Faster Algorithms for Bounded Knapsack and Bounded Subset Sum Via Fine-Grained Proximity Results

Lin Chen[*]   Jiayi Lian[†]   Yuchen Mao[‡]   Guochuan Zhang[§]

Abstract

We investigate pseudopolynomial-time algorithms for Bounded Knapsack and Bounded Subset Sum. Recent years have seen a growing interest in settling their fine-grained complexity with respect to various parameters. For Bounded Knapsack, the number of items n and the maximum item weight $w_{\max}$ are two of the most natural parameters that have been studied extensively in the literature. The previous best running time in terms of n and $w_{\max}$ is $O(n + w_{\max}^3)$ [Polak, Rohwedder, Węgrzycki '21]. There is a conditional lower bound of $(n + w_{\max})^{2-o(1)}$ based on (min, +)-convolution hypothesis [Cygan, Mucha, Węgrzycki, Włodarczyk '17]. We narrow the gap significantly by proposing an $\widetilde{O}(n + w_{\max}^{12/5})$-time algorithm. Our algorithm works for both 0-1 Knapsack and Bounded Knapsack. Note that in the regime where $w_{\max} \approx n$, our algorithm runs in $\widetilde{O}(n^{12/5})$ time, while all the previous algorithms require $\Omega(n^3)$ time in the worst case.

For Bounded Subset Sum, we give two algorithms running in $\widetilde{O}(nw_{\max})$ and $\widetilde{O}(n + w_{\max}^{3/2})$ time, respectively. These results match the currently best running time for 0-1 Subset Sum. Prior to our work, the best running times (in terms of n and $w_{\max}$) for Bounded Subset Sum are $\widetilde{O}(n + w_{\max}^{5/3})$ [Polak, Rohwedder, Węgrzycki '21] and $\widetilde{O}(n + \mu_{\max}^{1/2} w_{\max}^{3/2})$ [implied by Bringmann '19 and Bringmann, Wellnitz '21], where $\mu_{\max}$ refers to the maximum multiplicity of item weights.

1 Introduction

Knapsack and Subset Sum are two of the most fundamental problems in combinatorial optimization. In 0-1 Knapsack, we are given a set of n items and a knapsack of capacity t. Each item i has a weight w_i and a profit p_i. Assume $t, p_i, w_i \in \mathbb{N}$. We should select items so as to maximize the total profit subject to the capacity constraint. Subset Sum is a special case of Knapsack where every item has its profit equal to its weight. In 0-1 Subset Sum, given a set of n items with weight $\{w_i\}_{i \in [n]}$ and a target t, we are asked whether there is some subset of items whose total weight is t. The two problems can be naturally generalized to the bounded case where each item i can be selected up to u_i times. We investigate Bounded Knapsack and Bounded Subset Sum. Throughout the rest of the paper, Knapsack and Subset Sum always refer to the bounded case unless stated otherwise.

Knapsack and Subset Sum are both weakly NP-hard, and can be solved in pseudopolynomial time using standard dynamic programming [5]. In recent years, there has been a series of works trying to settle the best possible pseudopolynomial running time for these problems with respect to various parameters, including n, t, the total number of item copies $N = \sum_i u_i$, the maximum weight $w_{\max}$, the maximum profit $p_{\max}$, and the optimal total profit OPT. Table 1 and Table 2 list the known pseudopolynomial-time algorithms for Knapsack and Subset Sum, respectively. Of particular interest are those algorithms that have only n and $w_{\max}$ as parameters, because n and $w_{\max}$ can be much smaller than N and t, but not vice versa.

For Knapsack, the standard dynamic programming has a running time of $O(N^2 w_{\max})$ as $t \leqslant N w_{\max}$. Tamir [28] gave an $O(n^3 w_{\max}^2)$-time algorithm, which is the first algorithm with running time depending only on n and $w_{\max}$. The results via fast (min, +)-convolution in Reference [3, 4, 21] imply an $\widetilde{O}(N w_{\max}^2)$-time[1] algorithm. Bateni et al. [4] also gave an improved $\widetilde{O}(n \cdot w_{\max}^2 \cdot \min\{n, w_{\max}\})$-time algorithm. Eisenbrand and Weismantel [16]

Table 1: Pseudopolynomial-time algorithms for Bounded Knapsack.

Bounded Knapsack	Reference
$\widetilde{O}(n \cdot \min\{t, OPT\})$	Bellman [5]
$O(N \cdot w_{\max} \cdot p_{\max})$	Pisinger [25]
$O(n^3 \cdot w_{\max}^2)$	Tamir [28]
$\widetilde{O}(n + w_{\max} \cdot t)$	Kellerer and Pferschy [21], also [3, 4]
$\widetilde{O}(n + p_{\max} \cdot t)$	Bateni et al. [4]
$\widetilde{O}(n \cdot w_{\max}^2 \cdot \min\{n, w_{\max}\})$	Bateni et al. [4]
$O(N \cdot \min\{w_{\max}^2, p_{\max}^2\})$	Axiotis and Tzamos [3]
$\widetilde{O}(n \cdot \min\{w_{\max}^2, p_{\max}^2\})$	Eisenbrand and Weismantel [16]
$O(n + \min\{w_{\max}^3, p_{\max}^3\})$	Polak et al. [26]
$\widetilde{O}(n + (t + OPT)^{1.5})$	Bringmann and Cassis [8]
$\widetilde{O}(N \cdot \min\{w_{\max} \cdot p_{\max}^{2/3}, p_{\max} \cdot w_{\max}^{2/3}\})$	Bringmann and Cassis [9]
$\widetilde{O}(n + \min\{w_{\max}^{12/5}, p_{\max}^{12/5}\})$	This Paper

Table 2: Pseudopolynomial-time algorithms for Bounded Subset Sum. In (*), $\mu_{\max} = \max_w\{\sum_{i:w_i=w} u_i\}$.

Bounded Subset Sum	Reference
$\widetilde{O}(n \cdot \min\{t, OPT\})$	Bellman [5]
$O(N \cdot w_{\max})$	Pisinger [25]
$\widetilde{O}(n + \min\{\sqrt{n} \cdot t, t^{5/4}\})$	Koiliaris and Xu [22]
$\widetilde{O}(n + t)$	Bringmann [6], Jin and Wu [20]
$\widetilde{O}(N + \mu_{\max}^{1/2} \cdot w_{\max}^{3/2})$*	Bringmann and Wellnitz [11] + [6]
$\widetilde{O}(n + w_{\max}^{5/3})$	Polak et al. [26]
$\widetilde{O}(n \cdot w_{\max})$	This Paper
$\widetilde{O}(n + w_{\max}^{3/2})$	This Paper

proved a proximity result, with which an $\widetilde{O}(nw_{\max}^2)$-time algorithm for Knapsack can be obtained. Combining the proximity result [16] and the convolution framework [3, 4, 21], Polak et al. [26] showed that Knapsack can be solved in $O(n + w_{\max}^3)$ time. There is also a conditional lower bound (for 0-1 Knapsack) of $(n + w_{\max})^{2-o(1)}$ based on the $(\min, +)$-convolution conjecture [14, 23]. It remains open whether Knapsack can be solved in $O(n + w_{\max}^2)$ time [8].

For Subset Sum, Pisinger [25] improved the standard dynamic programming and obtained a running time of $O(Nw_{\max})$. We remark that the dense Subset Sum result by Galil and Margalit [17] implies an $\widetilde{O}(n + w_{\max}^{3/2})$-time algorithm, but their algorithm requires all w_i to be distinct and $u_{\max} = 1$ and is not applicable for Bounded Subset Sum. Hence, we do not include it in Table 2. Nevertheless, Bringmann and Wellnitz [11] later refined and extended the dense Subset Sum result, and their result implies an $\widetilde{O}(N + \mu_{\max}^{1/2} w_{\max}^{3/2})$-time algorithm where $\mu_{\max} = \max_w \sum_{i:w_i=w} u_i$. (Note that $\sum_{i:w_i=w} u_i$ is the total number of item copies whose weight is w). Polak et al. [26] gave the first algorithm for Bounded Subset Sum that depends only on n and $w_{\max}$. Their algorithm runs in $\widetilde{O}(n + w_{\max}^{5/3})$-time. There is a conditional lower bound of $\Omega(n + w_{\max}^{1-o(1)})$ implied by the Set Cover Hypothesis [6] and Strong Exponential Time Hypothesis [1].

1.1 Our Results

THEOREM 1.1. *There is an $\widetilde{O}(n + w_{\max}^{12/5})$-time algorithm for Bounded Knapsack.*

Our result significantly narrows the gap between the previous upper bound of $O(n + w_{\max}^3)$ and the lower bound of $\Omega(n + w_{\max}^{2-o(1)})$. Moreover, in the regime where $w_{\max}$ is $O(n)$, our algorithm outperforms all the previous

algorithms that have only n and $w_{\max}$ as parameters. In particular, when $w_{\max} \approx n$ and $t \approx n^2$, our algorithm is the first to guarantee a subcubic running time in this regime, while all previous algorithms require $\Omega(n^3)$ time[2].

Due to the symmetry of weights and profits, we can also obtain an $\widetilde{O}(n + p_{\max}^{12/5})$-time algorithm by exchanging the roles of weights and profits [3, 26].

COROLLARY 1.1. *There is an $\widetilde{O}(n + p_{\max}^{12/5})$-time algorithm for Bounded Knapsack.*

THEOREM 1.2. *There is an $\widetilde{O}(nw_{\max})$-time algorithm for Bounded Subset Sum.*

THEOREM 1.3. *There is an $\widetilde{O}(n + w_{\max}^{3/2})$-time randomized algorithm for Bounded Subset Sum.*

The best algorithms (in terms of n and $w_{\max}$) for 0-1 Subset Sum run in $O(nw_{\max})$ and $\widetilde{O}(n + w_{\max}^{3/2})$ time, but they cannot be generalized to Bounded Subset Sum. Indeed, prior to our work, the best-known algorithm for Bounded Subset Sum in terms of these two parameters is that of Polak et al. [26] running in $\widetilde{O}(n + w_{\max}^{5/3})$ time. Pisinger's algorithm [25] runs in $O(nw_{\max})$ time on 0-1 Subset Sum but requires $O(Nw_{\max})$ on Bounded Subset Sum[3]. Meanwhile, the works of Galil and Margalit [17] and Bringmann and Wellnitz [11] imply a running time of $\widetilde{O}(n + w_{\max}^{3/2})$ only when all $\{w_1, \ldots, w_n\}$ are distinct and all $u_i = 1$. For Bounded Subset Sum, their algorithms either do not work or have additional factors in the running time. It is natural to ask whether Bounded Subset Sum can be solved as fast as 0-1 Subset Sum. Our results provide an affirmative answer to this question (at least with respect to the known algorithms).

1.2 Technique Overview

Knapsack A proximity result [16] states that a certain greedy solution $\mathbf{g}$ and an optimal solution $\mathbf{z}$ differ in $O(w_{\max})$ items. It directly follows that the total weight Δ of these items is $O(w_{\max}^2)$. Therefore, it is possible to obtain $\mathbf{z}$ from $\mathbf{g}$ by deleting and adding only $O(w_{\max}^2)$ volume of items. The upper bound on Δ plays an important role in accelerating the $(\min, +)$-convolution framework [3, 4, 21] for Knapsack. The $(\min, +)$-convolution-based algorithm first partitions items by their weights, and for each group, computes an all-capacities solution which is a vector of length $t + 1$. These vectors are combined one by one using a linear-time concave $(\min, +)$-convolution algorithm [2], and every intermediate vector is also truncated to be of length t. Since the concave $(\min, +)$-convolution are performed for at most $w_{\max}$ times, each taking $O(t)$ time, the total running time is $O(n + w_{\max}t)$. Applying the proximity result, every solution vector and intermediate vector can be truncated to be of length roughly $w_{\max}^2$. Therefore, a running time of $O(n + w_{\max}^3)$ can be obtained in [26].

The upper bound of $O(w_{\max}^2)$ on Δ is tight in the sense that there are instances on which $\Delta = \Theta(w_{\max}^2)$. Nevertheless, Δ only characterizes the total difference of $\mathbf{g}$ and $\mathbf{z}$ on all groups, and it does not provide satisfactory answers to questions like: *how much can a particular group (or a particular set of groups) contribute to Δ?* Intuitively, since the total contribution of all groups is $O(w_{\max}^2)$, only a few groups can have large contribution. Our first fine-grained proximity result states that only $\widetilde{O}(w_{\max}^{1/2})$ groups can contribute as much as $O(w_{\max}^2)$, and rest of the groups together can contribute at most $O(w_{\max}^{3/2})$. Then except for $\widetilde{O}(w_{\max}^{1/2})$ convolutions, every convolution can be done in $O(w_{\max}^{3/2})$ time, which leads to an $\widetilde{O}(n + w_{\max}^{5/2})$-time algorithm. Then we further improve the proximity result and obtain an $\widetilde{O}(n + w_{\max}^{12/5})$ algorithm.

We define the efficiency of an item to be the ratio of its profit to weight. Our proximity result is based on a natural intuition that only the choice of items with median efficiency can differ a lot in $\mathbf{g}$ and $\mathbf{z}$, while the items with high efficiency or low efficiency are so good or so bad that both $\mathbf{g}$ and $\mathbf{z}$ tend to make the same decision. Formal proof of this idea requires additive combinatorics tools developed by a series of works including [6, 17, 24, 27]. Basically, when we select enough items with median efficiency, the multiset of weights of these items becomes "dense". Then we can use tools from dense subset sum due to Bringmann and Wellnitz [11] to show that items with high and low efficiency cannot differ too much in $\mathbf{g}$ and $\mathbf{z}$.

We remark that a recent work of Deng et al. [15] also utilized the additive combinatorics result in Subset Sum [11] to show a proximity result in the context of approximation algorithms for Knapsack. Both our approach

---

[2]The only exception is the $\widetilde{O}(Nw_{\max}p_{\max}^{2/3})$-time algorithm by Bringmann [9], but they additionally require $p_{\max} \leqslant O(n^{3/2})$ and $N \approx n$.

[3]Note that the idea of binary encoding each u_i as $1 + 2 + 4 + \ldots + 2^{\log u_i}$ (bundling item copies) does not help to reduce the running time to $O(nw_{\max})$, because although it reduces u_i to a constant, it increases $w_{\max}$ by a factor of u_i.

and theirs share a similar high-level idea, that is, if the optimal solution selects many low-efficiency items, then the high-efficiency items not selected by the optimal solution cannot form a dense set (in terms of their weights or profits), for otherwise it is possible to utilize additive combinatorics results to conduct an exchange argument on low- and high-efficiency items. There are, however, two major differences in techniques: (i). The density requirement on the high-efficiency item set in Deng et al.'s work [15] is very strong. In particular, they require any large subset of this set to be dense. This works in approximation algorithms where the input instance can be reduced to a bounded instance in which item profits differ by a constant factor. For exact algorithms, such kind of density requirement is not achievable. Nevertheless, we give a similar exchange argument that only requires a weaker density condition. (ii). Multiplicity of item weights or profits has not been considered in Deng et al.'s work [15]. We remark that the additive combinatorics result in Subset Sum by Bringmann and Wellnitz [11] is for multisets, which actually provides a trade-off between "dense threshold" and item multiplicity. Very roughly speaking, for a multiset X that consists of integers up to w, it can become dense if it contains $\widetilde{\Theta}(\sqrt{w})$ or more distinct integers, but it can also become dense if it only contains $\widetilde{\Theta}(w^{0.4})$ distinct integers where each integer has $\widetilde{\Theta}(w^{0.2})$ multiplicities. Deng et al.'s work only utilized the former result. We achieve an $\widetilde{O}(n + w_{\max}^{5/2})$-time algorithm using the former result, and show that it can be further improved to $\widetilde{O}(n + w_{\max}^{12/5})$-time by exploiting the multiplicity to build a stronger proximity result.

Subset Sum Observe that the gap between $O(nw_{\max})$ and $O(Nw_{\max})$ arises from $u_{\max}$ as $N \leqslant nu_{\max}$. A standard method for reducing $u_{\max}$ is to bundle item copies. For example, by bundling every k copies of each item, we reduce $u_{\max}$ by a factor of k. Doing this, however, increases $w_{\max}$ by the same factor. As a consequence, any trivial treatment of the resulting instance will not improve the running time. Let X be the input instance. Let $\mathcal{S}(X)$ be the set of all subset sums of X. The effect of bundling is essentially decomposing X into $X = X_0 + kX_1$ where X_0 stands for the set of the items that are not bundled (called residual items), and kX_1 stands for the set of the bundled items. We can prove that $\mathcal{S}(X) = \mathcal{S}(X_0) + k\mathcal{S}(X_1)$. Since both the residual items and the bundled items have smaller multiplicities, computing $\mathcal{S}(X_0)$ and $\mathcal{S}(X_1)$ is faster than directly computing $\mathcal{S}(X)$. Computing $\mathcal{S}(X_0) + k\mathcal{S}(X_1)$ using standard FFT, however, still requires $\widetilde{O}(\mathtt{m}_{\mathcal{S}(X_0)} + k \cdot \mathtt{m}_{\mathcal{S}(X_1)})$ time where $\mathtt{m}_{\mathcal{S}(X_i)}$ is the maximum element in $\mathcal{S}(X_i)$. Note that $\mathtt{m}_{\mathcal{S}(X_0)} + k \cdot \mathtt{m}_{\mathcal{S}(X_1)}$ can be as large as $Nw_{\max}$, so it leads to the failure of the bundling idea. We propose an FFT-based algorithm that can determine whether $t \in \mathcal{S}(X_0) + k\mathcal{S}(X_1)$ in $\widetilde{O}(\mathtt{m}_{\mathcal{S}(X_0)} + \mathtt{m}_{\mathcal{S}(X_1)})$ time for arbitrary t. This algorithm can be extended to arbitrary many levels of bundling. After recursively applying the bundling idea for logarithmic levels, we can reduce the multiplicities of items in each level to a constant. Therefore, the $\mathcal{S}(X_i)$ for each level can be computed in $\widetilde{O}(nw_{\max})$ time. Then we can determine whether $t \in \sum_i k^i \mathcal{S}(X_i)$ using our FFT-based algorithm in $\widetilde{O}(nw_{\max})$ total time.

Then we observe that when the number of distinct weights in the input is bounded by $\widetilde{O}(w_{\max}^{1/2})$, after $O(n)$-time preprocessing, Subset sum can be solved in $\widetilde{O}(w_{\max}^{3/2})$ time using our $\widetilde{O}(nw_{\max})$-time algorithm. It remains to tackle the case where the number of distinct weights is at least $\widetilde{\Theta}(w_{\max}^{1/2})$. In this case, the set of item weights is "dense", so we can apply additive combinatorics tools as we did for Knapsack. Moreover, since every item has the same efficiency in Subset Sum, we can get stronger proximity result. Indeed, we show that there is a greedy solution $\mathbf{g}$ and an optimal solution $\mathbf{z}$ that differ by at most $\Delta = \widetilde{O}(w_{\max}^{3/2})$ volume of items. Obtaining $\mathbf{z}$ from $\mathbf{g}$ can be basically reduced to two Subset Sum problems with $t = \widetilde{O}(w_{\max}^{3/2})$. Then using Bringmann's $\widetilde{O}(n + t)$-time algorithm [6] for Subset Sum, the dense case can be solved in $\widetilde{O}(n + w_{\max}^{3/2})$ time.

1.3 Further Related Work For Knapsack, $N, t, p_{\max}, OPT$ are alternative parameters that have been used in the literature. The standard dynamic programming due to Bellman together with the idea of bundling item copies gives a running time of $\widetilde{O}(nt)$ [5, 25]. Using a balancing technique, Pisinger obtained a dynamic programming algorithm that runs in $O(Nw_{\max}p_{\max})$ time [25]. Several recent advances in Knapsack build upon $(\min, +)$-convolution [3, 4, 8, 12, 21]. In particular, Bringmann and Cassis used the partition and convolve paradigm to develop an algorithm that runs in $\widetilde{O}(Nw_{\max}p_{\max}^{2/3}\})$ time [9]. The additive combinatorics techniques used in these papers also inspired progress in approximation algorithms of Knapsack [15]. Pseudopolynomial-time algorithms have also been studied extensively for Knapsack and Subset Sum in the unbounded case where $u_i = \infty$. See Reference [4, 6, 8, 13, 18, 28].

For Subset Sum, when taking the target t as a parameter, the best-known deterministic algorithm is due to Koiliaris and Xu [22], which runs in $\widetilde{O}(n + \min\{\sqrt{n}t, t^{5/4}\})$ time. The best-known randomized algorithm is due to

Bringmann [6], which runs in $\widetilde{O}(n + t)$ time. An alternative randomized algorithm with almost the same running time is given by Jin and Wu [20].

1.4 Paper Outline In Section 2, we introduce necessary terminology and preliminaries. In Section 3, we prove two exchange arguments that are crucial to our algorithms. In Section 4, we present a simpler algorithm to illustrate our main idea. The $\widetilde{O}(n + w_{\max}^{12/5})$-time algorithm for Knapsack is presented in Section 5. In Section 6, we give two algorithms for Subset Sum. Omitted proofs can be found in the appendix.

2 Preliminaries

2.1 Notation We use $\mathcal{I} = \{1, \ldots, n\}$ to denote the set of all items, each with weight w_i and profit p_i. We assume that items are labeled in decreasing order of efficiency. That is, $p_1/w_1 \geqslant \ldots \geqslant p_n/w_n$. Item i has u_i copies. Let $W = \{w_i : i \in \mathcal{I}\}$ be the set of all item weights. The maximum weight is denoted by $w_{\max}$.

Let i be some item. We denote the set of item $\{1, ..., i-1\}$ as $\mathcal{I}_{<i}$ and $\{i, \ldots, n\}$ as $\mathcal{I}_{\geqslant i}$. Let i and j be two items with $i < j$, we say i is to the left of j, and j is to the right.

For $w \in W$, we denote the set of items with weight w as

$$\mathcal{I}^w = \{i \in \mathcal{I} : w_i = w\}.$$

For any subset W' of W, we write $\bigcup_{w \in W'} \mathcal{I}^w$ as $\mathcal{I}^{W'}$. We also write $\mathcal{I}^{W'} \cap \mathcal{I}_{<i}$ as $\mathcal{I}_{<i}^{W'}$, and $\mathcal{I}^{W'} \cap \mathcal{I}_{\geqslant i}$ as $\mathcal{I}_{\geqslant i}^{W'}$. Basically, the superscript restricts the weights of items in the set, while the subscript restricts the indices.

For any subset $\mathcal{I}'$ of items, we refer to the multiset $\{w_i : i \in \mathcal{I}'\}$ as the weight multiset of $\mathcal{I}'$.

A solution to Bounded Knapsack can be represented by a vector $\mathbf{x}$ where x_i is the number of selected copies of item i.

2.2 The Previous Proximity Result Let $\mathbf{g}$ be a maximal prefix solution that can be obtained greedily as follows. Starting with an empty knapsack, we process items in increasing order of index (i.e., in decreasing order of efficiency). If the knapsack has enough room for all copies of the current item, we select all these copies, and process the next item. Otherwise, we select as many copies as possible, and then stop immediately. The item at which we stop is called the *break item*, denoted by b. Without loss of generality, we assume that no copies of the break item b are selected by $\mathbf{g}$ (i.e., $g_b = 0$). If $g_b \neq 0$, we can view those copies selected by $\mathbf{g}$ as a new item i' with $u_{i'} = g_b$, and set $u_b = u_b - g_b$. It is easy to observe that $g_i = u_i$ for all $i < b$ and that $g_i = 0$ for all $i \geqslant b$.

The following proximity result states that there is an optimal solution $\mathbf{z}$ that differs from $\mathbf{g}$ only in a few items.

LEMMA 2.1. ([16, 26]) *Let $\mathbf{g}$ be a maximal prefix solution of Knapsack. There exists an optimal solution $\mathbf{z}$ such that*

$$\|\mathbf{z} - \mathbf{g}\|_1 = \sum_{i=1}^{n} |z_i - g_i| \leqslant 2w_{\max}.$$

Throughout the rest of the paper, we fix $\mathbf{g}$ to be a maximal prefix solution and $\mathbf{z}$ to be an optimal solution that minimizes $\|\mathbf{z} - \mathbf{g}\|_1$. For any subset $\mathcal{I}'$ of items, we define the weighted ℓ_1-distance from $\mathbf{g}$ to $\mathbf{z}$ on $\mathcal{I}'$ to be

$$\Delta(\mathcal{I}') = \sum_{i \in \mathcal{I}'} w_i |g_i - z_i|.$$

2.3 Proximity-based (min, +)-Convolution Framework Both the algorithm of Polak et al. [26] and our algorithm use the proximity-based (min, +)-convolution framework. We present a general form of the framework through the following Lemma. Recall that W is the set of all item weights.

LEMMA 2.2. *Let $\mathbf{g}$ be a maximal prefix solution to Bouned Knapsack. Let $W_1 \cup \cdots \cup W_k$ be a partition of the set W. For $j \in \{1, ..., k\}$, let U_j be an upper bound for $\Delta(\mathcal{I}^{W_j})$. Then Bounded Knapsack can be solved in $\widetilde{O}(n + k\sum_{j=1}^{k} |W_j| \cdot U_j)$ time.*

We only give a sketch of the proof. The complete proof is deferred to Appendix A.

Proof Sketch. We construct an optimal solution $\mathbf{x}$ through modifying $\mathbf{g}$. Specifically, we have $\mathbf{x} = \mathbf{g} - \mathbf{x}^- + \mathbf{x}^+$, where $\mathbf{x}^-$ represents the copies of items in $\mathcal{I}_{<b}$ that are deleted from $\mathbf{g}$, and $\mathbf{x}^+$ represents the copies of items in $\mathcal{I}_{\geqslant b}$ that are added to $\mathbf{g}$, while noting that b is the break item. We shall compute two sequences, namely: (i). $\boldsymbol{x}^- = \langle x_0^-, ..., x_{k \cdot U_k}^- \rangle$, where $x_{t'}^-$ is the minimum total profit of copies in $\mathcal{I}_{<b}$ whose total weight is exactly t'; (ii). $\boldsymbol{x}^+ = \langle x_0^+, ..., x_{k \cdot U_k}^+ \rangle$, where $x_{t'}^+$ is the maximum total profit of copies in $\mathcal{I}_{\geqslant b}$ whose total weight is exactly t'. The optimal solution can be found easily using these two sequences in $O(kU_k)$ time.

The computation of $\boldsymbol{x}^-$ and $\boldsymbol{x}^+$ follows the same method as Polak et al. [26]: we consider all copies of items whose weight is exactly w, and let $\boldsymbol{s}^w$ denote the sequence whose t'-th entry is the maximum total profit of these items when their total weight is exactly t'. After computing all $\boldsymbol{s}^w$'s, we iteratively update $\boldsymbol{x}^-$ and $\boldsymbol{x}^+$ by computing their convolutions with each $\boldsymbol{s}^w$. The key point is that we can strategically pick an order of the $\boldsymbol{s}^w$'s to perform convolution. In particular, assuming that $U_1 \leqslant ... \leqslant U_k$, we have $\Delta(\mathcal{I}^{W_1 \cup ... \cup W_{k'}}) \leqslant \sum_{j=1}^{k'} U_j \leqslant k' \cdot U_{k'}$ for any $1 \leqslant k' \leqslant k$. We first convolve $\boldsymbol{s}^w$'s in W_1, then W_2, etc. When we compute the convolution with $\boldsymbol{s}^w$ where $w \in W_{k'}$, we can truncate the sequence after the $k' \cdot U_{k'}$-th entry, and thus the convolution takes $O(k' \cdot U_{k'})$ time via SMAWK algorithm [2]. It is easy to verify that the total time is $\widetilde{O}(n + \sum_{j=1}^k |W_j| \cdot j \cdot U_j) = \widetilde{O}(n + k \sum_{j=1}^k |W_j| \cdot U_j)$. □

2.4 Additive Combinatorics Additive combinatorics has been a useful tool for Subset Sum. With additive combinatorics tools, Galil and Margalit [17] gave a characterization of the regime within which Subset Sum can be solved in linear time. Their result was later improved by Bringmann and Wellnitz [11]. Our exchange argument in Section 3 heavily utilizes the dense properties by Bringmann and Wellnitz. We briefly descibe these properties in this subsection.

Let X be a multiset of positive integers. We denote the sum of X as Σ_X, the maximum element in X as $\mathtt{m}_X$, the maximum multiplicity of elements in X as μ_X, and the support of X as $\mathtt{supp}_X$. The number of elements in X is denoted by $|X|$ (with multiplicities counted). We say X can *hit* an integer s if some subset of X sums to this integer, i.e., there is a subset $X' \subseteq X$ that $\Sigma_{X'} = s$.

DEFINITION 2.1. *(Definition 3.1 in [11]) We say that a multiset X is δ-dense if it satisfies $|X|^2 \geqslant \delta \cdot \mu_X \mathtt{m}_X$.*

DEFINITION 2.2. *(Definition 3.2 in [11]) Let X be a multiset. We denote by $X(d) := X \cap d\mathbb{Z}$ the multiset of all numbers in X that are divisible by d. Further, we write $\overline{X(d)} := X \backslash X(d)$ to denote the multiset of all numbers in X not divisible by d. We say an integer $d > 1$ is an α-almost divisor of X if $|\overline{X(d)}| \leqslant \alpha \cdot \mu_X \Sigma_X / |X|^2$.*

THEOREM 2.1. *(Theorem 4.2 in [11]) Let X be a multiset of positive integers and set*

$$c_\delta := 1699200 \cdot \log(2|X|) \log^2(2\mu_X),$$
$$c_\alpha := 42480 \cdot \log(2\mu_X),$$
$$c_\lambda := 169920 \cdot \log(2\mu_X).$$

If X is c_δ-dense and has no c_α-almost divisor, then for $\lambda_X := c_\lambda \mu_X \mathtt{m}_X \Sigma_X / |X|^2$, X can hit all the integers in range $[\lambda_X, \Sigma_X - \lambda_X]$.

THEOREM 2.2. *(Theorem 4.1 in [11]) Let X be a multiset of positive integers. Let δ, α be functions of n with $\delta \geq 1$ and $0 < \alpha \leqslant \delta/16$. Given a δ-dense set X of size n, there exists an integer $d \geqslant 1$ such that $X' := X(d)/d$ is δ-dense and has no α-almost divisor. Moreover, we have the following additional properties:*

(i) $d \leqslant 4\mu_X \Sigma_X / |X|^2$,

(ii) $|X'| \geqslant 0.75|X|$,

(iii) $\Sigma_{X'} \geq 0.75 \Sigma_X / d$.

3 A General Exchange Argument

In this section, we establish two additive combinatorics results that are crucial to proving our proximity result. Basically we show that given two (multi-)sets A and B of positive integers, if A is dense, and Σ_B is large, there must be some non-empty subset A' of A and a subset B' of B such that $\Sigma_{A'} = \Sigma_{B'}$. We first give a result where multiplicities of integers are not utilized.

LEMMA 3.1. *Let w be a positive integer. Let p be an arbitrary real number such that $0.5 \leqslant p < 1$. Let A be a multiset of integers from $\{1, ..., w\}$ such that $|\mathtt{supp}_A| \geqslant c_A w^p \log w$, B be a multiset of integers from $\{1, ..., w\}$ such that $\Sigma_B \geqslant c_B w^{2-p}$, where c_A and c_B are two sufficiently large constants. Then there must exist a non-empty subset A' of A and a non-empty subset B' of B such that $\Sigma_{A'} = \Sigma_{B'}$.*

Proof. Let $S = \mathtt{supp}_A$. Note that $|S| \geqslant c_A w^p \log w$. What we actually prove is that there is a non-empty subset S' of S such that some subset B' of B have the same total weight as S'. We first characterize the set of integers that can be hit by S. Then we show that B can hit at least one of these integers.

Note that $\mathtt{m}_S \leqslant w$, $\mu_S = 1$, and $|S| \leqslant w$. Since c_A is sufficiently large, we have that

$$|S|^2 \geqslant c_A^2 w^{2p} \log^2 w \geqslant c_A^2 w \log w \geqslant c_A^2 \mu_S \mathtt{m}_S \log |S| \geqslant c_\delta \mu_S \mathtt{m}_S.$$

By definition, S is c_δ-dense. By Theorem 2.2, there exists an integer d such that $S' := S(d)/d$ is c_δ-dense and has no c_α-almost divisor. And the followings hold.

(i) $d \leqslant 4\mu_S \Sigma_S / |S|^2$,

(ii) $|S'| \geqslant 0.75|S|$.

(iii) $\Sigma_{S'} \geqslant 0.75 \Sigma_S / d$.

Note that $\mu_{S'} = 1$, $\mathtt{m}_{S'} \leqslant w/d$, and $\Sigma_{S'} \leqslant \Sigma_S/d$. Applying Theorem 2.1 on S', we get S' can hit any integer in the range $[\lambda_{S'}, \Sigma_{S'} - \lambda_{S'}]$ where

$$\lambda_{S'} = \frac{c_\lambda \mu_{S'} \mathtt{m}_{S'} \Sigma_{S'}}{|S'|^2} \leqslant \frac{c_\lambda w \Sigma_S}{(0.75|S|)^2 d^2} \leqslant \frac{\min\{c_A, c_B\}}{2} \cdot \frac{w \Sigma_S}{d^2 |S|^2}.$$

The last inequality holds since c_A and c_B are sufficiently large constants. We can conclude that S can hit any multiple of d in the range $[d\lambda_{S'}, d(\Sigma_{S'} - \lambda_{S'})]$. We also have that the left endpoint of this interval

$$d\lambda_{S'} \leqslant \frac{c_B}{2} \cdot \frac{w \Sigma_S}{d|S|^2} \leqslant \frac{c_B}{2} \cdot \frac{w^2 |S|}{|S|^2} \leqslant \frac{c_B}{2} \cdot \frac{w^2}{|S|} \leqslant \frac{c_B}{2} \cdot w^{2-p},$$

and that the length of the interval

$$
\begin{aligned}
d(\Sigma_{S'} - 2\lambda_{S'}) &\geqslant \frac{3\Sigma_S}{4} - c_A \cdot \frac{w \Sigma_S}{d|S|^2} \\
&\geqslant \frac{\Sigma_S}{|S|^2}\left(\frac{3|S|^2}{4} - c_A \cdot \frac{w}{d}\right) && \text{(since } |S| \geqslant c_A w^{1/2} \log w \text{ and } d \geqslant 1) \\
&\geqslant \frac{\Sigma_S}{|S|^2}\left(\frac{3c_A^2}{4} \cdot w - c_A \cdot w\right) && \text{(since } c_A \text{ is sufficiently large)} \\
&\geqslant 4 \cdot \frac{\Sigma_S}{|S|^2} \cdot w && \text{(since } d \leqslant 4\mu_S \Sigma_S / |S|^2 \text{ and } \mu_S = 1) \\
&\geqslant dw
\end{aligned}
$$

To complete the proof, it suffices to show that there is a subset B' of B whose sum is a multiple of d and is within the interval $[d\lambda_{S'}, d(\Sigma_{S'} - \lambda_{S'})]$. We claim that as long as B has at least d numbers, there must be a non-empty subset of B whose sum is at most dw and is a multiple of d. Assume the claim is true. We can repeatedly extract such subsets from B until B has less than d numbers. Note that the total sum of these subsets is at least

$$\Sigma_B - wd \geqslant c_B w^{2-p} - w \cdot \frac{4\Sigma_S}{|S|^2} \geqslant c_B w^{2-p} - \frac{4w^2}{|S|} \geqslant c_B w^{2-p} - w^{2-p} \geqslant \frac{c_B w^{2-p}}{2}.$$

That is, the total sum of these subsets is at least the left endpoint of $[d\lambda_{S'}, d(\Sigma_{S'} - \lambda_{S'})]$. Also note that the sum of each subset is at most dw, which does not exceed the length of the interval. As a result, there must be a collection of subsets whose total sum is within the interval. Since the sum of each subset is a multiple of d, so is any collection of these subsets.

To see why the claim is true, take d arbitrary numbers from B. For $i \in \{1, ..., d\}$, Let h_i be the sum of the first i numbers. If any of $h_i \equiv 0 \pmod{d}$, then we are done. Otherwise, by the pigeonhole principle, there must be $i < j$ such that $h_i \equiv h_j \pmod{d}$. This implies that there is a subset of $j - i$ numbers whose sum is $h_j - h_i \equiv 0 \pmod{d}$. Note that $0 < j - i \leqslant d$. So this subset is non-empty and has its sum at most dw. $\qquad\square$

Lemma 3.1 has no requirement on the multiplicities of elements in A. Therefore, in order to be dense, A must contain at least $c_A w^{1/2} \log w$ distinct integers. The following Lemma 3.2 generalizes Lemma 3.1 by taking the multiplicity of integers into consideration. Its proof is similar to that of Lemma 3.1, so we defer it to the appendix B.

LEMMA 3.2. *Let w be a positive integer. Let A and B be two multisets of integers from $\{1, ..., w\}$ such that*

(i) *at least $c_A w^{2/5} \log^2 w$ distinct integers in A have multiplicity of at least $w^{1/5}$,*

(ii) $\Sigma_B \geqslant c_B w^{8/5}$,

where c_A and c_B are two sufficiently large constants. Then there must exist a non-empty subset A' of A and a non-empty subset B' of B such that $\Sigma_{A'} = \Sigma_{B'}$.

4 An $\widetilde{O}(n + w_{\max}^{5/2})$-time Algorithm for Knapsack

To illustrate our main idea, we first present a simpler algorithm that runs in $\widetilde{O}(n + w_{\max}^{5/2})$. The algorithm uses the proximity-based $(\min, +)$-convolution framework (See Subsection 2.3). By Lemma 2.2, the key to obtaining a fast algorithm is to find a good partition of W.

Polak et al. [26] did not partition W at all. That is, they used $k = 1$ and $W_1 = W$. Lemma 2.1 implies an upper bound of $2w_{\max}^2$ on $\Delta(\mathcal{I}^W)$. Together with the fact that $|W| \leqslant w_{\max}$, they obtained a running time of $O(n + w_{\max}^3)$. Although the upper bound of $O(w_{\max}^2)$ on $\Delta(\mathcal{I}^W)$ is actually tight, an improvement in the running time is still possible. Indeed, our first proximity result states that $\Delta(\mathcal{I}^W)$ concentrates at the small set W^* of roughly $w_{\max}^{1/2}$ weights, and that the rest of the weights together contribute at most $O(w_{\max}^{3/2})$. Moreover, W^* can be identified in $\widetilde{O}(n)$ time. Then an $\widetilde{O}(n + w_{\max}^{5/2})$-time algorithm directly follows by Lemma 2.2.

This section is divided into two parts: the structural part and the algorithmic part. The structural part proves the existence of W^*, and the algorithmic part gives algorithms for finding W^* and solving Knapsack.

4.1 Structural Part – Fine-Grained Proximity In the structural part, Bounded Knapsack and 0-1 Knapsack are equivalent, in the sense that every item with u_i copies can be viewed as u_i items. Therefore, only 0-1 Knapsack is discussed in the structural part. In 0-1 Knapsack, each item i has only one copy, so there is no difference between items and copies of items. We can use them interchangeably. As a result, any solution to 0-1 Knapsack can be represented as a subset of $\mathcal{I}$. Let $\mathbf{x} \in \{0, 1\}^{|\mathcal{I}|}$ be a solution to 0-1 Knapsack. With slight abuse of notation, we also use $\mathbf{x}$ to denote the set of items selected by $\mathbf{x}$. For any subset $\mathcal{I}'$ of items, $\mathbf{x}(\mathcal{I}') = \mathcal{I}' \cap \mathbf{x}$ and $\overline{\mathbf{x}}(\mathcal{I}') = \mathcal{I}' \setminus \mathbf{x}$ denote the sets of items from $\mathcal{I}'$ that are selected and not selected by $\mathbf{x}$, respectively.

Our first proximity result is the following.

LEMMA 4.1. *There exists a partition $(W^*, \overline{W^*})$ of W such that*

(i) $|W^*| = 4c_A w_{\max}^{1/2} \log w_{\max}$, *and*

(ii) $\Delta(\mathcal{I}^{\overline{W^*}}) \leqslant 4c_B w_{\max}^{3/2}$,

where c_A and c_B are two large constants used in Lemma 3.1.

Proof. We will define W^* via a partition of $\mathcal{I}$, and then show that W^* satisfies properties in the lemma.

Defining W^* via a partition of $\mathcal{I}$. Recall that we label the items in decreasing order of efficiency. That is, $p_1/w_1 \geqslant \ldots \geqslant p_n/w_n$. Without loss of generality, we assume that for any two items $i < j < b$, if i and j have the same efficiency, then $w_i \leqslant w_j$, and that for any two items $b < i < j$, if i and j have the same efficiency, then $w_i \geqslant w_j$. We partition the items into four groups $(\mathcal{I}_1, \mathcal{I}_2, \mathcal{I}_3, \mathcal{I}_4)$ according to their indices. See Figure 1 for an illustration. We shall define $\mathcal{I}_2$ and $\mathcal{I}_3$ in such a way that (i) they are "close" to the break item b and that (ii)

items within them have $\Theta(w_{\max}^{1/2} \log w_{\max})$ distinct weights separately. Because the items in $\mathcal{I}_2$ and $\mathcal{I}_3$ have their efficiency close to that of b, it is difficult to tell how many of them should be selected by the optimal solution. In other words, $\mathbf{z}$ and $\mathbf{g}$ may differ a lot in these items. In contrast, the items in $\mathcal{I}_1$ and $\mathcal{I}_4$ have very high and very low efficiency, respectively, so $\mathbf{z}$ tends to select most of $\mathcal{I}_1$ and few of $\mathcal{I}_4$. As a result, $\mathbf{z}$ and $\mathbf{g}$ are similar in $\mathcal{I}_1$ and $\mathcal{I}_4$.

Figure 1: $\mathcal{I}_1$, $\mathcal{I}_2$, $\mathcal{I}_3$, and $\mathcal{I}_4$

Now we formally describe $(\mathcal{I}_1, \mathcal{I}_2, \mathcal{I}_3, \mathcal{I}_4)$. For any $i < j$, let $\mathcal{I}_{[i,j)}$ be the set of items $\{i, \ldots, j-1\}$. Let i^* be the minimum index i such that the items in $\mathcal{I}_{[i,b)}$ have exactly $2c_A w_{\max}^{1/2} \log w_{\max}$ distinct weights. Let $\mathcal{I}_2 = \mathcal{I}_{[i^*,b)}$, and let $\mathcal{I}_1 = \mathcal{I}_{<i^*}$. When no such i^* exists, let $\mathcal{I}_2 = \mathcal{I}_{<b}$, and let $\mathcal{I}_1 = \emptyset$. $\mathcal{I}_3$ and $\mathcal{I}_4$ are defined similarly as follows. Let j^* be the maximum index j such that the items in $\mathcal{I}_{[b,j)}$ have exactly $2c_A w_{\max}^{1/2} \log w_{\max}$ distinct weights. Let $\mathcal{I}_3 = \mathcal{I}_{[b,j^*)}$, and let $\mathcal{I}_4 = \mathcal{I}_{\geqslant j^*}$. When no such j^* exists, let $\mathcal{I}_3 = \mathcal{I}_{\geqslant b}$, and let $\mathcal{I}_4 = \emptyset$.

W^* is defined as the set of the weights of the items in $\mathcal{I}_2 \cup \mathcal{I}_3$.

Verifying properties. It is straightforward that $|W^*| \leqslant 4c_A w_{\max}^{1/2} \log w_{\max}$. We are left to show $\Delta(\mathcal{I}^{\overline{W^*}}) \leqslant 2c_B w_{\max}^{3/2}$. Note that $\mathcal{I}^{\overline{W^*}} \subseteq \mathcal{I}_1 \cup \mathcal{I}_4$, so $\mathcal{I}^{\overline{W^*}}$ can be partitioned into $\mathcal{I}_1^{\overline{W^*}} = \mathcal{I}^{\overline{W^*}} \cap \mathcal{I}_1$ and $\mathcal{I}_4^{\overline{W^*}} = \mathcal{I}^{\overline{W^*}} \cap \mathcal{I}_4$, we have $\Delta(\mathcal{I}^{\overline{W^*}}) = \Delta(\mathcal{I}_1^{\overline{W^*}}) + \Delta(\mathcal{I}_4^{\overline{W^*}})$. It suffices to show that $\Delta(\mathcal{I}_1^{\overline{W^*}})$ and $\Delta(\mathcal{I}_4^{\overline{W^*}})$ are both bounded by $2c_B w_{\max}^{3/2}$. In the following, we only provide proof for $\Delta(\mathcal{I}_1^{\overline{W^*}})$. Bounds for $\Delta(\mathcal{I}_4^{\overline{W^*}})$ can be proved similarly due to symmetry.

Suppose, for the sake of contradiction, that $\Delta(\mathcal{I}_1^{\overline{W^*}}) > 2c_B w_{\max}^{3/2}$. That is, a great volume of items in $\mathcal{I}_1^{\overline{W^*}}$ are deleted when obtaining the optimal solution $\mathbf{z}$ from the greedy solution $\mathbf{g}$. (Note that $\mathbf{g}$ selects all items in $\mathcal{I}_1^{\overline{W^*}}$ as $\mathcal{I}_1^{\overline{W^*}} \subseteq \mathcal{I}_{<b}$, so $\Delta(\mathcal{I}_1^{\overline{W^*}})$ is exactly the total weight of items in $\mathcal{I}_1$ that are deleted.) Obviously, $\mathcal{I}_1^{\overline{W^*}}$ is non-empty since otherwise $\Delta(\mathcal{I}_1^{\overline{W^*}})$ would be 0. This implies that $\mathcal{I}_2$ has exactly $2c_A w_{\max}^{1/2} \log w_{\max}$ distinct weights. The greedy solution $\mathbf{g}$ picks all the items in $\mathcal{I}_2$ as $\mathcal{I}_2 \subseteq \mathcal{I}_{<b}$. When obtaining $\mathbf{z}$ from $\mathbf{g}$, at least one of the following two cases must be true.

(i) $\mathbf{z}$ keeps lots of items in $\mathcal{I}_2$ so that $\mathbf{z}(\mathcal{I}_2)$ have at least $c_A w_{\max}^{1/2} \log w_{\max}$ distinct weights.

(ii) lots of the items in $\mathcal{I}_2$ are deleted so that $\mathbf{z}(\mathcal{I}_2)$ have at most $c_A w_{\max}^{1/2} \log w_{\max}$ distinct weights. In other words, $\overline{\mathbf{z}}(\mathcal{I}_2)$ have at least $c_A w_{\max}^{1/2} \log w_{\max}$ distinct weights.

We will show that both cases lead to a contradiction. The high-level idea is the following. In case (i), when obtaining $\mathbf{z}$, lots of items in $\mathcal{I}_2$ are kept while a great volume of items in $\mathcal{I}_1^{\overline{W^*}}$ are deleted. We will show that there is a subset $\mathcal{Z} \subseteq \mathbf{z}(\mathcal{I}_2)$ of items kept by $\mathbf{z}$ and a subset $\mathcal{D} \subseteq \overline{\mathbf{z}}(\mathcal{I}_1^{\overline{W^*}})$ of the deleted items such that $\mathcal{D}$ and $\mathcal{Z}$ have the same total weight. Note that items in $\mathcal{D}$ have higher efficiency than those in $\mathcal{Z}$ as $\mathcal{D}$ is to the left of $\mathcal{Z}$. But then, when obtaining $\mathbf{z}$ from $\mathbf{g}$, why not delete $\mathcal{Z}$ rather than $\mathcal{D}$? In case (ii), lots of items in $\mathcal{I}_2$ are deleted. Also, since a great volume of items (in $\mathcal{I}_1^{\overline{W^*}}$) are deleted, at least the same volume of items should be added in order to obtain the optimal solution $\mathbf{z}$. We will show that there is a subset $\mathcal{D} \subseteq \overline{\mathbf{z}}(\mathcal{I}_2)$ of the deleted items and a subset $\mathcal{A} \subseteq \mathbf{z}(\mathcal{I}_{\geqslant b})$ of the added items such that $\mathcal{A}$ and $\mathcal{D}$ have the same total weight. Since the items in $\mathcal{D}$ have higher efficiency than those in $\mathcal{A}$, when obtaining $\mathbf{z}$ from $\mathbf{g}$, why bother to delete $\mathcal{D}$ and add $\mathcal{A}$?

Case (i). $\mathbf{z}(\mathcal{I}_2)$ has at least $c_A w_{\max}^{1/2} \log w_{\max}$ distinct weights. Note that $\overline{\mathbf{z}}(\mathcal{I}_1^{\overline{W^*}})$ is exactly the set of items in $\mathcal{I}_1^{\overline{W^*}}$ that are deleted when obtaining $\mathbf{z}$ and the total weight of $\overline{\mathbf{z}}(\mathcal{I}_1^{\overline{W^*}})$ is $\Delta(\mathcal{I}_1^{\overline{W^*}}) > c_B w_{\max}^{3/2}$. The weight multisets of $\mathbf{z}(\mathcal{I}_2)$ and $\overline{\mathbf{z}}(\mathcal{I}_1^{\overline{W^*}})$ satisfy the condition of Lemma 3.1, which implies that there is a non-empty subset $\mathcal{Z}$ of $\mathbf{z}(\mathcal{I}_2)$ and a non-empty subset $\mathcal{D}$ of $\overline{\mathbf{z}}(\mathcal{I}_1^{\overline{W^*}})$ such that $\mathcal{Z}$ and $\mathcal{D}$ have the same total weight. Let $\hat{\mathbf{z}} = (\mathbf{z} \setminus \mathcal{Z}) \cup \mathcal{D}$ be the solution obtained from $\mathbf{z}$ by deleting $\mathcal{Z}$ and adding $\mathcal{D}$ back. Clearly $\hat{\mathbf{z}}$ is feasible. Note $\mathcal{D}$ is to the left of $\mathcal{Z}$ (see Figure 1 for an illustration). According to the way we label items and break ties, we have that $\mathcal{D}$ has either

strictly higher average efficiency than $\mathcal{Z}$, or — if the efficiency is the same — strictly smaller average weight than $\mathcal{Z}$. In the former case, $\hat{\mathbf{z}}$ has a larger profit than $\mathbf{z}$. In the latter case, $|\mathcal{D}| > |\mathcal{Z}|$. Since $\mathbf{g}$ selects all items in $\mathcal{D}$ and $\mathcal{Z}$, it follows that $\hat{\mathbf{z}}$ is an optimal solution with $\|\hat{\mathbf{z}} - \mathbf{g}\|_1 < \|\mathbf{z} - \mathbf{g}\|_1$. Both cases contradict our choice of $\mathbf{z}$.

Case (ii). $\overline{\mathbf{z}}(\mathcal{I}_2)$ has at least $c_A w_{\max}^{1/2} \log w_{\max}$ distinct weights. When obtaining $\mathbf{z}$ from $\mathbf{g}$, the total weight of items that are added is exactly $\Delta(\mathcal{I}_{\geqslant b})$, while the total weight of items that are deleted is at least $\Delta(\mathcal{I}_1^{\overline{W^*}})$. Since the total weight of $\mathbf{z}$ and $\mathbf{g}$ differ by at most $w_{\max}$ (they are both maximal solutions), we have the following.

$$\Delta(\mathcal{I}_{\geqslant b}) \geqslant \Delta(\mathcal{I}_1^{\overline{W^*}}) - w_{\max} \geqslant (2c_B - 1)w_{\max}^{3/2} \geqslant c_B w_{\max}^{3/2}.$$

Also note that $\Delta(\mathcal{I}_{\geqslant b})$ is exactly the total weight of items in $z(\mathcal{I}_{\geqslant b})$. Again, the weight multisets of $\overline{\mathbf{z}}(\mathcal{I}_2)$ and $\mathbf{z}(\mathcal{I}_{\geqslant b})$ satisfies the conditions of Lemma 3.1. By Lemma 3.1, there is a non-empty subset $\mathcal{D}$ of $\overline{\mathbf{z}}(\mathcal{I}_2)$ and a non-empty subset $\mathcal{Z}$ of $\mathbf{z}(\mathcal{I}_{\geqslant b})$ such that $\mathcal{D}$ and $\mathcal{Z}$ have the same total weight. Moreover, $\mathcal{D}$ is to the left of $\mathcal{Z}$. By an argument similar to that in case (i), we can obtain a solution $\hat{\mathbf{z}} = (\mathbf{z} \setminus \mathcal{Z}) \cup \mathcal{D}$ that is better than $\mathbf{z}$, which contradicts with our choice of $\mathbf{z}$. This completes the proof. □

4.2 Algorithmic Part

LEMMA 4.2. *In $\widetilde{O}(n)$ time, we can compute the partition $(W^*, \overline{W^*})$ of W in Lemma 4.1.*

It is easy to see that the process of identifying W^* in the proof of Lemma 4.1 can be generalized to Bounded Knapsack and can be done in $\widetilde{O}(n)$ time.

The following theorem directly follows by Lemma 4.2 and Lemma 2.2.

THEOREM 4.1. *There is an $\widetilde{O}(n + w_{\max}^{5/2})$-time algorithm for Bounded Knapsack.*

5 An $\widetilde{O}(n + w^{12/5})$-time Algorithm for Knapsack

In the structural part of the $\widetilde{O}(n + w_{\max}^{5/2})$-time algorithm, in order to make the weight multiset of $\mathbf{z}(\mathcal{I}_2)$ (or $\overline{\mathbf{z}}(\mathcal{I}_2)$) dense, we require that items in it have $\Theta(w_{\max}^{1/2} \log w_{\max})$ distinct weights. The multiplicities of these weights are not considered. In this section, we develop a better proximity result by exploiting the multiplicities of weights, and therefore obtain an algorithm with improved running time. As before, this section is divided into the structural part and the algorithmic part.

5.1 Structural Part - Fine-Grained Proximity
As in Subsection 4.1, it suffices to consider 0-1 Knapsack, and the notations defined at the beginning of that subsection will be used.

LEMMA 5.1. *There exists a partition of W into W^* and $\overline{W^*}$ such that*

(i) $|W^*| \leqslant 4c_A w_{\max}^{3/5} \log w_{\max}$,

(ii) $\Delta(\mathcal{I}^{\overline{W^*}}) \leqslant 4c_B w_{\max}^{7/5}$,

where c_A and c_B are two large constants used in Lemma 3.1.

Lemma 5.1 does not exploit the multiplicities of weights, and can be proved in exactly the same way as Lemma 4.1. The trade-off between $|W^*|$ and $\Delta(\mathcal{I}^{\overline{W^*}})$ essentially results from Lemma 3.1. When $|W^*|$ is $\widetilde{\Theta}(w_{\max}^p)$ for some $p \geqslant 1/2$, $\Delta(\mathcal{I}^{\overline{W^*}})$ is $O(w_{\max}^{2-p})$.

$\overline{W^*}$ already meets the requirement since $|\overline{W^*}|\Delta(\mathcal{I}^{\overline{W^*}}) \leqslant c_B w_{\max}^{12/5}$. W^*, however, may be problematic since $\Delta(\mathcal{I}^{W^*})$ can be as large as $O(w_{\max}^2)$. We further partition W^* into two sets and prove a similar result as before. But this time, we will take advantage of the multiplicities of weights and use Lemma 3.2 instead of Lemma 3.1.

LEMMA 5.2. *There exits a partition of W^* into W^+ and $\overline{W^+} = W^* \setminus W^+$ such that*

(i) $|W^+| \leqslant 4c_A w_{\max}^{2/5} \log^2 w_{\max}$,

(ii) $\Delta(\mathcal{I}^{\overline{W^+}}) \leqslant 8c_B w_{\max}^{9/5}$,

where c_A and c_B are two large constants used in Lemma 3.2.

5.2 Algorithmic Part The construction of W^* and W^+ can be easily generalized to Bounded Knapsack and can be done in $\widetilde{O}(n)$ time. So we have:

LEMMA 5.3. *In $\widetilde{O}(n)$ time, we can compute the partition $(W^+, \overline{W^+}, \overline{W^*})$ of W such that*

(i) $|W^+| \leqslant 4c_A w_{\max}^{2/5} \log^2 w_{\max}$,

(ii) $|\overline{W^+}| \leqslant 4c_A w_{\max}^{3/5} \log w_{\max}$ *and* $\Delta(\mathcal{I}^{\overline{W^+}}) \leqslant 8c_B w_{\max}^{9/5}$,

(iii) $\Delta(\mathcal{I}^{\overline{W^*}}) \leqslant 4c_B w_{\max}^{7/5}$.

It is easy to verify that

$$|W^+|\Delta(\mathcal{I}^{W+}) + |\overline{W^+}|\Delta(\mathcal{I}^{\overline{W+}}) + |\overline{W^*}|\Delta(\mathcal{I}^{\overline{W*}}) \leqslant \widetilde{O}(w_{\max}^{12/5}).$$

Therefore, Theorem 1.1 follows by Lemma 5.3 and 2.2.

6 Algorithms for Bounded Subset Sum

We consider Bounded Subset Sum. We have the following observation by Lemma 2.1.

OBSERVATION 6.1. *Without loss of generality, we may assume $u_i \leqslant 4w_{\max}$ for all $1 \leqslant i \leqslant n$.*

6.1 An $\widetilde{O}(nw_{\max})$-time Algorithm In this subsection, we present an $\widetilde{O}(nw_{\max})$-time algorithm. It builds upon two ingredients: (i). a faster algorithm for computing the sumset of sets with a special structure, and (ii). a layering technique that transforms the input into a special structure.

6.1.1 Faster sumset algorithm for sets with a special structure We first introduce some notations for integer sets. Given an integer (multi-)set X and a number a, we let $X - a := \{x - a : x \in X\}$. We let $\mathtt{m}_X$ denote the largest number in X. We let $rX := \{rx : x \in X\}$ for any real number r. Given two integer multisets A, B, we let $A + B := \{a + b : a \in A \cup \{0\}, b \in B \cup \{0\}\}$. It is easy to verify that $r(A + B) = rA + rB$.

It is known that the sumset $A + B$ can be computed via Fast Fourier Transform (FFT) in $\widetilde{O}(\mathtt{m}_A + \mathtt{m}_B)$ time. More generally, the following is true:

LEMMA 6.1. [10] *Given sets $S_1, S_2, \cdots, S_\ell \subset \mathbb{N}$, we can compute $S_1 + S_2 + \cdots + S_\ell$ in $O(\sigma \log \sigma \log \ell)$ time, where $\sigma = \mathtt{m}_{S_0} + \mathtt{m}_{S_1} + \cdots + \mathtt{m}_{S_\ell}$.*

For our purpose, given a target value t, we want to decide whether $t \in A + kB$ quickly for some positive integer k. Using FFT, it takes $O(\mathtt{m}_A + k\mathtt{m}_B)$ time, which is too much when k is large. We show that $O(\mathtt{m}_A + \mathtt{m}_B)$ time is sufficient, and the result can be further generalized by the following lemma.

LEMMA 6.2. *Given sets $S_0, S_1, \cdots, S_\ell \subset \mathbb{N}$, an integer $k \in \mathbb{Z}_{>0}$ and a target number t, we can determine whether $t \in S_0 + kS_1 + \cdots + k^\ell S_\ell$ in $O(\sigma(\ell + 1)\log \sigma)$ time, where $\sigma = \mathtt{m}_{S_0} + \mathtt{m}_{S_1} + \cdots + \mathtt{m}_{S_\ell}$.*

Proof. We prove by induction. The lemma is obviously true for $\ell = 0$. Suppose it is true for $\ell = h - 1$. That is, we can determine whether $t \in S_0 + kS_1 + \cdots + k^{h-1}S_{h-1}$ in $c_1(\sum_{i=0}^{h-1} \mathtt{m}_{S_i})(h - 1)\log(\sum_{i=0}^{h-1} \mathtt{m}_{S_i})$ time for some sufficiently large constant c_1. Now we consider the case $\ell = h$.

Note that $t \in S_0 + kS_1 + \cdots + k^h S_h$ if and only if $t = t_0 + t'$ where $t_0 \in S_0$ and $t' \in kS_1 + \cdots + k^h S_h$. It follows that t' is a multiple of k. Therefore, $t \equiv t_0 \pmod{k}$. Let $r_0 \in [0, k - 1]$ be the residue of t modulo k, $\bar{S}_0 \subseteq S_0$ be the subset of integers in S_0 whose residue is r_0. It is clear that every element in $\bar{S}_0 - r_0$ is a multiple of k, so $\frac{1}{k}(\bar{S}_0 - r_0)$ is an integer set. Moreover, it follows that $t \in S_0 + kS_1 + \cdots + k^h S_h$ if and only if $t \in \bar{S}_0 + kS_1 + \cdots + k^h S_h$, or equivalently, if and only if

$$\frac{t - r_0}{k} \in \frac{1}{k}\left((\bar{S}_0 - r_0) + kS_1 + \cdots + k^h S_h\right) = \frac{1}{k}(\bar{S}_0 - r_0) + S_1 + kS_2 + \cdots + k^{h-1}S_h.$$

Now we compute $\hat{S}_1 := \frac{1}{k}(\bar{S}_0 - r_0) + S_1$ via FFT, which takes $c_2\sigma_2 \log \sigma_2$ time for some constant c_2 and $\sigma_2 \leqslant \mathtt{m}_{S_0}/k + \mathtt{m}_{S_1}$. It thus remains to determine whether

$$\frac{t - r_0}{k} \in \hat{S}_1 + kS_2 + \cdots + k^{h-1}S_h.$$

By the induction hypothesis, this takes $c_1(h-1)\sigma' \log \sigma'$ time where $\sigma' = \mathtt{m}_{\hat{S}_1} + \mathtt{m}_{S_2} + \cdots + \mathtt{m}_{S_h} \leqslant \mathtt{m}_{S_0}/k + \mathtt{m}_{S_1} + \cdots + \mathtt{m}_{S_h}$. Therefore, it is easy to verify that the overall running time is bounded by $c_2\sigma_2 \log \sigma_2 + c_1(h-1)\sigma' \log \sigma' \leqslant c_1 h\sigma \log \sigma$ when $c_1 \geqslant c_2$. □

6.1.2 Item Grouping We follow a standard idea to bundle item copies into groups. A solution to the grouped instance will take item copies within a group as a whole. Thus, the key point is to show that, after grouping, we do not lose any solution, and thus the optimal solution.

Recall that u_j refers to the copy number of item j. It is obvious that u_j can be represented as $u_j[0] + u_j[1] \cdot 2^1 + \cdots + u_j[l_j] \cdot 2^{l_j}$ where $2 \leqslant u_j[i] < 4$ for all $0 \leqslant i \leqslant \ell_j - 1$ and $u_j[\ell_j] < 4$. This can be easily achieved recursively: let r be the residue of u_j modulo 2; set $u_j[0] = 2 + r$, $u_j \leftarrow \frac{u_j - (2+r)}{2}$ and repeat the above procedure.

As $u_j \leqslant 4w_{\max}$ for all $1 \leqslant j \leqslant n$, $\ell_j \leqslant 2 + \log w_{\max}$. For ease of discussion, we let $u_j[i] = 0$ for $i > \ell_j$ and $\ell = \max_j \ell_j$. Now we define X_i as a multiset that consists of exactly $u_j[i]$ copies of weight w_j for every j, and note that by $2^i X_i$ we mean the multiset that consists of exact $u_j[i]$ copies of weight $2^i w_j$, which represents a multiset of groups (where each group contains 2^i copies of weight w_j but has to be selected as a whole).

Let X denote the multiset where every w_j occurs u_j times. Recall that $\mathcal{S}(X)$ denotes the set of all possible subset-sums of multiset X. We show the following in the appendix.

LEMMA 6.3.

$$\mathcal{S}(X) = \sum_{i=0}^{\ell} 2^i \mathcal{S}(X_i) = 2^0 \mathcal{S}(X_0) + \cdots + 2^\ell \mathcal{S}(X_\ell).$$

That is, for any $t \in \mathcal{S}(X)$, there exist $t_i \in \mathcal{S}(X_i)$ such that $t = \sum_{i=0}^{\ell} 2^i t_i$.

Polak et al. [26] derived a similar lemma for $\ell = 2$.

6.1.3 Finalizing the $\widetilde{O}(nw_{\max})$-time algorithm Now we are ready to present our $\widetilde{O}(nw_{\max})$-time algorithm. By Observation 6.1, we have $u_j \leqslant 4w_{\max}$. Consequently, for $\ell = O(\log w_{\max})$, the given instance can be grouped as $\cup_{i=0}^{\ell} 2^i X_i$. Note that X_i contains at most $4n$ elements as there are n item weights, and each item has $u_j[i] < 4$ copies. Our goal is to determine whether $t \in \mathcal{S}(\cup_{i=0}^{\ell} 2^i X_i)$ for an arbitrary target value t. By Lemma 6.3, it suffices to determine whether $t \in \sum_{i=0}^{\ell} 2^i \mathcal{S}(X_i)$. We first compute each $\mathcal{S}(X_i)$. This is equivalent to computing sumset $\sum_{e \in X} \{0, e\}$. According to Lemma 6.1, this can be achieved in $\widetilde{O}(\sigma_1)$ time, where $\sigma_1 \leqslant 4nw_{\max}$. Next, we apply Lemma 6.2 to determine whether $t \in \sum_{i=0}^{\ell} 2^i \mathcal{S}(X_i)$, which takes $\widetilde{O}(\sigma\ell)$ time, where $\sigma = \sum_{i=0}^{\ell} \mathtt{m}_{\mathcal{S}(X_i)}$, where $\mathtt{m}_{\mathcal{S}(X_i)}$ refers to the largest integer in $\mathcal{S}(X_i)$, which is bounded by $4nw_{\max}$. Consequently, the overall running time is $\widetilde{O}(\ell^2 nw_{\max}) = \widetilde{O}(nw_{\max})$.

REMARK 6.1. *We do not require that in the input instance, item weights are distinct. If, however, there are items of the same weight, we can simply merge them and update u_j's. After preprocessing, we apply our algorithm. Hence, the running time of our algorithm can also be bounded by $\widetilde{O}(n + |W|w_{\max})$ where W stands for the set of distinct weights.*

6.2 An $\widetilde{O}(n + w_{\max}^{3/2})$-time Algorithm for Subset Sum We present an alternative algorithm for Bounded Subset Sum of running time $\widetilde{O}(n + w_{\max}^{3/2})$. We start with the following observation. Recall that W is the set of all distinct weights.

OBSERVATION 6.2. *We may assume without loss of generality that $|W| \geqslant 4c_A w_{\max}^{1/2} \log w_{\max}$, and $2c_A w_{\max}^{3/2} \log w_{\max} \leqslant t \leqslant \Sigma_{\mathcal{I}}/2$.*

Proof. If $|W| \leqslant 4c_A w_{\max}^{1/2} \log w_{\max}$, our $\widetilde{O}(n + |W| w_{\max})$ algorithm in the previous subsection already runs in $\widetilde{O}(n + w_{\max}^{3/2})$ time. It remains to consider the case when $|W| \geqslant 4c_A w_{\max}^{1/2} \log w_{\max}$.

If $t \geqslant \Sigma_{\mathcal{I}}/2$, we let $t' = \Sigma_{\mathcal{I}} - t$. It is straightforward that the instance $(\mathcal{I}, t)$ is equivalent to the instance $(\mathcal{I}, t')$ and that $t' \leqslant \Sigma_{\mathcal{I}}/2$. If $t < 2c_A w_{\max}^{3/2} \log w_{\max}$, the $\widetilde{O}(n + t)$-time algorithm already runs in $\widetilde{O}(n + w_{\max}^{3/2})$ time. $\quad\square$

We will use our technique developed for Knapsack to deal with Subset Sum. Like Knapsack, this whole subsection is divided into two parts: the structural part and the algorithm part. In the structural part, we will conceptually take the given instance as a special Knapsack instance, and transform it into a 0-1 Subset Sum instance. This allows us to carry over the proximity results from Knapsack. In the algorithm part, we take the original instance and show that it is possible to leverage the proximity results to design our algorithm without involving the instance transformation.

6.2.1 The structural part In this part, we conceptually view Bounded Subset Sum as a 0-1 Knapsack. Recall that our Knapsack algorithm starts with a greedy solution. For Subset Sum, every item has the same efficiency, which means we may construct a greedy solution with respect to any order of items. For technical reasons, however, we need to start with a special greedy solution $\mathbf{g}$. Towards this, we will first define item sets $\mathcal{I}_2$ and $\mathcal{I}_3$, use them to define $\mathbf{g}$, and then use $\mathbf{g}$ to further define item sets $\mathcal{I}_1$ and $\mathcal{I}_4$. These four sets will be treated in a similar way as we treat $\mathcal{I}_1$ to $\mathcal{I}_4$ in Knapsack.

Let W be the set of all item weights. Let

- W_2 be the set of the largest $2c_A w_{\max}^{1/2} \log w_{\max}$ weights in W;

- W_3 be the set of the smallest $2c_A w_{\max}^{1/2} \log w_{\max}$ weights in W.

Clearly, $W_2 \cap W_3 = \emptyset$. For each w in W_2, we put an arbitrary item of weight w into $\mathcal{I}_2$. For each w in W_3, we put an arbitrary item of weight w into $\mathcal{I}_3$. The following statements hold.

(i) the total weight of items in $\mathcal{I}_2$ is at most $2c_A w_{\max}^{3/2} \log w_{\max} \leqslant t$.

(ii) the total weight of items not in $\mathcal{I}_3$ is at least $\Sigma_{\mathcal{I}}/2 \geqslant t$.

Let $\mathbf{g}$ be an arbitrary maximal solution such that $\mathbf{g}$ selects every item in $\mathcal{I}_2$ but no item in $\mathcal{I}_3$. In the algorithm part, we shall construct such a solution $\mathbf{g}$.

Assuming $\mathbf{g}$, let $\mathcal{I}_1 = \mathbf{g}(\mathcal{I}) \setminus \mathcal{I}_2$, and let $\mathcal{I}_4 = \overline{\mathbf{g}}(\mathcal{I}) \setminus \mathcal{I}_3$. We relabel the items so that for any $i < j$, every item in $\mathcal{I}_i$ has smaller index than any item in $\mathcal{I}_j$.

Recall that Subset Sum is a decision problem. Since we treat it as a special Knapsack problem, we define its optimal solution as the solution that returns a subset-sum $t^* \leqslant t$ and is closest to t. In particular, if the answer to Subset Sum is "yes", then $t^* = t$. Let $\mathbf{z}$ be the optimal solution with the largest lexicographical order. That is, $\mathbf{z}$ always prefers items with smaller indices. We have the following fine-grained proximity between $\mathbf{g}$ and $\mathbf{z}$.

LEMMA 6.4. $\Delta(\mathcal{I}) = \sum_{i \in \mathcal{I}} w_i |g_i - z_i| \leqslant (4c_A + 2c_B) w_{\max}^{3/2} \log w_{\max}$.

Proof. Note that $\Delta(\mathcal{I}) = \sum_{i=1}^{4} \Delta(\mathcal{I}_i)$. It is obvious that $\Delta(\mathcal{I}_2) + \Delta(\mathcal{I}_3)$ is at most $4c_A w_{\max}^{3/2} \log w_{\max}$ since the total weight of items in $\mathcal{I}_2 \cup \mathcal{I}_3$ is bounded by this amount. Next we show that $\Delta(\mathcal{I}_1) \leqslant c_B w_{\max}^{3/2}$. $\Delta(\mathcal{I}_4)$ can be proved similarly due to symmetry.

Suppose, for the sake of contradiction, that $\Delta(\mathcal{I}_1) > c_B w_{\max}^{3/2}$. As in the proof of Lemma 4.1, we can show that either there is a subset $\mathcal{D}$ of $\overline{\mathbf{z}}(\mathcal{I}_1)$ and a subset $\mathcal{Z}$ of $\mathbf{z}(\mathcal{I}_2)$ such that $\mathcal{D}$ and $\mathcal{Z}$ have the same total weight, or there is a subset $\mathcal{D}$ of $\overline{\mathbf{z}}(\mathcal{I}_2)$ and a subset $\mathcal{A}$ of $\mathbf{z}(\mathcal{I}_3 \cup \mathcal{I}_4)$ such that $\mathcal{D}$ and $\mathcal{A}$ have the same total weight. Note that in Subset Sum, every item has the same efficiency of 1. In the former case, $\hat{\mathbf{z}} = (\mathbf{z} \setminus \mathcal{Z}) \cup \mathcal{D}$ is an optimal solution whose lexicographical order is larger than $\mathbf{z}$. In the latter case, $\hat{\mathbf{z}} = (\mathbf{z} \setminus \mathcal{A}) \cup \mathcal{D}$ is an optimal solution whose lexicographical order is larger than $\mathbf{z}$. Both contradict our choice of $\mathbf{z}$. $\quad\square$

6.2.2 The algorithmic part Note that the input consists of n pairs (w_i, u_i), meaning that item i has a weight w_i and has u_i copies. Despite that $\mathbf{g}$ is defined on the transformed 0-1 instance, as we can get and sort W in $O(n \log n)$, constructing it can be done efficiently.

LEMMA 6.5. $\mathbf{g}$ *can be computed in* $\widetilde{O}(n)$ *time.*

LEMMA 6.6. *Assuming* $\mathbf{g}$, *with* $1 - o(1)$ *probability we can determine whether there exists* $\mathbf{z} \in \mathbb{Z}^n$, $0 \leqslant z_i \leqslant u_i$ *such that* $\sum_{i \in \mathcal{I}} w_i z_i = t$ *in* $\widetilde{O}(w_{\max}^{3/2})$ *time.*

Proof. From $\mathbf{g}$ we define two sets as follows: $G^- = \{(w_i, u_i^-) : u_i^- = g_i\}$, $G^+ = \{(w_i, u_i^+) : u_i^+ = u_i - g_i\}$. Intuitively, when changing $\mathbf{g}$ to $\mathbf{z}$, G^- and G^+ represent copies that may need to be deleted and added, respectively. By Lemma 6.4, the total weight of item copies to be deleted in G^- is at most $\widetilde{O}(w_{\max}^{3/2})$, and the total weight of copies to be added from G^+ is also at most $\widetilde{O}(w_{\max}^{3/2})$.

Let $t' = (4c_A + 2c_B) w_{\max}^{3/2} \log w_{\max}$ and $t_0 = \sum_{i \in \mathcal{I}} w_i g_i$. Define $\mathcal{S}(G^+, t') = \{y \leqslant t' : y = \sum_{i=1}^n w_i x_i, 0 \leqslant x_i \leqslant u_i^+\}$ and $\mathcal{S}(G^-, t') = \{y \leqslant t' : y = \sum_{i=1}^n w_i x_i, 0 \leqslant x_i \leqslant u_i^-\}$.

We first compute $\mathcal{S}(G^+, t')$ and $\mathcal{S}(G^-, t')$ using Bringmann's algorithm [6]. Note that it is a randomized algorithm. We will compute $S^+ \subseteq \mathcal{S}(G^+, t')$ and $S^- \subseteq \mathcal{S}(G^-, t')$ in $\widetilde{O}(t') = \widetilde{O}(w_{\max}^{3/2})$ time such that every number in $\mathcal{S}(G^-, t')$ and $\mathcal{S}(G^+, t')$ is contained in S^- and S^+ with probability $1 - (n + t')^{-\Omega(1)}$, respectively. Therefore, with probability $1 - o(1)$ we get $\mathcal{S}(G^-, t')$ and $\mathcal{S}(G^+, t')$. Next, we compute $\mathcal{S}(G^+, t') - \mathcal{S}(G^-, t')$ via FFT in $\widetilde{O}(t') = \widetilde{O}(w_{\max}^{3/2})$ time. Finally, we test whether $t \in t_0 + \mathcal{S}(G^+, t') - \mathcal{S}(G^-, t')$, which takes $\widetilde{O}(t') = \widetilde{O}(w_{\max}^{3/2})$ time. Thus, Lemma 6.6 is proved. $\square$

Remark

We note that very recently, Bringmann [7] and Jin [19] independently improved the running time to $\widetilde{O}(n + w_{\max}^2)$ by generalizing our technique.

A Proof of Lemma 2.2

LEMMA 2.2. *Let* $\mathbf{g}$ *be a maximal prefix solution to Bouned Knapsack. Let* $W_1 \cup \cdots \cup W_k$ *be a partition of the set* W. *For* $j \in \{1, ..., k\}$, *let* U_j *be an upper bound for* $\Delta(\mathcal{I}^{W_j})$. *Then Bounded Knapsack can be solved in* $\widetilde{O}(n + k \sum_{j=1}^{k} |W_j| \cdot U_j)$ *time.*

Proof. Without loss of generality, assume that $U_1 \leqslant ... \leqslant U_k$. We construct an optimal solution $\mathbf{x}$ from $\mathbf{g}$, which can be interpreted as $\mathbf{x} = \mathbf{g} - \mathbf{x}^- + \mathbf{x}^+$. Here $\mathbf{x}^-$ stands for the copies in $\mathbf{g}$ but not in $\mathbf{x}$, which must be copies in $\mathcal{I}_{<b}$. Likewise, $\mathbf{x}^+$ stands for copies in $\mathbf{x}$ but not in $\mathbf{g}$, which must be copies in $\mathcal{I}_{\geqslant b}$. It is obvious that $\Delta(\mathcal{I}_{<b}) = \sum_{j=1}^{k} \Delta(\mathcal{I}_{<b}^{W_j}) \leqslant k \cdot U_k$ and $\Delta(\mathcal{I}_{\geqslant b}) \leqslant k \cdot U_k$.

From now on we shall use the bold symbol $\boldsymbol{x}$ to represent a sequence. We shall compute 3 sequences:

- $\boldsymbol{x}^- = \langle x_0^-, ..., x_{k \cdot U_k}^- \rangle$, where $x_{t'}^-$ is the minimum total profit of copies in $\mathcal{I}_{<b}$ whose total weight is exactly t';

- $\boldsymbol{x}^+ = \langle x_0^+, ..., x_{k \cdot U_k}^+ \rangle$, where $x_{t'}^+$ is the maximum total profit of copies in $\mathcal{I}_{\geqslant b}$ whose total weight is exactly t'.

- $\boldsymbol{x}_{\leqslant}^+ = \langle x_{\leqslant 0}^+, ..., x_{\leqslant k \cdot U_k}^+ \rangle$, where $x_{\leqslant t'}^+$ means the maximum value we can obtain by choosing copies from $\mathcal{I}_{\geqslant b}$ whose total weight does not exceed t'. To simply the writing up, we let $x_{\leqslant k \cdot U_k + h}^+ = x_{\leqslant k \cdot U_k}^+$ for all $1 \leqslant h \leqslant \Delta$, where $\Delta = t - \sum_{i \in \mathcal{I}} w_i g_i$.

Suppose we have computed the above three sequences, then Knapsack can be solved as follows: For every $t' \in \{0, ..., k \cdot U_k\}$, we compute $y_{t'} = \sum_{i \in \mathcal{I}} v_i g_i - x_{t'}^- + x_{\leqslant (t' + \Delta)}^+$, and select $\max_{t'} y_{t'}$.

It remains to compute the three sequences. We show that, $\boldsymbol{x}^+ = \langle x_0^+, ..., x_{k \cdot U_k}^+ \rangle$ can be computed in $\widetilde{O}(n + k \sum_{j=1}^{k} |W_j| U_j)$ time. The computation of $\boldsymbol{x}^-$ is similar. Moreover, once we have computed $\boldsymbol{x}^+$, $\boldsymbol{x}_{\leqslant}^+$ can be computed in a straightforward way using that $x_{\leqslant t'}^+ = \max_{1 \leqslant h \leqslant t'} x_h^+$.

Now we describe our algorithm for computing $\boldsymbol{x}^+ = \langle x_0^+, ..., x_{k \cdot U_k}^+ \rangle$. Towards this, we use the fast structured $(\min, +)$-convolution as a basic operation. The $(\max, +)$-convolution $\boldsymbol{a} \oplus \boldsymbol{b}$ between two sequence $\boldsymbol{a}$ and $\boldsymbol{b}$ is a sequence $c_0, ..., c_{n+m}$ such that for any i,

$$c_i = \max_{0 \leqslant j \leqslant i} (a_j + b_{i-j}).$$

Let $\boldsymbol{s}^w = \langle s_0^w, ..., s_{U_j}^w \rangle$, where $s_{t'}^w$ denotes the optimal objective value of copies in $\mathcal{I}_{\geqslant b}^w$ whose total weight is exactly t'. Note that copies in $\mathcal{I}_{\geqslant b}^w$ have the same weight w. Hence, $s_{t'}^w$ equals the total profit of the i most valuable copies in $\mathcal{I}_{\geqslant b}^w$ when $t' = i \cdot w$ for some i, or equals $-\infty$ otherwise.

Given $\boldsymbol{s}^w$, $\boldsymbol{x}^+$ can be computed by iteratively convolving $\boldsymbol{s}^w$, see Algorithm 1. Due to the special structure of $\boldsymbol{s}^w$, a single convolution step $\boldsymbol{x}^+ \oplus \boldsymbol{s}^w$ can be performed in linear time by SMAWK algorithm [2] (also see Lemma 2.2 in [26]). As $\Delta(\mathcal{I}_{\geqslant b}^{W_1 \cup ... \cup W_{k'}}) \leqslant \sum_{j=1}^{k'} U_j \leqslant k' \cdot U_{k'}$, we can truncate $\boldsymbol{x}^+$ after the $k' \cdot U_{k'}$-th entry when we compute the weights in $W_{k'}$. We can compute $\boldsymbol{x}^+$ in time: $\widetilde{O}(n + \sum_{j=1}^{k} j \cdot |W_j| \cdot U_j) \leqslant \widetilde{O}(n + k \sum_{j=1}^{k} |W_j| U_j)$. □

Algorithm 1 The Algorithm to Compute $\boldsymbol{x}^+$

1: sort $W_1, ..., W_k$ that $U_1 \leqslant ... \leqslant U_k$
2: $\boldsymbol{x}^+ :=$ empty sequence
3: **for** $j = 1, ..., k$ **do**
4:   **for** $w \in W_j$ **do**
5:     $\boldsymbol{s}^w :=$ the sequence of maximum value get from $\mathcal{I}_{\geqslant b}^w$
6:     $\boldsymbol{x}^+ := \boldsymbol{x}^+ \oplus \boldsymbol{s}^w$ ▷ using SMAWK algorithm
7:     Truncate $\boldsymbol{x}^+$ after the $j \cdot U_j$-th entry
8: **return** $\boldsymbol{x}^+$

B Proof of Lemma 3.2

LEMMA 3.2. *Let* w *be a positive integer. Let* A *and* B *be two multisets of integers from* $\{1, ..., w\}$ *such that*

(i) *at least $c_A w^{2/5} \log^2 w$ distinct integers in A have multiplicity of at least $w^{1/5}$,*

(ii) $\Sigma_B \geqslant c_B w^{8/5}$,

where c_A and c_B are two sufficiently large constants. Then there must exist a non-empty subset A' of A and a non-empty subset B' of B such that $\Sigma_{A'} = \Sigma_{B'}$.

Proof. Let S a maximal subset of A such that each integer has multiplicity $w^{1/5}$ in S. Note that $|\mathtt{supp}_S| \geqslant c_A w^{2/5} \log^2 w$. What we actually prove is that there is a non-empty subset S' of S such that some subset B' of B have the same total weight as S. We characterize the set of integers that can be hit by S. Then we show that B can hit at least one of these integers.

Since the items in S has the same multiplicity, we obtain the following two equalities that will be frequently used in the proof.

$$\mu_S \Sigma_{\mathtt{supp}_S} = \Sigma_S,$$
$$\mu_S |\mathtt{supp}_S| = |S|.$$

Note that $\mathtt{m}_S \leqslant w$, $\mu_S = w^{1/5}$, $|\mathtt{supp}_S| = |\mathtt{supp}_A| \geqslant c_A w^{2/5} \log^2 w$. Since c_A is sufficiently large, we have that

$$|S|^2 = \mu_S^2 |\mathtt{supp}_S|^2 \geqslant \mu_S^2 \cdot c_A^2 w^{4/5} \log^4 w \geqslant c_A^2 \log^4 w \cdot \mathtt{m}_S \mu_S.$$

The last inequality is due to $\mu_A \geqslant w^{1/5}$. Since $|\mathtt{supp}_A| \leqslant w$ and $\mu_A \leqslant w$,

$$\begin{aligned}
c_\delta &= \Theta\left(\log(2|S|) \log^2(2\mu_S)\right) \\
&= \Theta\left(\log(\mu_S |\mathtt{supp}_S|) \log^2(\mu_S)\right) \\
&= \Theta\left(\log(w^2) \log^2(w)\right) \\
&= \Theta(\log^3 w).
\end{aligned}$$

When c_A is sufficiently large, $c_S^2 \log^4 w \geqslant c_\delta$, so S is c_δ-dense.

By Theorem 2.2, there exists an integer d such that $S' := S(d)/d$ is c_δ-dense and has no c_α-almost divisor. And the followings hold.

(i) $d \leqslant 4\mu_S \Sigma_S / |S|^2$.

(ii) $|S'| \geqslant 0.75|S|$.

(iii) $\Sigma_{S'} \geqslant 0.75 \Sigma_S / d$.

Note that $\mu_{S'} \leqslant \mu_S$, $\mathtt{m}_{S'} \leqslant w/d$, and $\Sigma_{S'} \leqslant \Sigma_S/d$. Applying Theorem 2.1 on S', we get S' can hit any integer in $[\lambda_{S'}, \Sigma_{S'} - \lambda_{S'}]$ where

$$\lambda_{S'} = \frac{c_\lambda \mu_{S'} \mathtt{m}_{S'} \Sigma_{S'}}{|S'|^2} \leqslant \frac{c_\lambda \mu_S w \Sigma_S}{(0.75|S|)^2 d^2} \leqslant \frac{\min\{c_A, c_B\}}{2} \cdot \frac{w}{d^2} \cdot \frac{\mu_S \Sigma_S}{|S|^2}.$$

The last inequality holds since c_A and c_B are sufficiently large constants. We can conclude that S can hit any multiple of d in $[d\lambda_{S'}, d(\Sigma_{S'} - \lambda_{S'})]$. We also have that the left endpoint of this interval

$$d\lambda_{S'} \leqslant \frac{c_B}{2} \cdot \frac{w}{d} \cdot \frac{\mu_S \Sigma_S}{|S|^2} = \frac{c_B}{2} \cdot \frac{w}{d} \cdot \frac{\Sigma_{\mathtt{supp}_S}}{|\mathtt{supp}_S|^2} \leqslant \frac{c_B}{2} \cdot \frac{w^2}{|\mathtt{supp}_S|} \leqslant \frac{c_B}{2} \cdot w^{8/5},$$

and that the length of the interval

$$d(\Sigma_{S'} - 2\lambda_{S'}) \geqslant \frac{3\Sigma_S}{4} - c_A \cdot \frac{w\mu_S\Sigma_S}{d|S|^2}$$

$$= \frac{\mu_S\Sigma_S}{|S|^2}\left(\frac{3|S|^2}{4\mu_S} - c_A \cdot \frac{w}{d}\right) \qquad \text{(since } |S| = \mu_S|\mathsf{supp}_S| \text{ and } d \geqslant 1)$$

$$\geqslant \frac{\mu_S\Sigma_S}{|S|^2}\left(\frac{3|\mathsf{supp}_S|^2\mu_S^2}{4\mu_S} - c_A \cdot w\right) \qquad \text{(since } |\mathsf{supp}_S| \geqslant c_A w^{2/5}\log^2 w \text{ and } \mu_S = w^{1/5})$$

$$\geqslant \frac{\mu_S\Sigma_S}{|S|^2}\left(\frac{3c_A^2 w}{4} - c_A w\right) \qquad \text{(since } c_A \text{ is sufficiently large)}$$

$$\geqslant 4 \cdot \frac{\mu_S\Sigma_S}{|S|^2} \cdot w \qquad \text{(since } d \leqslant 4\mu_S\Sigma_S/|S|^2)$$

$$\geqslant dw.$$

To complete the proof, it suffices to show that there is a subset B' of B whose sum is a multiple of d and is within the interval $[d\lambda_{S'}, d(\Sigma_{S'} - \lambda_{S'})]$. We claim that as long as B has at least d numbers, there must be a non-empty subset of B whose sum is at most dw and is a multiple of d. Assume the claim is true. We can repeatedly extract such subsets from B until B has less than d numbers. Note that the total sum of these subsets is at least

$$\Sigma_B - wd \geqslant c_B w^{8/5} - w \cdot \frac{4\mu_S\Sigma_S}{|S|^2} \geqslant c_B w^{8/5} - \frac{16w\Sigma_{\mathsf{supp}_S}}{|\mathsf{supp}_S|^2} \geqslant c_B w^{8/5} - 16w^{8/5} \geqslant \frac{c_B}{2}w^{8/5}.$$

That is, the total sum of these subset is at least the left endpoint of $[d\lambda_{S'}, d(\Sigma_{S'} - \lambda_{S'})]$. Also note that the sum of each subset is at most dw, which does not exceed the length of the interval. As a result, there must be a collection of subsets whose total sum is within the interval. Since the sum of each subset is a multiple of d, so is the any collection of these subset. □

C Proof of Lemma 5.2

LEMMA 5.2. *There exits a partition of W^* into W^+ and $\overline{W^+} = W^* \setminus W^+$ such that*

(i) $|W^+| \leqslant 4c_A w_{\max}^{2/5}\log^2 w_{\max}$,

(ii) $\Delta(\mathcal{I}^{\overline{W^+}}) \leqslant 8c_B w_{\max}^{9/5}$,

where c_A and c_B are two large constants used in Lemma 3.2.

Proof. Recall that we label the items in decreasing order of efficiency. Without loss of generality, we assume that for any two items $i < j < b$, if i and j have the same efficiency, then $w_i \leqslant w_j$, and that for any two items $b < i < j$, if i and j have the same efficiency, then $w_i \geqslant w_j$. For a set $\mathcal{I}'$ of items, we say a weight w is *frequent* in $\mathcal{I}'$ if $\mathcal{I}'$ contains at least $2w_{\max}^{1/5}$ items with weight w.

Defining W^+ via Partition of $\mathcal{I}^{W^*}$. W^+ will be defined via a partition of $\mathcal{I}^{W^*}$. We partition them into four subsets $(\mathcal{I}_1, \mathcal{I}_2, \mathcal{I}_3, \mathcal{I}_4)$ as follows. For $i < b$, let $\mathcal{I}_{[i,b)}$ be the set of items $\{i, \ldots, b-1\}$ whose weight is in W^*. Let i^* be the minimum index i such that exactly $2c_A w_{\max}^{2/5}\log^2 w_{\max}$ weights are frequent in $\mathcal{I}_{[i,b)}$. Let $\mathcal{I}_2 = \mathcal{I}_{[i^*,b)}$, and let $\mathcal{I}_1 = \mathcal{I}_{<i^*}$. When no such i^* exists, let $\mathcal{I}_2 = \mathcal{I}_{<b}$, and let $\mathcal{I}_1 = \emptyset$. $\mathcal{I}_3$ and $\mathcal{I}_4$ are defined similarly as follows. For any $j \geqslant b$, define $\mathcal{I}_{[b,j]}$ to be the set of items $\{b, \ldots, j\}$ whose weight is in W^*. Let j^* be the maximum index j such that exactly $2c_A w_{\max}^{2/5}\log^2 w_{\max}$ weights are frequent. Let $\mathcal{I}_3 = \mathcal{I}_{[b,j^*]}$, and let $\mathcal{I}_4 = \mathcal{I}_{>j^*}$. When no such j^* exists, let $\mathcal{I}_3 = \mathcal{I}_{\geqslant b}$, and let $\mathcal{I}_4 = \emptyset$.

W^+ is define to be the set of weights that are frequent in $\mathcal{I}_2$ or $\mathcal{I}_3$, and $\overline{W^+} = W^* \setminus W^+$.

Verifying Properties. It is straightforward that $|W^+| \leqslant 4c_A w_{\max}^{2/5}\log^2 w_{\max}$. It is left to show $\Delta(\mathcal{I}^{\overline{W^+}}) \leqslant 8c_B w_{\max}^{9/5}$. We first partition $\mathcal{I}^{\overline{W^+}}$ into $\mathcal{I}_k^{\overline{W^+}} = \mathcal{I}^{\overline{W^+}} \cap \mathcal{I}_k$ for $k \in \{1,2,3,4\}$. Next we show $\Delta(\mathcal{I}_k^{\overline{W^+}}) \leqslant 2c_B w_{\max}^{9/5}$ for $k \in \{1,2\}$.

Consider $\mathcal{I}_2^{\overline{W^+}}$ first. For any $w \in \overline{W^+}$, w is not frequent in $\mathcal{I}_2$. That is, $|\mathcal{I}_2 \cap \mathcal{I}^w| < w_{\max}^{1/5}$. We have

$$\Delta(\mathcal{I}_2^{\overline{W^+}}) \leqslant \sum_{i \in \mathcal{I}_2^{\overline{W^+}}} w_i = \sum_{w \in \overline{W^+}} w|\mathcal{I}_2 \cap \mathcal{I}^w| \leqslant |\overline{W^+}|w_{\max}w_{\max}^{1/5} = w_{\max}^{9/5}.$$

Now consider $\mathcal{I}_1^{\overline{W^+}}$. Suppose, for the sake of contradiction, that $\Delta(\mathcal{I}_1^{\overline{W^+}}) > 2c_B w_{\max}^{9/5} > 2c_B w_{\max}^{8/5}$. That is, we delete a large volume of items in $\mathcal{I}_1^{\overline{W^+}}$. Clearly, $\mathcal{I}_1^{\overline{W^+}}$ is not empty, so there are exactly $2c_A w_{\max}^{2/5} \log^2 w_{\max}$ weights that are frequent in $\mathcal{I}_2$. Recall that, by frequent, we mean $|\mathcal{I}^w \cap \mathcal{I}_2| \geqslant 2w_{\max}^{1/5}$. Then at least one of the following two cases are true.

(i) there are at least $c_A w_{\max}^{2/5} \log^2 w_{\max}$ weights w such that $|\mathcal{I}^w \cap \mathbf{z}(\mathcal{I}_2)| \geqslant w_{\max}^{1/5}$.

(ii) there are at least $c_A w_{\max}^{2/5} \log^2 w_{\max}$ weights w such that $|\mathcal{I}^w \cap \overline{\mathbf{z}}(\mathcal{I}_2)| \geqslant w_{\max}^{1/5}$.

Case (i). $\overline{\mathbf{z}}(\mathcal{I}_1^{\overline{W^+}})$ is a set of deleted items whose total weight is $\Delta(\mathcal{I}_1^{\overline{W^+}}) > 2c_B w_{\max}^{8/5}$. $\mathbf{z}(\mathcal{I}_2)$ is a set of items remaining in $\mathbf{z}$, and there are at least $c_A w_{\max}^{2/5} \log^2 w_{\max}$ weights w such that $|\mathcal{I}^w \cap \mathbf{z}(\mathcal{I}_2)| \geqslant w_{\max}^{1/5}$. One can verify that the weight multisets of $\overline{\mathbf{z}}(\mathcal{I}_1^{\overline{W^+}})$ and $\mathbf{z}(\mathcal{I}_2)$ satisfy the conditions of Lemma 3.2. By Lemma 3.2, there is a subset $\mathcal{D}$ of $\overline{\mathbf{z}}(\mathcal{I}_1^{\overline{W^+}})$ and a subset $\mathcal{Z}$ of $\mathbf{z}(\mathcal{I}_2)$ such that $\mathcal{D}$ and $\mathcal{Z}$ have the same total weight. Then $\hat{\mathbf{z}} = (\mathbf{z} \setminus \mathcal{Z}) \cup \mathcal{D}$ would be a better solution than $\mathbf{z}$. Contradiction.

Case (ii). there are at least $c_A w_{\max}^{2/5} \log^2 w_{\max}$ weights w such that $|\mathcal{I}^w \cap \overline{\mathbf{z}}(\mathcal{I}_2)| \geqslant w_{\max}^{1/5}$. $\Delta(\mathcal{I}_1^{\overline{W^+}}) > 2c_B w_{\max}^{8/5}$ implies that

$$\Delta(\mathcal{I}_{\geqslant b}) \geqslant \Delta(\mathcal{I}_1^{\overline{W^+}}) - w_{\max} \geqslant c_B w_{\max}^{8/5}.$$

Note that the total weight of $\mathbf{z}(\mathcal{I}_{\geqslant b})$ is exactly $\Delta(\mathcal{I}_{\geqslant b})$. One can verify that the weight multisets of $\overline{\mathbf{z}}(\mathcal{I}_2)$ and $\mathbf{z}(\mathcal{I}_{\geqslant b})$ satisfy the condition of Lemma 3.2. By Lemma 3.2, there is a subset $\mathcal{D}$ of $\overline{\mathbf{z}}(\mathcal{I}_2)$ and a subset $\mathcal{A}$ of $\mathbf{z}(\mathcal{I}_{\geqslant b})$ such that $\mathcal{D}$ and $\mathcal{A}$ have the same total weight. Then $\hat{\mathbf{z}} = (\mathbf{z} \setminus \mathcal{A}) \cup \mathcal{D}$ would be a better solution than $\mathbf{z}$. Contradiction.

Due to symmetry, it can be similarly proved that $\Delta(\mathcal{I}_k^{\overline{W^+}}) \leqslant 2c_B w_{\max}^{9/5}$ for $k \in \{3, 4\}$. We provide a full proof for completeness. Consider $\mathcal{I}_3^{\overline{W^+}}$. For any $w \in \overline{W^+}$, w is not frequent in $\mathcal{I}_2$. That is, $|\mathcal{I}_3 \cap \mathcal{I}^w| < w_{\max}^{1/5}$.

$$\Delta(\mathcal{I}_3^{\overline{W^+}}) \leqslant \sum_{i \in \mathcal{I}_3^{\overline{W^+}}} w_i = \sum_{w \in \overline{W^+}} w|\mathcal{I}_3 \cap \mathcal{I}^w| \leqslant |\overline{W^+}|w_{\max}w_{\max}^{1/5} = w_{\max}^{9/5}.$$

Consider $\mathcal{I}_4^{\overline{W^+}}$. Suppose, for the sake of contradiction, that $\Delta(\mathcal{I}_4^{\overline{W^+}}) > 2c_B w_{\max}^{9/5} > 2c_B w_{\max}^{8/5}$. That is, we add a large volume of items from $\mathcal{I}_4^{\overline{W^+}}$ when obtaining $\mathbf{z}$ from $\mathbf{g}$. Clearly, $\mathcal{I}_4^{\overline{W^+}}$ is not empty, so there are exactly $2c_A w_{\max}^{2/5} \log^2 w_{\max}$ weights that are frequent in $\mathcal{I}_3$. Recall that, by frequent, we mean $|\mathcal{I}^w \cap \mathcal{I}_3| \geqslant 2w_{\max}^{1/5}$. Then at least one of the following two cases are true.

(iii) there are at least $c_A w_{\max}^{2/5} \log^2 w_{\max}$ weights w such that $|\mathcal{I}^w \cap \overline{\mathbf{z}}(\mathcal{I}_3)| \geqslant w_{\max}^{1/5}$.

(iv) there are at least $c_A w_{\max}^{2/5} \log^2 w_{\max}$ weights w such that $|\mathcal{I}^w \cap \mathbf{z}(\mathcal{I}_3)| \geqslant w_{\max}^{1/5}$.

Case (iii). $\mathbf{z}(\mathcal{I}_4^{\overline{W^+}})$ has a total weight of $\Delta(\mathcal{I}_4^{\overline{W^+}}) > 2c_B w_{\max}^{8/5}$. For $\overline{\mathbf{z}}(\mathcal{I}_3)$, there are at least $c_A w_{\max}^{2/5} \log^2 w_{\max}$ weights w such that $|\mathcal{I}^w \cap \overline{\mathbf{z}}(\mathcal{I}_3)| \geqslant w_{\max}^{1/5}$. One can verify that the weight multisets of $\mathbf{z}(\mathcal{I}_4^{\overline{W^+}})$ and $\overline{\mathbf{z}}(\mathcal{I}_3)$ satisfy the conditions of Lemma 3.2. By Lemma 3.2, there is a subset $\mathcal{Z}$ of $\mathbf{z}(\mathcal{I}_4^{\overline{W^+}})$ and a subset $\mathcal{D}$ of $\overline{\mathbf{z}}(\mathcal{I}_3)$ such that $\mathcal{D}$ and $\mathcal{Z}$ have the same total weight. Then $\hat{\mathbf{z}} = (\mathbf{z} \setminus \mathcal{Z}) \cup \mathcal{D}$ would be a better solution than $\mathbf{z}$. Contradiction.

Case (iv). there are at least $c_A w_{\max}^{2/5} \log^2 w_{\max}$ weights w such that $|\mathcal{I}^w \cap \mathbf{z}(\mathcal{I}_3)| \geqslant w_{\max}^{1/5}$. $\Delta(\mathcal{I}_4^{\overline{W^+}}) > 2c_B w_{\max}^{8/5}$ implies that

$$\Delta(\mathcal{I}_{<b}) \geqslant \Delta(\mathcal{I}_4^{\overline{W^+}}) - w_{\max} \geqslant c_B w_{\max}^{8/5}.$$

Note that the total weight of $\overline{\mathbf{z}}(\mathcal{I}_{<b})$ is exactly $\Delta(\mathcal{I}_{<b})$. One can verify that the weight multisets of $\mathbf{z}(\mathcal{I}_3)$ and $\overline{\mathbf{z}}(\mathcal{I}_{<b})$ satisfy the conditions of Lemma 3.2. By Lemma 3.2, there is a subset $\mathcal{A}$ of $\mathbf{z}(\mathcal{I}_3)$ and a subset $\mathcal{D}$ of $\overline{\mathbf{z}}(\mathcal{I}_{<b})$ such that $\mathcal{D}$ and $\mathcal{A}$ have the same total weight. Then $\hat{\mathbf{z}} = (\mathbf{z} \setminus \mathcal{A}) \cup \mathcal{D}$ would be a better solution than $\mathbf{z}$. Contradiction. $\qquad \square$

D Proof of Lemma 6.3

LEMMA 6.3.

$$\mathcal{S}(X) = \sum_{i=0}^{\ell} 2^i \mathcal{S}(X_i) = 2^0 \mathcal{S}(X_0) + \cdots + 2^{\ell} \mathcal{S}(X_{\ell}).$$

That is, for any $t \in \mathcal{S}(X)$, there exist $t_i \in \mathcal{S}(X_i)$ such that $t = \sum_{i=0}^{\ell} 2^i t_i$.

Proof. It is obvious that $\mathcal{S}(X) \supseteq \sum_{i=0}^{\ell} 2^i \mathcal{S}(X_i)$. We prove below that $\mathcal{S}(X) \subseteq \sum_{i=0}^{\ell} 2^i \mathcal{S}(X_i)$, and Lemma 6.3 follows. Take an arbitrary $t' \in \mathcal{S}(X)$, by definition there exist $\eta_j \leqslant u_j = \sum_{i=0}^{\ell} u_j[i] \cdot 2^i$ for all $1 \leqslant j \leqslant n$ such that

$$t' = \sum_j w_j \eta_j.$$

We claim that

CLAIM D.1. *If $\eta_j \leqslant u_j = \sum_{i=0}^{\ell_j} u_j[i] \cdot 2^i$, then there exist $\eta_j[i]$'s such that $\eta_j = \sum_{i=0}^{\ell_j} \eta_j[i] \cdot 2^i$ and $\eta_j[i] \leqslant u_j[i]$ for all $0 \leqslant i \leqslant \ell_j$.*

Suppose the claim is true, then since X_i contains $u_j[i]$ copies of weight w_j, we know that $\sum_j \eta_j[i] w_j \in \mathcal{S}(X_i)$. Thus, $\sum_j \eta_j[i] \cdot 2^i w_j \in 2^i \mathcal{S}(X_i)$, and consequently $t' = \sum_i \sum_j \eta_j[i] \cdot 2^i w_j \in \sum_i 2^i \mathcal{S}(X_i)$, and Lemma 6.3 is proved.

It remains to prove Claim D.1. We prove by induction on ℓ_j. Claim D.1 is obviously true for $\ell_j = 0$. Suppose it is true for $\ell_j = h - 1$, we prove that it is also true for $\ell_j = h$. Towards this, let $r_t \in \{0, 1\}$ be the residue of t modulo 2. Recall that $u_j[i] \in [2, 4)$ for $0 \leqslant i \leqslant \ell_j - 1$. There are two possibilities: (i). If $r_t \leqslant u_j[0] - 2$, then we let $\eta_j[0] = r_t + 2$; (ii). If $u_j[0] - 2 < r_t \leqslant u_j[0]$, then we let $\eta_j[0] = r_t$. It is clear that $\eta_j[0] \leqslant u_j[0]$ is always true. Meanwhile, we also have $\eta_j[0] + 2 > u_j[0]$.

By the fact that $\eta_j[0] \equiv \eta_j \pmod{2}$ and $u_j[0] \equiv u_j \pmod{2}$, $\eta_j - \eta_j[0]$ and $u_j - u_j[0]$ are multiples of 2. Using that $\eta_j \leqslant u_j$ and $\eta_j[0] + 2 > u_j[0]$, we have

$$\eta_j - \eta_j[0] < u_j - u_j[0] + 2.$$

Thus,

$$\frac{\eta_j - \eta_j[0]}{2} < \frac{u_j - u_j[0]}{2} + 1.$$

Since $\frac{\eta_j - \eta_j[0]}{2}$ and $\frac{u_j - u_j[0]}{2}$ are both integers, we have that

$$\frac{\eta_j - \eta_j[0]}{2} \leqslant \frac{u_j - u_j[0]}{2} = \sum_{i=0}^{\ell-1} u_j[i+1] \cdot 2^i.$$

Using the induction hypothesis, there exists $\bar{\eta}_j[i]$ such that $\frac{\eta_j - \eta_j[0]}{2} = \sum_{i=0}^{\ell-1} \bar{\eta}_j[i] \cdot 2^i$ where $\bar{\eta}_j[i] \leqslant u_j[i+1]$. Recall that $\eta_j[0] \leqslant u_j[0]$. Consequently, Claim D.1 is true. □

References

[1] Amir Abboud, Karl Bringmann, Danny Hermelin, and Dvir Shabtay. SETH-based Lower Bounds for Subset Sum and Bicriteria Path. *ACM Transactions on Algorithms*, 18(1):1–22, January 2022. `doi:10.1145/3450524`. 2

[2] Alok Aggarwal, Maria M. Klawe, Shlomo Moran, Peter Shor, and Robert Wilber. Geometric applications of a matrix-searching algorithm. *Algorithmica*, 2(1):195–208, 1987. `doi:10.1007/BF01840359`. 3, 6, 15

[3] Kyriakos Axiotis and Christos Tzamos. Capacitated Dynamic Programming: Faster Knapsack and Graph Algorithms. In *46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)*, volume 132, pages 19:1–19:13, Dagstuhl, Germany, 2019. `doi:10.4230/LIPIcs.ICALP.2019.19`. 1, 2, 3, 4

[4] MohammadHossein Bateni, MohammadTaghi Hajiaghayi, Saeed Seddighin, and Cliff Stein. Fast algorithms for knapsack via convolution and prediction. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing (STOC 2018)*, pages 1269–1282, New York, NY, USA, June 2018. `doi:10.1145/3188745.3188876`. 1, 2, 3, 4

[5] Richard Bellman. *Dynamic Programming*, volume 33. Princeton University Press, 1957. `arXiv:j.ctv1nxcw0f`, `doi:10.2307/j.ctv1nxcw0f`. 1, 2, 4

[6] 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, January 2017. `doi:10.1137/1.9781611974782.69`. 2, 3, 4, 5, 14

[7] Karl Bringmann. Knapsack with small items in near-quadratic time, 2023. `arXiv:2308.03075`. 14

[8] 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)*, volume 229, pages 31:1–31:21, Dagstuhl, Germany, 2022. `doi:10.4230/LIPIcs.ICALP.2022.31`. 2, 4

[9] Karl Bringmann and Alejandro Cassis. Faster 0-1-Knapsack via Near-Convex Min-Plus-Convolution. In *31st Annual European Symposium on Algorithms (ESA 2023)*, volume 274, pages 24:1–24:16, Dagstuhl, Germany, 2023. `doi:10.4230/LIPIcs.ESA.2023.24`. 2, 3, 4

[10] Karl Bringmann and Vasileios Nakos. A fine-grained perspective on approximating subset sum and partition. In *Proceedings of the 2021 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2021)*, pages 1797–1815. SIAM, 2021. 11

[11] Karl Bringmann and Philip Wellnitz. On Near-Linear-Time Algorithms for Dense Subset Sum. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA 2021)*, pages 1777–1796, January 2021. `doi:10.1137/1.9781611976465.107`. 2, 3, 4, 6

[12] Timothy M. Chan. Approximation Schemes for 0-1 Knapsack. In *1st Symposium on Simplicity in Algorithms (SOSA 2018)*, volume 61, pages 5:1–5:12, Dagstuhl, Germany, 2018. `doi:10.4230/OASIcs.SOSA.2018.5`. 4

[13] Timothy M. Chan and Qizheng He. More on change-making and related problems. *Journal of Computer and System Sciences*, 124:159–169, 2022. `doi:10.1016/j.jcss.2021.09.005`. 4

[14] 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`. 2

[15] 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)*, Proceedings, pages 2961–2979, January 2023. `doi:10.1137/1.9781611977554.ch113`. 3, 4

[16] 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`. 1, 2, 3, 5

[17] Zvi Galil and Oded Margalit. An Almost Linear-Time Algorithm for the Dense Subset-Sum Problem. *SIAM Journal on Computing*, 20(6):1157–1189, December 1991. `doi:10.1137/0220072`. 2, 3, 6

[18] Klaus Jansen and Lars Rohwedder. On Integer Programming and Convolution. In *10th Innovations in Theoretical Computer Science Conference (ITCS 2019)*, 2019. `doi:10.4230/LIPIcs.ITCS.2019.43`. 4

[19] Ce Jin. 0-1 knapsack in nearly quadratic time, 2023. `arXiv:2308.04093`. 14

[20] Ce Jin and Hongxun Wu. A Simple Near-Linear Pseudopolynomial Time Randomized Algorithm for Subset Sum. In *2nd Symposium on Simplicity in Algorithms (SOSA 2019)*, volume 69, pages 17:1–17:6, Dagstuhl, Germany, 2018. `doi:10.4230/OASIcs.SOSA.2019.17`. 2, 5

[21] Hans Kellerer and Ulrich Pferschy. Improved Dynamic Programming in Connection with an FP-TAS for the Knapsack Problem. *Journal of Combinatorial Optimization*, 8(1):5–11, March 2004. `doi:10.1023/B:JOCO.0000021934.29833.6b`. 1, 2, 3, 4

[22] 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`. 2, 4

[23] 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)*, volume 80, pages 21:1–21:15, 2017. `doi:10.4230/LIPIcs.ICALP.2017.21`. 2

[24] Vsevolod F. Lev. Blocks and progressions in subset sum sets. *Acta Arithmetica*, 106(2):123–142, 2003. `doi:10.4064/aa106-2-3`. 3

[25] David Pisinger. Linear Time Algorithms for Knapsack Problems with Bounded Weights. *Journal of Algorithms*, 33(1):1–14, October 1999. `doi:10.1006/jagm.1999.1034`. 2, 3, 4

[26] 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)*, volume 198, pages 106:1–106:19, Dagstuhl, Germany, 2021. `doi:10.4230/LIPIcs.ICALP.2021.106`. 2, 3, 5, 6, 8, 12, 15

[27] A. Sárközy. Fine Addition Theorems, II. *Journal of Number Theory*, 48(2):197–218, August 1994. `doi:10.1006/jnth.1994.1062`. 3

[28] Arie Tamir. New pseudopolynomial complexity bounds for the bounded and other integer Knapsack related problems. *Operations Research Letters*, 37(5):303–306, 2009. `doi:10.1016/j.orl.2009.05.003`. 1, 2, 4