An Improved FPTAS for 0-1 Knapsack

Ce Jin
Institute for Interdisciplinary Information Sciences, Tsinghua University, Beijing, China
jinc16@mails.tsinghua.edu.cn

Abstract

The 0-1 knapsack problem is an important NP-hard problem that admits fully polynomial-time approximation schemes (FPTASs). Previously the fastest FPTAS by Chan (2018) with approximation factor $1+\varepsilon$ runs in $\tilde{O}(n+(1/\varepsilon)^{12/5})$ time, where $\tilde{O}$ hides polylogarithmic factors. In this paper we present an improved algorithm in $\tilde{O}(n+(1/\varepsilon)^{9/4})$ time, with only a $(1/\varepsilon)^{1/4}$ gap from the quadratic conditional lower bound based on $(\min,+)$-convolution. Our improvement comes from a multi-level extension of Chan's number-theoretic construction, and a greedy lemma that reduces unnecessary computation spent on cheap items.

2012 ACM Subject Classification Theory of computation → Algorithm design techniques

Keywords and phrases approximation algorithms, knapsack, subset sum

Category Track A: Algorithms, Complexity and Games

Acknowledgements Part of this research was done while visiting Harvard University. I would like to thank Professor Jelani Nelson for introducing this problem to me, advising this project, and giving many helpful comments on my writeup.

1 Introduction

1.1 Background

In the *0-1 knapsack* problem, we are given a set I of n items where each item $i \in I$ has weight w_i and profit p_i, and we want to select a subset $J \subseteq I$ such that $\sum_{j\in J} w_j \le W$ and $\sum_{j\in J} p_j$ is maximized.

The 0-1 knapsack problem is a fundamental optimization problem in computer science and is one of Karp's 21 NP-complete problems [8]. An important field of study on NP-hard problems is to find efficient approximation algorithms. A $(1+\varepsilon)$-approximation algorithm (for a maximization problem) outputs a value SOL such that $\text{SOL} \le \text{OPT} \le (1+\varepsilon)\cdot\text{SOL}$, where OPT denotes the optimal answer. The 0-1 knapsack problem is one of the first problems that were shown to have fully polynomial-time approximation schemes (FPTASs), i.e., algorithms with approximation factor $1+\varepsilon$ for any given $0<\varepsilon<1$ and running time polynomial in both n and $1/\varepsilon$.

There has been a long line of research on finding faster FPTASs for the 0-1 knapsack problem, as summarized in Table 1. The first algorithm with only subcubic dependence on $1/\varepsilon$ was due to Rhee [15]. Very recently, Chan [3] gave an elegant algorithm for the 0-1 knapsack problem in deterministic $O(n\log\frac{1}{\varepsilon}+(\frac{1}{\varepsilon})^{5/2}/2^{\Omega(\sqrt{\log(1/\varepsilon)})})$ via simple combination of the SMAWK algorithm [1] and a standard divide-and-conquer technique. The speedup of superpolylogarithmic factor $2^{\Omega(\sqrt{\log(1/\varepsilon)})}$ is due to recent progress on $(\min,+)$-convolution [2, 16, 4]. Using an elementary number-theoretic lemma, Chan further improved the algorithm to $O(n\log\frac{1}{\varepsilon}+(\frac{1}{\varepsilon})^{12/5}/2^{\Omega(\sqrt{\log(1/\varepsilon)})})$ time, and obtained two new algorithms running in $\tilde{O}(\frac{1}{\varepsilon}n^{3/2})$ and $O((\frac{1}{\varepsilon})^{4/3}n+(\frac{1}{\varepsilon})^2)/2^{\Omega(\sqrt{\log(1/\varepsilon)})})$ time respectively, which are faster for small n.

FPTASs on several special cases of 0-1 knapsack are also of interest. For the *unbounded knapsack* problem, where every item has infinitely many copies, Jansen and Kraft [7] obtained an $O(n+(\frac{1}{\varepsilon})^2\log^3\frac{1}{\varepsilon})$-time algorithm; the unbounded version can be reduced

Table 1 FPTASs for 0-1 knapsack.

$O(n \log n + (\frac{1}{\varepsilon})^4 \log \frac{1}{\varepsilon})$	Ibarra and Kim [6]	1975
$O(n \log \frac{1}{\varepsilon} + (\frac{1}{\varepsilon})^4)$	Lawler [13]	1979
$O(n \log \frac{1}{\varepsilon} + (\frac{1}{\varepsilon})^3 \log^2 \frac{1}{\varepsilon})$	Kellerer and Pferschy [11]	2004
$O(n \log \frac{1}{\varepsilon} + (\frac{1}{\varepsilon})^{5/2} \log^3 \frac{1}{\varepsilon})$ (randomized)	Rhee [15]	2015
$O(n \log \frac{1}{\varepsilon} + (\frac{1}{\varepsilon})^{12/5}/2^{\Omega(\sqrt{\log(1/\varepsilon)})})$	Chan [3]	2018
$O(n \log \frac{1}{\varepsilon} + (\frac{1}{\varepsilon})^{9/4}/2^{\Omega(\sqrt{\log(1/\varepsilon)})})$	**This work**	
$O(\frac{1}{\varepsilon}n^3)$	Textbook algorithm	
$O(\frac{1}{\varepsilon}n^2)$	Lawler [13]	1979
$O((\frac{1}{\varepsilon})^2 n \log \frac{1}{\varepsilon})$	Kellerer and Pferschy [10]	1999
$\tilde{O}(\frac{1}{\varepsilon}n^{3/2})$ (randomized, Las Vegas)	Chan [3]	2018
$O(((\frac{1}{\varepsilon})^{4/3}n + (\frac{1}{\varepsilon})^2)/2^{\Omega(\sqrt{\log(1/\varepsilon)})})$	Chan [3]	2018
$O(((\frac{1}{\varepsilon})^{3/2}n^{3/4} + (\frac{1}{\varepsilon})^2)/2^{\Omega(\sqrt{\log(1/\varepsilon)})} + n \log \frac{1}{\varepsilon})$	**This work**	

to 0-1 knapsack with only a logarithmic blowup in the problem size [5]. For the *subset sum* problem, where every item has $p_i = w_i$, Kellerer et al. [9] obtained an algorithm with $O(\min\{n/\varepsilon, n + (\frac{1}{\varepsilon})^2 \log \frac{1}{\varepsilon}\})$ running time, which will be used in our algorithm as a subroutine. For the *partition* problem, which is a special case of the subset sum problem where $W = \frac{1}{2}\sum_{i \in I} w_i$, Mucha et al. [14] obtained an algorithm with a subquadratic $\tilde{O}(n + (\frac{1}{\varepsilon})^{5/3})$ running time.

On the lower bound side, recent reductions showed by Cygan et al. [5] and Künnemann et al. [12] imply that 0-1 knapsack and unbounded knapsack have no FPTAS in $O((n+\frac{1}{\varepsilon})^{2-\delta})$ time, unless $(\min, +)$-convolution has truly subquadratic algorithm [14]. It remains open whether 0-1 knapsack has a matching upper bound.

1.2 Our results

In this paper we present improved FPTASs for the 0-1 knapsack problem. Our results are summarized in the following two theorems.

▶ **Theorem 1.** *There is a deterministic* $(1 + \varepsilon)$*-approximation algorithm for 0-1 knapsack with running time* $O(n \log \frac{1}{\varepsilon} + (\frac{1}{\varepsilon})^{9/4}/2^{\Omega(\sqrt{\log(1/\varepsilon)})})$.

▶ **Theorem 2.** *For* $n = O(\frac{1}{\varepsilon})$*, there is a deterministic* $(1 + \varepsilon)$*-approximation algorithm for 0-1 knapsack with running time* $O\Big(\big(n^{3/4}(\frac{1}{\varepsilon})^{3/2} + (\frac{1}{\varepsilon})^2\big)/2^{\Omega(\sqrt{\log(1/\varepsilon)})}\Big)$.

Theorem 2 gives the current best time bound for $(\frac{1}{\varepsilon})^{2/3} \ll n \ll \frac{1}{\varepsilon}$, improving upon the previous $O(((\frac{1}{\varepsilon})^{4/3}n + (\frac{1}{\varepsilon})^2)/2^{\Omega(\sqrt{\log(1/\varepsilon)})})$ algorithm by Chan [3]. For $n \ll (\frac{1}{\varepsilon})^{2/3}$, Chan's $\tilde{O}(\frac{1}{\varepsilon}n^{3/2})$ time randomized algorithm [3] remains the fastest.

For $n \gg \frac{1}{\varepsilon}$, Theorem 1 gives a better time bound, improving upon the previous $O(n \log \frac{1}{\varepsilon} + (\frac{1}{\varepsilon})^{12/5}/2^{\Omega(\sqrt{\log(1/\varepsilon)})})$ algorithm by Chan [3].

1.3 Outline of our algorithm

We give an informal overview of our improved algorithm for 0-1 knapsack.

Using a known reduction [3], it suffices to solve an easier instance of 0-1 knapsack where profits of all items satisfy $p_i \in [1, 2]$. Here "solving an instance" means approximating the function $f(x) :=$ [maximum total profit of items with at most x total weight] for all $x \geq 0$, rather than for just a single point $x = W$. In this restricted case, simple greedy (sorting according to *unit profits* p_i/w_i) gives an additive error of at most $\max_j p_j = O(1)$, so it suffices to approximate the capped function $\min\{\varepsilon^{-1}, f(x)\}$ with approximation factor $1+O(\varepsilon)$. Chan gave an algorithm that $(1+\varepsilon)$-approximates $\min\{B, f(x)\}$ in $\tilde{O}(n+\varepsilon^{-2}B^{1/2})$ time (implied by [3, Lemma 7]), which immediately implies an $\tilde{O}(n+\varepsilon^{-5/2})$ time FPTAS by setting $B=\varepsilon^{-1}$.

Greedy. Now we explain how to use a greedy argument (described in detail in Section 5) to improve this algorithm to $\tilde{O}(n+\varepsilon^{-7/3})$ time. We sort all items (with $p_i \in [1,2]$) in non-increasing order of unit profits p_i/w_i, and divide them into three subsets H, M, L (items with high, medium, low unit profits), where H contains the top $\Theta(\varepsilon^{-1})$ items, and L contains all items i for which $p_i/w_i \le (1-\varepsilon^{2/3}) \cdot \min_{h\in H}\{p_h/w_h\}$, so there is a gap between the unit profits of H-items and L-items. Intuitively, there are sufficiently many H-items available, so it's not optimal to include too many cheap L-items when the knapsack capacity is not very big. To be more precise, we prove that in any optimal solution we care about (i.e., having optimal total profit smaller than ε^{-1}), the total profit contributed by L-items cannot exceed $O(\varepsilon^{-2/3})$. Hence, for subset L we only need to approximate up to $B=\Theta(\varepsilon^{-2/3})$ in $\tilde{O}(n+\varepsilon^{-2}B^{1/2}) = \tilde{O}(n+\varepsilon^{-7/3})$ time. Subset H has only $O(\varepsilon^{-1})$ items and can be solved using Chan's $\tilde{O}(\varepsilon^{-4/3}n+\varepsilon^{-2})$ algorithm in $\tilde{O}(\varepsilon^{-7/3})$ time. To solve subset M, we round down the profit value p_i for every item $i \in M$, so that the unit profit p_i/w_i becomes a power of $(1+\varepsilon)$. Then there are $O(\varepsilon^{-1/3})$ distinct unit profit values in M. Items with the same unit profit can be solved together using the efficient FPTAS for *subset sum* by Kellerer et al. [9] in $\tilde{O}(n+\varepsilon^{-2})$ time. Finally we merge the results for H, M, L. The total time complexity is $\tilde{O}(n+\varepsilon^{-7/3})$.

Multi-level number-theoretic construction. The above approach invokes two of Chan's algorithms: an $\tilde{O}(n+\varepsilon^{-2}B^{1/2})$ algorithm (useful for small B) and an $\tilde{O}(\varepsilon^{-4/3}n+\varepsilon^{-2})$ algorithm (useful for small n). The key ingredient in these algorithms is a number-theoretic lemma: we can $(1+\varepsilon)$-approximate all profit values $p_i \in [1,2]$ by multiples of elements from a small set $\Delta \subset [\delta, 2\delta]$ of size $|\Delta| = \tilde{O}(\frac{\delta}{\varepsilon})$ (small $|\Delta|$ can reduce the additive error incurred from rounding).

Chan obtained an $\tilde{O}(n+\varepsilon^{-2}B^{2/5})$ time algorithm using some additional tricks. First, evenly partition Δ into r subsets $\Delta^{(1)}, \ldots, \Delta^{(r)}$, and divide the items into $P = P^{(1)} \cup \cdots \cup P^{(r)}$ accordingly, so that profit values from $P^{(j)}$ are approximated by $\Delta^{(j)}$-multiples. To $(1+\varepsilon)$-approximate the profit function f_j for each $P^{(j)}$, pick a threshold $B_0 \ll B$, and return the combination of a $(1+\varepsilon)$-approximation of $\min\{f_j, B_0\}$ and an εB_0-additive-approximation of $\min\{f_j, B\}$. Since the size of $\Delta^{(j)}$ is only $|\Delta|/r$, the latter function can be approximated faster when $r \gg 1$. Finally, merge f_j over all $1 \le j \le r$. By fine-tuning the parameters r, δ, B_1, the time complexity is improved to $\tilde{O}(n+\varepsilon^{-2}B^{2/5})$.

Our new algorithm extends this technique to *multiple levels*. To $(1+\varepsilon)$-approximate the profit function f_j for each $P^{(j)}$, we will pick $B_0 \ll B_1 \ll \cdots \ll B_{d-1} \ll B_d \approx B$, and compute the εB_{i-1}-additive-approximation of $\min\{f_j, B_i\}$, for all $i \in [d]$. An issue of this multi-level approach is that, different levels have different optimal parameters δ_i and different $\Delta_i^{(1)}, \ldots, \Delta_i^{(r)}$, but we have to stick to the same partition of items $P = P^{(1)} \cup \cdots \cup P^{(r)}$ over all levels. We overcome this issue by enforcing that $\Delta_i^{(j)}$ at level i must be generated by multiples of elements from $\Delta_{i-1}^{(j)}$ at level $i-1$, so that $P^{(j)}$ can be approximated by $\Delta_i^{(j)}$-multiples for all levels. To achieve this, we need a multi-level version of the number-theoretic lemma. We will discuss this part in detail in Section 4.

Using this multi-level construction, we obtain algorithms in $\tilde{O}(n + \varepsilon^{-2}B^{1/3})$ time and $\tilde{O}(\varepsilon^{-3/2}n^{3/4} + \varepsilon^{-2})$ time. Combining these improved algorithms with the greedy argument previously described (the threshold which splits M and L needs to be adjusted accordingly), we obtain an algorithm in $\tilde{O}(n + \varepsilon^{-9/4})$ time as claimed in Theorem 1.

2 Preliminaries

Throughout this paper, $\log x$ stands for $\log_2 x$, and $\tilde{O}(f)$ stands for $O(f \cdot \operatorname{poly}\log(f))$.

We will describe our algorithm with approximation factor $1 + O(\varepsilon)$, which can be lowered to $1 + \varepsilon$ if we scale down ε by a constant factor at the beginning.

We are only interested in the case where $n = O(\varepsilon^{-4})$. For greater n, Lawler's $O(n \log \frac{1}{\varepsilon} + (\frac{1}{\varepsilon})^4)$ algorithm [13] is already near-optimal. Hence we assume $\log n = O(\log \varepsilon^{-1})$.

Assume $0 < w_i \le W$ 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 discard all items i with $p_i \le \frac{\varepsilon}{n} \max_j p_j$. Since the total profit of discarded items is at most $\varepsilon \max_j p_j$, the optimal total profit is only reduced by a factor of $1 + O(\varepsilon)$. So we can assume that $\frac{\max_j p_j}{\min_j p_j} \le \frac{n}{\varepsilon}$.

We adopt Chan's terminology in presenting our algorithm. For a set I of items, define the profit function

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

over non-negative real numbers $x \ge 0$. Note that f_I is a monotone (nondecreasing) step function. The *complexity* of a monotone step function refers to the number of its steps.

We say that a function $\tilde{f}$ approximates a function f with factor $1 + \varepsilon$ if $\tilde{f}(x) \le f(x) \le (1 + \varepsilon)\tilde{f}(x)$ for all $x \ge 0$. We say that $\tilde{f}$ approximates f with additive error δ if $\tilde{f}(x) \le f(x) \le \tilde{f}(x) + \delta$ for all $x \ge 0$. Our goal is to approximate f_I with factor $1 + O(\varepsilon)$ on the input item set I.

Let I_1, I_2 be two disjoint subsets of items, and $I = I_1 \cup I_2$. We have $f_I = f_{I_1} \oplus f_{I_2}$, where $\oplus$ denotes the $(\max, +)$-convolution, defined by $(f \oplus g)(x) = \max_{0 \le x' \le x}(f(x') + g(x - x'))$. If two non-negative monotone step functions f, g are approximated with factor $1 + \varepsilon$ by functions $\tilde{f}, \tilde{g}$ respectively, then $f \oplus g$ is also approximated by $\tilde{f} \oplus \tilde{g}$ with factor $1 + \varepsilon$.

For a monotone step function f with range[1] contained in $\{0\} \cup [A, B]$, we can obtain a function $\tilde{f}$ with complexity only $O(\varepsilon^{-1} \log(B/A))$ which approximates f with factor $1 + \varepsilon$, by simply rounding f down to powers of $(1 + \varepsilon)$. For our purposes, B/A will be bounded by polynomial of n and $1/\varepsilon$, hence we may always assume that the approximation results are monotone step functions with complexity $\tilde{O}(\varepsilon^{-1})$.

For an item set I with the same profit $p_i = p$ for every item $i \in I$, the step function f_I can be exactly computed in $O(n \log n)$ time by simple greedy: the function values are $0, p, 2p, \ldots, np$ and the x-breakpoints are $w_1, w_1 + w_2, \ldots, w_1 + \cdots + w_n$, after sorting all w_i's in nondecreasing order. We say that a monotone step function is *p-uniform* if its function values are of the form $0, p, 2p, \ldots, lp$ for some l. We say that a p-uniform function is *pseudo-concave* if the differences of consecutive x-breakpoints are nondecreasing from left to right. In the previous case where all p_i's are equal to p, f_I is indeed p-uniform and pseudo-concave.

[1] Here *range* refers to the set of possible output values of the function.

3 Chan's techniques

In this section we review several useful lemmas by Chan [3].

3.1 Merging profit functions

▶ **Lemma 3** ([3, Lemma 2(i)]). *Let $f_1, \dots, f_m$ 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 approximates $f_1 \oplus \cdots \oplus f_m$ with factor $1 + O(\varepsilon)$ and complexity $\tilde{O}(\frac{1}{\varepsilon} \log B/A)$ in $O(n) + \tilde{O}((\frac{1}{\varepsilon})^2 m / 2^{\Omega(\sqrt{\log(1/\varepsilon)})} \log B/A)$ time.*

▶ Remark 4. Lemma 3 is proved using a divide-and-conquer method, which was also used previously in [10]. The speedup of superpolylogarithmic factor $2^{\Omega(\sqrt{\log(1/\varepsilon)})}$ is due to recent progress on $(\min, +)$-convolution [2, 16, 4].

Lemma 3 enables us to focus on a simpler case where all $p_i \in [1, 2]$. For the general case, we divide the items into $O(\log \frac{\max_j p_j}{\min_j p_j}) = O(\log \varepsilon^{-1})$ groups, each containing items with $p_i \in [2^j, 2^{j+1}]$ for some j (which can be rescaled to $[1, 2]$), and finally merge the profit functions of all groups by using Lemma 3 in $\tilde{O}(n + \varepsilon^{-2})$ time.

Assuming ε^{-1} is an integer and every $p_i \in [1, 2]$, we can round every p_i down to a multiple of ε, introducing only a $1 + \varepsilon$ error factor. Then there are only $m = O(\varepsilon^{-1})$ distinct values of p_i. For every value of p_i, the corresponding profit function f_i is p_i-uniform and pseudo-concave, and can be obtained by simple greedy (as discussed in Section 2).

3.2 Approximating big profit values using greedy

When all p_i's are small, simple greedy gives good approximation guarantee when the answer is big enough.

▶ **Lemma 5.** *Suppose $p_i \in [1, 2]$ for all $i \in I$. For $B = \Omega(\varepsilon^{-1})$, the function f_I can be approximated with additive error $O(\varepsilon B)$ in $O(n \log n)$ time.*

Proof. Sort the items in nonincreasing order of unit profit p_i / w_i. Let $\tilde{f}$ be the monotone step function resulting from greedy, with function values $0, p_1, p_1 + p_2, \dots, p_1 + \cdots + p_n$ and x-breakpoints $0, w_1, w_1 + w_2, \dots, w_1 + \cdots + w_n$. It approximates f_I with an additive error of $\max_i p_i \le 2 \le O(\varepsilon B)$ for $B = \Omega(\varepsilon^{-1})$. ◀

When every $p_i \in [1, 2]$, we set $B = \varepsilon^{-1}$ and let f_H denote the result from greedy (Lemma 5). Then we only need to obtain a function f_L which $1 + O(\varepsilon)$ approximates $\min\{f_I, B\}$, and finally return $\max\{f_L, f_H\}$ as a $1 + O(\varepsilon)$ approximation of f_I (because when $f_I(x)$ exceeds B, an additive error $O(\varepsilon B)$ implies $1 + O(\varepsilon)$ approximation factor).

3.3 Approximation using Δ-multiples of small set Δ

For a set Δ of numbers, we say that p is a Δ-multiple if it is a multiple of δ for some $\delta \in \Delta$.

▶ **Lemma 6** ([3, Lemma 5]). *Let $f_1, \dots, f_m$ 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 approximates $\min\{f_1 \oplus \cdots \oplus f_m, B\}$ with additive error $O(|\Delta| \delta)$ in $\tilde{O}(Bm/\delta)$ time.*

▶ Remark 7. An intuition of Lemma 6 is as follows. When p_i's are exact multiples of δ, standard dynamic programming algorithm maintains a result array of length B/δ, and adding a new f_i to the result can be done in linear time (by exploiting the pseudo-concavity of f_i using the SMAWK algorithm[2]). Now if the next p_i to be considered is a multiple of $\delta' \neq \delta$, we first have to round down the current results to multiples of δ', introducing an additive error of δ'. We round our results for $|\Delta| - 1$ times, so smaller $|\Delta|$ implies smaller overall additive error.

▶ **Corollary 8.** *Let $f_1, \ldots, f_m$ be monotone step functions with ranges contained in $[0, B]$. If every f_i is p_i-uniform and pseudo-concave for some $p_i \in [1, 2]$, then we can compute a monotone step function that approximates $\min\{f_1 \oplus \cdots \oplus f_m, B\}$ with factor $1 + O(\varepsilon)$ in $\tilde{O}(\varepsilon^{-1}Bm)$ time.*

Proof. Assuming ε^{-1} is an integer, adjust every p_i down to the nearest multiple of ε, and adjust f_i accordingly. This introduces a $1 + \varepsilon$ overall error factor. Then use Lemma 6 with $\delta = \varepsilon, \Delta = \{\varepsilon\}$ to compute the desired function in $\tilde{O}(Bm\varepsilon^{-1})$ time. ◀

4 Extending Chan's number-theoretic construction

As mentioned in Section 1.3, the main results of this section are two approximation algorithms in $\tilde{O}(n + \varepsilon^{-2}B^{1/3})$ and $\tilde{O}(\varepsilon^{-3/2}n^{3/4} + \varepsilon^{-2})$ time respectively (the latter time bound assumes $n = O(1/\varepsilon)$). These results rely on Lemma 6.

4.1 Number-theoretic construction

To avoid checking boundary conditions, from now on we assume $\varepsilon > 0$ is sufficiently small.

Our algorithm extends Chan's technique by using a multi-level structure defined as follows.

▶ **Definition 9.** *For fixed parameters $\delta_1, \delta_2, \ldots, \delta_d$ satisfying condition*

$$\varepsilon \leq \delta_1, \;\; \delta_i \leq \delta_{i+1}/2, \;\; \delta_d \leq 1/8 \tag{1}$$

and a finite real number set $\Delta_1 \subset [\delta_1, 8\delta_1]$, a set tower *$(\Delta_1, \Delta_2, \ldots, \Delta_d)$ generated by Δ_1 is defined by recurrence*[3]

$$\Delta_{i+1} := [\delta_{i+1}, 8\delta_{i+1}] \cap \bigcup_{k \in \mathbb{Z}} k\Delta_i, \quad i = 1, 2, \ldots, d-1. \tag{2}$$

We refer to Δ_1 as the base set *and Δ_d as the* top set *of this set tower. We also say that the base set Δ_1 generates the top set Δ_d.*

If Δ_d^* is the top set generated by a singleton base set $\Delta_1^* = \{x\}$, then for every $y \in \Delta_d^*$ we say x generates *y.*

We have the following simple facts about set towers.

▶ **Proposition 10.** *If x generates y then $x \in \Delta_1$ implies $y \in \Delta_d$. Conversely, for every $y \in \Delta_d$, there exists $x \in \Delta_1$ which generates y, and for every $1 \leq i \leq d$ there exists $z \in \Delta_i$ such that both y/z and z/x are integers.*

---

[2] The SMAWK algorithm [1] finds all row-minima in an $n \times n$ matrix A satisfying the *Monge property* $A[i, j] + A[i + 1, j + 1] \leq A[i, j + 1] + A[i + 1, j]$ using only $O(n)$ queries.

[3] For a number k and a set A of numbers, $kA := \{ka : a \in A\}$.

▶ **Proposition 11.** *For any* $1 \le i \le d$, $|\Delta_i| \le 8^{i-1}(\delta_i/\delta_1)|\Delta_1|$, *and we can compute* Δ_i *in* $\tilde{O}(8^{i-1}(\delta_i/\delta_1)|\Delta_1|)$ *time given* Δ_1 *as input.*

Proof. For $2 \le i \le d$, we have

$$|\Delta_i| = \left|[\delta_i, 8\delta_i] \cap \bigcup_{k\in\mathbb{Z}} k\Delta_{i-1}\right| \le \sum_{x\in\Delta_{i-1}} 8\delta_i/x \le |\Delta_{i-1}| 8\delta_i/\delta_{i-1}.$$

The proof of size upper bounds follows by induction. Elements of Δ_i can be generated straightforwardly within the time bound. ◀

▶ **Lemma 12.** *Let* $T_1, T_2, \dots, T_d$ *be positive real numbers satisfying* $T_1 \ge 2$ *and* $T_{i+1} \ge 2T_i$. *There exist at least* $T_d/(\log T_d)^{O(d)}$ *integers* t *satisfying the following condition:* t *can be written as a product of integers* $t = n_1 n_2 \cdots n_d$, *such that* $n_1 n_2 \cdots n_i \in (T_i/2, T_i]$ *for every* $1 \le i \le d$.

The proof of Lemma 12 is deferred to Appendix A. Lemma 12 helps us prove the following fact, which is a multi-level extension of [3, Lemma 6].

▶ **Lemma 13.** *For any parameters* $\delta_1, \dots, \delta_d$ *satisfying condition (1), there exists a base set* Δ_1 *of size* $\frac{\delta_1}{\varepsilon} \cdot (\log \varepsilon^{-1})^{O(d)}$, *such that every* $p \in [1, 2]$ *can be approximated by a* Δ_d*-multiple with additive error* $O(\varepsilon)$, *where* Δ_d *is the top set generated by* Δ_1.

This base set Δ_1 *can be constructed in* $\tilde{O}(\varepsilon^{-1}\delta_1^{-1})$ *time deterministically.*

Proof. Let $P = \{1, 1+\varepsilon, 1+2\varepsilon, \dots, 1+\lfloor \frac{1}{\varepsilon} \rfloor \varepsilon\}$. It suffices to approximate every value $p \in P$ with additive error ε using Δ_d-multiples. For any $p \in P$ and $y \in \Delta_d \subset [\delta_d, 8\delta_d]$, p is approximated with additive error ε by a multiple of y if and only if $y \in \bigcup_{j\in\mathbb{Z}} \left[\frac{p-\varepsilon}{j}, \frac{p}{j}\right]$.

Our constructed base set Δ_1 will satisfy $\Delta_1 \subset [\delta_1, 4\delta_1]$. Suppose integers $k_1, k_2, \dots, k_{d-1}$ satisfy

$$k_1 k_2 \cdots k_{i-1} \in [\delta_i/\delta_1, 2\delta_i/\delta_1], \quad \text{for every } 2 \le i \le d. \tag{3}$$

Then by Definition 9, for any $x \in \Delta_1 \subset [\delta_1, 4\delta_1]$, we have $x k_1 k_2 \cdots k_{i-1} \in \Delta_i$ for every $2 \le i \le d$.

For any integer j satisfying

$$k_1 k_2 \cdots k_{d-1} j \in [p/(4\delta_1), p/(2\delta_1)], \tag{4}$$

the interval $[\frac{p-\varepsilon}{k_1 k_2 \cdots k_{d-1} j}, \frac{p}{k_1 k_2 \cdots k_{d-1} j}]$ is contained in $[\delta_1, 4\delta_1]$.

We say an integer K is *good* for p, if K can be expressed as a product of integers $k_1 k_2 \cdots k_{d-1} j$ satisfying conditions (3) and (4). For such K, any $x \in [\frac{p-\varepsilon}{K}, \frac{p}{K}] \cap \Delta_1$ generates an element $y = x k_1 k_2 \cdots k_{d-1} \in \Delta_d \cap [\frac{p-\varepsilon}{j}, \frac{p}{j}]$ such that p can be approximated by a multiple of y with additive error ε.

By Lemma 12, the number of good integers K for p is at least

$$\frac{p/(4\delta_1)}{\big(\log(p/(4\delta_1))\big)^{O(d)}} = \Omega\Big(\frac{\delta_1^{-1}}{(\log \varepsilon^{-1})^{O(d)}}\Big),$$

and at most $p/(2\delta_1) = O(\delta_1^{-1})$, by (4). Using conditions (3) and (4) we can compute all these K's by simple dynamic programming. We denote the union of their associated intervals by

$$I_p := \bigcup_{K \text{ good for } p} \left[\frac{p-\varepsilon}{K}, \frac{p}{K}\right] \subset [\delta_1, 4\delta_1]. \tag{5}$$

Note that these intervals are disjoint since $p/(K+1) \le (p-\varepsilon)/K$, so the total length of I_p can be lower-bounded as

$$\lambda(I_p) \ge \frac{\delta_1^{-1}}{(\log \varepsilon^{-1})^{O(d)}} \cdot \frac{p-(p-\varepsilon)}{\max K} \ge \frac{\varepsilon}{(\log \varepsilon^{-1})^{O(d)}}. \tag{6}$$

We have seen that p is approximated by a Δ_d-multiple with additive error ε as long as $\Delta_1 \cap I_p \neq \emptyset$. We compute I_p for every $p \in P$, and use the standard greedy algorithm (for Hitting Set problem) to construct a base set $\Delta_1 \subset [\delta_1, 4\delta_1]$ which intersects with every I_p: in each round we find a point $x \in [\delta_1, 4\delta_1]$ that hits the most I_p's, include x into Δ_1, and remove the I_p's that are hit by x. In each round the number of remaining I_p's decreases by

$$s := \frac{\min_p \lambda(I_p)}{4\delta_1 - \delta_1} \ge \frac{\varepsilon/\delta_1}{(\log \varepsilon^{-1})^{O(d)}},$$

so the solution size $|\Delta_1|$ is upper-bounded by

$$1 + \log_{1/(1-s)} |P| = O\left(\frac{\log |P|}{s}\right) = \frac{\delta_1}{\varepsilon}(\log \varepsilon^{-1})^{O(d)}.$$

To implement this greedy algorithm, we use standard data structures (for example, segment trees) that support finding x which hits the most intervals, reporting an interval hit by x, removing an interval, all in logarithmic time per operation. The number of operations is bounded by the total number of small intervals, so the running time is at most $\tilde{O}(|P| \cdot \frac{p}{2\delta_1}) = \tilde{O}(\delta_1^{-1}\varepsilon^{-1})$. ◀

The following lemma evenly partitions the base set Δ_1 into r subsets $\Delta_1^{(1)}, \dots, \Delta_1^{(r)}$, and partitions the profit values $P = \{p_1, \dots, p_m\}$ into $P^{(1)} \cup \dots \cup P^{(r)}$, so that $P^{(j)}$ can be approximated by $\Delta_d^{(j)}$-multiples. An additional requirement is that $P^{(1)}, \dots, P^{(r)}$ should have size $O(|P|/r)$ each.

▶ **Lemma 14.** *Let $\delta_1, \dots, \delta_d$ be parameters satisfying condition (1). Let $P = \{p_1, \dots, p_m\} \subset [1, 2]$ with $m = O(\varepsilon^{-1})$. Given a positive integer parameter $r = O(\min\{\frac{\delta_1}{\varepsilon}, m\})$, there exist r base sets $\Delta_1^{(1)}, \Delta_1^{(2)}, \dots, \Delta_1^{(r)}$ each of size $\frac{\delta_1}{\varepsilon r} \cdot (\log \varepsilon^{-1})^{O(d)}$, and a partition of $P = P^{(1)} \cup P^{(2)} \cup \dots \cup P^{(r)}$ each of size $O(m/r)$, such that for every $1 \le j \le r$, every $p \in P^{(j)}$ can be approximated by a $\Delta_d^{(j)}$-multiple with additive error $O(\varepsilon)$, where $\Delta_d^{(j)}$ is the top set generated by $\Delta_1^{(j)}$.*

These r base sets and the partition of P can be computed by a deterministic algorithm in $\tilde{O}(\varepsilon^{-2}/r)$ time .

Proof. First construct the base set Δ_1 of size $\frac{\delta_1}{\varepsilon}(\log \varepsilon^{-1})^{O(d)}$ from Lemma 13 in $\tilde{O}(\delta_1^{-1}\varepsilon^{-1}) = \tilde{O}(\varepsilon^{-2}/r)$ time, and compute the top set Δ_d that it generates. By Proposition 11, $|\Delta_d| \le 8^{d-1}\frac{\delta_d}{\delta_1}|\Delta_1| \le \frac{\delta_d}{\varepsilon}(\log \varepsilon^{-1})^{O(d)}$. Generate and sort all Δ_d-multiples in interval $[1, 2]$. For every $p \in P$, use binary search to find the Δ_d-multiple $ky \le p \, (y \in \Delta_d)$ closest to p, and then add p to the set Q_x, where $x \in \Delta_1$ is an element that generates y. (Q_x is initialized as empty for every $x \in \Delta_1$.) Then remove every x with $Q_x = \emptyset$ from Δ_1, so that $|\Delta_1| \le m$, while every $p \in P$ can still be approximated with $O(\varepsilon)$ additive error by a Δ_d-multiple.

Let $D := \max\{r, |\Delta_1|\}$ and let $s := \lceil m/D \rceil$. For every $x \in \Delta_1$ we divide Q_x evenly into small subsets each having size at most s. The total number of these small subsets is

$$\sum_{x \in \Delta_1} \lceil |Q_x|/s \rceil \le |\Delta_1| + \sum_{x \in \Delta_1} |Q_x|/s = |\Delta_1| + m/s \le 2D.$$

We merge these small subsets into r groups, each having at most $\lceil 2D/r \rceil$ small subsets. Then, define set $P^{(j)}$ as the union of small subsets from the j-th group, and let base set $\Delta_1^{(j)}$ contain $x \in \Delta_1$ if any of these small subsets comes from Q_x. So $|\Delta_1^{(j)}| \leq \lceil 2D/r \rceil = \frac{\delta_1}{\varepsilon r}(\log \varepsilon^{-1})^{O(d)}$, and $|P^{(j)}| \leq s \cdot \lceil 2D/r \rceil = O(m/D) \cdot O(D/r) = O(m/r)$. ◀

4.2 Approximation using set towers

We first prove a slightly improved version of Corollary 8. The only purpose of this lemma is to get rid of the $(\log \varepsilon^{-1})^{O(\log\log \varepsilon^{-1})}$ factor in the final running time.

▶ **Lemma 15.** *Let $f_1, \dots, f_m$ be monotone step functions with ranges contained in $[0, B]$ for some $1 \leq B \leq O(\varepsilon^{-1})$. If every f_i is p_i-uniform and pseudo-concave for some $p_i \in [1, 2]$, then we can compute a monotone step function that approximates $\min\{f_1 \oplus \cdots \oplus f_m, B\}$ with factor $1 + O(\varepsilon)$ in $\tilde{O}(\varepsilon^{-1}(Bm + \varepsilon^{-1})/B^{0.01})$ time.*

Proof. Using Lemma 13 with parameters $d = 1, \delta_1 = \varepsilon B^{0.01}$, we get $\Delta \subset [\delta_1, 8\delta_1]$ with size $|\Delta| \leq \tilde{O}(\delta_1/\varepsilon) = \tilde{O}(B^{0.01})$, in $\tilde{O}(\varepsilon^{-2}/B^{0.01})$ time. Adjust every p_i down to the nearest Δ-multiple, and adjust f_i accordingly. This introduces at most $1+O(\varepsilon)$ error factor. Then use Lemma 6 to compute a monotone step function f_H that approximates $\min\{f_1 \oplus \cdots \oplus f_m, B\}$ with additive error $e = O(|\Delta|\delta_1) = \tilde{O}(\varepsilon B^{0.02})$, in $\tilde{O}(B^{0.99} m\varepsilon^{-1})$ time.

Let $B_L := e/\varepsilon$, and use Corollary 8 to compute a monotone step function f_L that approximates $\min\{f_1 \oplus \cdots \oplus f_m, B_L\}$ with factor $1+O(\varepsilon)$ in only $\tilde{O}(B_L m\varepsilon^{-1}) = \tilde{O}(B^{0.02} m\varepsilon^{-1})$ time.

Since f_H approximates $\min\{f_1 \oplus \cdots \oplus f_m, B\}$ with additive error εB_L, $\max\{f_L, f_H\}$ is a $1 + O(\varepsilon)$ approximation of $\min\{f_1 \oplus \cdots \oplus f_m, B\}$. ◀

Now we can approximate the profit function $\min\{B, \bigoplus_{p_k \in P^{(j)}} f_k\}$ for each group $P^{(j)}$, using the multi-level approach described in Section 1.3.

▶ **Lemma 16.** *Let $f_1, \dots, f_m$ be given monotone step functions with ranges contained in $[0, B]$, and every f_k is p_k-uniform and pseudo-concave for some $p_k \in [1, 2]$. Assume $m = O(\varepsilon^{-1})$, $\Omega(\varepsilon^{-0.01}) \leq B \leq O(\varepsilon^{-1})$. Let r be a given positive integer parameter with $r = O(m), r = o(B)$.*

We can set $d = O(\log\log \varepsilon^{-1})$ and choose d parameters $\delta_1, \dots, \delta_d$ satisfying condition (1), such that the following statement holds:

Let $P^{(1)} \cup \cdots \cup P^{(r)}$ be the partition of set $P = \{p_1, \dots, p_m\}$ returned by the algorithm in Lemma 14 with the above parameters. Then for any $1 \leq j \leq r$, using the base set $\Delta_1^{(j)}$ from Lemma 14, we can compute a monotone step function that approximates $\min\{B, \bigoplus_{p_k \in P^{(j)}} f_k\}$ with factor $1 + O(\varepsilon)$, in $(\varepsilon^{-2}/r^{0.01} + m\varepsilon^{-1}B^{1/2}/r^{3/2})(\log \varepsilon^{-1})^{O(d)}$ time.

Proof. We can assume $B \geq 4r$, and define d to be the unique positive integer such that

$$2^{2^{d-1}} \leq \frac{\sqrt{B}}{\sqrt{r}} < 2^{2^d} = 4^{2^{d-1}}.$$

Then $d = O(\log\log \frac{\sqrt{B}}{\sqrt{r}}) = O(\log\log \varepsilon^{-1})$. Pick $\alpha \in [2, 4)$ such that

$$\alpha^{2^{d-1}} = \frac{\sqrt{B}}{\sqrt{r}}. \tag{7}$$

Define

$$\delta_i := \varepsilon\sqrt{Br} \Big/ \alpha^{2^{d-i}}, \ 0 \leq i \leq d. \tag{8}$$

Then

$$\delta_d = \frac{\varepsilon\sqrt{Br}}{\alpha},\ \delta_1 = \varepsilon r \tag{9}$$

Note that $\delta_d = \varepsilon\sqrt{B} \cdot O(\sqrt{r}) = \varepsilon\sqrt{B} \cdot o(\sqrt{B}) = \varepsilon \cdot o(B) = o(1)$. Hence the parameters $\delta_1, \ldots, \delta_d$ satisfy condition (1) for sufficiently small ε.

The base set $\Delta_1^{(j)}$ from Lemma 14 has size $\frac{\delta_1}{\varepsilon r}(\log \varepsilon^{-1})^{O(d)}$. We compute the generated set tower $\Delta_1^{(j)}, \Delta_2^{(j)}, \ldots, \Delta_d^{(j)}$. By Proposition 11, $|\Delta_i^{(j)}| \le \frac{\delta_i}{\varepsilon r}(\log \varepsilon^{-1})^{O(d)}$. Let

$$t := \max\left\{\alpha,\ \max_j |\Delta_i^{(j)}| \Big/ \frac{\delta_i}{\varepsilon r}\right\} = (\log \varepsilon^{-1})^{O(d)} \tag{10}$$

and define

$$B_i := Bt \Big/ \alpha^{2^{d-i}},\ \ 0 \le i \le d. \tag{11}$$

Then $B \le B_d \le B \cdot (\log \varepsilon^{-1})^{O(d)}$, and it's easy to verify that

$$|\Delta_i^{(j)}| \cdot \delta_i \le B_{i-1}\varepsilon,\ \ (1 \le i \le d). \tag{12}$$

For every $1 \le i \le d$, adjust every $p_k \in P^{(j)}$ down to the nearest $\Delta_i^{(j)}$-multiple and adjust f_k accordingly, which introduces a $1 + O(\varepsilon)$ error factor. Then use Lemma 6 to obtain a monotone step function g_i which approximates $\min\{\bigoplus_{p_k \in P^{(j)}} f_k, B_i\}$ with additive error $O(|\Delta_i^{(j)}|\delta_i) = O(\varepsilon B_{i-1})$ in $\tilde{O}(|P^{(j)}|B_i/\delta_i)$ time.

Then we use Lemma 15 to obtain a monotone step function g_0 which approximates $\min\{\bigoplus_{p_k \in P^{(j)}} f_k, B_0\}$ with $1 + O(\varepsilon)$ factor, in $\tilde{O}(\varepsilon^{-1}(|P^{(j)}|B_0 + \varepsilon^{-1})B_0^{-0.01})$ time. Notice that $B_0 = rt$.

Finally, $\max\{g_0, g_1, g_2, \ldots, g_d\}$ is a $1 + O(\varepsilon)$ approximation of $\min\{\bigoplus_{p_k \in P^{(j)}} f_k, B_d\}$, where $B_d \ge B$. Overall running time is

$$\begin{aligned}
&\tilde{O}(\varepsilon^{-1}(|P^{(j)}|B_0 + \varepsilon^{-1})B_0^{-0.01}) + \sum_{1 \le j \le d} \tilde{O}(|P^{(j)}|B_j/\delta_j) \\
&= \tilde{O}\big(\varepsilon^{-1}(\frac{m}{r} \cdot (rt) + \varepsilon^{-1})(rt)^{-0.01}\big) + d \cdot \tilde{O}\big(\frac{m}{r}B_d/\delta_d\big) \\
&= (\varepsilon^{-2}/r^{0.01} + m\varepsilon^{-1}B^{1/2}/r^{3/2})(\log \varepsilon^{-1})^{O(d)}.
\end{aligned}$$ ◀

Now we merge the results from all r groups, and obtain an approximation of the final result $\min\{f_1 \oplus \cdots \oplus f_m, B\}$.

▶ **Lemma 17.** *Let $f_1, \ldots, f_m$ be given monotone step functions with ranges contained in $[0, B]$, and every f_k is p_k-uniform and pseudo-concave for some $p_k \in [1, 2]$. Assume $m = O(1/\varepsilon), \Omega(\varepsilon^{-0.01}) \le B \le O(\varepsilon^{-1})$. We can approximate $\min\{f_1 \oplus \cdots \oplus f_m, B\}$ with factor $1 + O(\varepsilon)$ in $O(\varepsilon^{-2}B^{1/3}/2^{\Omega(\sqrt{\log(1/\varepsilon)})})$ time.*

Proof. Assume $m \ge \varepsilon^{-1}$, by adding zero functions which do not change the answer.

Let $r = o(B)$ be a positive integer parameter to be determined later.

Using Lemma 14 and Lemma 16, we can get a partition of $\{p_1, \ldots, p_m\} = P^{(1)} \cup \cdots \cup P^{(r)}$ and then get an $1 + O(\varepsilon)$ approximation of $\min\{\bigoplus_{p_k \in P^{(j)}} f_k, B\}$ for every $1 \le j \le r$, in $r \cdot (\varepsilon^{-2}/r^{0.01} + m\varepsilon^{-1}B^{1/2}/r^{3/2})(\log \varepsilon^{-1})^{O(d)} = (r^{0.99} + \sqrt{B/r})\varepsilon^{-2}(\log \varepsilon^{-1})^{O(\log\log \varepsilon^{-1})}$ overall time.

Then we use Lemma 3 to merge all these r functions in $\tilde{O}((\frac{1}{\varepsilon})^2 r/2^{\Omega(\sqrt{\log(1/\varepsilon)})})$ time.

Setting $r = B^{1/3}2^{c\sqrt{\log(1/\varepsilon)}}$, where $c > 0$ is some small enough constant, the total complexity is

$$O(\varepsilon^{-2}B^{1/3}/2^{\Omega(\sqrt{\log(1/\varepsilon)})}).$$ ◀

▶ **Lemma 18.** *Let I be a set of m items with $p_i \in [1,2]$ for every $i \in I$, where $\Omega(\varepsilon^{-2/3}) \le m \le O(\varepsilon^{-1})$. One can approximate f_I with factor $1 + O(\varepsilon)$ in $O(\varepsilon^{-3/2}m^{3/4}/2^{\Omega(\sqrt{\log(1/\varepsilon)})})$ time.*

Proof. Let $f_1, \ldots, f_m$ denote the profit functions of the m items.

Let $r = o(m^{1/2})$ be a positive integer parameter to be determined later. Obtain a partition of $\{p_1, \ldots, p_m\} = P^{(1)} \cup \cdots \cup P^{(r)}$ using Lemma 14. Let $B := \max_i \sum_{p \in P^{(i)}} p \le 2\max_i |P^{(i)}| = \Theta(m/r)$. Then $r = o(B)$. Use Lemma 16 to get an $1 + O(\varepsilon)$ approximation of $\bigoplus_{p_k \in P^{(j)}} f_k = \min\{\bigoplus_{p_k \in P^{(j)}} f_k, B\}$ for every $1 \le j \le r$, in $r \cdot (\varepsilon^{-2}/r^{0.01} + m\varepsilon^{-1}B^{1/2}/r^{3/2})(\log \varepsilon^{-1})^{O(d)} = (\varepsilon^{-2}r^{0.99} + m^{3/2}\varepsilon^{-1}/r)(\log \varepsilon^{-1})^{O(\log\log \varepsilon^{-1})}$ overall time.

Then we use Lemma 3 to merge all these r functions in $\tilde{O}((\frac{1}{\varepsilon})^2 r/2^{\Omega(\sqrt{\log(1/\varepsilon)})})$ time.

Setting $r = m^{3/4}\varepsilon^{1/2}2^{c\sqrt{\log(1/\varepsilon)}}$, where $c > 0$ is some small enough constant, the total complexity is

$$O(\varepsilon^{-3/2}m^{3/4}/2^{\Omega(\sqrt{\log(1/\varepsilon)})}).$$ ◀

▶ **Corollary 19** (restated Theorem 2)**.** *For $n = O(\frac{1}{\varepsilon})$, there is a deterministic $(1+\varepsilon)$-approximation algorithm for 0-1 knapsack in $O\Big((n^{3/4}(\frac{1}{\varepsilon})^{3/2} + (\frac{1}{\varepsilon})^2)/2^{\Omega(\sqrt{\log(1/\varepsilon)})}\Big)$ time.*

Proof. Divide the items into $O(\log \frac{n}{\varepsilon})$ groups, each containing items with $p_i \in [2^j, 2^{j+1}]$ for some j. Use Lemma 18 to solve each group, and merge them using Lemma 3. ◀

5 Main algorithm

5.1 A greedy lemma

Our improved algorithm uses the following lemma, which gives an upper bound on the total profit of cheap items (with low p_i/w_i) in an optimal knapsack solution.

▶ **Lemma 20.** *Let H, L be two subsets of items with $p_i \in [1,2]$. Let $W = \sum_{h \in H} w_h$ and $q = \min_{h \in H} \frac{p_h}{w_h}$. Suppose $\max_{l \in L} \frac{p_l}{w_l} \le q(1-\alpha)$ for some $0 < \alpha < 1$. Let $f = f_H \oplus f_L, \tilde{f} = f_H \oplus \min\{\frac{2}{\alpha}, f_L\}$. Then for every $x \le W$, $f(x) = \tilde{f}(x)$.*

Proof. By greedy, $f(W) = \sum_{h \in H} p_h = \tilde{f}(W)$ clearly holds. Now consider $0 \le x < W$. Suppose $f_L(x') + f_H(x - x')$ achieves its maximum value at $x' = w_L$, i.e., $f(x) = f_L(w_L) + f_H(x - w_L)$. It suffices to prove $f_L(w_L) \le \frac{2}{\alpha}$.

Let $J \subseteq H$ be a subset of items with total weight $w_J \le x - w_L$ and total profit achieving optimal value $f_H(x - w_L)$. Let $K \subseteq H \backslash J$ be a subset of items with total weight w_K, such that $w_K \le w_L$, and $w_K + w_i > w_L$ for every remaining item $i \in H \backslash (J \cup K)$. Such K can be constructed by a simple greedy algorithm.

Since $w_J + w_K \le (x - w_L) + w_L < W = \sum_{h \in H} w_h$, the remaining set $H \backslash (J \cup K)$ contains at least one item h_0. Hence, $w_L - w_K < w_{h_0} = p_{h_0}/\frac{p_{h_0}}{w_{h_0}} \le 2/q$, and equivalently $qw_K > qw_L - 2$.

Since $J \cup K$ is a subset of H with total weight bounded by x, we have $f_H(x) \geq \sum_{k\in K} p_k + \sum_{j\in J} p_j$, and thus $f_H(x) - f_H(x-w_L) = f_H(x) - \sum_{j\in J} p_j \geq \sum_{k\in K} p_k \geq qw_K > qw_L - 2$.

Hence $qw_L - 2 < f_H(x) - f_H(x-w_L) \leq f(x) - f_H(x-w_L) = f_L(w_L) \leq q(1-\alpha)w_L$, which shows that $q\alpha w_L \leq 2$. So $f_L(w_L) \leq q(1-\alpha)w_L \leq qw_L \leq 2/\alpha$, which concludes the proof. ◀

5.2 FPTAS for Subset Sum

We will use the efficient FPTAS for the subset sum problem by Kellerer et al. [9] as a subroutine in our algorithm.

▶ **Lemma 21** ([9], implicit)**.** *Let I be a set of n items and W be a number. We can obtain a list S of $O(\frac{1}{\varepsilon})$ numbers in $O(n + (\frac{1}{\varepsilon})^2 \log \frac{1}{\varepsilon})$ time, such that for every $s \leq W$ that is the subset sum $s = \sum_{j\in J} w_j$ of some subset $J \subseteq I$, there exists $s' \in S$ with $s - \varepsilon W \leq s' \leq s$.*

▶ Remark 22. This result wasn't explicitly stated in [9], but can be easily seen from their analysis of the correctness of the FPTAS.

▶ **Corollary 23.** *Let I be a set of n items with $p_i \in [1,2]$ and $p_i = w_i$ for every item $i \in I$. We can approximate f_I with factor $1 + O(\varepsilon)$ in $O(n \log n + \varepsilon^{-2} \log \frac{1}{\varepsilon} \log n)$ time.*

Proof. Notice that approximating s with additive error εW implies approximation factor $1+O(\varepsilon)$ for $W/2 \leq s \leq W$. So we simply apply Lemma 21 with $W = 2^j$ for $0 \leq j \leq 1 + \log n$, and merge all obtained lists. ◀

5.3 Improved algorithm

▶ **Lemma 24.** *Let I be a set of n items with $p_i \in [1,2]$ for every $i \in I$. We can approximate f_I with factor $1 + O(\varepsilon)$ in $O(n \log \frac{1}{\varepsilon} + (\frac{1}{\varepsilon})^{9/4}/2^{\Omega(\sqrt{\log(1/\varepsilon)})})$ time.*

Proof. Let $B = \lceil \varepsilon^{-1} \rceil$ and assume $n \geq B$ (if $n < B$, we can simply apply Lemma 18). By Lemma 5, we can approximate f_I with additive error $O(\varepsilon B)$ in $O(n \log \frac{1}{\varepsilon})$ time, so we only need to approximate $\min\{f_I, B\}$ with factor $1 + O(\varepsilon)$.

We sort the items by their unit profits p_i/w_i. Let set H contain the top B items with the highest unit profits. Define $q = \min_{h\in H} \frac{p_h}{w_h}$, and let M be the set of remaining items i with $q(1-\alpha) \leq \frac{p_i}{w_i} \leq q$, where parameter $0 < \alpha < 1$ is to be determined later. Let set L contain the remaining items not included in H or M.

Using Lemma 18, we can compute $\tilde{f}_H$ which approximates f_H with factor $1 + O(\varepsilon)$ in time $O(B^{3/4}\varepsilon^{-3/2}/2^{\Omega(\sqrt{\log(1/\varepsilon)})}) = O(\varepsilon^{-9/4}/2^{\Omega(\sqrt{\log(1/\varepsilon)})})$.

Since $\max_{l\in L} \frac{p_l}{w_l} < q(1-\alpha)$, Lemma 20 states that $f_H \oplus f_L$ and $f_H \oplus \min\{2/\alpha, f_L\}$ agree when $x \leq W_H = \sum_{h\in H} w_h$. Since $(f_H \oplus f_L)(W_H) = \sum_{h\in H} p_h \geq B$, this implies $\min\{B, f_H \oplus f_L\} = \min\{B, f_H \oplus \min\{2/\alpha, f_L\}\}$. For every item $l \in L$, we round down p_l to a power of $1+\varepsilon$, so that there are only $\log_{1+\varepsilon} 2 = O(\varepsilon^{-1})$ distinct values. This only multiplies the approximation factor by $1 + \varepsilon$. Then we use Lemma 17 to compute an approximation of $\min\{2/\alpha, f_L\}$ with factor $1 + O(\varepsilon)$ in $\tilde{O}(\varepsilon^{-2}(2/\alpha)^{1/3}/2^{\Omega(\sqrt{\log(1/\varepsilon)})})$ time. We merge it with $\tilde{f}_H$ and obtain an approximation of $\min\{f_H \oplus f_L, B\}$ with factor $1 + O(\varepsilon)$.

For every $m \in M$, we round down p_m so that the unit profit p_m/w_m becomes a power of $1 + \varepsilon$. After rounding, the approximation factor is only multiplied by $1 + \varepsilon$, and there are at most $\log_{1+\varepsilon} \frac{q}{q(1-\alpha)} = O(\alpha/\varepsilon)$ distinct unit profits in M. Let M_q denote the set of items in M with unit profit q. For each q, we use Lemma 23 to obtain a $1 + \varepsilon$ approximation of the function f_{M_q} in $O(|M_q| + \varepsilon^{-2})$ time. Then we use Lemma 3 to merge these functions and obtain a $1 + \varepsilon$ approximation of f_M. The total time is $O(|M| \log n) + \tilde{O}(\alpha\varepsilon^{-3})$.

Finally we merge the functions and get an approximation of $\min\{B, f_L \oplus f_H \oplus f_M\}$ with factor $1 + O(\varepsilon)$. The total time is $O(n \log \frac{1}{\varepsilon}) + \tilde{O}(\alpha\varepsilon^{-3} + \varepsilon^{-2}(2/\alpha)^{1/3}/2^{\Omega(\sqrt{\log(1/\varepsilon)})})$, which is $O(n \log \frac{1}{\varepsilon} + \varepsilon^{-9/4}/2^{\Omega(\sqrt{\log(1/\varepsilon)})})$ if we choose $\alpha = \varepsilon^{3/4}/2^{c\sqrt{\log(1/\varepsilon)}}$ for a sufficiently small constant c. ◀

▶ **Corollary 25** (restated Theorem 1). *There is a deterministic* $(1+\varepsilon)$*-approximation algorithm for 0-1 knapsack with running time* $O(n \log \frac{1}{\varepsilon} + (\frac{1}{\varepsilon})^{9/4}/2^{\Omega(\sqrt{\log(1/\varepsilon)})})$.

Proof. Divide the items into $O(\log \frac{n}{\varepsilon})$ groups, each containing items with $p_i \in [2^j, 2^{j+1}]$ for some j. Use Lemma 24 to solve each group, and merge them using Lemma 3. ◀

References

1. Alok Aggarwal, Maria M. Klawe, Shlomo Moran, Peter Shor, and Robert Wilber. Geometric applications of a matrix-searching algorithm. *Algorithmica*, 2(1):195–208, November 1987. doi:10.1007/BF01840359.
2. 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.
3. 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.
4. 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.
5. 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.
6. 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.
7. 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.
8. Richard M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, pages 85–103. Springer US, 1972. doi:10.1007/978-1-4684-2001-2_9.
9. Hans Kellerer, Renata Mansini, Ulrich Pferschy, and Maria Grazia Speranza. An efficient fully polynomial approximation scheme for the Subset-Sum Problem. *Journal of Computer and System Sciences*, 66(2):349–370, 2003. doi:10.1016/S0022-0000(03)00006-0.
10. 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.
11. 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.
12. 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.
13. 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.

14 Marcin Mucha, Karol Węgrzycki, and Michał Włodarczyk. Subquadratic Approximation Scheme for Partition. In *Proceedings of the 30th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 70–88, 2019. doi:10.1137/1.9781611975482.5.
15 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.
16 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.

A Proof of Lemma 12

▶ **Theorem 26** (Reminder of Lemma 12). *Let $T_1, T_2, \dots, T_d$ be positive real numbers satisfying $T_1 \geq 2$ and $T_{i+1} \geq 2T_i$. There exist at least $T_d/(\log T_d)^{O(d)}$ integers t satisfying the following condition: t can be written as a product of integers $t = n_1 n_2 \cdots n_d$, such that $n_1 n_2 \cdots n_i \in (T_i/2, T_i]$ for every $1 \leq i \leq d$.*

Proof. For every $1 \leq k \leq d$, we say an ordered k-tuple $(p_1, p_2, \dots, p_k)$ is *valid* if every p_i is prime, and $p_1 p_2 \cdots p_i \in (T_i/2, T_i]$ for every $1 \leq i \leq k$. Then the product $t = p_1 p_2 \cdots p_d$ of any valid d-tuple $(p_1, \dots, p_d)$ satisfies our condition. For any integer t, there are at most $d!$ different valid d-tuples with product t (which could be obtained by permuting t's prime factors). Let N_k denote the number of valid k-tuples. Then it suffices to show $N_d/(d!) \geq T_d/(\log T_d)^{O(d)}$.

By the prime number theorem and Bertrand-Chebyshev theorem, there exists a positive constant C such that

$$\pi(x) - \pi(x/2) \geq x/(C \log x), \text{ for all } x \geq 2,$$

where $\pi(x)$ denotes the number of primes less than or equal to x. We will prove $N_k \geq T_k/(C \log T_k)^k$ for all $1 \leq k \leq d$ by induction.

First note that this statement is trivial for $k = 1$. For $k \geq 2$, a valid k-tuple $(p_1, \dots, p_k)$ can be obtained by appending any prime $p_k \in \big(T_k/(2P), T_k/P\big]$ to any valid $(k-1)$-tuple $(p_1, \dots, p_{k-1})$ with product $P = p_1 \cdots p_{k-1} \leq T_{k-1}$. The number of such primes p_k is

$$\pi(T_k/P) - \pi\big(T_k/(2P)\big) \geq \frac{T_k/P}{C \log(T_k/P)} \geq \frac{T_k/T_{k-1}}{C \log T_k}.$$

Summing over all valid $(k-1)$-tuples, we have

$$N_k \geq N_{k-1} \cdot \frac{T_k/T_{k-1}}{C \log T_k} \geq \frac{T_{k-1}}{(C \log T_{k-1})^{k-1}} \cdot \frac{T_k/T_{k-1}}{C \log T_k} \geq \frac{T_k}{(C \log T_k)^k}.$$

Hence, $N_d \geq T_d/(C \log T_d)^d$ by induction. Observe that $T_d \geq 2^d$ and we have

$$\frac{N_d}{d!} \geq \frac{T_d}{(Cd \log T_d)^d} \geq \frac{T_d}{(C \log^2 T_d)^d} \geq \frac{T_d}{(\log T_d)^{O(d)}},$$

which finishes the proof. ◀