Approximating Knapsack and Partition via Dense Subset Sums

Mingyang Deng [*]   Ce Jin [†]   Xiao Mao [‡]
        MIT   MIT   MIT

January 24, 2023

Abstract

Knapsack and *Partition* are two important additive problems whose fine-grained complexities in the $(1 - \varepsilon)$-approximation setting are not yet settled. In this work, we make progress on both problems by giving improved algorithms.

- *Knapsack* can be $(1 - \varepsilon)$-approximated in $\tilde{O}(n + (1/\varepsilon)^{2.2})$ time, improving the previous $\tilde{O}(n + (1/\varepsilon)^{2.25})$ by Jin (ICALP'19). There is a known conditional lower bound of $(n + 1/\varepsilon)^{2-o(1)}$ based on $(\min, +)$-convolution hypothesis.

- *Partition* can be $(1 - \varepsilon)$-approximated in $\tilde{O}(n + (1/\varepsilon)^{1.25})$ time, improving the previous $\tilde{O}(n + (1/\varepsilon)^{1.5})$ by Bringmann and Nakos (SODA'21). There is a known conditional lower bound of $(1/\varepsilon)^{1-o(1)}$ based on Strong Exponential Time Hypothesis.

Both of our new algorithms apply the additive combinatorial results on dense subset sums by Galil and Margalit (SICOMP'91), Bringmann and Wellnitz (SODA'21). Such techniques have not been explored in the context of Knapsack prior to our work. In addition, we design several new methods to speed up the divide-and-conquer steps which naturally arise in solving additive problems.

---

[*]dengm@mit.edu

[†]cejin@mit.edu. Partially supported by NSF Grant CCF-2129139

[‡]matthew99a@gmail.com

1 Introduction

1.1 Background

Knapsack, *Subset Sum*, and *Partition* are three fundamental problems in computer science and mathematical optimization, and are actively being studied in fields such as integer programming and fine-grained complexity. In the Knapsack problem (sometimes also called 0-1 Knapsack), we are given a set I of n items where each item $i \in I$ has weight $w_i > 0$ and profit $p_i > 0$, as well as a knapsack capacity W, and we want to choose a subset $J \subseteq I$ satisfying the weight constraint $\sum_{j \in J} w_j \leq W$ such that the total profit $\sum_{j \in J} p_j$ is maximized. The Subset Sum problem is a special case of Knapsack, where the weight of an item is always equal to their profit. The Partition problem is a special case of Subset Sum, where the capacity equals half of the total weight of the items. In other words, in Partition we want to partition the input items into two parts so that their sums is as close as possible.

These three problems are well-known to be hard: they appeared in Karp's original list of 21 NP-hard problems [Kar72]. To cope with NP-hardness, a natural direction is to study their *approximation algorithms*. Given a parameter $\varepsilon > 0$, and an input instance with optimal value OPT, a $(1-\varepsilon)$-approximation algorithm is required to output a number SOL such that $(1-\varepsilon)\text{OPT} \leq \text{SOL} \leq \text{OPT}$. Fortunately, these three problems are well-known to have *fully polynomial-time approximation schemes (FPTASes)*, namely $(1-\varepsilon)$-approximation algorithm that runs in $\text{poly}(n, 1/\varepsilon)$ time, for any $\varepsilon > 0$.

There has been a long line of research since the 70's on getting approximation schemes for these problems with improved time complexities in terms of n and $1/\varepsilon$ [IK75, Law79, GL79, KP99, KMPS03, KP04, Rhe15, JK18, Cha18, MWW19, Jin19, BN21, BC22]. On the other hand, recent advances in fine-grained complexity have pointed out the limit of such improvements, under well-believed hardness assumptions [CMWW19, KPS17, ABHS19, BN21]. Here, we briefly describe the most recent results along this line.

- **Knapsack:** The best known algorithm by Jin [Jin19] runs in $\tilde{O}(n + \varepsilon^{-2.25})$ time, and is based on the previous algorithm of Chan [Cha18] in $\tilde{O}(n + \varepsilon^{-2.4})$ time. [CMWW19] and [KPS17] showed a conditional lower bound of $(n + \frac{1}{\varepsilon})^{2-o(1)}$, based on the $(\min, +)$-convolution hypothesis.

- **Subset Sum:** The best known algorithm by Bringmann and Nakos [BN21] runs in $\tilde{O}(n + \varepsilon^{-2}/2^{\Omega(\sqrt{\log(1/\varepsilon)})})$ time (improving [KMPS03] by low-order factors). Bringmann and Nakos [BN21] showed a matching lower bound based on the $(\min, +)$-convolution hypothesis.

- **Partition:** The first breakthrough by Mucha, Węgrzycki and Włodarczyk [MWW19] gave a randomized algorithm in $\tilde{O}(n + 1/\varepsilon^{5/3})$ time. Later, Bringmann and Nakos [BN21] improved it to deterministic $\tilde{O}(n + 1/\varepsilon^{-3/2})$ time. Abboud, Bringmann, Hermelin, and Shabtay [ABHS19] showed that Partition cannot be approximated in $\text{poly}(n)/\varepsilon^{1-\delta}$ time for any $\delta > 0$, under the Strong Exponential Time Hypothesis.

We can see that the complexity of Subset Sum is already settled, but for Knapsack and Partition there still remain gaps between the best known algorithms and their conditional lower bounds.

1.2 Our Results

In this work, we make progress on this direction, by giving improved approximation schemes for Knapsack and Partition.

Theorem 1.1. *There is a randomized algorithm for $(1 - \varepsilon)$-approximating Knapsack with running time* [1]

$$\tilde{O}\left(n + \varepsilon^{-11/5}/2^{\Omega(\sqrt{\log(1/\varepsilon)})}\right).$$

Theorem 1.2. *There is a deterministic algorithm for $(1-\varepsilon)$-approximating Partition with running time*

$$\tilde{O}\left(n + \varepsilon^{-5/4}\right).$$

1.3 Technical Overview

A useful result in additive combinatorics for many subset sum related problems is the one from by Galil and Margalit [GM91], which was later improved by Bringmann and Wellnitz [BW21] (based on results of Sárközy [Sár94]). These combinatorial results reveal structural properties on the set $\mathcal{S}(X)$ of subset sums of a positive integer set X, defined as $\mathcal{S}(X) := \{\sum_{y \in Y} y : Y \subseteq X\}$, in the case when X is *"dense"*. Intuitively, it states that if the total number of items is large, then a large portion of the subset sum can be computed very efficiently, so only a small margin of the sumsets with value up to some λ needs to be approximated. We apply these combinatorial results to Partition and, surprisingly, to the "two-dimensional" problem of Knapsack where each item has both weights and values.

1.3.1 Knapsack

Faster knapsack via dense subset sums Our improved approximation scheme for knapsack relies on multiple technical components from the previous algorithms by Chan [Cha18] and Jin [Jin19]. However, one novel key idea that makes our improvement possible is a technique previously not explored in the context of knapsack algorithms: the additive combinatorics result for dense subset sum by Galil and Margalit [GM91]. One particular result useful to us roughly says the following: when X consists of n distinct integers in $[\ell, 2\ell]$ for a small enough $\ell \ll n^2$, then there is a long enough interval $[\lambda, (\sum_{x \in X} x) - \lambda]$ that is *densely filled* with elements in $\mathcal{S}(X)$, in the sense that every two adjacent elements must be very close to each other. A formal version of the statement is in Lemma 2.6. As we will see, such density statements will be useful in the framework of Jin [Jin19]. Jin's approximation algorithm for Knapsack separately deals with items with high and low *efficiency*, defined as the profit-to-weight ratio p_i/w_i. Intuitively, it is not very profitable to include too many low-efficiency items in the solution. Indeed, after some technical steps, Jin managed to place an upper bound B on the total profit contributed by low-efficiency items in any optimal solution, so that it is still correct to only compute the answers for low-efficiency items up to B (which would take much shorter time than original). The way Jin proved such a bound B was by a certain greedy exchange argument: given a solution set with too many low-efficiency items (which occupies a total capacity of W_L), remove them and try to fill in the freed up space of W_L using high-efficiency items instead. This would potentially lead to a better solution, contradicting the optimality of the given solution.

---

[1] Throughout this paper, we use $\tilde{O}(f)$ to denote $O(f \cdot \operatorname{poly} \log(f))$.

Naturally, such an exchange is not always profitable, since the high-efficiency items may not be able to fill up the entire space W_L. Jin's argument accounts for this issue by additionally making sure that all items have size in an interval $[\ell, 2\ell]$, so that the wasted space after the exchange cannot be larger 2ℓ (otherwise one can always fit in another high-efficiency item). In our new algorithm, we refine this argument using combinatorial results on dense subset sums: observe that the task of minimizing the wasted space is equivalent to a subset sum problem on the sizes of high-efficiency items. By setting up parameters appropriately, we can make sure that the dense subset sum result applies, leading to a much smaller wasted space.

Having refined this argument, we can improve Jin [Jin19] by putting a stricter upper bound on the total profit contributed by low-efficiency items, leading to an improved running time.

1.3.2 Partition

Densified divide and conquer   There have been many FPTAS algorithms for problems such as Subset Sum, Partition, KNAPSACK that employ the technique of divide and conquer, e.g. [Cha18, Jin19, BN21]. Unlike previous methods, our improvement crucially relies on performing divide-and-conquer on the *sorted* list of items. To motivate our idea, we note that in most divide and conquer based algorithms, the bottleneck to the running time is incurred at the bottom levels, where we need to merge two sets of answers often with the same complexity of those at the top levels. However, intuitively, if our list is sorted, at the bottom levels the items have values contained in a small interval, and hence the sumsets of these items are clustered in several small intervals with large gaps between them. To exploit this property, instead of using the usual 1D FFT to merge the sumsets, we "densify" the sumsets and merge them using 2D FFT, so that our running time is only dependent on the total length of these small intervals.

We note that the idea of 2D-FFT has been employed before to Subset Sum by Koiliaris and Xu [KX19], but it is used in a different spirit: while we use 2D-FFT to "densify" sumsets, in [KX19] it is used to bound the size of the solution for each sum.[2]

Combining with additive combinatorics result   The additive combinatorics result for dense subset sum by Galil and Margalit [GM91] has also been used to an extent in the $\tilde{O}(n + 1/\varepsilon^{5/3})$ algorithm by [MWW19]. In our algorithm we combine this with densified divide and conquer. Note that λ can be much smaller than the total sum σ of the items, and this would mean that an $\varepsilon\sigma$ additive error is an $\varepsilon'\lambda$ additive error for a much larger $\varepsilon' = \varepsilon\sigma/\lambda$, so we only need to ensure $(1 - \varepsilon')$-multiplicative approximation instead of $(1 - \varepsilon)$ during the computation.

1.4   Paper Organization

We will give useful definitions and lemmas in Section 2. In Section 3 we present our algorithm for Knapsack. In Section 4 we present our algorithm for Partition. Some standard reductions and known lemmas from previous works are deferred to appendix.

---

[2]In fact, it can be verified that by combining our way of doing 2D-FFT with the other techniques in [KX19] we can get an alternative deterministic solution for Subset Sum running in $\tilde{O}(\sqrt{n}t)$ time. It is interesting to see if the two ways of using 2D-FFT can be combined to improve the running time for Subset Sum deterministically.

2 Preliminaries

We write $\mathbb{N} = \{0, 1, 2, \dots\}$ and $\mathbb{N}^+ = \{1, 2, \dots\}$. For $n \in \mathbb{N}$ we write $[n] = \{1, 2, \dots, n\}$.

2.1 Problem Statements

In the Knapsack problem, the input is a list of n items $(p_1, w_1), \dots, (p_n, w_n) \in \mathbb{N} \times \mathbb{N}$ together with a knapsack capacity $W \in \mathbb{N}$, and the optimal value is

$$\text{OPT} := \max_{J \subseteq [n]} \Big\{ \sum_{j \in J} p_j \,\Big|\, \sum_{j \in J} w_j \leq W \Big\}.$$

In the easier Partition problem, the input is a list of n integers $x_1, \dots, x_n \in \mathbb{N}$, and the optimal value is

$$\text{OPT} := \max_{J \subseteq [n]} \Big\{ \sum_{j \in J} x_j \,\Big|\, \sum_{j \in J} x_j \leq \frac{1}{2} \sum_{i \in [n]} x_i \Big\}.$$

Given a Knapsack (or a Partition) instance and a parameter $\varepsilon \in (0, 1)$, an $(1 - \varepsilon)$-*approximation algorithm* is required to output a number SOL such that $(1 - \varepsilon)\text{OPT} \leq \text{SOL} \leq \text{OPT}$.

In both problems, we can assume $n = O(\varepsilon^{-4})$ and hence $\log n = O(\log \varepsilon^{-1})$. For larger n, Lawler's algorithm [Law79] for Knapsack in $O(n \log \frac{1}{\varepsilon} + (\frac{1}{\varepsilon})^4)$ time is already near-optimal.

We will sometimes describe algorithms with approximation ratio $1 - O(\varepsilon)$ (or $1 - \varepsilon \cdot \text{poly} \log(1/\varepsilon)$), which can be made $1 - \varepsilon$ by scaling down ε by a constant factor (or a logarithmic factor) at the beginning.

2.2 Sumsets and Subset Sums

In a multiset A, an element a could appear multiple times (the number of times it appears is the *multiplicity* of a in A). We use $A \uplus B$ to denote union without removing duplicates (i.e., possibly resulting in a multiset).

For a multiset $Y \subset \mathbb{N}$, let $\Sigma(Y) = \sum_{y \in Y} y$ denote the sum of its elements (without removing duplicates).

For a multiset $X \subset \mathbb{N}$, let $\mathcal{S}(X) = \{\Sigma(Y) : Y \subseteq X\}$ be the set of its subset sums, and let $\mathcal{S}(X; t) = \mathcal{S}(X) \cap [0, t]$ be the set of its subset sums up to t.

For a number c and a set X, define $c \cdot X = \{cx : x \in X\}$. For two sets X, Y, define their *sumset* $X + Y = \{x + y : x \in X, y \in Y\}$. Given sets $X \subseteq [n], Y \subseteq [n]$, the sumset $X + Y$ can be computed in $O(n \log n)$ time using FFT. This simple fact has a straightforward generalization to 2 dimension, which we state below.

Lemma 2.1 (2-dimensional FFT, e.g., [Bla10, Chapter 12.8]). *Given two sets $A_1, A_2 \subseteq [n] \times [m]$, one can compute*

$$A_1 + A_2 := \big\{ (x_1 + x_2, y_1 + y_2) : (x_1, y_1) \in A_1, (x_2, y_2) \in A_2 \big\}$$

in $O(nm \log(nm))$ time deterministically.

2.3 Knapsack Problem and Profit functions

In the knapsack problem, assume $0 < w_i \leq W$ and $p_i > 0$ for every item i. Then a trivial lower bound of the maximum total profit is $\max_j p_j$. At the beginning, we can discard all items i with $p_i \leq \frac{\varepsilon}{n} \max_j p_j$, reducing the total profit by at most $\varepsilon \max_j p_j$, which is only an $O(\varepsilon)$ fraction of the optimal total profit. So we can assume $\frac{\max_j p_j}{\min_j p_j} \leq \frac{n}{\varepsilon}$.

For a set I of items, we use f_I to denote its *profit function*, defined as

$$f_I(x) = \max \left\{ \sum_{i \in J} p_i : \sum_{i \in J} w_i \leq x, \ J \subseteq I \right\}$$

over $x \in [0, +\infty)$. Note that f_I is a monotone nondecreasing step function. Adopting the terminology of Chan [Cha18], the *complexity* of a monotone step function refers to the number of its steps.

Let I_1, I_2 be two disjoint subsets of items, and $I = I_1 \uplus I_2$. It is straightforward to see that $f_I = f_{I_1} \oplus f_{I_2}$, where $\oplus$ denotes $(\max, +)$-convolution, defined by $(f \oplus g)(x) = \max_{0 \leq x' \leq x}(f(x') + g(x - x'))$.

2.4 $(1 - \delta, \Delta)$ approximation up to t

Both our algorithms for Knapsack and Partition frequently use the notion of $(1 - \delta, \Delta)$-*approximation up to t*. Their definitions are analogous, as stated below.

Definition 2.2 (Approximation for Profit Functions). *For functions $\tilde{f}, f$ and real numbers $t, \Delta \in \mathbb{R}_{\geq 0}, \delta \in [0, 1)$, we say that $\tilde{f}$ is a $(1 - \delta, \Delta)$ approximation of f up to t, if*

$$\tilde{f}(w) \leq f(w)$$

holds for all $w \geq 0$, and

$$\tilde{f}(w) \geq (1 - \delta)f(w) - \Delta$$

holds whenever $f(w) \leq t, w \geq 0$.

The following notion of approximation will be useful in our Partition algorithm. Similar notions have been termed as "weak approximation" in the literature [MWW19, BN21], in contrast to "strong approximation" that would be required for approximating general Subset Sum instances.

Definition 2.3 (Approximation for Integer Sets). *For integer sets $A, B \subseteq \mathbb{N}$, and real numbers $t, \Delta \in \mathbb{R}_{\geq 0}, \delta \in [0, 1)$, we say that A is a $(1 - \delta, \Delta)$ approximation of B up to t, if*

1. *for every $b \in B \cap [0, t]$, there exists $a \in A$ such that $(1 - \delta)b - \Delta \leq a \leq b$, and,*

2. *for every $a \in A$, there exists $b \in B$ such that $(1 - \delta)b - \Delta \leq a \leq b$.*

One can assume $A \subseteq \mathbb{N} \cap [0, t]$ in this case without loss of generality.

For the case of $t = +\infty$, we simply omit the phrase "up to t".

We also refer to $(1, \Delta)$ approximation as Δ-*additive approximation*, and refer to $(1 - \delta, 0)$ approximation as $(1 - \delta)$-*multiplicative approximation*, or simply $(1 - \delta)$ *approximation*.

We have the following simple facts regarding approximating merged sumsets and profit functions.

Proposition 2.4. *For $i \in \{1, 2\}$, suppose A_i is a $(1 - \delta, \Delta_i)$ approximation of $\mathcal{S}(X_i)$ up to t. Then, $(A_1 + A_2) \cap [0, t]$ is a $(1 - \delta, \Delta_1 + \Delta_2)$ approximation of $\mathcal{S}(X_1 \uplus X_2)$ up to t.*

Proof. For any $b \in \mathcal{S}(X_1 \uplus X_2) \cap [0, t]$ where $b = b_1 + b_2$ for $b_i \in \mathcal{S}(X_i) \cap [0, t]$ $(i \in \{1, 2\})$, there exits $a_i \in A_i$ such that $(1 - \delta)b_i - \Delta_i \le a_i \le b_i$. Hence, $a_1 + a_2 \le b_1 + b_2 = b \le t$, and $a_1 + a_2 \ge (1 - \delta)b_1 - \Delta_1 + (1 - \delta)b_2 - \Delta_2 = (1 - \delta)b - (\Delta_1 + \Delta_2)$.

The converse direction can be verified similarly. □

The following fact can be proved similarly.

Proposition 2.5. *For $i \in \{1, 2\}$, suppose $\tilde{f}_i$ is a $(1 - \delta, \Delta_i)$ approximation of the profit function f_{I_i} up to t. Then, $(\tilde{f}_1 \oplus \tilde{f}_2)$ is a $(1 - \delta, \Delta_1 + \Delta_2)$ approximation of $f_{I_1 \uplus I_2}$ up to t.*

Following Chan [Cha18] and Jin [Jin19], given a monotone step function f (we sometimes also call it a profit function, although it might not be equal to the profit function f_I of any particular item set I) with range contained in $\{0\} \cup [A, B]$, one can round f down to powers of $1/(1 - \varepsilon)$, and obtain another profit function $\tilde{f}$ which has complexity only $O(\varepsilon^{-1} \log(B/A))$, and $(1 - \varepsilon)$-approximates f. In our algorithm we will always have $B/A \le \mathrm{poly}(n/\varepsilon)$, so we may always assume that the intermediate profit functions computed during our algorithm are monotone step functions with complexity $\tilde{O}(\varepsilon^{-1})$, by incurring $(1 - \varepsilon)$ approximation factor each time.

2.5   Additive Combinatorics

We need several results on dense subset sums developed by a series of works including [Sár94, GM91, Lev03, BW21]. The following structural lemma follows from Theorem 4.1 and Theorem 4.2 of Bringmann and Wellnitz [BW21].

Lemma 2.6. *Let n distinct positive integers $X = \{x_1, \ldots, x_n\} \subseteq [\ell, 2\ell]$ be given, where $\ell = o(n^2 / \log n)$.*

Then, for a universal constant $c \ge 1$, for every $c\ell^2/n \le t \le \Sigma(X)/2$, there exists $t' \in \mathcal{S}(X)$ such that $0 \le t' - t \le 8\ell/n$.

A proof of Lemma 2.6 is included in Appendix B.

The following algorithmic lemma follows from the main theorem of [BW21], and will be used in our Partition algorithm.

Lemma 2.7 (Follows from [BW21]). *Given n distinct positive integers $X = \{x_1, \ldots, x_n\} \subseteq [\ell, 2\ell]$, there exists $\lambda = \tilde{\Theta}(\ell^2/n)$ such that, if $\lambda \le \Sigma(X)/2$, then in $\tilde{O}(n)$ time we can construct a deterministic data structure supporting the following query in $O(1)$ time: given L, R such that $\lambda \le L \le R \le \Sigma(X)/2$, report whether there exists $t \in [L, R]$ such that $t \in \mathcal{S}(X)$.*

Remark 2.8. *We remark that the main theorem stated in [BW21] only supports querying whether a given $\lambda \le t \le \Sigma(X)/2$ is a subset sum. In our application, we require a version supporting range queries. This is easy to achieve by building an additional prefix sum array in the proof of [BW21, Theorem 4.6], which supports range sum queries.*

3 Approximating Knapsack

3.1 Known Lemmas

By known reductions (e.g., [Cha18, Jin19]), we can focus on solving the following cleaner problem, which already captures the main difficulty of knapsack.

Problem 1. *Assume $\varepsilon \in (0, 1/2)$ and $1/\varepsilon \in \mathbb{N}^+$. Given a list I of items $(p_1, w_1), \ldots, (p_n, w_n)$ with weights $w_i \in \mathbb{N}$ and profits p_i being multiples of ε in the interval $[1, 2)$, compute a profit function that $(1 - \varepsilon)$-approximates f_I up to $2/\varepsilon$.*

Lemma 3.1. *If for some $c \geq 2$, Problem 1 can be solved in $\tilde{O}(n + 1/\varepsilon^c)$ time, then $(1 - \varepsilon)$-approximating Knapsack can also be solved in $\tilde{O}(n + 1/\varepsilon^c)$ time.*

Lemma 3.1 will be proved in the appendix.

Based on Chan's techniques [Cha18], Jin [Jin19] obtained the following lemmas for $(1 - \varepsilon)$-approximating knapsack up to a small B or when there are few distinct values p_i.

Lemma 3.2 (Follows from Lemma 17 of [Jin19])**.** *Given a list I of items $(p_1, w_1), \ldots, (p_n, w_n)$ with weights $w_i \in \mathbb{N}$ and profits p_i being multiples of ε in the interval $[1, 2)$, one can $(1 - \varepsilon)$-approximate the profit function f_I up to B in $\tilde{O}(n + \varepsilon^{-2} B^{1/3}/2^{\Omega(\sqrt{\log(1/\varepsilon)})})$ time.*

Lemma 3.3 (Follows from Theorem 3 of [Cha18])**.** *Given a list I of items $(p_1, w_1), \ldots, (p_n, w_n)$ with weights $w_i \in \mathbb{N}$ and profits p_i being multiples of ε in the interval $[1, 2)$, if there are only m distinct profit values p_i, then one can $(1-\varepsilon)$-approximate the profit function f_I in $\tilde{O}(n+\varepsilon^{-3/2}m/2^{\Omega(\sqrt{\log(1/\varepsilon)})})$ time.* [3]

The following useful lemma allows us to merge multiple profit functions, which was proved by Chan using divide-and-conquer and improved algorithms for $(\min, +)$-convolution [BCD+14, Wil14, CW16].

Lemma 3.4 ([Cha18, Lemma 2(i)])**.** *Let $f_1, \ldots, f_m$ be monotone step functions with total complexity $O(n)$ and ranges contained in $\{0\} \cup [A, B]$. Then we can compute a monotone step function that has complexity $\tilde{O}(\frac{1}{\varepsilon} \log B/A)$ and $(1 - O(\varepsilon))$-approximates $f_1 \oplus \cdots \oplus f_m$, in $O(n) + \tilde{O}((\frac{1}{\varepsilon})^2 m/2^{\Omega(\sqrt{\log(1/\varepsilon)})} \log B/A)$ time.*

3.2 Greedy Exchange Argument via Dense Subset Sum

The goal of this section is to prove the following Lemma 3.5. Our algorithm is based on a greedy exchange argument similar to [Jin19, Lemma 20], but we can obtain better bounds by combining with number theoretic results on dense subset sums.

Lemma 3.5. *Given a list I of n items with p_i being multiples of ε in interval $[1, 2)$, and integer $1 \leq m \leq n$ with $m = O(1/\varepsilon)$, one can compute in $O(n + \varepsilon^{-11/5}/2^{\Omega(\sqrt{\log(1/\varepsilon)})})$ time a profit function that $(m\varepsilon)$-additively approximates f_I up to $2m$.*

---

[3]In the proceedings version of our paper, we incorrectly claimed that the task in Lemma 3.3 can be done in $\tilde{O}(n + \varepsilon^{-3/2}m^{3/4}/2^{\Omega(\sqrt{\log(1/\varepsilon)})})$ time. Here, the statement of Lemma 3.3 has been corrected. As a result, several parameters in Section 3.2 have been adjusted accordingly. This correction did not affect the final time bound of our main result (Theorem 1.1), since the step that invokes Lemma 3.3 is not a bottleneck in our algorithm.

The proof of Lemma 3.5 assumes the following ingredient, which will be proved in later sections using random partitioning.

Lemma 3.6. *Given a list I of $n = O(1/\varepsilon)$ items with p_i being multiples of ε in interval $[1, 2)$, one can compute in $\tilde{O}(n^{4/5}\varepsilon^{-7/5}/2^{\Omega(\sqrt{\log(1/\varepsilon)})})$ time a profit function that $(n\varepsilon)$-additively approximates f_I.*

Now we proceed to describe the algorithm for Lemma 3.5. Given items $(p_1, w_1), \ldots, (p_n, w_n)$, where $p_i \in [1, 2)$ are multiples of ε, we sort them by non-increasing order of efficiency, $p_1/w_1 \geq p_2/w_2 \geq \cdots \geq p_n/w_n$. Then, we consider prefixes of this sequence of items, and define the following measure of diversity:

Definition 3.7 ($D(i)$). *For $1 \leq i \leq n$, let $D(i) = \min_J C([i] \setminus J)$, where the minimization is over all subsets $J \subseteq [i]$ with $|J| \leq 2m$, and $C([i] \setminus J)$ denote the number of distinct values in $\{p_j : j \in [i] \setminus J\}$.*

We have the following immediate observations about $D(i)$.

Observation 3.8.   *1. For all $2 \leq i \leq n$, $0 \leq D(i) - D(i-1) \leq 1$.*

2. $D(i)$ (and the minimizer J) can be computed in $\tilde{O}(i)$ time by the following greedy algorithm: Start with all values $p_1, p_2, \ldots, p_i$. Repeat the following up to $2m$ times: remove the value p_j with the minimum multiplicity, and add j into J.

Now, we set parameter $\Delta = \lfloor \varepsilon^{-5/8} \rfloor$. Define $i \in \{1, 2, \ldots, n\}$ to be the maximum such that $D(i) \leq \Delta$, which can be found using Observation 3.8 with a binary search in $\tilde{O}(n)$ time.

The following lemma is the key component in our proof of Lemma 3.5.

Lemma 3.9 (Greedy Exchange Lemma). *Let $S \subseteq [n]$ be any item set with total profit $\sum_{s \in S} p_s \leq 2m$. Let $B := 9c\varepsilon^{-1}/\Delta$, where $c \geq 1$ is the universal constant in Lemma 2.6.*

Then, there exists an item set $\tilde{S} \subseteq [n]$, such that the total profit $\tilde{p}$ contributed by items $[n] \setminus [i]$ in $\tilde{S}$ satisfies

$$\tilde{p} := \sum_{s \in \tilde{S} \cap ([n] \setminus [i])} p_s \leq B, \tag{1}$$

and

$$\sum_{s \in \tilde{S}} p_s \geq (1 - \varepsilon) \sum_{s \in S} p_s, \tag{2}$$

and

$$\sum_{s \in \tilde{S}} w_s \leq \sum_{s \in S} w_s. \tag{3}$$

Proof. If $D(i) < \Delta$, then by the definition of i we have $i = n$, and we can simply let $\tilde{S} = S$, since $\tilde{p} = 0$ always holds. So in the following we assume $D(i) = \Delta$.

We define $\tilde{S} \subseteq [n]$ as the maximizer of

$$\sum_{s \in \tilde{S} \cap [i]} p_s + \sum_{s \in \tilde{S} \cap ([n] \setminus [i])} (1 - \varepsilon) p_s$$

8

among all $\tilde{S}$ satisfying $\sum_{s\in\tilde{S}} w_s \leq \sum_{s\in S} w_s$ and $\sum_{s\in\tilde{S}} p_s \leq \sum_{s\in S} p_s$. We claim that $\tilde{S}$ satisfies the properties (1), (2), (3). Observe that (2), (3) immediately follow from the definition of $\tilde{S}$. The main part is to verify (1).

Suppose for contradiction that (1) does not hold. Then, we can find a subset $K \subseteq \tilde{S} \cap ([n] \setminus [i])$ with total profit $p^* = \sum_{k\in K} p_k \in (B, B+2]$, which can be obtained by removing items from $\tilde{S} \cap ([n] \setminus [i])$ (recall that each item has profit in $[1,2)$).

Define item set $I' := [i] \setminus \tilde{S}$. Since $|\tilde{S}| < \sum_{s\in\tilde{S}} p_s / \min_{s\in\tilde{S}} p_s \leq \sum_{s\in\tilde{S}} p_s \leq \sum_{s\in S} p_s \leq 2m$, by the definition of $D(i)$, we know that $\{p_i : i \in I'\}$ contains at least $D(i) = \Delta$ distinct elements.

We apply Lemma 2.6 on the set of integers $X = \{p_i/\varepsilon : i \in I'\} \subseteq [1/\varepsilon, 2/\varepsilon)$ which contains at least Δ distinct integers, where the premise $1/\varepsilon = o(\Delta^2/\log\Delta)$ in Lemma 2.6 is satisfied by our choice of $\Delta = \lfloor \varepsilon^{-5/8} \rfloor$. Lemma 2.6 states that for every $t \in [c\varepsilon^{-2}/\Delta, 0.5\Delta/\varepsilon]$, there exists $t' \in \mathcal{S}(X)$ such that $0 \leq t' - t \leq 8\varepsilon^{-1}/\Delta$. Here we set

$$t := \frac{(1-\varepsilon)p^*}{\varepsilon} + \frac{\varepsilon^{-1}}{\Delta},$$

which satisfies $t > p^*(1-\varepsilon)/\varepsilon > (1-\varepsilon)B/\varepsilon = (1-\varepsilon)(9c\varepsilon^{-1}/\Delta)/\varepsilon > c\varepsilon^{-2}/\Delta$, and $t < (B+2)/\varepsilon + \varepsilon^{-1}/\Delta = 9c\varepsilon^{-2}/\Delta + 2/\varepsilon + \varepsilon^{-1}/\Delta \leq O(\varepsilon^{-11/8}) \leq 0.5\Delta/\varepsilon$. Then the conclusion of Lemma 2.6 says that there is a subset $R \subseteq I'$ of items with total profit $\tilde{p} := \varepsilon \cdot t'$, satisfying

$$1/\Delta \leq \tilde{p} - p^*(1-\varepsilon) \leq 9/\Delta. \tag{4}$$

Note that (4) implies

$$\begin{aligned}
p^* - \tilde{p} &\geq \varepsilon \cdot p^* - 9/\Delta \\
&> \varepsilon \cdot B - 9/\Delta \\
&= \varepsilon \cdot 9c\varepsilon^{-1}/\Delta - 9/\Delta \\
&\geq 0.
\end{aligned}$$

Recall that $R \subseteq I' = [i] \setminus \tilde{S}$ and $K \subseteq \tilde{S} \cap ([n] \setminus [i])$, which must both be non-empty. Since the efficiency of items are sorted in non-increasing order, we have $\min_{r\in R} p_r/w_r \geq \max_{k\in K} p_k/w_k$. Now we define the set of items

$$\tilde{S}' := (\tilde{S} \setminus K) \cup R.$$

Then, we have

$$\begin{aligned}
\sum_{s\in\tilde{S}} p_s - \sum_{s\in\tilde{S}'} p_s &= \sum_{k\in K} p_k - \sum_{r\in R} p_r \\
&= p^* - \tilde{p} \\
&\geq 0,
\end{aligned}$$

and

$$\sum_{s\in\tilde{S}} w_s - \sum_{s\in\tilde{S}'} w_s = \sum_{k\in K} w_k - \sum_{r\in R} w_r$$

$$\geq \frac{\sum_{k\in K} p_k}{\max_{k\in K}(p_k/w_k)} - \frac{\sum_{r\in R} p_r}{\min_{r\in R}(p_r/w_r)}$$

$$\geq \frac{1}{\min_{r\in R}(p_r/w_r)} \cdot \left(\sum_{k\in K} p_k - \sum_{r\in R} p_r\right)$$

$$= \frac{1}{\min_{r\in R}(p_r/w_r)} \cdot (p^* - \tilde{p})$$

$$\geq 0.$$

Hence, $\sum_{s\in\tilde{S}'} p_s \leq \sum_{s\in\tilde{S}} p_s$ and $\sum_{s\in\tilde{S}'} w_s \leq \sum_{s\in\tilde{S}} w_s$. On the other hand, by (4), we know that

$$\left(\sum_{s\in\tilde{S}'\cap[i]} p_s + \sum_{s\in\tilde{S}'\cap([n]\setminus[i])} (1-\varepsilon)p_s\right) - \left(\sum_{s\in\tilde{S}\cap[i]} p_s + \sum_{s\in\tilde{S}\cap([n]\setminus[i])} (1-\varepsilon)p_s\right)$$

$$= \sum_{r\in R} p_r - \sum_{k\in K} (1-\varepsilon)p_k$$

$$= \tilde{p} - (1-\varepsilon)p^*$$

$$\geq 1/\Delta > 0,$$

contradicting the definition of $\tilde{S}$ being a maximizer.

Hence, we have established that $\tilde{S}$ satisfies (1). $\qquad\square$

Now we are ready to prove Lemma 3.5.

Proof of Lemma 3.5. Recall that $i \in \{1, 2, \ldots, n\}$ is the maximum such that $D(i) \leq \Delta$, which can be found using Observation 3.8 with a binary search in $\tilde{O}(n)$ time. Let $J \subset [i]$ with $|J| \leq 2m$ be the minimizer for $D(i)$.

Now, we approximately compute the profit functions $f_J, f_{[i]\setminus J}, f_{[n]\setminus[i]}$ for three item sets $J, [i] \setminus J, [n] \setminus [i]$ using different algorithms, described as follows:

1. Use Lemma 3.6 to compute f_1 that $(2m\varepsilon)$-additively approximates f_J, in $O(m^{\frac{4}{5}}\varepsilon^{-\frac{7}{5}}/2^{\Omega(\sqrt{\log(1/\varepsilon)})}) \leq O(\varepsilon^{-\frac{11}{5}}/2^{\Omega(\sqrt{\log(1/\varepsilon)})})$ time.

2. By definition of i, items in $[i] \setminus J$ have no more than Δ distinct profit values. Hence we can use Lemma 3.3 to compute f_2 that $(1-\varepsilon)$-approximates $f_{[i]\setminus J}$, in $\tilde{O}(\Delta\varepsilon^{-3/2}) = \tilde{O}(\varepsilon^{-17/8})$ time.

3. Use Lemma 3.2 to compute f_3 that $(1-\varepsilon)$-approximates the $f_{[n]\setminus[i]}$ up to $B = \Theta(\varepsilon^{-1}/\Delta)$ (defined in Lemma 3.9), in $\tilde{O}(B^{1/3}\varepsilon^{-2}) \leq \tilde{O}(\varepsilon^{-17/8})$ time.

Finally, merge the three parts f_1, f_2, f_3 using Lemma 3.4 in $\tilde{O}(\varepsilon^{-2})$ time, and return the result.[4]

---

[4]Although the running time of the second and third algorithm is dominated by the first algorithm, a simple rebalancing of parameters does not seem to yield better complexity, due to various constraints in the parameter settings for Lemma 3.6.

In the third part, the correctness of only computing up to B is justified by Lemma 3.9, which shows that if we only consider approximating sets with total profit up to $2m$, then we can assume the items in $[n] \setminus [i]$ only contributes profit at most B (1), at the cost of only incurring an $(1 - \varepsilon)$ approximation factor (2).

To analyze the error, notice that in the first part we incur an additive error of $(2m\varepsilon)$. In the second and third part and the final merging step we incur $(1 - O(\varepsilon))$ multiplicative error, which turns into $O(m\varepsilon)$ additive error since we only care about approximating up to $2m$. Hence the overall additive error is $O(m\varepsilon)$, which can be made $m\varepsilon$ by lowering the value of ε. □

Now we show that Lemma 3.5 can be used to solve Problem 1, which is sufficient for proving Theorem 1.1.

Proof of Theorem 1.1. To solve Problem 1, we divide $[1, 2\varepsilon^{-1})$ into $O(\log(1/\varepsilon))$ many intervals $[m, 2m)$ where m are powers of 2, and use Lemma 3.5 to obtain profit functions achieving $m\varepsilon$-additive approximation up to $2m$. Then, taking their pointwise minima yields an $(1 - O(\varepsilon))$ approximation. □

In the following sections, we will prove Lemma 3.6.

3.3 Approximation using Δ-multiples of small set Δ

We first introduce several additional tools borrowed from previous works that will be used in our final proof of Lemma 3.6.

Following [Cha18]'s terminology, we say a monotone step function is *p-uniform* if its function values are $0, p, 2p, \ldots, lp$ for some l. A p-uniform function is said to be *pseudo-concave*, if the differences of consecutive x-breakpoints are nondecreasing from left to right. An example of a p-uniform and pseudo-concave function is the profit function f_I of a set I of items with the same profit $p_i = p$, which can be exactly computed by simple greedy: the function f_I takes values $0, p, 2p, \ldots, np$, with x-breakpoints $w_1, w_1 + w_2, \ldots, w_1 + \cdots + w_n$, where w_i's are sorted in nondecreasing order.

As in [Cha18] and [Jin19], we will use the method of approximation via Δ-multiples. For a set Δ of numbers, we say that p is a Δ-multiple if it is a multiple of δ for some $\delta \in \Delta$. Chan [Cha18] used the SMAWK algorithm [AKM+87] and suitable rounding to prove the following lemma:

Lemma 3.10 ([Cha18, Lemma 5]). *Let $f_1, \ldots, f_m$ be monotone step functions with ranges contained in $[0, B]$. Let $\Delta \subset [\delta, 8\delta]$. If every f_i is p_i-uniform and pseudo-concave for some $p_i \in [1, 2]$ which is a Δ-multiple, then we can compute a monotone step function that $O(|\Delta|\delta)$-additively approximates $\min\{f_1 \oplus \cdots \oplus f_m, B\}$ in $\tilde{O}(Bm/\delta)$ time.*

Chan [Cha18] gave a construction of a small set Δ such that every real number in $[1, 2]$ can be approximated by a Δ-multiple. Here, we present a more simplified construction.

Lemma 3.11. *For parameters $0 < \varepsilon < \delta < 1/2$, let $r = \lceil \log_{1+\varepsilon}(1 + 2\delta) \rceil = O(\delta/\varepsilon)$, and define $a_i = \delta(1 + \varepsilon)^i$ for $0 \le i \le r + 1$. Let $\Delta = \{a_i\}$ be the set of a_i. Then for any $t \in [1, 2]$, there is a multiple of some a_i in the range $[t, t + 2\varepsilon]$. Thus, every real number in $[1, 2]$ can be approximated by a Δ multiple with $O(\varepsilon)$ additive error, where $|\Delta| = r + 2 = O(\delta/\varepsilon)$ and all elements in Δ are within $[\delta, 8\delta]$.*

Proof. Let c be the largest integer such that $(t+2\varepsilon)/c \geq a_0$. Since c is largest, $c+1 \geq (t+2\varepsilon)/a_0 \geq 1/\delta$, so $(c+1)/c \leq (1/\delta)/(1/\delta - 1) = 1 + \delta/(1-\delta) \leq 1 + 2\delta \leq a_{r+1}/a_0$. Since $(t+2\varepsilon)/(c+1) < a_0$, we know $(t+2\varepsilon)/c < a_{r+1}$. Let k be the largest integer in $[0, r]$ such that $a_k \leq (t+2\varepsilon)/c$. Note $a_{k+1} > (t+2\varepsilon)/c$, so $a_k = a_{k+1}/(1+\varepsilon) \geq t/c$ using the fact that $t \leq 2$. As a result, $a_k \in [t/c, (t+2\varepsilon)/c]$, thus $ca_k \in [t, t+2\varepsilon]$. $\qquad\square$

3.4 Random Partitioning

Assume that $n < 1/\varepsilon$. In the section, we will use random partitioning to prove Lemma 3.6, restated below.

Lemma 3.6. *Given a list I of $n = O(1/\varepsilon)$ items with p_i being multiples of ε in interval $[1, 2)$, one can compute in $\tilde{O}(n^{4/5}\varepsilon^{-7/5}/2^{\Omega(\sqrt{\log(1/\varepsilon)})})$ time a profit function that $(n\varepsilon)$-additively approximates f_I.*

Proof. Set $\Delta_1 = \Theta(\sqrt{n})$ and $\Delta_0 = \Theta(n^{\frac{7}{10}}\varepsilon^{\frac{2}{5}}2^{c\sqrt{\log(1/\varepsilon)}})$ for some small constant $c > 0$. Assume that Δ_0 is a power of 2 without loss of generality. Note that $\Delta_0 = O(\Delta_1)$, which follows from $n = O(1/\varepsilon)$.

Claim 3.12. *We can partition elements of I into $\Theta(\Delta_1)$ groups $G_1, G_2, \ldots, G_k$, each of size $O(n/\Delta_1)$, while all elements within group G_i are $(1+\varepsilon)$-approximated by multiples of p_i for some $p_i = \Theta(\Delta_1\varepsilon)$.*

Proof. In Lemma 3.11, plugging in $\delta = \varepsilon\Delta_1$, we obtain a set A of size $O(\Delta_1)$ whose elements are of order $\Theta(\Delta_1\varepsilon)$, and each item in I can be $(1+\varepsilon)$-approximated by A-multiples.

We group the elements in I by their divisor in A. We then evenly split groups with size more than n/Δ_1 into two until all groups have sizes of at most n/Δ_1. $\qquad\square$

From now on, assume that $G_1, G_2, \ldots, G_k$ are groups satisfying conditions in Claim 3.12.

We now randomly partition $\{1, 2, \ldots, k\}$ into Δ_0 parts, $I_1, \ldots, I_{\Delta_0}$, by assigning each $1 \leq i \leq k$ into some $I_j (1 \leq j \leq \Delta_0)$ independently and uniformly. Then, set $X_j = \bigcup_{i \in I_j} G_i$ for every $1 \leq j \leq \Delta_0$. It is easy to see $\{X_j\}$ is a partition of I.

Claim 3.13. *With probability $\geq 3/4$, $|I_j| = O(\Delta_1/\Delta_0)$, and hence $|X_j| \leq O(n/\Delta_0)$.*

Proof. By Chernoff bound[5], for some large constant $c > 0$, $|I_j| \geq ck/\Delta_0$ happens with probability at most $1/(4\Delta_0)$. Thus $|I_j| = O(\Delta_1/\Delta_0)$ holds for all j with probability $\geq 3/4$ by union bound. By Claim 3.12, $|X_j| \leq |I_j|O(n/\Delta_1) = O(m/\Delta_0)$. $\qquad\square$

Now assume the event in Claim 3.13 happens.

Claim 3.14. *We can approximate $\bigoplus_{x \in X_j} f_x$ with additive error $O(n\varepsilon/\Delta_0)$ for all $1 \leq j \leq \Delta_0$ in $\tilde{O}(n^2\varepsilon^{-1}/(\Delta_0\Delta_1)) = \tilde{O}(n^{4/5}\varepsilon^{-7/5}/2^{\Omega(\sqrt{\log(1/\varepsilon)})})$ time.*

---

[5]For independent random variables $x_1, \ldots, x_n \in \{0, 1\}$ and $\delta > 0, 0 \leq w_1, \ldots, w_n \leq 1$, let $X = \sum_{i=1}^n w_i x_i$ and $\mu = \mathbb{E}[x]$, then $\mathbf{Pr}[|x - \mu| \geq \delta\mu] \leq 2e^{-\delta^2\mu/3}$.

Proof. Fix a single j. By Claim 3.13, $\bigoplus_{x \in X_j} f_x$ is the convolution of $O(n/\Delta_0)$ elements, each being a multiple of order $\Theta(\Delta_1 \varepsilon)$. By applying Lemma 3.10 with parameters $B = O(n/\Delta_0), \delta = \Theta(\Delta_1 \varepsilon), |\Delta| = |I_j| = O(\Delta_1/\Delta_0)$, we can approximate $\bigoplus_{x \in X_j} f_x$ with additive error $O(\Delta_1^2 \varepsilon/\Delta_0) = O(n\varepsilon/\Delta_0)$ within time $\tilde{O}((n/\Delta_0)^2/(\Delta_1 \varepsilon))$.

We can do so for all $1 \le j \le \Delta_0$, with running time $\tilde{O}(n^2/(\Delta_0 \Delta_1 \varepsilon)) = \tilde{O}(n^{4/5} \varepsilon^{-7/5}/2^{\Omega(\sqrt{\log(1/\varepsilon)})})$. $\square$

Now we can replace $\bigoplus_{x \in X_j} f_x$ by the approximation obtained in Claim 3.14, since the total additive error inflicted will be $O(n\varepsilon/\Delta_0)\Delta_0 = O(n\varepsilon)$.

We use divide and conquer to combine the answer of $\bigoplus_{x \in X_j} f_x$. The merge process can be viewed as a complete binary tree with Δ_0 leaves. For $S \subseteq \{1, 2, \ldots, \Delta_0\}$, define $F(S) = \bigoplus_{x \in \cup_{(s \in S)} X_s} f_x$. Claim 3.14 allows us to approximate $F(S)$ for all $|S| = 1$. Now we have the following claim regarding combining two subtrees S_1 and S_2.

Claim 3.15. *Assume $i \le \log_2 \Delta_0$ and $|S_1| = |S_2| = 2^i$, where $S_1, S_2 \subseteq \{1, 2, \ldots, \Delta_0\}$ and $S_1 \cap S_2 = \emptyset$. Assume that A_1 is an approximation of $F(S_1)$ with additive error err_1, A_2 is an approximation of $F(S_2)$ with additive error err_2. Then with probability $\ge 1 - 1/(5\Delta_0)$, we can compute an approximation of $F(S_1 \cup S_2)$ with additive error $err_1 + err_2 + O(2^{0.9i} n\varepsilon/\Delta_0)$ in time $O(\varepsilon^{-2}\Delta_0^{0.5}/(\Delta_1^{0.5} 2^{\Theta(\sqrt{\log(1/\varepsilon)})}))$.*

Proof. Define $\delta_i = 2^{0.9i} n\varepsilon/\Delta_0$. A naive way to approximate $F(S_1 \cup S_2)$ is to round each value in A_1 and A_2 to a multiple of δ_i, and then invoke the $(\min, +)$ convolution as in Lemma 3.4. In the following we will show a better method exploiting the fact that $\{X_i\}$ is a random partition.

Let the global optimal solution be to choose the subset T of items. Define $H_1 = \bigcup_{i \in S_1} X_i, H_2 = \bigcup_{i \in S_2} X_i$. Note that the groups $G_1, \ldots, G_k$ are assigned into $X_1, \ldots, X_{\Delta_0}$ uniformly randomly. Pick $u = Cn\sqrt{2^i/(\Delta_1 \Delta_0)}\log n$ for a large constant $C > 0$. By Chernoff bound, the probability that $\mathbf{Pr}(|\sum_{x \in T \cap H_1} x - \sum_{x \in T \cap H_2} x| \ge u) \le 1/(5n)$.[6]

Now assume that $|\sum_{x \in T \cap H_1} x - \sum_{x \in T \cap H_2} x| \le u$, and we show how to approximate $F(S_1 \cup S_2)$ under the assumption. During the $(\min, +)$ convolution, we first round the values of $F(S_1), F(S_2)$ to multiples of δ_i. Then we only need to consider the pairs that differ in value by at most u. We then divide the arrays into blocks with values within a difference of u from each other, and do $(\min, +)$-convolution between the pairs of blocks with indices differing by at most 1. The block sizes are at most $u/\delta_i = \tilde{O}(\varepsilon^{-1}\Delta_0^{0.5}\Delta_1^{-0.5}/2^{0.4i})$, so the running time for each $(\min, +)$-convolution is $O(\varepsilon^{-2}\Delta_0/(\Delta_1 2^{0.8i} 2^{\Omega(\sqrt{\log(1/\varepsilon)})}))$ using Williams's $O(n^2/2^{\Omega(\sqrt{\log(1/\varepsilon)})})$-time algorithm for length-n $(\min, +)$-convolution [Wil14]. Since the value in the merged answer is bounded by $\tilde{O}(2^i n/\Delta_0)$ by Claim 3.13, there are $\tilde{O}(2^i n/(\Delta_0 u))$ min-plus convolutions in total, with total complexity $\tilde{O}(\varepsilon^{-2} 2^i n/(\Delta_1 2^{0.8i} 2^{\Omega(\sqrt{\log(1/\varepsilon)})} u)) = O(\varepsilon^{-2}\Delta_0^{0.5}/(\Delta_1^{0.5} 2^{\Theta(\sqrt{\log(1/\varepsilon)})}))$. $\square$

Now we conclude the proof by applying Claim 3.15 to the divide and conquer process. Assume all the $\le \Delta_0$ many calls to Claim 3.15 yield correct approximations, which happens with success

---

[6]We apply Chernoff bound with $w_j = R_j/(2n/\Delta_1)$ where $R_j = \sum_{x \in T \cap G_j} x$. Now consider S_1. We set $x_j = 1$ if $j \in \cup_{t \in S_1} I_t$ and $x_j = 0$ otherwise. Since the partition $\{I_t\}_{1 \le t \le \Delta_0}$ is random, the expected value of $\sum_{j=1}^k w_j x_j$ will be $\Theta(\Delta_1 2^i/\Delta_0)$. From Chernoff bound, this value will be $u/(4n/\Delta_1)$ away from expected value with probability $\le 2e^{\Theta(-(u/(4n/\Delta_1))^2/(\Delta_1 2^i/\Delta_0))} \le 1/(10n)$. By union bound, both $\sum_{x \in T \cap H_2} x$ and $\sum_{x \in T \cap H_1} x$ will be within difference $u/2$ from the expected value with probability $\ge 1 - 1/(5n)$, in which case their difference will be bounded by u.

probability $\geq 3/4$ by union bound.

To analyze the error term, note that there are $O(\Delta_0/2^i)$ merges of two subtrees with 2^i parts each, where Claim 3.15 inflicts additive error $O(2^{0.9i} n\varepsilon/\Delta_0)$ for each such a merge. Thus the total additive error is bounded by $\sum_{2^i \leq \Delta_0} (2^{0.9i} n\varepsilon/\Delta_0)(\Delta_0/2^i) = O(n\varepsilon)$.

Now we analyze the time complexity. Note that the total complexity for the i-th layer is

$$O((\Delta_0/2^i) \cdot \varepsilon^{-2}\Delta_0^{0.5}\Delta_1^{-0.5}/2^{\Omega(\sqrt{\log(1/\varepsilon)})})$$
$$= O(\varepsilon^{-2}\Delta_0^{1.5}\Delta_1^{-0.5}/2^{\Omega(\sqrt{\log(1/\varepsilon)})})$$
$$= O(n^{\frac{4}{5}}\varepsilon^{-\frac{7}{5}}/2^{\Omega(\sqrt{\log(1/\varepsilon)})}).$$

As there are logarithmically many layers, the total complexity for the divide and conquer part is $O(n^{\frac{4}{5}}\varepsilon^{-\frac{7}{5}}/2^{\Omega(\sqrt{\log(1/\varepsilon)})})$.

Thus our total complexity is $O(n^{\frac{4}{5}}\varepsilon^{-\frac{7}{5}}/2^{\Omega(\sqrt{\log(1/\varepsilon)})})$, and with a success probability of $\geq 1/2$ (which can be amplified by repetition) by union bound.

A small detail is that when n is so small that $n^{7/10} < \varepsilon^{-2/5}$, $\Delta_0 < 1$ and our reasoning falls apart. In such cases, one can simply set $\Delta_0 = 1$ and the running time still holds. □

4 Approximating Partition

In this section, we will solve the following problem.

Problem 2. *Assume $\varepsilon \in (0, 1/2)$ and $1/\varepsilon \in \mathbb{N}^+$. Given a set X of n distinct integers in the interval $[1/\varepsilon, 2/\varepsilon)$, compute a set $A \subset \mathbb{N}$ that n-additively approximates $\mathcal{S}(X)$.*

By a tedious reduction that is heavily based on known techniques, one can show the following.

Lemma 4.1. *If for some $c \geq 1$, Problem 2 can be solved in $\tilde{O}(n + 1/\varepsilon^c)$ time, then $(1-\varepsilon)$-approximating Partition can also be solved in $\tilde{O}(n + 1/\varepsilon^c)$ time.*

Lemma 4.1 will be proved in the appendix.

Now we proceed to describe our main algorithm for solving Problem 2.

In the following lemma, we merge the approximations of $\mathcal{S}(X_1), \mathcal{S}(X_2)$ and obtain an approximation of $\mathcal{S}(X_1 \uplus X_2)$. When X_1, X_2 come from a short interval $[\ell, \ell + d]$, we can use densification via 2D FFT to obtain a speedup over the straightforward algorithm.

Lemma 4.2. *Let $\delta \in (0, 1/2)$, and $\ell, d, t, \Delta \in \mathbb{N}^+$ such that $d \leq \ell \leq t$.*

Let $X_1, X_2 \subseteq \mathbb{N}^+ \cap [\ell, \ell + d]$ be two integer sets. Given $A_1, A_2 \subset \mathbb{N}$ as input where for $i \in \{1, 2\}$, A_i is an $(1 - \delta)$ approximation of $\mathcal{S}(X_i)$ up to t, one can compute a set $A \subset \mathbb{N}^+$ of size $|A| \leq Z$ that $(1 - \delta, \Delta - 1)$-approximates $\mathcal{S}(X_1 \uplus X_2)$ up to t, in $\tilde{O}(Z + |A_1| + |A_2|)$ time, where

$$Z \leq O\left(\min\left\{\left\lceil \frac{t}{\Delta}\right\rceil, \ \frac{t}{\ell} \cdot \left\lceil\frac{td}{\ell\Delta}\right\rceil\right\}\right).$$

Proof. Let $\bar{\Delta} := \lceil \Delta/2 \rceil$. We will run one of the following two algorithms that minimizes Z.

14

Algorithm 1 (1D FFT). For $i \in \{1, 2\}$, by rounding the integers in A_i down to multiples of $\bar{\Delta}$, we obtain set $A_i' \subset \bar{\Delta} \cdot \mathbb{N}$ that $(\bar{\Delta} - 1)$-additively approximates A_i. Then, since $A_i' \subseteq [0, t]$, their sumset $A_1' + A_2'$ can be computed by FFT in $\tilde{O}(\lceil t/\bar{\Delta} \rceil) \leq \tilde{O}(\lceil t/\Delta \rceil)$ time. Note that $A := A_1' + A_2'$ approximates $A_1 + A_2$ with additive error at most $2(\bar{\Delta} - 1) \leq \Delta - 1$, so A is a $(1 - \delta, \Delta - 1)$-approximation of $\mathcal{S}(X_1 \uplus X_2)$ up to t, due to Proposition 2.4.

Algorithm 2 (Densification with 2D FFT). For every $a \in A_i$, there exists $s \in \mathcal{S}(X_i; t)$ such that $0 \leq s - a \leq s\delta$. Note that s is the sum of at most t/ℓ many integers from $[\ell, \ell + d]$, so s can be expressed as $s = k\ell + b'$ for some $k \in \mathbb{N} \cap [0, t/\ell]$ and $0 \leq b' \leq dt/\ell$. Hence, $a \in A_i$ can be expressed as $a = k\ell + b$ for some $k \in \mathbb{N} \cap [0, t/\ell]$ and $-s\delta \leq b \leq dt/\ell$. Then, by rounding b down to integer multiples of $\bar{\Delta}$, we obtain $A_i' \subset \mathbb{N}$ that $(\bar{\Delta} - 1)$-additively approximates A_i, such that every $a' \in A_i'$ can be expressed as

$$a' = k\ell + j\bar{\Delta},$$

for some $k \in \mathbb{N} \cap [0, t/\ell]$ and $j \in \mathbb{Z} \cap [-1 - s\delta/\bar{\Delta}, dt/(\ell\bar{\Delta})]$. Using this 2-dimensional (k, j) representation of A_i', we can compute $A_1' + A_2'$ using 2D FFT (Lemma 2.1): the first dimension has size $O(t/\ell)$, and the second dimension has size at most

$$dt/(\ell\bar{\Delta}) + s\delta/\bar{\Delta} + O(1) \leq O\left(\left\lceil \frac{td}{\ell\Delta} \right\rceil\right),$$

where the inequality follows from $s \leq t$ and an assumption

$$\delta \leq O(d/\ell), \tag{5}$$

which will be justified later. Hence, the running time of this 2D FFT is

$$\tilde{O}\left(\frac{t}{\ell} \cdot \left\lceil \frac{td}{\ell\Delta} \right\rceil\right).$$

Similarly to Algorithm 1, one also can show that in this case $A := A_1' + A_2'$ is a $(1 - \delta, \Delta - 1)$-approximation of $\mathcal{S}(X_1 \uplus X_2)$ up to t.

To justify assumption (5), observe that if $\delta \geq d/(\ell + d)$ holds instead, or equivalently, $(1 - \delta)(\ell + d) \leq \ell$, then one can round every integer in $X_1, X_2 \subset [\ell, \ell + d]$ down to exactly ℓ while still ensuring $(1 - \delta)$ approximation, and hence immediately obtain an $A \subset \ell \cdot \mathbb{N}$ of size $|A| \leq \lceil t/\ell \rceil$ that $(1 - \delta)$-approximates $\mathcal{S}(X_1 \uplus X_2)$ up to t. $\square$

We then apply Lemma 4.2 with scaling, and obtain the following lemma that has purely multiplicative approximation.

Lemma 4.3. *Let $\delta, \delta_0 \in (0, 1/2)$, and $\ell, d, T \in \mathbb{N}^+$ such that $d \leq \ell \leq T$.*
Let $X_1, X_2 \subseteq \mathbb{N}^+ \cap [\ell, \ell + d]$ be two integer sets. Given $A_1, A_2 \subset \mathbb{N}$ as input where for $i \in \{1, 2\}$, A_i is an $(1 - \delta)$ approximation of $\mathcal{S}(X_i)$ up to T, one can compute a set $A \subset \mathbb{N}^+$ of size $|A| \leq Z$ that $(1 - \delta - \delta_0)$-approximates $\mathcal{S}(X_1 \uplus X_2)$ up to T, in $\tilde{O}\big(Z + (|A_1| + |A_2|) \log(2T/\ell)\big)$ time, where

$$Z \leq O\left(\min\left\{\frac{\log(2T/\ell)}{\delta_0}, \frac{T}{\ell} \cdot \left\lceil \frac{d}{\ell\delta_0} \right\rceil\right\}\right).$$

Proof. Initialize set $A = \{0\}$. We iterate over all r being integer powers of 2 such that $\ell/6 \le r \le T$. For each r, apply Lemma 4.2 to A_1 and A_2 with $t := 6r$ and $\Delta := \lceil \delta_0 r \rceil$, and obtain a set $A_r \subseteq \mathbb{N} \cap [0, 6r]$ that $(1 - \delta, \lceil \delta_0 r \rceil - 1)$-approximates $\mathcal{S}(X_1 \uplus X_2)$ up to $6r$. We then insert all elements in $A_r \cap [r, 6r]$ into A. We will show that eventually A is a $(1 - \delta_0 - \delta)$-approximation of $\mathcal{S}(X_1 \uplus X_2)$ up to T.

Observe that for every $a \in A_r \cap [r, 6r]$, there exists $s \in \mathcal{S}(X_1 \uplus X_2)$ such that $a \le s$ and

$$
\begin{aligned}
a &\ge (1 - \delta)s - (\lceil \delta_0 r \rceil - 1) \\
&> (1 - \delta)s - \delta_0 r \\
&\ge (1 - \delta - \delta_0)s,
\end{aligned}
$$

where the last step follows from $s \ge a \ge r$.

Conversely, for every positive $s \in \mathcal{S}(X_1 \uplus X_2; T)$ (which must satisfy $\ell \le s \le T$), let r be a power of two such that $3r \le s \le 6r$. Then there exists $a \in A_r$ such that $a \le s \le 6r$ and

$$
\begin{aligned}
a &\ge (1 - \delta)s - (\lceil \delta_0 r \rceil - 1) \\
&\ge (1 - \delta)s - \delta_0 r \\
&\ge s/2 - r/2 \\
&\ge r,
\end{aligned}
$$

so $a \in A_r \cap [r, 6r]$ and hence will be included in A, and similarly as before we have $a \ge (1 - \delta_0 - \delta)s$. Hence, we have established that A is a $(1 - \delta_0 - \delta)$-approximation of $\mathcal{S}(X_1 \uplus X_2)$ up to T.

It remains to bound the total running time and the size of A. There are $O(\log(2T/L))$ many iterations of r, where for each $r \in [\ell/6, T]$ with $t := 6r$ and $\Delta := \lceil \delta_0 r \rceil$, Lemma 4.2 gives the upper bound

$$
\begin{aligned}
Z_r &\le O\left(\min\left\{\left\lceil \frac{t}{\Delta} \right\rceil, \frac{t}{\ell} \cdot \left\lceil \frac{td}{\ell \Delta} \right\rceil\right\}\right) \\
&\le O\left(\min\left\{\left\lceil \frac{r}{\delta_0 r} \right\rceil, \frac{r}{\ell} \cdot \left\lceil \frac{rd}{\ell \delta_0 r} \right\rceil\right\}\right) \\
&\le O\left(\min\left\{\left\lceil \frac{1}{\delta_0} \right\rceil, \frac{r}{\ell} \cdot \left\lceil \frac{d}{\ell \delta_0} \right\rceil\right\}\right).
\end{aligned}
$$

Hence, summing over all powers of two in the range $[\ell/6, T]$, we have

$$
Z \le \sum_r Z_r \le O\left(\min\left\{\frac{\log(2T/\ell)}{\delta_0}, \frac{T}{\ell} \cdot \left\lceil \frac{d}{\ell \delta_0} \right\rceil\right\}\right). \qquad \square
$$

Lemma 4.3 implies the following immediate corollary by dropping the upper bound T.

Corollary 4.4. *Let $\delta, \delta_0 \in (0, 1/2)$, and $\ell, d \in \mathbb{N}^+$ such that $d \le \ell$.*

Let $X_1, X_2 \subseteq \mathbb{N}^+ \cap [\ell, \ell + d]$ be two integer sets of total size $|X_1| + |X_2| = n$. Given $A_1, A_2 \subset \mathbb{N}$ as input where for $i \in \{1, 2\}$, A_i is an $(1 - \delta)$ approximation of $\mathcal{S}(X_i)$, one can compute a set $A \subset \mathbb{N}^+$ of size $|A| \le Z$ that $(1 - \delta_0 - \delta)$-approximates $\mathcal{S}(X_1 \uplus X_2)$, in $\tilde{O}\big(Z + (|A_1| + |A_2|) \log n\big)$ time, where

$$
Z \le O\left(\min\left\{\frac{1}{\delta_0}, \frac{nd}{\ell \delta_0} + n\right\} \cdot \log n\right).
$$

16

Proof. Immediately follows from Lemma 4.3 by setting $T = n \cdot (\ell + d)$, which is an upper bound on the largest element of $\mathcal{S}(X_1 \uplus X_2)$. $\quad\square$

Now, we apply Corollary 4.4 in a divide-and-conquer fashion, to approximate the subset sums of $X \subseteq \mathbb{N}^+ \cap [\ell, 2\ell]$.

Lemma 4.5. *Let $\delta \in (0, 1/2)$ and $\ell \in \mathbb{N}^+$.*

Given an integer set $X \subseteq \mathbb{N}^+ \cap [\ell, 2\ell]$ of n integers, one can compute a set $A \subset \mathbb{N}^+$ that $(1 - \delta)$-approximates $\mathcal{S}(X)$, in $\tilde{O}(n + \sqrt{n}/\delta)$ time.

Proof. Let $X = \{x_1, x_2, \ldots, x_n\}$ where $\ell \leq x_1 < x_2 < \cdots < x_n \leq 2\ell$. Set $\delta_0 := \delta/\lceil \log_2 n \rceil$.

We will use a divide-and-conquer approach to merge the items of X using Corollary 4.4. Build a balanced binary tree with n leaf nodes representing the items $x_1, \ldots, x_n$ from left to right. At each internal node representing $x_{[l..r]}$, we use Corollary 4.4 to merge the results of the two child nodes (representing $x_{[l..m]}$ and $x_{[m+1..r]}$ respectively, where $m = \lfloor (l+r)/2 \rfloor$), and obtain an approximation of $\mathcal{S}(\{x_l, x_{l+1}, \ldots, x_r\})$. Finally we obtain an approximation of $\mathcal{S}(X)$ at the root node.

The binary tree has $\lceil \log_2 n \rceil$ levels, where each level of applying Corollary 4.4 worsens the approximation factor by δ_0. Hence, the overall approximation factor of $\mathcal{S}(X)$ is $1 - \delta_0 \cdot \lceil \log_2 n \rceil = 1 - \delta$ as required.

It remains to bound the total running time of all invocations of Corollary 4.4. Note that in each invocation, the $(|A_1| + |A_2|) \log n$ summand in the stated time complexity is always absorbed (up to $\log n$ factors) by the output sizes of the two child nodes, which are in turn bounded by the running times of these two child invocations. So it suffices to bound the sum of the Z quantity stated in Corollary 4.4.

We separately bound for each level of the binary tree. At the i-th level ($0 \leq i < \lceil \log_2 n \rceil$), there are at most $m_i = 2^i$ invocations of Corollary 4.4, where each invocation involves at most $n_i = \lceil n/2^i \rceil$ items in X. Note that $n_i m_i \leq 2n$. Suppose these m_i invocations involve $x_{[1..k_1]}, x_{[k_1+1..k_2]}, \ldots,$ $x_{[k_{m_i-1}+1..n]}$ respectively. Then the j-th invocation has d value (stated in Corollary 4.4) at most $d_j \leq x_{k_j} - x_{k_{j-1}}$. Hence, the sum of these d values is at most

$$\sum_{j=1}^{m_i} d_j \leq \sum_{j=1}^{m_i} (x_{k_j} - x_{k_{j-1}}) \leq x_n - x_1 \leq \ell. \tag{6}$$

Now we are ready to bound the sum of the Z quantity over the m_i invocations at level i ($0 \leq i < \lceil \log_2 n \rceil$). We consider two cases.

- **Case 1**: $n_i \leq \sqrt{n}$.

  Then, by Corollary 4.4,

$$\begin{aligned}
\sum_{j=1}^{m_i} Z_j &\leq \sum_{j=1}^{m_i} \left(\frac{n_i d_j}{\ell \delta_0} + n_i \right) \cdot \log n \\
&= \left(\frac{n_i \sum_{j=1}^{m_i} d_j}{\ell \delta_0} + m_i n_i \right) \cdot \log n \\
&\leq \left(\frac{n_i}{\delta_0} + m_i n_i \right) \cdot \log n \qquad \text{(by (6))} \\
&\leq \tilde{O}\left(\frac{\sqrt{n}}{\delta_0} + n \right).
\end{aligned}$$

17

- **Case 2:** $n_i > \sqrt{n}$.

  Then, $m_i \leq 2n/n_i < 2\sqrt{n}$. By Corollary 4.4,

$$\sum_{j=1}^{m_i} Z_j \leq m_i \cdot \frac{1}{\delta_0} \cdot \log n$$

$$\leq \tilde{O}(\sqrt{n}/\delta_0).$$

Hence, in either case we have $\sum_{j=1}^{m_i} Z_j \leq \tilde{O}(n + \sqrt{n}/\delta)$. Hence, the total running time over all levels $0 \leq i < \lceil \log_2 n \rceil$ is also $\tilde{O}(n + \sqrt{n}/\delta)$. □

Finally, we solve Problem 2 by combining Lemma 4.5 with the additive combinatorics results of [GM91, BW21].

Lemma 4.6. *We can solve Problem 2 in $\tilde{O}\left(n + \min\{\varepsilon^{-1}n^{1/2}, \ \varepsilon^{-1} + \varepsilon^{-2}/n^{3/2}\}\right)$ time, which is at most $\tilde{O}(n + 1/\varepsilon^{5/4})$.*

Proof. Recall that in Problem 2, for $\varepsilon > 0$ where $\ell = 1/\varepsilon$ is an integer, we are given a set $X \subseteq \mathbb{N}^+ \cap [\ell, 2\ell]$ of n distinct integers, and need to compute a set $A \subset \mathbb{N}$ that n-additively approximates $\mathcal{S}(X)$.

We will choose to run one of the following two algorithms depending on the parameters.

Algorithm 1. Directly apply Lemma 4.5 with $\delta := \varepsilon$, in $\tilde{O}(n + \sqrt{n}/\varepsilon)$ time.
    When $n \leq \tilde{O}(1/\varepsilon^{1/2})$, the running time of Algorithm 1 is $\tilde{O}(n + 1/\varepsilon^{5/4})$.

Algorithm 2. Let $\sigma = \Sigma(X)$, and let λ be the threshold value from Theorem 2.7 satisfying $\lambda = \tilde{\Theta}(\ell^2/n)$. The following algorithm applies when $\lambda \leq \sigma/2$, which holds in particular when $1/\varepsilon \ll n^2$.

Initialize $A = \emptyset$. We set $\delta := n/(n + \lambda)$, and apply Lemma 4.5 in $\tilde{O}(n + \sqrt{n}/\delta)$ time to compute a set A_δ that $(1 - \delta)$-approximates $\mathcal{S}(X)$. Observe that $A_\delta \cap [0, \lambda]$ is an n-additive approximation of $\mathcal{S}(X)$ up to λ. Hence, we insert all elements in $A_\delta \cap [0, \lambda]$ to A.

Then, using the data structure from Lemma 2.7, we compute an n-additive approximation of $\mathcal{S}(X) \cap [\lambda, \sigma/2]$ and insert them into A. To do this, we start from the left endpoint λ of the interval $[\lambda, \sigma/2]$, and each time use binary search (implementable using range queries supported by Lemma 2.7) to find the next subset sum in the interval, and then jump n steps to the right since we allow an additive error of n. The time complexity is $O(\lceil \frac{\sigma/2 - \lambda}{n} \rceil \cdot \log \sigma) \leq \tilde{O}(\ell)$.

Now we have constructed A that n-additively approximates $\mathcal{S}(X)$ up to $\sigma/2$. Using the simple fact that $t \in \Sigma(X)$ if and only if $\sigma - t \in \Sigma(X)$, we can symmetrically use A to obtain an approximation of the remaining half. Specifically, letting $A' := \{\sigma - a - n : a \in A\}$, it is straightforward to verify that $A \cup A'$ is an n-additive approximation of $\mathcal{S}(X)$ (up to σ). So we return $A \cup A'$.

The overall time complexity of Algorithm 2 is

$$\tilde{O}(n + \ell + \sqrt{n}/\delta) \leq \tilde{O}\left(n + 1/\varepsilon + \frac{\sqrt{n}(n + \lambda)}{n}\right)$$

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

When $n \gg 1/\varepsilon^{1/2}$, the running time is $\tilde{O}(n + 1/\varepsilon^{5/4})$. □

Combined with the reduction in Lemma 4.1, this proves our main Theorem 1.2.

References

[ABHS19] Amir Abboud, Karl Bringmann, Danny Hermelin, and Dvir Shabtay. SETH-based lower bounds for subset sum and bicriteria path. In *Proceedings of the 30th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 41–57, 2019. doi:10.1137/1.9781611975482.3. 1

[AKM+87] Alok Aggarwal, Maria M. Klawe, Shlomo Moran, Peter Shor, and Robert Wilber. Geometric applications of a matrix-searching algorithm. *Algorithmica*, 2(1):195–208, November 1987. doi:10.1007/BF01840359. 11

[BC22] 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, July 4-8, 2022, Paris, France*, volume 229 of *LIPIcs*, pages 31:1–31:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.ICALP.2022.31. 1

[BCD+14] David Bremner, Timothy M. Chan, Erik D. Demaine, Jeff Erickson, Ferran Hurtado, John Iacono, Stefan Langerman, Mihai Pătraşcu, and Perouz Taslakian. Necklaces, convolutions, and x+y. *Algorithmica*, 69(2):294–314, June 2014. doi:10.1007/s00453-012-9734-3. 7

[Bla10] Richard E Blahut. *Fast algorithms for signal processing*. Cambridge University Press, 2010. 4

[BN21] Karl Bringmann and Vasileios Nakos. A fine-grained perspective on approximating subset sum and partition. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021*, pages 1797–1815. SIAM, 2021. doi:10.1137/1.9781611976465.108. 1, 3, 5

[BW21] 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, Virtual Conference, January 10 - 13, 2021*, pages 1777–1796. SIAM, 2021. doi:10.1137/1.9781611976465.107. 2, 6, 18, 22

[Cha18] Timothy M. Chan. Approximation Schemes for 0-1 Knapsack. In *Proceedings of the 1st Symposium on Simplicity in Algorithms (SOSA)*, pages 5:1–5:12, 2018. doi:10.4230/OASIcs.SOSA.2018.5. 1, 2, 3, 5, 6, 7, 11

[CMWW19] Marek Cygan, Marcin Mucha, Karol Węgrzycki, and Michał Włodarczyk. On problems equivalent to (min,+)-convolution. *ACM Trans. Algorithms*, 15(1):14:1–14:25, January 2019. doi:10.1145/3293465. 1

[CW16] Timothy M. Chan and Ryan Williams. Deterministic APSP, orthogonal vectors, and more: Quickly derandomizing Razborov-Smolensky. In *Proceedings of the 27th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1246–1255, 2016. doi:10.1137/1.9781611974331.ch87. 7

[GL79]   George Gens and Eugene Levner. Computational complexity of approximation algorithms for combinatorial problems. In *Mathematical Foundations of Computer Science 1979, Proceedings, 8th Symposium, Olomouc, Czechoslovakia, September 3-7, 1979*, volume 74 of *Lecture Notes in Computer Science*, pages 292–300. Springer, 1979. `doi:10.1007/3-540-09526-8\_26`. 1

[GM91]   Zvi Galil and Oded Margalit. An almost linear-time algorithm for the dense subset-sum problem. *SIAM J. Comput.*, 20(6):1157–1189, 1991. `doi:10.1137/0220072`. 2, 3, 6, 18, 22

[IK75]   Oscar H. Ibarra and Chul E. Kim. Fast approximation algorithms for the knapsack and sum of subset problems. *Journal of the ACM (JACM)*, 22(4):463–468, October 1975. `doi:10.1145/321906.321909`. 1

[Jin19]   Ce Jin. An improved FPTAS for 0-1 knapsack. In *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019, July 9-12, 2019, Patras, Greece*, volume 132 of *LIPIcs*, pages 76:1–76:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. `doi:10.4230/LIPIcs.ICALP.2019.76`. 1, 2, 3, 6, 7, 11

[JK18]   Klaus Jansen and Stefan E.J. Kraft. A faster fptas for the unbounded knapsack problem. *European Journal of Combinatorics*, 68:148 – 174, 2018. `doi:10.1016/j.ejc.2017.07.016`. 1

[Kar72]   Richard M Karp. Reducibility among combinatorial problems. In *Complexity of computer computations*, pages 85–103. Springer, 1972. 1

[KMPS03] Hans Kellerer, Renata Mansini, Ulrich Pferschy, and Maria Grazia Speranza. An efficient fully polynomial approximation scheme for the subset-sum problem. *J. Comput. Syst. Sci.*, 66(2):349–370, 2003. `doi:10.1016/S0022-0000(03)00006-0`. 1

[KP99]   Hans Kellerer and Ulrich Pferschy. A new fully polynomial time approximation scheme for the knapsack problem. *Journal of Combinatorial Optimization*, 3(1):59–71, July 1999. `doi:10.1023/A:1009813105532`. 1

[KP04]   Hans Kellerer and Ulrich Pferschy. Improved dynamic programming in connection with an fptas for the knapsack problem. *Journal of Combinatorial Optimization*, 8(1):5–11, March 2004. `doi:10.1023/B:JOCO.0000021934.29833.6b`. 1

[KPS17]   Marvin Künnemann, Ramamohan Paturi, and Stefan Schneider. On the fine-grained complexity of one-dimensional dynamic programming. In *Proceedings of the 44th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 21:1–21:15, 2017. `doi:10.4230/LIPIcs.ICALP.2017.21`. 1

[KX19]   Konstantinos Koiliaris and Chao Xu. Faster pseudopolynomial time algorithms for subset sum. *ACM Trans. Algorithms*, 15(3):40:1–40:20, June 2019. `doi:10.1145/3329863`. 3, 24

[Law79]   Eugene L. Lawler. Fast approximation algorithms for knapsack problems. *Mathematics of Operations Research*, 4(4):339–356, 1979. `doi:10.1287/moor.4.4.339`. 1, 4

[Lev03]   Vsevolod F Lev. Blocks and progressions in subset sum sets. *ACTA ARITHMETICA-WARSZAWA-*, 106(2):123–142, 2003. 6, 22

[MWW19] Marcin Mucha, Karol Węgrzycki, and Michał Włodarczyk. A subquadratic approximation scheme for partition. In *Proceedings of the 30th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 70–88, 2019. Full version at https://arxiv.org/abs/1804.02269v2. doi:10.1137/1.9781611975482.5. 1, 3, 5, 23, 24

[Rhe15]   Donguk Rhee. Faster fully polynomial approximation schemes for knapsack problems. Master's thesis, Massachusetts Institute of Technology, 2015. URL: http://hdl.handle.net/1721.1/98564. 1

[Sár94]   A. Sárközy. Fine addition theorems, II. *Journal of Number Theory*, 48(2):197–218, 1994. 2, 6, 22

[Wil14]   Ryan Williams. Faster all-pairs shortest paths via circuit complexity. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing (STOC)*, pages 664–673, 2014. doi:10.1145/2591796.2591811. 7, 13

A   Know reductions from the Knapsack Problem

Recall that we defined the following simpler problem.

Problem 1. *Assume $\varepsilon \in (0, 1/2)$ and $1/\varepsilon \in \mathbb{N}^+$. Given a list I of items $(p_1, w_1), \ldots, (p_n, w_n)$ with weights $w_i \in \mathbb{N}$ and profits p_i being multiples of ε in the interval $[1, 2)$, compute a profit function that $(1 - \varepsilon)$-approximates f_I up to $2/\varepsilon$.*

Lemma 3.1. *If for some $c \geq 2$, Problem 1 can be solved in $\tilde{O}(n + 1/\varepsilon^c)$ time, then $(1 - \varepsilon)$-approximating* Knapsack *can also be solved in $\tilde{O}(n + 1/\varepsilon^c)$ time.*

Proof. First, we can reduce ε so that $1/\varepsilon$ becomes an integer.

We will restrict the profit values into small intervals, as follows: divide the items into $O(\log \frac{\max_j p_j}{\min_j p_j}) = O(\log \varepsilon^{-1})$ groups (see Section 2.3), each containing items with $p_i \in [2^j, 2^{j+1}]$ for some j (which can be rescaled to $[1, 2]$). Finally, use the merging lemma Lemma 3.4 to merge the profit functions of all groups, in $\tilde{O}(n + \varepsilon^{-2})$ overall time.

Now, having restricted the profit values into $[1, 2)$, we can round every profit value to a multiple of ε, which incurs only $(1 - O(\varepsilon))$ approximation factor in total.

Finally, the following greedy lemma takes care of the case with total profit above $\Omega(\varepsilon^{-1})$.

Lemma A.1. *Suppose $p_i \in [1, 2]$ for all $i \in I$. For $B = \Omega(\varepsilon^{-1})$, the profit function f_I can be approximated with additive error $O(\varepsilon B)$ in $O(n \log n)$ time.*

Proof. Simply sort the items in nonincreasing order of efficiency p_i/w_i, and define the profit function $\tilde{f}$ resulting from greedy, with function values $0, p_1, p_1 + p_2, \ldots, p_1 + \cdots + p_n$ and x-breakpoints $0, w_1, w_1 + w_2, \ldots, w_1 + \cdots + w_n$. It clearly approximates f_I with an additive error of $\max_i p_i \leq 2 \leq O(\varepsilon B)$ for $B = \Omega(\varepsilon^{-1})$. $\qquad\square$

This greedy approach achieves $(1 - O(\varepsilon))$-approximation for large profit values. Hence, it is sufficient to approximate f_I up to $2/\varepsilon$. $\qquad\square$

B Proof of Lemma 2.6

We need several results on dense subset sums developed by a series of works including [Sár94, Lev03, GM91, BW21]. The following definitions and theorems are from [BW21]. The sets considered here contain *distinct positive integers*.

Definition B.1 (Density). *A set $X \subset \mathbb{N}^+$ is δ-dense if it satisfies $|X|^2 \geq \delta \cdot \max X$.*

Definition B.2 (Almost Divisor). *Let $X(d) := X \cap d\mathbb{Z}$ denote the set of all numbers in X that are divisible by d. Let $\overline{X(d)} := X \setminus X(d)$ denote the set of all numbers in X not divisible by d. We say an integer $d > 1$ is an α-almost divisor of X if $|\overline{X(d)}| \leq \alpha \cdot \Sigma(X)/|X|^2$.*

Theorem B.3 ([BW21, Theorem 4.1]). *Let $\delta \geq 1$ and $\alpha \leq \delta/16$. Given an δ-dense set X of size n, there exists a positive integer d such that $X' := X(d)/d$ is δ-dense and has no α-almost divisor, and the following additional properties are satisfied:*

1. $d \leq 4\Sigma(X)/|X|^2$,

2. $|X'| \geq 0.75|X|$,

3. $\Sigma(X') \geq 0.75\,\Sigma(X)/d$.

Theorem B.4 ([BW21, Theorem 4.2]). *Let X be a multi-set and set*

$$C_\delta := 1699200 \cdot \log(2n) \log^2(2),$$
$$C_\alpha := 42480 \cdot \log(2),$$
$$C_\lambda := 169920 \cdot \log(2).$$

If X is C_δ-dense and has no C_α-almost divisor, then for $\lambda_X := C_\lambda \cdot (\max X) \cdot \Sigma(X)/|X|^2$ we have

$$\big([\lambda_X, \Sigma(X) - \lambda_X] \cap \mathbb{Z}\big) \subseteq \mathcal{S}(X).$$

Now we are ready to prove Lemma 2.6.

Lemma 2.6. *Let n distinct positive integers $X = \{x_1, \ldots, x_n\} \subseteq [\ell, 2\ell]$ be given, where $\ell = o(n^2/\log n)$.*

Then, for a universal constant $c \geq 1$, for every $c\ell^2/n \leq t \leq \Sigma(X)/2$, there exists $t' \in \mathcal{S}(X)$ such that $0 \leq t' - t \leq 8\ell/n$.

Proof. Let $C_\delta = \Theta(\log n), C_\alpha = \Theta(1), C_\lambda = \Theta(1)$ be defined as in Theorem B.4. Then, X is C_δ-dense since $C_\delta \cdot \ell = o(n^2)$. Let d be the positive integer guaranteed by Theorem B.3 such that $X' := X(d)/d$ is C_δ-dense and has no C_α-almost divisor.

Then, by Theorem B.4, for $\lambda_{X'} := C_\lambda \cdot (\max X') \cdot \Sigma(X')/|X'|^2$ we have

$$\big([\lambda_{X'}, \Sigma(X') - \lambda_{X'}] \cap \mathbb{Z}\big) \subseteq \mathcal{S}(X').$$

Since X' is C_δ-dense, we have $\lambda_{X'}/\Sigma(X') = C_\lambda \cdot (\max X')/|X'|^2 \leq C_\lambda/C_\delta < 0.1$.

Now, let $\lambda := d \cdot \lambda_{X'}$. From Property 3 in Theorem B.3, we know that

$$\frac{\Sigma(X)/2}{d} \leq \frac{2}{3}\Sigma(X') < \Sigma(X') - \lambda_{X'}.$$

Hence, given any $\lambda \leq t \leq \Sigma(X)/2$, we have

$$\lambda_{X'} \leq \lceil t/d \rceil \leq \Sigma(X') - \lambda_{X'}.$$

So $\lceil t/d \rceil \in \mathcal{S}(X')$, which implies $t' := d \cdot \lceil t/d \rceil \in \mathcal{S}(X)$. From Property 1 in Theorem B.3, we have

$$0 \leq t' - t < d \leq 4 \cdot |X| \cdot (2\ell)/|X|^2 = 8\ell/n.$$

Finally, we upper-bound λ as

$$
\begin{aligned}
\lambda &= d \cdot C_\lambda \cdot (\max X') \cdot \Sigma(X')/|X'|^2 \\
&\leq d \cdot C_\lambda \cdot (\max X')^2/|X'| \\
&\leq d \cdot C_\lambda \cdot (2\ell/d)^2/|X'| \\
&\leq d \cdot C_\lambda \cdot (2\ell/d)^2/(0.75n) \qquad\qquad\qquad\text{(by Property 2)} \\
&= (16C_\lambda/3) \cdot \frac{\ell^2}{dn} \\
&\leq O(\ell^2/n). \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \square
\end{aligned}
$$

C Known reductions from the Partition Problem

Recall that we defined the following simpler problem.

Problem 2. *Assume $\varepsilon \in (0, 1/2)$ and $1/\varepsilon \in \mathbb{N}^+$. Given a set X of n distinct integers in the interval $[1/\varepsilon, 2/\varepsilon)$, compute a set $A \subset \mathbb{N}$ that n-additively approximates $\mathcal{S}(X)$.*

In the following, we will reduce Partition to this problem.
By a simple greedy argument, we can assume $\mathrm{OPT} \geq t/2$.

Lemma C.1 (e.g., [MWW19, Lemma 4.3]). *One may assume w.l.o.g. that for any Subset Sum instance $\mathrm{OPT} \geq t/2$. Otherwise the instance can be solved exactly in $\tilde{O}(n)$ time.*

Then, we have the following important lemma about $(1-\varepsilon)$-approximating Partition. The key insight behind this lemma was first observed in [MWW19], indicating that approximating Partition is much easier than approximating general Subset Sum instances.

Lemma C.2 (c.f. [MWW19]). *Let $X \subset \mathbb{N}^+$ be a multiset with sum of elements $\sigma = \Sigma(X)$, and let $\varepsilon \in (0, 1/2)$. Given a set $A \subset \mathbb{N}$ that $\varepsilon\sigma/4$-additively approximates $\mathcal{S}(X)$, one can immediately solve $(1-\varepsilon)$-approximation Partition on X.*

Proof. Recall that $t = \sigma/2$, and $\mathrm{OPT} = \max\{\Sigma(Y) : \Sigma(Y) \leq t, Y \subseteq X\}$.
Given A, let $a := \max\{a \in A : a \leq t\}$. We claim that

$$(1-\varepsilon)\mathrm{OPT} \leq \min\{a, t(1-\varepsilon/2)\} \leq \mathrm{OPT},$$

which allows us to solve $(1-\varepsilon)$-approximation Partition on X.
We prove this claim by separately considering two cases.

- Case 1: $a \leq t(1 - \varepsilon/2)$.

  By definition of A, there exists $s \in \mathcal{S}(X)$ such that $s - \varepsilon\sigma/4 \leq a \leq s$. We have $s \leq a + \varepsilon\sigma/4 \leq t(1 - \varepsilon/2) + \varepsilon\sigma/4 = t$, so $s \in \mathcal{S}(X;t)$ and hence $\mathrm{OPT} \geq s \geq a$. By Lemma C.1 we can assume $t/2 \leq \mathrm{OPT} \leq t$. Then by definition of A there exists $a' \in A$ such that $a' \leq \mathrm{OPT} \leq t$ and $a' \geq \mathrm{OPT} - \varepsilon\sigma/4 \geq \mathrm{OPT} - \varepsilon\mathrm{OPT}$. Then, by definition of a, we have $a \geq a' \geq (1 - \varepsilon)\mathrm{OPT}$.

  Hence, we have established

  $$(1 - \varepsilon)\mathrm{OPT} \leq a = \min\{a, t(1 - \varepsilon/2)\} \leq \mathrm{OPT}.$$

- Case 2: $a > t(1 - \varepsilon/2)$.

  By definition of A, there exists $s \in \mathcal{S}(X)$ such that $s - \varepsilon\sigma/4 \leq a \leq s$. We have $s \leq a + \varepsilon\sigma/4 \leq t + \varepsilon\sigma/4 = t(1 + \varepsilon/2)$, and $s \geq a > t(1 - \varepsilon/2)$.

  By taking complement, we know $\sigma - s \in \mathcal{S}(X)$ as well. Using the crucial fact that $t = \sigma/2$, we see that $\min\{s, \sigma - s\} \in \mathcal{S}(X;t)$ and hence $\mathrm{OPT} \geq \min\{s, \sigma - s\}$. Then, since $s \in (t(1 - \varepsilon/2), t(1 + \varepsilon/2)]$, we have $\min\{s, \sigma - s\} \geq t(1 - \varepsilon/2)$.

  Hence, we have established

  $$(1 - \varepsilon)\mathrm{OPT} \leq t(1 - \varepsilon/2) = \min\{a, t(1 - \varepsilon/2)\} \leq \mathrm{OPT}. \qquad \square$$

Using Lemma C.2, we can solve $(1 - \varepsilon)$-approximation Partition by finding an additive approximation of $\mathcal{S}(X)$.

We are going to further simplify the input instance X. First we need the following lemma, which reduces the number of duplicate items in the input, by grouping them into powers of two. The proof of this lemma appeared in [MWW19], based on an earlier proof of a similar statement [KX19, Lemma 2.4].

Lemma C.3 ([MWW19, Lemma 4.1]). *Given a multiset S of n integers from $[t]$, one can compute a multiset T in $O(n \log n)$ time such that:*

- *$\mathcal{S}(S;t) = \mathcal{S}(T;t)$.*

- *$|T| \leq |S|$.*

- *No element in T has multiplicity exceeding two.*

- *For every $y \in T$, there is a corresponding $x \in S$ such that $y = 2^k \cdot x$ for some $k \in \mathbb{N}$.*

Now we prove the main lemma.

Lemma 4.1. *If for some $c \geq 1$, Problem 2 can be solved in $\tilde{O}(n + 1/\varepsilon^c)$ time, then $(1 - \varepsilon)$-approximating Partition can also be solved in $\tilde{O}(n + 1/\varepsilon^c)$ time.*

Proof. Let $X \subset \mathbb{N}^+$ be the input multiset of the $(1 - \varepsilon)$-Partition problem. We can without loss of generality assume $1/\varepsilon$ is an integer.

Recall that $t = \sigma/2 = \Sigma(X)/2$. We define a multiset $Y \subset \mathbb{N}^+$ as follows: for every $x \in X$, round x down to the nearest integer multiple of $\lceil \frac{\sigma}{100n/\varepsilon} \rceil$, denoted as y, and insert y into Y if y is nonzero.

Since the total incurred additive loss is at most $n \cdot (\lceil \frac{\sigma}{100n/\varepsilon} \rceil - 1) \leq \frac{\varepsilon\sigma}{100}$, we know that $\mathcal{S}(Y)$ is an $\varepsilon\sigma/100$-additive approximation of $\mathcal{S}(X)$.

Now, we can without loss of generality assume $y \in [1/\varepsilon, 100n/\varepsilon^2] \cap \mathbb{N}^+$ for all $y \in Y$, since otherwise we could simply scale all elements in X, Y (as well as σ, t).

Then, define another multiset $Z \subset \mathbb{N}^+$ as follows: for every $y \in Y$, round y down to $2^k \cdot z_0$ for some $k \in \mathbb{N} \cap [0, \log_2(100n/\varepsilon) + 1]$ and $z_0 \in \mathbb{N}^+ \cap [100/\varepsilon, 200/\varepsilon)$, and insert $2^k \cdot z_0$ into Z. Observe that, every $y \in Y$ incurs a multiplicative error of at most $\varepsilon/100$ after rounding. Hence, $\mathcal{S}(Z)$ is an $(1 - \varepsilon/100)$ approximation of $\mathcal{S}(Y)$. In particular, $\mathcal{S}(Z)$ approximates $\mathcal{S}(Y)$ with additive error at most $(\varepsilon/100) \cdot \Sigma(Y) \leq \varepsilon\sigma/100$. Combined with previous discussion, this means that $\mathcal{S}(Z)$ is an $\varepsilon\sigma/50$-additive approximation of $\mathcal{S}(X)$.

Then, we process Z using Lemma C.3, and obtain another set $Z' \subset \mathbb{N}^+$ so that $\mathcal{S}(Z') = \mathcal{S}(Z)$, and the multiplicity of any element in Z' is at most 2. Moreover, by the fourth property of Lemma C.3, we still have that every $z \in Z'$ can be expressed as $z = 2^k \cdot z_0$ for some non-negative integer $k \leq O(\log(n/\varepsilon))$ and integer $z_0 \in \mathbb{N}^+ \cap [100/\varepsilon, 200/\varepsilon)$. Now, we can partition Z' into $O(\log(n/\varepsilon))$ groups so that each group contains *distinct* integers from $2^k \cdot (\mathbb{N}^+ \cap [100/\varepsilon, 200/\varepsilon))$ for some non-negative integer $k \leq O(\log(n/\varepsilon))$.

Pick a smaller $\varepsilon' = \Theta\left(\frac{\varepsilon}{\log(n/\varepsilon)}\right)$ (assuming $\varepsilon/\varepsilon' \in \mathbb{N}^+$). For each group Z'_j mentioned above, we compute a set $A_j \subseteq \mathbb{N}$ that approximates $\mathcal{S}(Z'_j)$ with $\varepsilon'\Sigma(Z'_j)/100$ additive error. This can be done as follows: recall that Z'_j contains distinct integers from $2^k \cdot (\mathbb{N}^+ \cap [100/\varepsilon, 200/\varepsilon))$; we scale the integers in Z'_j to $(\varepsilon/\varepsilon') \cdot (\mathbb{N}^+ \cap [100/\varepsilon, 200/\varepsilon]))$ and then invoke the algorithm for Problem 2 which approximates $\mathcal{S}(Z'_j)$ with additive error $|Z'_j| \leq \frac{\Sigma(Z'_j)}{(\varepsilon/\varepsilon') \cdot 100/\varepsilon} = \varepsilon'\Sigma(Z'_j)/100$ as desired. The total running time for these invocations is (up to poly $\log(n/\varepsilon)$ factors) $\sum_j (|Z_j| + (100/\varepsilon')^c) \leq \tilde{O}(n + 1/\varepsilon^c)$.

Now, using the computed $A_j \subseteq \mathbb{N}$ that approximates $\mathcal{S}(Z'_j)$ with $\varepsilon'\Sigma(Z'_j)/100$ additive error, we will compute an approximation of $\mathcal{S}(Z')$ (recall that $Z' = \bigcup_j Z'_j$ is a partition). To do this, we first round every element in every A_j down to integer multiples of $\lceil \varepsilon'\sigma/100 \rceil$, and this rounded A'_j still approximates $\mathcal{S}(Z'_j)$ with additive error at most $\varepsilon'\Sigma(Z'_j)/100 + \varepsilon'\sigma/100 \leq \varepsilon'\sigma/50$. Finally, we use FFT to compute the sumset of all these A'_j (there are $O(\log(n/\varepsilon))$ of them), and this will be our approximation of $\mathcal{S}(Z')$. The accumulated additive error here is at most $O(\log(n/\varepsilon)) \cdot \varepsilon'\sigma/50 \leq \varepsilon\sigma/50$, and the running time of these FFTs is $O(\log(n/\varepsilon)) \cdot \tilde{O}\left(\frac{\sigma}{\lceil \varepsilon'\sigma/100 \rceil}\right) \leq \tilde{O}(1/\varepsilon)$.

We have obtained an $\varepsilon\sigma/50$-additive approximation of $\mathcal{S}(Z')$. Previously we established $\mathcal{S}(Z') = \mathcal{S}(Z)$ and $\mathcal{S}(Z)$ is an $\varepsilon\sigma/50$-additive approximation of $\mathcal{S}(X)$, so we have obtained an $\varepsilon\sigma/50$-additive approximation of $\mathcal{S}(X)$. By Lemma C.2, this is sufficient for solving $(1-\varepsilon)$-approximation Partition on X.

The overall running time of this reduction is $\tilde{O}(n + 1/\varepsilon^c)$. $\square$