximation Algorithms for the Knapsack and Sum of Subset Problems. *Journal of the ACM*, 22(4):463–468, October 1975. doi:10.1145/321906.321909.
- 23 Kazuo Iwama and Guochuan Zhang. Online knapsack with resource augmentation. *Information Processing Letters*, 110(22):1016–1020, 2010. doi:10.1016/j.ipl.2010.08.013.
- 24 Klaus Jansen and Stefan E. J. Kraft. A faster FPTAS for the Unbounded Knapsack Problem. *European Journal of Combinatorics*, 68:148–174, February 2018. doi:10.1016/j.ejc.2017.07.016.
- 25 Ce Jin. An improved FPTAS for 0-1 knapsack. In *Proceedings of 46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)*, pages 76:1–76:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.ICALP.2019.76.

- 26 Ce Jin. 0-1 Knapsack in Nearly Quadratic Time. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing (STOC 2024)*, pages 271–282. Association for Computing Machinery, 2024. doi:10.1145/3618260.3649618.
- 27 Richard M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, pages 85–103. Springer US, Boston, MA, 1972. doi:10.1007/978-1-4684-2001-2_9.
- 28 Hans Kellerer, Renata Mansini, Ulrich Pferschy, and Maria Grazia Speranza. An efficient fully polynomial approximation scheme for the Subset-Sum Problem. *Journal of Computer and System Sciences*, 66(2):349–370, March 2003. doi:10.1016/S0022-0000(03)00006-0.
- 29 Hans Kellerer and Ulrich Pferschy. A New Fully Polynomial Time Approximation Scheme for the Knapsack Problem. *Journal of Combinatorial Optimization*, 3(1):59–71, July 1999. doi:10.1023/A:1009813105532.
- 30 Hans Kellerer and Ulrich Pferschy. Improved Dynamic Programming in Connection with an FPTAS for the Knapsack Problem. *Journal of Combinatorial Optimization*, 8(1):5–11, March 2004. doi:10.1023/B:JOCO.0000021934.29833.6b.
- 31 Hans Kellerer, Ulrich Pferschy, and Pisinger David. *Knapsack Problems*. Springer, Berlin Heidelberg, 2004. doi:10.1007/978-3-540-24777-7.
- 32 Konstantinos Koiliaris and Chao Xu. Faster Pseudopolynomial Time Algorithms for Subset Sum. *ACM Transactions on Algorithms*, 15(3):1–20, July 2019. doi:10.1145/3329863.
- 33 Marvin Künnemann, Ramamohan Paturi, and Stefan Schneider. On the Fine-Grained Complexity of One-Dimensional Dynamic Programming. In *44th International Colloquium on Automata, Languages, and Programming (ICALP 2017)*, pages 21:1–21:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.ICALP.2017.21.
- 34 Eugene L. Lawler. Fast Approximation Algorithms for Knapsack Problems. *Mathematics of Operations Research*, 4(4):339–356, November 1979. URL: <https://www.jstor.org/stable/3689221>.
- 35 Xiao Mao. $(1 - \epsilon)$ -approximation of knapsack in nearly quadratic time. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing (STOC 2024)*, pages 295–306. Association for Computing Machinery, 2024. doi:10.1145/3618260.3649677.
- 36 Marcin Mucha, Karol Węgrzycki, and Michał Włodarczyk. A Subquadratic Approximation Scheme for Partition. In *Proceedings of the 2019 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2019)*, pages 70–88. Society for Industrial and Applied Mathematics, 2019. doi:10.1137/1.9781611975482.5.
- 37 Adam Polak, Lars Rohwedder, and Karol Węgrzycki. Knapsack and Subset Sum with Small Items. In *48th International Colloquium on Automata, Languages, and Programming (ICALP 2021)*, pages 106:1–106:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.ICALP.2021.106.
- 38 Donguk Rhee. *Faster fully polynomial approximation schemes for knapsack problems*. PhD thesis, Massachusetts Institute of Technology, 2015. URL: <https://dspace.mit.edu/handle/1721.1/98564>.
- 39 Ryan Williams. Faster all-pairs shortest paths via circuit complexity. In *Proceedings of the Forty-Sixth Annual ACM Symposium on Theory of Computing (STOC 2014)*, page 664–673. Association for Computing Machinery, 2014. doi:10.1145/2591796.2591811.

A

 Details of Approximating Convolution

We shall prove the following lemma.

► **Lemma 3.** *Let $S_1, S_2, \dots, S_m \subseteq (\{0\} \cup [A, B]) \times (\{0\} \cup [C, D])$ be sets of tuples with total size ℓ . In $\tilde{O}(\ell + (\frac{1}{\varepsilon})^{3/2}m)$ time⁶ and with high probability, we can compute a set \tilde{S} of size $\tilde{O}(\frac{1}{\varepsilon})$ that approximates $S_1 \oplus S_2 \oplus \dots \oplus S_m$ with factor $1 + O(\varepsilon)$. Moreover, \tilde{S} is dominated by $S_1 + S_2 + \dots + S_m$.*

We need Chi et al.'s algorithm for bounded monotone $(\min, +)$ -convolution [17]. We say a set S of tuples is monotone if for any two tuple $(w, p), (w', p')$ in S , $w' \geq w$ implies $p' \geq p$. We say S is bounded by $O(\ell)$ if the entries of the tuples in S are non-negative integers bounded by $O(\ell)$.

► **Lemma 23.** [17] *Let S_1 and S_2 be two sets of tuples. Suppose that both S_1 and S_2 are monotone and bounded by $O(\ell)$. In $\tilde{O}(\ell^{3/2})$ time and with high probability⁷, we can compute $S_1 \oplus S_2$.*

With this algorithm, we can approximately compute the $(\max, +)$ -convolution of two sets of tuples in $\tilde{O}((\frac{1}{\varepsilon})^{3/2})$ time. The following lemma is essentially the same as that in [6, Lemma 28 in the full version], and its proof follows the idea of [10, Lemma 1].

► **Lemma 24.** *Let $S_1, S_2 \subseteq (\{0\} \cup [A, B]) \times (\{0\} \cup [C, D])$ be two sets of tuples with total size ℓ . In $\tilde{O}(\ell + (\frac{1}{\varepsilon})^{3/2})$ time and with high probability, we can compute a set S of size $\tilde{O}(\frac{1}{\varepsilon})$ that approximates $S_1 \oplus S_2$ with factor $1 + \varepsilon$. Moreover, S is dominated by $S_1 + S_2$.*

Proof. TOPROVE 6 ◀

Now we are ready to present an approximation algorithm for computing the $(\max, +)$ -convolution of multiple sets. It follows the idea of [10, Lemma 2(i)].

Proof. TOPROVE 7 ◀

B

 Details of Approximation Errors

We prove the following two lemmas.

► **Lemma 4.** *Let S, S_1 , and S_2 be sets of tuples.*

- (i) *If S_1 approximates S with factor $1 + \varepsilon_1$ and S_2 approximates S_1 with factor $1 + \varepsilon_2$, then S_2 approximates S with factor $(1 + \varepsilon_1)(1 + \varepsilon_2)$.*
- (ii) *If S_1 approximates S with additive error $(\delta_{w1}, \delta_{p1})$ and S_2 approximates S_1 with factor $(\delta_{w2}, \delta_{p2})$, then S_2 approximates S with additive error $(\delta_{w1} + \delta_{w2}, \delta_{p1} + \delta_{p2})$.*

Proof. TOPROVE 8 ◀

► **Lemma 5.** *Let S_1, S_2, S'_1, S'_2 be sets of tuples.*

⁶ Here, the $\tilde{O}(\cdot)$ notation hides polylogarithmic factor not only in n and $\frac{1}{\varepsilon}$, but also in other parameters such as $\frac{B}{A}$ and $\frac{D}{C}$. These parameters will eventually be set to be polynomial in n and $\frac{1}{\varepsilon}$.

⁷ The original algorithm in [17] runs in $\tilde{O}(\ell^{3/2})$ expected time. It is easy to convert it into an algorithm that succeeds with high probability. Such an algorithm is sufficient for our purpose since other parts of our algorithm only succeed with high probability.

- (i) If S'_1 and S'_2 approximate S_1 and S_2 respectively with factor $1 + \varepsilon$, then $S'_1 \oplus S'_2$ and $S'_1 + S'_2$ approximate $S_1 \oplus S_2$ and $S_1 + S_2$ respectively with factor $1 + \varepsilon$.
- (ii) If S'_1 and S'_2 approximate S_1 and S_2 with additive error $(\delta_{w1}, \delta_{p1})$ and $(\delta_{w2}, \delta_{p2})$, respectively, then $S'_1 \oplus S'_2$ and $S'_1 + S'_2$ approximate $S_1 \oplus S_2$ and $S_1 + S_2$ respectively with additive error $(\delta_{w1} + \delta_{w2}, \delta_{p1} + \delta_{p2})$.

Proof. TOPROVE 9 ◀

C

 Details of Problem Reduction

Recall that the reduced problem is defined as follows.

The reduced problem $\text{RP}(\alpha)$: given a set I of n items from $[1, 2] \times [1, 2]$ and a real number $\alpha \in [\frac{1}{2}, \frac{1}{4\varepsilon}]$, compute a set S that approximates $\mathcal{S}^+(I) \cap ([0, \frac{1}{\alpha\varepsilon}] \times [0, \frac{1}{\alpha\varepsilon}])$ with additive error $(\tilde{O}(\frac{1}{\alpha}), \tilde{O}(\frac{1}{\alpha}))$. Moreover, S should be dominated by $\mathcal{S}(I)$.

We shall prove the following lemma.

► **Lemma 7.** *An $\tilde{O}(n + T(\frac{1}{\varepsilon}))$ -time algorithm for the reduced problem $\text{RP}(\alpha)$ implies an $\tilde{O}(n + (\frac{1}{\varepsilon})^{3/2} + T(\frac{1}{\varepsilon}))$ -time weak approximation scheme for Knapsack.*

Let (I, t) be a Knapsack instance. Let OPT be the optimal value. Recall that to obtain a weak approximation, it suffices to compute a set that approximates $\mathcal{S}^+(I) \cap ([0, t] \times [0, \text{OPT}])$ with additive error $(\varepsilon t, \varepsilon \cdot \text{OPT})$. (Strictly speaking, the set should also be dominated by $\mathcal{S}(I)$. One can verify that this property follows quite straightforwardly. So we omit this property.) We can slightly relax the permissible additive error to $(\tilde{O}(\varepsilon t), \tilde{O}(\varepsilon \cdot \text{OPT}))$ because an additive error of $(\tilde{O}(\varepsilon t), \tilde{O}(\varepsilon \cdot \text{OPT}))$ can be reduced to $(\varepsilon t, \varepsilon \cdot \text{OPT})$ by adjusting ε by a polylogarithmic factor and this increases the running time by only a logarithmic factor.

C.1 Preprocessing

We first preprocess the instance so that the following properties hold after the preprocessing.

- $(w, p) \in [1, \frac{1}{\varepsilon}] \times [1, \frac{1}{\varepsilon}]$ for every item $(w, p) \in I$;
- $\frac{1}{2\varepsilon} \leq \text{OPT} \leq \frac{1}{\varepsilon}$ and $t = \frac{1}{\varepsilon}$.

The preprocessing takes $O(n \log n)$ time.

C.1.1 Preprocessing Profits

Knapsack has an $O(n)$ -time greedy algorithm that achieves an approximation ratio of 2 [31, Section 2.5]. Let y be the value of this greedy solution. Then $y \leq \text{OPT} \leq 2y$. We say an item (w, p) is cheap if $p \leq 2\varepsilon y$. We shall merge cheap items into meta-items with large profits. Let I^c be the set of cheap items. We partition I^c into I_1^c, \dots, I_ℓ^c so that

- (i) For every integer $j \in [1, \ell - 1]$, the efficiency of an item in I_j^c is no smaller than that of any item in I_{j+1}^c . That is, $\frac{p}{w} \geq \frac{p'}{w'}$ for any $(w, p) \in I_j^c$ and any $(w', p') \in I_{j+1}^c$.
- (ii) $2\varepsilon y < p(I_j^c) \leq 4\varepsilon y$ for every integer $j \in [1, \ell - 1]$ and $p(I_\ell^c) \leq 4\varepsilon y$.

The partition can be done in $O(n \log n)$ time by scanning the items in I^c in decreasing order of efficiency. The items in I_ℓ^c can be safely discarded. This decreases the optimal value by at most $4\varepsilon y$. For every remaining group, $2\varepsilon y < p(I_j^c) \leq 4\varepsilon y$, and we replace it with a single meta-item $(p(I_j^c), w(I_j^c))$. We claim that replacing all I_j^c 's with meta-items decreases the optimal value by at most $4\varepsilon y$. To see this, consider any subset S of cheap items. By selecting

meta-items in decreasing order of efficiency, we can always obtain a subset S' of meta-items such that $p(S') \geq p(S) - 4\epsilon y$ and $w(S') \leq w(S)$.

Now every item has $p_i > 2\epsilon y$. Without loss of generality, we assume that $p_i \leq 2y$ (since $OPT \leq 2y$). Then we scale all the profit by $2\epsilon y$. That is, we set $p_i := \frac{p_i}{2\epsilon y}$ for every p_i . After scaling, every item has $p_i \in (1, \frac{1}{\epsilon}]$, and $\frac{1}{2\epsilon} \leq OPT \leq \frac{1}{\epsilon}$.

C.1.2 Preprocessing Weights

Preprocessing weights is similar to preprocessing profits. We say an item is small if $w_i \leq \epsilon t$. Let I^s be the set of small items. We partition I^s into I_1^s, \dots, I_ℓ^s so that

- (i) For every integer $j \in [1, \ell - 1]$, the efficiency of an item in I_j^s is no smaller than that of any item in I_{j+1}^s . That is, $\frac{p}{w} \geq \frac{p'}{w'}$ for any $(w, p) \in I_j^s$ and any $(w', p') \in I_{j+1}^s$.
- (ii) $\epsilon t < p(I_j^s) \leq 2\epsilon t$ for every integer $j \in [1, \ell - 1]$ and $p(I_\ell^s) \leq 2\epsilon t$.

It can be done in $O(n \log n)$ time by scanning the items in I^s in decreasing order of efficiency. The items in I_ℓ^s can be ignored, since we can always select these items using an additional capacity of at most $2\epsilon t$. For every remaining group, $\epsilon t < p(I_j^s) \leq 2\epsilon t$, and we replace it with a single meta-item $(p(I_j^s), w(I_j^s))$. We claim that replacing all I_j^s 's with meta-items increases the given capacity by at most $2\epsilon t$. To see this, consider any subset S of small items. By selecting meta-items in decreasing order of efficiency, we can always obtain a subset S' of meta-items such that $p(S') \geq p(S)$ and $w(S') \leq w(S) + 2\epsilon t$.

Now every item has $w_i > \epsilon t$. Without loss of generality, we assume that $w_i \leq t$. Then we scale all the weights by ϵt . That is, we set $w_i := \frac{w_i}{\epsilon t}$ for every w_i . After scaling, every item has $w_i \in (1, \frac{1}{\epsilon}]$, and $t = \Theta(\frac{1}{\epsilon})$.

C.2 Partitioning Items into Groups

After preprocessing, $\frac{1}{2\epsilon} \leq OPT \leq \frac{1}{\epsilon}$ and $t = \frac{1}{\epsilon}$. Our goal becomes computing a set that approximates $\mathcal{S}^+(I) \cap ([0, \frac{1}{\epsilon}] \times [0, \frac{1}{\epsilon}])$ with additive error $(\tilde{O}(1), \tilde{O}(1))$. We shall compute such a set by the following three steps.

- (i) Recall that, after preprocessing, all the tuples in I are in $[1, \frac{1}{\epsilon}] \times [1, \frac{1}{\epsilon}]$. Partition I into $O(\log^2 \frac{1}{\epsilon})$ groups so that within each group, all item weights differ by a factor of at most 2, and all item profits differ by a factor of at most 2.
- (ii) for each group I' , compute a set S' that approximates $\mathcal{S}^+(I') \cap ([0, \frac{1}{\epsilon}] \times [0, \frac{1}{\epsilon}])$ with additive error $(\tilde{O}(1), \tilde{O}(1))$.
- (iii) merging the resulting sets via Lemma 3.

Step (i) costs $\tilde{O}(n)$ time. Since there are only $O(\log^2 \frac{1}{\epsilon})$ groups, Step (iii) costs $\tilde{O}((\frac{1}{\epsilon})^{3/2})$ time and increases the additive error by polylogarithmic factors.

It remains to consider Step (ii). Let I' be an arbitrary group. The tuples in I' are from $[a, 2a] \times [b, 2b]$ for some $a, b \in [1, \frac{1}{2\epsilon}]$. Let $\alpha = \frac{\max\{a, b\}}{2}$. Note that $\alpha \in [\frac{1}{2}, \frac{1}{4\epsilon}]$. By scaling, we can assume that every $(w, p) \in [1, 2] \times [1, 2]$. After the scaling, the permissible additive error becomes $(\tilde{O}(1/a), \tilde{O}(1/b)) \geq (\tilde{O}(1/\alpha), \tilde{O}(1/\alpha))$. The part of $\mathcal{S}^+(I')$ that needs to be approximated becomes $\mathcal{S}^+(I') \cap ([0, \frac{1}{\alpha\epsilon}] \times [0, \frac{1}{b\epsilon}])$. Since the tuples in I' are from $[1, 2] \times [1, 2]$, we have $\frac{1}{2} \leq \frac{p}{w} \leq 2$ for every $(w, p) \in \mathcal{S}^+(I')$. Therefore,

$$(\mathcal{S}^+(I') \cap ([0, \frac{1}{\alpha\epsilon}] \times [0, \frac{1}{b\epsilon}])) \subseteq \mathcal{S}^+(I') \cap ([0, \frac{1}{\alpha\epsilon}] \times [0, \frac{1}{\alpha\epsilon}]).$$

In conclusion, all we need to do is to approximate $\mathcal{S}^+(I') \cap ([0, \frac{1}{\alpha\epsilon}] \times [0, \frac{1}{\alpha\epsilon}])$ with additive error $(\tilde{O}(\frac{1}{\alpha}), \tilde{O}(\frac{1}{\alpha}))$.

D

 Details of the Algorithm for Bounded Solution Size

We shall prove the following lemma.

► **Lemma 8.** *Let I be a set of items from $[1, 2] \times [1, 2]$. In $\tilde{O}(n + (\frac{1}{\varepsilon})^{5/2} \frac{1}{\alpha})$ time and with high probability, we can compute a set S of size $\tilde{O}(\frac{1}{\varepsilon})$ that approximates $\mathcal{S}^+(I; \frac{1}{\alpha\varepsilon})$ with factor $1 + O(\varepsilon)$.*

Since the items in I are from $[1, 2] \times [1, 2]$, we have that $|X| \leq \frac{1}{\alpha\varepsilon}$ for any $X \subseteq I$ with $(w(X), p(X)) \in \mathcal{S}^+(I; \frac{1}{\alpha\varepsilon})$. To deal with this case where the solution size is small, we use color coding. To reduce the running time, Bringmann [4] proposed a two-layer color-coding technique, leading to an $\tilde{O}(n + t)$ -time algorithm for Subset Sum. This technique was also used in later works for Subset Sum [13, 15].

Let X be a subset of I with $(w(X), p(X)) \in \mathcal{S}^+(I; \frac{1}{\alpha\varepsilon})$. Roughly speaking, if we randomly partition I into $\tilde{O}(\frac{1}{\alpha\varepsilon})$ subsets, then with high probability, each subset contains at most one element from X . After adding $(0, 0)$ to each subset, and merging them via Lemma 3, we can obtain a set that (approximately) contains $(w(X), p(X))$ in $\tilde{O}((\frac{1}{\varepsilon})^{5/2} \frac{1}{\alpha})$ time.

The standard color-coding is based on balls-and-bins analysis, and can be summarized by the following fact.

► **Fact 25.** *Let I be a set of items. Let $k > 0$ be an integer. Let $I_1 \cup \dots \cup I_{k^2}$ be a random partition of I . For any $X \subseteq I$ and $|X| \leq k$, with probability at least $1/4$, we have that $|X \cap I_i| \leq 1$ for all $i \in \{1, \dots, k^2\}$.*

► **Lemma 26.** *Let I be a set of items. Let $k > 0$ be an integer. In $\tilde{O}(n + (\frac{1}{\varepsilon})^{3/2} k^2)$ time, we can compute a set \tilde{S} of size $\tilde{O}(\frac{1}{\varepsilon})$ that for any $X \subseteq I$ with $|X| \leq k$ and $(w(X), p(X)) \in \mathcal{S}^+(I)$, with high probability, \tilde{S} approximates $(w(X), p(X))$ with factor $1 + O(\varepsilon)$.*

Algorithm 1

 ColorCoding(I, k, q) [4]

Input: A set of n tuples I , a positive integer k , and a error probability δ

Output: a set of S

- 1: **for** $j = 1, \dots, \lceil \log \frac{1}{q} \rceil$ **do**
 - 2: Randomly partition I into k^2 subsets $I_1^j, \dots, I_{k^2}^j$
 - 3: Compute \tilde{S}_j that approximates $(I_1^j \cup \{(0, 0)\}) \oplus \dots \oplus (I_{k^2}^j \cup \{(0, 0)\})$ with factor $1 + O(\varepsilon)$ by Lemma 3.
 - 4: Compute $\tilde{S} = \cup_j \tilde{S}_j$
 - 5: **return** \tilde{S}
-

Proof. TOPROVE 10 ◀

However, the running time of Algorithm 1 depends on k^2 , which is too large for our purpose. So we use the two-layer color-coding technique proposed by Bringmann [4], which reduces the running time by a factor of k .

► **Lemma 27 ([4]).** *Let I be a set of items. Let $k > 0$ be an integer, and let $q \in (0, 1)$. Let I_1, \dots, I_k be a random partition of I . For any $X \subseteq I$ with $|X| \leq k$, with probability at least $1 - q$, we have that $|X \cap I_j| \leq 6 \log \frac{k}{q}$ for all $j \in \{1, \dots, k\}$.*

► **Lemma 28.** *Let I be a set of items. Let $k > 0$ be an integer. In $\tilde{O}(n + (\frac{1}{\varepsilon})^{3/2} k)$ time, we can compute a set \tilde{S} of size $\tilde{O}(\frac{1}{\varepsilon})$ that for any $X \subseteq I$ with $|X| \leq k$ and $(w(X), p(X)) \in \mathcal{S}^+(I)$, with high probability, \tilde{S} approximates $(w(X), p(X))$ with factor $1 + O(\varepsilon)$.*

Proof. TOPROVE 11 ◀

Since for any $(w(X), p(X)) \in \mathcal{S}^+(I; \frac{1}{\alpha\varepsilon})$, $|X| \leq \frac{1}{\alpha\varepsilon}$, we can prove Lemma 8.

Proof. TOPROVE 12 ◀

E Details of the Proximity Bound

We shall prove the following lemma.

► **Lemma 9.** *Let I be a set of items from $[1, 2] \times [1, 2]$. Let $(w^*, p^*) \in \mathcal{S}^+(I)$. Let $I^* \subseteq I$ be a subset of items with $w(I^*) = w^*$ and $p(I^*) = p^*$. Let ρ_b be the efficiency of the breaking item with respect to w^* . For any $I' \subseteq I$,*

(i) *if every item in I' has efficiency at most $\rho_b - \Delta$ for some $\Delta > 0$, then*

$$w(I' \cap I^*) \leq \frac{2}{\Delta};$$

(ii) *if every item in I' has efficiency at least $\rho_b + \Delta$ for some $\Delta > 0$, then*

$$w(I' \setminus I^*) \leq \frac{2}{\Delta}.$$

Proof. TOPROVE 13 ◀

F An Improved Algorithm with Exponent 7/4 (Full Version)

In our $\tilde{O}(n + (\frac{1}{\varepsilon})^{11/6})$ -time algorithm, only the bad groups benefit from the proximity bound, which allows us to approximate with larger factor $1 + \frac{1}{\alpha\tau}$, while for the good groups, we simply approximate with factor $1 + \varepsilon$. To further improve the running time, we shall partition the items in a way that all the groups can benefit from the proximity bound.

We assume that $\frac{1}{2} \leq \alpha \leq (\frac{1}{\varepsilon})^{3/4}$ since the algorithm in Lemma 8 already runs in $\tilde{O}(n + (\frac{1}{\varepsilon})^{7/4})$ time for all $\alpha \geq (\frac{1}{\varepsilon})^{3/4}$. We assume that $I = \{(w_1, p_1), \dots, (w_n, p_n)\}$ and that $\frac{p_1}{w_1} \geq \dots \geq \frac{p_n}{w_n}$.

We first partition I into the head group I_{head} and the tail group I_{tail} where I_{head} contains the first $\lceil \frac{1}{\alpha\varepsilon} \rceil + 1$ items and I_{tail} contains the rest of the items.

F.1 Approximating Head Group

Let $n' = \lceil \frac{1}{\alpha\varepsilon} \rceil + 1$. We shall show that in $\tilde{O}((\frac{1}{\varepsilon})^{7/4})$ time, we can compute a set S of size $\tilde{O}(\frac{1}{\varepsilon})$ that approximate $\mathcal{S}^+(I_{\text{head}})$ with additive error $O(\frac{1}{\alpha})$.

Let τ be a parameter that will be specified later. Roughly speaking, we will further partition I_{head} into good groups and bad groups. The bad groups are the same as before: a bad group is a group of τ items whose efficiencies differ by at least $\frac{1}{\tau}$. For good groups, we will make them larger than before: when we obtain a good group I' of size τ , we will keep adding items to I' until the difference between the maximum and the minimum item efficiencies exceeds $\frac{1}{|I'|}$ with the next item. A more precise description of the partition process is given below.

We partition I_{head} into I_1, \dots, I_m as follows. Initially, $j = 1$ and $k = 1$, and the remaining items are $\{(w_k, p_k), \dots, (w_{n'}, p_{n'})\}$. Let k' be the minimum integer such that

$$(k' - k) \cdot \left(\frac{p_k}{w_k} - \frac{p_{k'}}{w_{k'}} \right) > 1.$$

If such k' does not exist, we set $I_j = \{(w_k, p_k), \dots, (w_{n'}, p_{n'})\}$, set $\Delta_j = \Delta'_j = \frac{p_k}{w_k} - \frac{p_{n'}}{w_{n'}}$, set $m := j$, and finish. Assume that k' exists.

- If $k' - k \geq \tau$, we set $I_j = \{(w_k, p_k), \dots, (w_{k'-1}, p_{k'-1})\}$, set $\Delta_j = \frac{p_k}{w_k} - \frac{p_{k'-1}}{w_{k'-1}}$, set $\Delta'_j = \frac{p_k}{w_k} - \frac{p_{k'}}{w_{k'}}$, and proceed with $j := j + 1$ and $k := k'$. In this case, we say I_j is a good group.
- Otherwise, we let $I_j = \{(w_k, p_k), \dots, (w_{k+\tau-1}, p_{k+\tau-1})\}$, set $\Delta_j = \frac{p_k}{w_k} - \frac{p_{k+\tau-1}}{w_{k+\tau-1}}$, set $\Delta'_j = \frac{p_k}{w_k} - \frac{p_{k+\tau}}{w_{k+\tau}}$, and proceed with $j := j + 1$ and $k := k + \tau$. In this case, we say that I_j is a bad group.

(We remark that Δ_j and Δ'_j are not required by the algorithm. They are maintained only for the purpose of analysis. Δ_j is the actual difference between the maximum and minimum item efficiencies in I_j . For technical reasons, we also need Δ'_j . Basically, in addition to Δ_j , the gap Δ'_j also includes the efficiency gap between the minimum efficiency in I_j and the maximum efficiency in I_{j+1} . We will use Δ_j to bound the time cost of I_j , and use Δ'_j to create an efficiency gap for other groups.)

We have the following observations.

► **Observation 29.** *The groups I_1, \dots, I_m satisfy the following properties.*

- (i) $m \leq \frac{1}{\alpha\epsilon\tau} + 1$
- (ii) $\sum_{j=1}^m |I_j| = |I_{\text{head}}| = \frac{1}{\alpha\epsilon} + 1$ and $\sum_{j=1}^m \Delta_j \leq \sum_{j=1}^m \Delta'_j \leq \frac{3}{2}$.
- (iii) For every group I_j , the efficiencies of the items in I_j differ by at most Δ_j .
- (iv) For every good group I_j , we have $|I_j|\Delta_j \leq 1$ and $|I_j|\Delta'_j \geq \frac{1}{2}$.
- (v) For every bad group I_j , we have $|I_j|\Delta'_j \geq |I_j|\Delta_j > 1$.
- (vi) For the last group I_m , we have $|I_m|\Delta_m \leq 1$.

For each group, we shall approximate it in exactly the same way as we did for the bad groups in the $\tilde{O}(n + (\frac{1}{\epsilon})^{11/6})$ -time algorithm, except that we shall use $\frac{1}{\alpha|I_j|}$ as the accuracy parameter. More specifically, we use Corollary 12 and Corollary 13 to compute a set S_j of size $\tilde{O}(\alpha|I_j|)$ that approximate each tuple $(w, p) \in \mathcal{S}^+(I_j)$ with additive error $\frac{1}{\alpha|I_j|} \cdot \min\{w, w(I) - w\}$. The time cost for I_j is $\tilde{O}(|I_j| + \alpha|I_j| + \alpha^2|I_j|^2\Delta_j)$.

Then we compute a set S_{head} of size $\tilde{O}(\frac{1}{\epsilon})$ that approximates $S_1 \oplus \dots \oplus S_m$ with factor $1 + O(\epsilon)$ via Lemma 3.

Bounding Time Cost

Let J_{good} be the set of indices of good groups, and let J_{bad} be the set of indices of bad groups.

For the good groups and the last group, the total time cost is

$$\sum_{j \in J_{\text{good}} \cup \{m\}} \tilde{O}(|I_j| + \alpha|I_j| + \alpha^2|I_j|^2\Delta_j) \leq \sum_{j \in J_{\text{good}} \cup \{m\}} \tilde{O}(\alpha^2|I_j|) \leq \tilde{O}(\frac{\alpha}{\epsilon}).$$

The first inequality is due to that $|I_j|\Delta_j \leq 1$ (Observation 29(iv) and (vi)), and the last inequality is due to that $\sum_j |I_j| = \frac{1}{\alpha\epsilon} + 1$ (Observation 29(ii)).

For the bad groups, the total time cost is

$$\sum_{j \in J_{\text{bad}}} \tilde{O}(|I_j| + \alpha|I_j| + \alpha^2|I_j|^2\Delta_j) \leq \sum_{j \in J_{\text{bad}}} \tilde{O}(\alpha\tau + \alpha^2\tau^2\Delta_j) \leq \tilde{O}(m\alpha\tau + \alpha^2\tau^2).$$

The first inequality is due to that every bad group has size exactly τ , and the second inequality is due to that $\sum_j \Delta_j \leq 3/2$ (Observation 29(ii)).

For the merging step, note that each set S_j is of size $\tilde{O}(\alpha|I_j|)$ except for that S_m is of size $\tilde{O}(\frac{1}{\varepsilon})$. (Recall that $\sum_j |I_j| = \frac{1}{\alpha\varepsilon} + 1$.) According to Lemma 3, the time cost is

$$\tilde{O}\left(\frac{1}{\varepsilon} + \alpha \sum_j |I_j| + m\alpha\tau + \left(\frac{1}{\varepsilon}\right)^{3/2}m\right) \leq \tilde{O}\left(\frac{1}{\varepsilon} + m\alpha\tau + \left(\frac{1}{\varepsilon}\right)^{3/2}m\right).$$

So the total time cost to approximate $\mathcal{S}^+(I_{\text{head}})$ is

$$\tilde{O}\left(\frac{\alpha}{\varepsilon}\right) + \tilde{O}(m\alpha\tau + \alpha^2\tau^2) + \tilde{O}\left(\frac{1}{\varepsilon} + m\alpha\tau + \left(\frac{1}{\varepsilon}\right)^{3/2}m\right) \leq \tilde{O}\left(\frac{\alpha}{\varepsilon} + m\alpha\tau + \alpha^2\tau^2 + \left(\frac{1}{\varepsilon}\right)^{3/2}m\right).$$

Recall our assumption that $\frac{1}{2} \leq \alpha \leq \left(\frac{1}{\varepsilon}\right)^{3/4}$ and Observation 29(i) that $m \leq \frac{1}{\alpha\tau\varepsilon} + 1$. By setting $\tau = \lceil (\frac{1}{\varepsilon})^{5/6} \frac{1}{\alpha} \rceil$, one can verify that all the above time costs are bounded by $\tilde{O}((\frac{1}{\varepsilon})^{7/4})$.

Bound Additive Error

We shall show that S_{head} approximates every tuple $(w^*, p^*) \in \mathcal{S}^+(I_{\text{head}})$ with additive error $\tilde{O}(\frac{1}{\alpha})$.

Let (w^*, p^*) be an arbitrary tuple in $\mathcal{S}^+(I_{\text{head}})$. Let (w_b, p_b) be the breaking item with respect to w^* . Let $\rho_b = \frac{p_b}{w_b}$ be the efficiency of the breaking item.

To apply the proximity bound, let $I_{j'}$ be the group containing the breaking item (w_b, p_b) . To apply the proximity bound, we will show that the groups can be divided into $O(\log \frac{1}{\varepsilon})$ collections so that for each collection $\{I_{j_k+1}, I_{j_k+2}, \dots, I_{j_{k+1}-1}\}$, the efficiency gap between ρ_b and the groups in this collection is at least the maximum Δ'_j of these groups. To do this, we need the following auxiliary lemma.

► **Lemma 30.** *Let $\Delta_1, \dots, \Delta_n$ be a sequence of positive real numbers. Let Δ_{\min} and Δ_{\max} be the minimum and maximum numbers in the sequence. There exists $h = O(\log \frac{\Delta_{\max}}{\Delta_{\min}})$ indices $1 = j_1 < j_2 < \dots < j_h = n$ such that for any $k \in \{1, \dots, h-1\}$, we have that*

$$\max\{\Delta_j : j_k < j < j_{k+1}\} \leq \sum \{\Delta_j : j_{k+1} \leq j \leq n\}, \quad (2)$$

where the maximum of an empty set is defined to be 0.

Proof. TOPROVE 14 ◀

► **Lemma 31.** $S_1 \oplus \dots \oplus S_m$ approximates (w^*, p^*) with additive error $\tilde{O}(\frac{1}{\alpha})$.

Proof. TOPROVE 15 ◀

The above lemma immediately implies the following since S_{head} approximates $S_1 \oplus \dots \oplus S_m$ with factor $1 + O(\varepsilon)$.

► **Corollary 32.** S_{head} approximates (w^*, p^*) with additive error $\tilde{O}(\frac{1}{\alpha})$.

We summarize this subsection by the following lemma.

► **Lemma 33.** *In $\tilde{O}((\frac{1}{\varepsilon})^{7/4})$ time and with high probability, we can compute a set of size $\tilde{O}(\frac{1}{\varepsilon})$ that approximates $\mathcal{S}^+(I_{\text{head}})$ with additive error $\tilde{O}(\frac{1}{\alpha})$.*

Algorithm 2 PartitionTailGroup(I_{tail})

```

1:  $i = \lceil \frac{1}{\alpha\epsilon} \rceil + 2, j = 0$  and  $I_j = \emptyset$ 
2: while  $i \leq n$  do
3:   if the item efficiencies of  $I_j \cup \{(w_i, p_i)\}$  differ by at most  $2^{j-1}\alpha\epsilon$  then
4:      $I_j := I_j \cup \{(w_i, p_i)\}$  and  $i := i + 1$ 
5:   else
6:      $j := j + 1$  and  $I_j := \emptyset$ 
7:  $m := j$ 
8: return  $I_1, \dots, I_m$ 

```

F.2 Approximating the Tail Group

Recall that the tail group I_{tail} contains $\{(w_{\lceil \frac{1}{\alpha\epsilon} \rceil + 2}, p_{\lceil \frac{1}{\alpha\epsilon} \rceil + 2}), \dots, (w_n, p_n)\}$. We partition I_{tail} into $m = O(\log \frac{1}{\epsilon})$ groups I_0, \dots, I_m so that I_j is a maximal group of items whose efficiencies differ by at most $2^j\alpha\epsilon$. See Algorithm 2 for details.

For each group I_j , we approximate it with accuracy parameter $2^j\epsilon$. More precisely, we compute a set S_j of size $\tilde{O}(\frac{1}{2^j\epsilon})$ that approximates each tuple $(w, p) \in \mathcal{S}^+(I_j)$ with additive error $2^j\epsilon w$ using Corollary 12 in time $\tilde{O}(|I_j| + \frac{1}{2^j\epsilon} + (\frac{1}{2^j\epsilon})^2 \cdot 2^j\alpha\epsilon)$.

Then we compute a set S_{tail} that approximates $S_0 \oplus \dots \oplus S_m$ with factor $1 + O(\epsilon)$ via Lemma 3. Note that the total size of S_1, \dots, S_m is $\tilde{O}(\frac{1}{\epsilon})$. Since $m = O(\log \frac{1}{\epsilon})$, the time cost of Lemma 3 is $\tilde{O}((\frac{1}{\epsilon})^{3/2})$.

Bounding Time Cost

It is easy to see that the total time cost is

$$\tilde{O}((\frac{1}{\epsilon})^{3/2}) + \sum_j \tilde{O}(|I_j| + \frac{1}{2^j\epsilon} + (\frac{1}{2^j\epsilon})^2 \cdot 2^j\alpha\epsilon) = \tilde{O}(n + (\frac{1}{\epsilon})^{3/2} + \frac{\alpha}{\epsilon}).$$

Recall that $\frac{1}{2} \leq \alpha \leq (\frac{1}{\epsilon})^{3/4}$. The total time cost is $\tilde{O}(n + (\frac{1}{\epsilon})^{7/4})$.

Bounding Additive Error

Let (w^*, p^*) be an arbitrary tuple in $\mathcal{S}^+(I; \frac{1}{\alpha\epsilon})$. Let (w_b, p_b) be the breaking item with respect to w^* . Let $\rho_b = \frac{p_b}{w_b}$ be the efficiency of the breaking item. Note that $b \leq \frac{1}{\alpha\epsilon} + 1$, so $b \in I_{\text{head}}$. Recall that S_{head} is the set we computed by Lemma 33.

► **Lemma 34.** $S_{\text{head}} \oplus S_0 \oplus S_1 \oplus \dots \oplus S_m$ approximates (w^*, p^*) with additive error $\tilde{O}(\frac{1}{\alpha})$.

Proof. TOPROVE 16 ◀

The above lemma immediately implies the following since S_{tail} approximates $S_0 \oplus S_1 \oplus \dots \oplus S_m$ with factor $1 + O(\epsilon)$.

► **Corollary 35.** $S_{\text{head}} \oplus S_{\text{tail}}$ approximates (w^*, p^*) with additive error $\tilde{O}(\frac{1}{\alpha})$.

F.3 Putting Things Together

We summarize this section with the following lemma.

► **Lemma 36.** *There is an $\tilde{O}(n + (\frac{1}{\epsilon})^{7/4})$ -time algorithm for the reduced problem, which is randomized and succeeds with high probability.*