# Approximation Schemes for 0-1 Knapsack

**Timothy M. Chan**

**Dept. of Computer Science, University of Illinois at Urbana-Champaign, USA**
tmc@illinois.edu

## Abstract

We revisit the standard 0-1 knapsack problem. The latest polynomial-time approximation scheme by Rhee (2015) with approximation factor $1+\varepsilon$ has running time near $\widetilde{O}(n+(1/\varepsilon)^{5/2})$ (ignoring polylogarithmic factors), and is randomized. We present a simpler algorithm which achieves the same result and is deterministic.

With more effort, our ideas can actually lead to an improved time bound near $\widetilde{O}(n+(1/\varepsilon)^{12/5})$, and still further improvements for small $n$.

**1998 ACM Subject Classification** F.2.2 Nonnumerical Algorithms and Problems

**Keywords and phrases** knapsack problem, approximation algorithms, optimization, (min,+)-convolution

**Digital Object Identifier** 10.4230/OASIcs.SOSA.2018.5

## 1 Introduction

In the *0-1 knapsack* problem, we are given a set of $n$ items where the $i$-th item has weight $w_i \leq W$ and profit $p_i > 0$, and we want to select a subset of items with total weight bounded by $W$ while maximizing the total profit. In other words, we want to maximize $\sum_{i=1}^n p_i\xi_i$ subject to the constraint that $\sum_{i=1}^n w_i\xi_i \leq W$ over all $\xi_1, \ldots, \xi_n \in \{0,1\}$.

This classic textbook problem is among the most fundamental in combinatorial optimization and approximation algorithms, and is important for being one of the first NP-hard problems shown to possess *fully polynomial-time approximation schemes (FPTASs)*, i.e., algorithms with approximation factor $1+\varepsilon$ for any given parameter $\varepsilon \in (0,1)$, taking time polynomial in $n$ and $\frac{1}{\varepsilon}$.

Despite all the attention the problem has received, the "fine-grained complexity" of FPTASs remains open: we still do not know the best running time as a function of $n$ and $\frac{1}{\varepsilon}$. An $O(\frac{1}{\varepsilon}n^3)$-time algorithm via dynamic programming is perhaps the most often taught in undergraduate algorithm courses. The first published FPTAS by Ibarra and Kim [6] from 1975 required $\widetilde{O}(n+(\frac{1}{\varepsilon})^4)$ time, where the $\widetilde{O}$ notation hides polylogarithmic factors in $n$ and $\frac{1}{\varepsilon}$. Lawler [12] subsequently obtained a small improvement, but only in the hidden polylogarithmic factors. For a long time, the record time bound was $\widetilde{O}(n+(\frac{1}{\varepsilon})^3)$ by Kellerer and Pferschy [10]. Recently, in a (not-too-well-known) Master's thesis, Rhee [14] described a new randomized algorithm running in $\widetilde{O}(n+(\frac{1}{\varepsilon})^{2.5})$ time. (Note that improved time bounds of this form tell us how much accuracy we can guarantee while keeping near-linear running time; for example, Rhee's result implies that a $(1+n^{-2/5})$-approximate solution can be found in $\widetilde{O}(n)$ time.)

In this paper, we give a new presentation of an algorithm that has the same running time as Rhee's, with the added advantages of being deterministic and simpler: One part of Rhee's algorithm relied on solving several linear programs with two constraints, using a Lagrangian relaxation and some sophisticated form of randomized binary search (although I suspect known low-dimensional linear programming techniques might help). In contrast,


**Table 1** FPTASs for the 0-1 knapsack problem.

| running time | reference | year |
|---|---|---|
| $n^{O(1/\varepsilon)}$ | Sahni [15] | 1975 |
| $O(n \log n + (\frac{1}{\varepsilon})^4 \log \frac{1}{\varepsilon})$ | Ibarra and Kim [6] | 1975 |
| $O(n \log \frac{1}{\varepsilon} + (\frac{1}{\varepsilon})^4)$ | Lawler [12] | 1979 |
| $O(n \log \frac{1}{\varepsilon} + (\frac{1}{\varepsilon})^3 \log^2 \frac{1}{\varepsilon})$ | Kellerer and Pferschy [10] | 2004 |
| $O(n \log \frac{1}{\varepsilon} + (\frac{1}{\varepsilon})^{5/2} \log^3 \frac{1}{\varepsilon})$ (randomized) | Rhee [14] | 2015 |
| $O(n \log \frac{1}{\varepsilon} + (\frac{1}{\varepsilon})^{5/2} / 2^{\Omega(\sqrt{\log(1/\varepsilon)})})$ (deterministic) | Section 4 | |
| $O(n \log \frac{1}{\varepsilon} + (\frac{1}{\varepsilon})^{12/5} / 2^{\Omega(\sqrt{\log(1/\varepsilon)})})$ (deterministic) | Appendix A | |
| $O(\frac{1}{\varepsilon} n^2)$ | Lawler [12] | 1979 |
| $O((\frac{1}{\varepsilon})^2 n \log \frac{1}{\varepsilon})$ | Kellerer and Pferschy [9] | 1999 |
| $\widetilde{O}(\frac{1}{\varepsilon} n^{3/2})$ (randomized) | Appendix B | |
| $O(((\frac{1}{\varepsilon})^{4/3} n + (\frac{1}{\varepsilon})^2) / 2^{\Omega(\sqrt{\log(1/\varepsilon)})})$ (deterministic) | Appendix B | |

our approach bypasses this part completely. Ironically, the "new" approach is just a simple combination of two previous approaches. Along the way, we also notice that the hidden polylogarithmic factors in the second term can be eliminated; in fact, we can get speedup of a *super*polylogarithmic factor ($2^{\Omega(\sqrt{\log(1/\varepsilon)})}$) by using the latest results on $(\min, +)$-*convolution* [2, 16], if we give up on simplicity.

In research, simplifying previous solutions can often serve as a starting point to obtaining new improved solutions. Indeed, by combining our approach with a few extra ideas, we can actually obtain a faster FPTAS for 0-1 knapsack running in $\widetilde{O}(n + (\frac{1}{\varepsilon})^{2.4})$ time. These extra ideas are interesting (relying on an elementary number-theoretic lemma), but since the incremental improvement is small and the algorithm is more complicated, we feel it is of secondary importance compared to the simpler $\widetilde{O}(n + (\frac{1}{\varepsilon})^{2.5})$ algorithm (in the true spirit of SOSA), and thus defer that result to the appendix. The appendix also describes some further improved bounds for small $n$ (see the bottom half of Table 1).

In passing, we should mention two easier special cases. First, for the *subset sum* problem, corresponding to the case when $p_i = w_i$, Kellerer et al. [8] obtained algorithms with $\widetilde{O}(\frac{1}{\varepsilon} n)$ and $\widetilde{O}(n + (\frac{1}{\varepsilon})^2)$ running time. For the *unbounded* knapsack problem, where the variables $\xi_i$ are unbounded nonnegative integers, Jansen and Kraft [7] obtained an $\widetilde{O}(n + (\frac{1}{\varepsilon})^2)$-time algorithm; the unbounded problem can be reduced to the 0-1 case, ignoring logarithmic factors [5]. These methods do not adapt to the general 0-1 knapsack problem.

## 2 Preliminaries

First we may discard all items with $p_i \le \frac{\varepsilon}{n} \max_j p_j$; this changes the optimal value by at most $\varepsilon \max_j p_j$, and thus at most a factor of $1 + \varepsilon$. So we may assume that $\frac{\max_j p_j}{\min_j p_j} \le \frac{n}{\varepsilon}$. By rounding, we may assume that all $p_i$'s are powers of $1 + \varepsilon$. In particular, there are at most $m = O(\frac{1}{\varepsilon} \log \frac{n}{\varepsilon})$ distinct $p_i$ values.

We adopt a "functional" approach in presenting our algorithms, which does not need explicit reference to dynamic programming, and makes analysis of approximation factors more elegant:

Given input $I = \{(w_1, p_1), \ldots, (w_n, p_n)\}$, we consider the more general problem of approximating the function

$$f_I(x) := \max\left\{\sum_{i=1}^n p_i\xi_i \;:\; \sum_{i=1}^n w_i\xi_i \le x, \;\; \xi_1, \ldots, \xi_n \in \{0,1\}\right\}$$

for *all* $x \in \mathbb{R}$. Note that $f_I$ is a monotone step function (in this paper, "monotone" always means "nondecreasing"). It is more convenient to define approximation from below: we say that a function $\widetilde{f}$ *approximates* a function $f$ *with factor* $1+\varepsilon$ if $1 \le \frac{f(x)}{\widetilde{f}(x)} \le 1+\varepsilon$ for all $x \in \mathbb{R}$. We say that $\widetilde{f}$ *approximates* $f$ *with additive error* $\delta$ if $0 \le f(x) - \widetilde{f}(x) \le \delta$.

We can merge $f_I$ functions by the following easy observation: if $I$ is the disjoint union of $I_1$ and $I_2$, then $f_I = f_{I_1} \oplus f_{I_2}$, where the operator $\oplus$ denotes the $(\max, +)$-*convolution*, defined by the formula

$$(f \oplus g)(x) = \max_{x' \in \mathbb{R}} (f(x') + g(x - x')).$$

In the "base case" when the $p_i$'s are all equal to a common value $p$, the function $f_I$ is easy to compute, by the obvious greedy algorithm: the function values are $-\infty, 0, p, 2p, \ldots, np$ and the $x$-breakpoints are $0, w_1, w_1 + w_2, \ldots, w_1 + \cdots + w_n$, after arranging the items in nondecreasing order of $w_i$. We say that a step function is *$p$-uniform* if the function values are of the form $-\infty, 0, p, 2p, \ldots, \ell p$ for some $\ell$. Furthermore, we say that a $p$-uniform function is *pseudo-concave* if the sequence of differences of consecutive $x$-breakpoints is nondecreasing. When the $p_i$'s are all equal, $f_I$ is indeed uniform and pseudo-concave. Thus, the original problem reduces to computing a monotone step function that is a $(1 + O(\varepsilon))$-factor approximation of the $\oplus$ of $m = O(\frac{1}{\varepsilon} \log \frac{n}{\varepsilon})$ uniform, pseudo-concave, monotone step functions.

The following facts provides the building blocks for all our algorithms.

▶ **Fact 1.** *Let $f$ and $g$ be monotone step functions with total complexity $O(\ell)$ (i.e., with $O(\ell)$ steps). We can compute $f \oplus g$ in*

- **(i)** *$\ell^2/2^{\Omega(\sqrt{\log \ell})}$ time if $f$ and $g$ are $p$-uniform;*
- **(ii)** *$O(\ell)$ time if $f$ is $p$-uniform, and $g$ is $p$-uniform and pseudo-concave;*
- **(iii)** *$O((\ell + \ell' \cdot \frac{p'}{p}) \log \frac{p'}{p})$ time if $f$ is $p$-uniform, and $g$ is $p'$-uniform and pseudo-concave with complexity $\ell'$, and $p'$ is a multiple of $p$.*

**Proof.** Without loss of generality, assume that the ranges of $f$ and $g$ are $\{-\infty, 0, 1, 2, \ldots, \ell\}$.

**(i)** Define $f^{-1}(y)$ to be the smallest $x$ with $f(x) = y$ (if no such $x$ exists, define $f^{-1}(y)$ to be supremum of all $x$ with $f(x) < y$). Define $g^{-1}(y)$ similarly. Both $f^{-1}$ and $g^{-1}$ can be generated in $O(\ell)$ time. We can compute the $(\min, +)$-convolution

$$(f \oplus g)^{-1}(y) = \min_{y' \in \{0,1,\ldots,\ell\}} (f^{-1}(y') + g^{-1}(y - y'))$$

for all $y \in \{0, 1, \ldots, 2\ell\}$ in $O(\ell^2)$ time naively. From $(f \oplus g)^{-1}$, we can obtain $f \oplus g$ in $O(\ell)$ time.

A slight speedup to $\ell^2/2^{\Omega(\sqrt{\log \ell})}$ time is known for the $(\min, +)$-convolution problem, by using Bremner et al.'s reduction to $(\min, +)$-matrix multiplication [2] and Williams' algorithm for the latter problem [16] (which was originally randomized but was derandomized later [4]). This improvement is not simple, however.

**(ii)** For this part, Kellerer and Pferschy [10] have already described an $O(\ell \log \ell)$-time algorithm (the extra logarithmic factor does not matter to us in the end), but actually we can directly reduce to a standard *matrix searching* problem [1]: computing the row minima in an $O(\ell) \times O(\ell)$ matrix $A$ satisfying the *Monge property*. To compute the above $(\min, +)$-convolution, we can set $A[y, y'] = f^{-1}(y) + g^{-1}(y' - y)$, and observe that the Monge property $A[y, y'] + A[y + 1, y' + 1] \leq A[y, y' + 1] + A[y + 1, y']$ is equivalent to $g^{-1}(y' - y) - g^{-1}(y' - y - 1) \leq g^{-1}(y' - y + 1) - g^{-1}(y' - y)$, which corresponds precisely to the definition of pseudo-concavity of $g$. The well-known SMAWK algorithm [1] solves the matrix searching problem in $O(\ell)$ time.

**(ii′)** This part can be directly reduced to (ii) as follows. Say that a function $h$ is *shifted-$p$-uniform* if $h + a$ is $p$-uniform for some value $a$. The *upper envelope* of $h_1, \ldots, h_m$ refers to the function $h(x) = \max\{h_1(x), \ldots, h_m(x)\}$.

We can express the given $p$-uniform function $f$ as an upper envelope of $\frac{p'}{p}$ shifted-$p'$-uniform functions $f_i$, each with complexity $O(\ell \frac{p}{p'})$. For each $i$, we can compute $f_i \oplus g$ by (ii) (after shifting $f_i$) in $O(\ell \frac{p}{p'} + \ell')$ time. The total time is $O(\frac{p'}{p} \cdot (\ell \frac{p}{p'} + \ell'))$. We can then return the upper envelope of all these functions $f_i \oplus g$. Note that the upper envelope of $\frac{p'}{p}$ step functions can be constructed in time linear in their total complexity times $\log \frac{p}{p'}$, by sweeping the breakpoints from left to right, using a priority queue to keep track of the current maximum. ◀

## 3 Two Known Methods with Exponent 3

We begin with two simple approximation approaches, one of which uses Fact 1(i) and the other uses Fact 1(ii′).

▶ **Lemma 1.** *Let $f$ and $g$ be monotone step functions with total complexity $\ell$ and ranges contained in $\{-\infty, 0\} \cup [A, B]$. Then we can compute a monotone step function that approximates $f \oplus g$ with factor $1 + O(\varepsilon)$ and complexity $\widetilde{O}(\frac{1}{\varepsilon})$ in*

**(i)** *$O(\ell) + \widetilde{O}((\frac{1}{\varepsilon})^2 / 2^{\Omega(\sqrt{\log(1/\varepsilon)})})$ time in general;*

**(ii)** *$O(\ell) + \widetilde{O}(\frac{1}{\varepsilon})$ time if $g$ is $p$-uniform and pseudo-concave.*[1]

**Proof.** For a given $b \in [A, B]$, we first describe how to compute an approximation[2] of $\min\{f \oplus g, b\}$ with additive error $O(\varepsilon b)$ and complexity $O(\frac{1}{\varepsilon})$:

**(i)** In the general case, we just round the function values of $\min\{f, b\}$ and $\min\{g, b\}$ down to multiples of $\varepsilon b$ (in $O(\ell)$ time). The new functions $\min\{f, b\}$ and $\min\{g, b\}$ are $(\varepsilon b)$-uniform with complexity $O(\frac{1}{\varepsilon})$. We can then compute $\min\{f \oplus g, b\}$ by Fact 1(i) in $O((\frac{1}{\varepsilon})^2 / 2^{\Omega(\sqrt{\log(1/\varepsilon)})})$ time.

**(ii)** In the case when $g$ is $p$-uniform and pseudo-concave, we consider two subcases:

- Case 1: $p \geq \varepsilon b$. We may assume that $p$ is a multiple of $\varepsilon b$, by adjusting $\varepsilon$ by an $O(1)$ factor. We round the function values of $\min\{f, b\}$ down to multiples of $\varepsilon b$. The new function $f$ is $(\varepsilon b)$-uniform. We can then compute $\min\{f \oplus g, b\}$ by Fact 1(ii′) in $\widetilde{O}(\frac{1}{\varepsilon} + \frac{b}{p} \cdot \frac{p}{\varepsilon b}) = \widetilde{O}(\frac{1}{\varepsilon})$ time.

---

[1] Throughout, we use the $\widetilde{O}$ notation to hide polylogarithmic factors not just in $n$ and $\frac{1}{\varepsilon}$, but also other parameters such as $\frac{B}{A}$ and $\frac{1}{\delta_0}$. Eventually, these parameters will be set to values which are polynomial in $n$ and $\frac{1}{\varepsilon}$.

[2] $\min\{f, b\}$ denotes the function $F$ with $F(x) = \min\{f(x), b\}$.

- Case 2: $\varepsilon b > p$. We may assume that $\varepsilon b$ is a multiple of $p$, by adjusting $\varepsilon$ by an $O(1)$ factor. We can round the function values of $\min\{g, b\}$ down to multiples of $\varepsilon b$ while preserving pseudo-concavity (since each difference of consecutive $x$-breakpoints in the new function is the sum of $\frac{\varepsilon b}{p}$ differences in the old function); the rounding causes additive error $O(\varepsilon b)$. We have now effectively made $p$ equal to $\varepsilon b$, and so Case 1 applies.

To finish the proof of (i) or (ii), we apply the above procedure to every $b \in [A, B]$ that is a power of 2, and return the upper envelope of the resulting $O(\log \frac{B}{A})$ functions. This gives a $(1 + O(\varepsilon))$-factor approximation of $f \oplus g$ (since in the case when the function value lies between $b/2$ and $b$, the $O(\varepsilon b)$ additive error for $\min\{f \oplus g, b\}$ implies approximation factor $1 + O(\varepsilon)$). The running time increases by a factor of $O(\log \frac{B}{A})$. ◀

▶ **Lemma 2.** *Let $f_1, \ldots, f_m$ be monotone step functions with total complexity $O(n)$ and ranges contained in $\{-\infty, 0\} \cup [A, B]$. Then we can compute a monotone step function that approximates $f_1 \oplus \cdots \oplus f_m$ with complexity $\widetilde{O}(\frac{1}{\varepsilon})$ in*

**(i)** *$O(n) + \widetilde{O}((\frac{1}{\varepsilon})^2 m / 2^{\Omega(\sqrt{\log(1/\varepsilon)})})$ time in general;*

**(ii)** *$O(n) + \widetilde{O}(\frac{1}{\varepsilon} m^2)$ time if every $f_i$ is $p_i$-uniform and pseudo-concave for some $p_i$.*

**Proof.**

**(i)** We use a simple divide-and-conquer algorithm: recursively approximate $f_1 \oplus \cdots \oplus f_{m/2}$ and $f_{m/2+1} \oplus \cdots \oplus f_m$, and return a $(1 + O(\varepsilon))$-factor approximation of the $\oplus$ of the two resulting functions, by using Lemma 1(i). Since the recursion tree has $O(m)$ nodes each with cost $\widetilde{O}((\frac{1}{\varepsilon})^2 / 2^{\Omega(\sqrt{\log(1/\varepsilon)})})$ (except for the leaf nodes, which have a total additional cost $O(n)$), the total time is $O(n) + \widetilde{O}(m(\frac{1}{\varepsilon})^2 / 2^{\Omega(\sqrt{\log(1/\varepsilon)})})$. However, since the depth of the recursion is $\log m$, the approximation factor increases to $(1+O(\varepsilon))^{\log m} = 1 + O(\varepsilon \log m)$. We can adjust $\varepsilon$ by a factor of $\log m$, which increases the running time only by polylogarithmic factors.

**(ii)** We use a simple incremental algorithm: initialize $f = f_1$; for each $i = 2, \ldots, m$, compute a $(1 + O(\varepsilon))$-factor approximation of $f \oplus f_i$, by using Lemma 1(ii), and reset $f$ to this new function. The total time is $O(n) + \widetilde{O}(m \cdot \frac{1}{\varepsilon})$. However, the approximation factor increases to $(1 + O(\varepsilon))^m = 1 + O(\varepsilon m)$. We can adjust $\varepsilon$ by a factor of $m$, which increases the running time to $O(n) + \widetilde{O}(m \cdot \frac{1}{\varepsilon/m})$. ◀

Both the divide-and-conquer and incremental methods in Lemmas 2(i) and (ii) are known, or are reinterpretations of known methods [9, 10, 14]. The divide-and-conquer method is similar to the "merge-and-reduce" technique often used in streaming (and in fact immediately implies a space-efficient streaming algorithm for the 0-1 knapsack problem). As $m = \widetilde{O}(\frac{1}{\varepsilon})$, both method happen to yield an 0-1 knapsack algorithm with roughly the same time bound, near $\widetilde{O}(n + (\frac{1}{\varepsilon})^3)$.

## 4 A Simpler Algorithm with Exponent 5/2

To improve the running time, we use a very simple idea: just combine the two methods!

▶ **Theorem 3.** *Let $f_1, \ldots, f_m$ be monotone step functions with total complexity $O(n)$ and ranges contained in $\{-\infty, 0\} \cup [A, B]$. If every $f_i$ is $p_i$-uniform and pseudo-concave for some $p_i$, then we can compute a monotone step function that approximates $f_1 \oplus \cdots \oplus f_m$ with factor $1 + O(\varepsilon)$ and complexity $\widetilde{O}(\frac{1}{\varepsilon})$ in $O(n) + \widetilde{O}((\frac{1}{\varepsilon})^{3/2} m / 2^{\Omega(\sqrt{\log(1/\varepsilon)})})$ time.*

**Proof.** Divide the set of given functions into $r$ subsets of $\frac{m}{r}$ functions, for a parameter $r$ to be specified later. For each subset, approximate the $\oplus$ of its $\frac{m}{r}$ pseudo-concave functions by Lemma 2(ii). Finally, return an approximation of the $\oplus$ of the $r$ resulting functions, by using Lemma 2(i). The total time is

$$O(n) + \widetilde{O}\left(r\frac{1}{\varepsilon}\left(\frac{m}{r}\right)^2 + (r-1)\left(\frac{1}{\varepsilon}\right)^2 / 2^{\Omega(\sqrt{\log(1/\varepsilon)})}\right).$$

Setting $r = \left\lceil \sqrt{\varepsilon} m 2^{c\sqrt{\log(1/\varepsilon)}} \right\rceil$ for a sufficiently small constant $c$ yields the theorem. ◀

▶ **Corollary 4.** *There is a $(1+\varepsilon)$-approximation algorithm for 0-1 knapsack with running time $O(n \log \frac{1}{\varepsilon} + (\frac{1}{\varepsilon})^{5/2}/2^{\Omega(\sqrt{\log(1/\varepsilon)})})$.*

**Proof.** We apply the theorem with $m = \widetilde{O}(\frac{1}{\varepsilon})$ and $\frac{B}{A} = O(\frac{n^2}{\varepsilon})$. Initial sorting of the $w_i$'s takes $O(n \log n)$ time. (Note that we may assume $n \le (\frac{1}{\varepsilon})^{O(1)}$, for otherwise we can switch to Lawler's algorithm [12]. In particular, $\log n = O(\log \frac{1}{\varepsilon})$.) ◀

This completes the description of our new simpler algorithm.

## 5 Closing Remarks

We have described how to compute approximations of the optimal value, but not a corresponding subset of items. To output the subset, we can modify the algorithms to record extra information whenever we apply Fact 1 to compute the $\oplus$ of two functions $f$ and $g$. Namely, for each step in the step function $f \oplus g$, we store the corresponding steps from $f$ and $g$ that define its $y$-value. Then a solution achieving the returned profit value can be retrieved by proceeding backwards in a straightforward way (as in most dynamic programming algorithms). Since we have performed a total of $\widetilde{O}(m)$ $\oplus$ operations to functions with complexity $\widetilde{O}(\frac{1}{\varepsilon})$, the total space usage is $O(n) + \widetilde{O}(\frac{1}{\varepsilon}m) = O(n) + \widetilde{O}((\frac{1}{\varepsilon})^2)$. (The space bound can probably be reduced by known space-reduction techniques [13, 9] on dynamic programming.)

The main open question is whether the running time can be improved to near $O(n+(\frac{1}{\varepsilon})^2)$. Our improvements in the appendix will hopefully inspire future work. Note that any improved subquadratic algorithm for $(\min,+)$-convolution would automatically lead to further improvements on the time bounds of our algorithms. The truly subquadratic algorithm by Chan and Lewenstein [3] for bounded monotone integer sequences does not seem applicable here for arbitrary weights, unfortunately. In the opposite direction, a variant of a recent reduction of Cygan et al. [5] or Künnemann et al. [11] shows that there is no algorithm for 0-1 (or unbounded) knapsack with $O((n+\frac{1}{\varepsilon})^{2-\delta})$ running time, assuming the conjecture that there is no truly subquadratic algorithm for $(\min,+)$-convolution.

## References


1 Alok Aggarwal, Maria M. Klawe, Shlomo Moran, Peter W. Shor, and Robert E. Wilber. Geometric applications of a matrix-searching algorithm. *Algorithmica*, 2:195–208, 1987. doi:10.1007/BF01840359.

2 David Bremner, Timothy M. Chan, Erik D. Demaine, Jeff Erickson, Ferran Hurtado, John Iacono, Stefan Langerman, Mihai Patrascu, and Perouz Taslakian. Necklaces, convolutions, and X+Y. *Algorithmica*, 69(2):294–314, 2014. doi:10.1007/s00453-012-9734-3.

3 Timothy M. Chan and Moshe Lewenstein. Clustered integer 3sum via additive combinatorics. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 31–40. ACM, 2015. doi:10.1145/2746539.2746568.

4 Timothy M. Chan and Ryan Williams. Deterministic apsp, orthogonal vectors, and more: Quickly derandomizing razborov-smolensky. In Robert Krauthgamer, editor, *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1246–1255. SIAM, 2016. doi:10.1137/1.9781611974331.ch87.

5 Marek Cygan, Marcin Mucha, Karol Wegrzycki, and Michal Wlodarczyk. On problems equivalent to (min, +)-convolution. In Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl, editors, *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*, volume 80 of *LIPIcs*, pages 22:1–22:15. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017. doi:10.4230/LIPIcs.ICALP.2017.22.

6 Oscar H. Ibarra and Chul E. Kim. Fast approximation algorithms for the knapsack and sum of subset problems. *J. ACM*, 22(4):463–468, 1975. doi:10.1145/321906.321909.

7 Klaus Jansen and Stefan Erich Julius Kraft. A faster FPTAS for the unbounded knapsack problem. In Zsuzsanna Lipták and William F. Smyth, editors, *Combinatorial Algorithms - 26th International Workshop, IWOCA 2015, Verona, Italy, October 5-7, 2015, Revised Selected Papers*, volume 9538 of *Lecture Notes in Computer Science*, pages 274–286. Springer, 2015. doi:10.1007/978-3-319-29516-9_23.

8 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.

9 Hans Kellerer and Ulrich Pferschy. A new fully polynomial time approximation scheme for the knapsack problem. *J. Comb. Optim.*, 3(1):59–71, 1999. doi:10.1023/A:1009813105532.

10 Hans Kellerer and Ulrich Pferschy. Improved dynamic programming in connection with an FPTAS for the knapsack problem. *J. Comb. Optim.*, 8(1):5–11, 2004. doi:10.1023/B:JOCO.0000021934.29833.6b.

11 Marvin Künnemann, Ramamohan Paturi, and Stefan Schneider. On the fine-grained complexity of one-dimensional dynamic programming. In Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl, editors, *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*, volume 80 of *LIPIcs*, pages 21:1–21:15. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017. doi:10.4230/LIPIcs.ICALP.2017.21.

12 Eugene L. Lawler. Fast approximation algorithms for knapsack problems. *Math. Oper. Res.*, 4(4):339–356, 1979. doi:10.1287/moor.4.4.339.

13 M. J. Magazine and O. Oguz. A fully polynomial approximation algorithm for the 0-1 knapsack problem. *Europ. J. Oper. Res.*, 123:325–332, 2000. doi:10.1016/0377-2217(84)90286-8.

14 Donguk Rhee. Faster fully polynomial approximation schemes for knapsack problems. Master's thesis, MIT, 2015. URL: https://dspace.mit.edu/bitstream/handle/1721.1/98564/920857251-MIT.pdf.

15 Sartaj Sahni. Approximate algorithms for the 0/1 knapsack problem. *J. ACM*, 22(1):115–124, 1975. doi:10.1145/321864.321873.

16 Ryan Williams. Faster all-pairs shortest paths via circuit complexity. In David B. Shmoys, editor, *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 664–673. ACM, 2014. doi:10.1145/2591796.2591811.

# A An Improved Algorithm with Exponent 12/5

In the appendix, we show how the ideas in our $\widetilde{O}(n + (\frac{1}{\varepsilon})^{5/2})$ algorithm can lead to further improvements.

In what follows, we make an extra input assumption that all the $p_i$'s are within a constant factor of each other. This case is sufficient to solve the general problem, because we can divide the input items into $O(\log \frac{n}{\varepsilon})$ classes with the stated property, and then merge via Lemma 2(i). By rescaling, we assume that all $p_i$'s are in $[1, 2]$. In this case, the optimal fractional solution approximates the optimal integral solution with $O(1)$ additive error (since rounding the fractional solution causes the loss of at most one item), and the optimal fractional solution can be found by the standard greedy algorithm. In other words, with $O(1)$ additive error, we can approximate $f_I$ by the step function with function values $-\infty, 0, p_1, p_1 + p_2, \ldots, p_1 + \cdots + p_n$ and the $x$-breakpoints $0, w_1, w_1 + w_2, \ldots, w_1 + \cdots + w_n$, after arranging the items in nondecreasing order of $w_i/p_i$. A solution with $O(1)$ additive error has approximation factor $1 + O(\varepsilon)$ if the optimal value is $\Omega(\frac{1}{\varepsilon})$. Thus, we may assume that the optimal value is upper-bounded by $B = O(\frac{1}{\varepsilon})$.

## A.1 Refining the Second Method

To obtain further improvement, we will refine the second incremental method in Lemma 2(ii). Recall that the inefficency of that method is due to the need to round in every iteration. We observe that if all the $p_i$'s are integer multiples of a small set of values, we do not need to round as often, as explained in the following lemma.

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

▶ **Lemma 5.** *Let $f_1, \ldots, f_m$ be monotone step functions and ranges contained in $\{-\infty, 0\} \cup [1, B]$. Let $\Delta \subset [\delta_0, 2\delta_0]$ and let $b \in [1, B]$. If every $f_i$ is $p_i$-uniform and pseudo-concave for some $p_i \in [1, 2]$ which is a $\Delta$-multiple, then we can compute a monotone step function that approximates $\min\{f_1 \oplus \cdots \oplus f_m, b\}$ with additive error $O(|\Delta|\delta_0)$ in $\widetilde{O}(\frac{1}{\delta_0} bm)$ time.*

**Proof.** We use a simple incremental algorithm: Initialize $f = -\infty$. In each iteration, take one $\delta \in \Delta$. Round the function values of $\min\{f, b\}$ down to multiples of $\delta$, which incurs an additive error of $O(\delta) = O(\delta_0)$. The new function $\min\{f, b\}$ is now $\delta$-uniform, with complexity $O(\frac{b}{\delta})$. For each not-yet-considered function $f_i$ with $p_i$ being a multiple of $\delta$, reset $f$ to $\min\{f \oplus f_i, b\}$, which can be computed exactly by Lemma 1(ii′) in $\widetilde{O}(\frac{b}{\delta} + \frac{b}{p_i} \cdot \frac{p_i}{\delta}) = \widetilde{O}(\frac{b}{\delta_0})$ time. Repeat for the next $\delta \in \Delta$. The total time is $\widetilde{O}(\frac{b}{\delta_0} m)$. The total additive error is $O(|\Delta|\delta_0)$. ◀

## A.2 A Number-Theoretic Lemma

To use the above lemma efficiently, we need the following combinatorial/number-theoretic lemma, stating that all numbers can be approximated well by integer multiples of a small set of values.

▶ **Lemma 6.** *Given $\varepsilon < \delta_0 < 1$, there exists a set $\Delta \subset [\delta_0, 2\delta_0]$ of size $\widetilde{O}(\frac{\delta_0}{\varepsilon})$, such that every value $p \in [1, 2]$ can be approximated by a $\Delta$-multiple with additive error $O(\varepsilon)$.*

*The set can be constructed in $\widetilde{O}(\frac{1}{\varepsilon})$ time by a randomized algorithm.*

**Proof.** Let $P = \{1, 1 + \varepsilon, 1 + 2\varepsilon, 1 + 3\varepsilon, \ldots, 1 + \lfloor \frac{1}{\varepsilon} \rfloor \varepsilon\}$. Then $|P| = O(\frac{1}{\varepsilon})$. In the stated property, it suffices to consider only values $p$ in $P$.

Given $p \in P$ and $\delta \in [\delta_0, 2\delta_0]$, $p$ is approximated by a multiple of $\delta$ with additive error $\varepsilon$ iff $0 \le p - i\delta \le \varepsilon$ for some integer $i$, i.e., iff $\delta$ lies in the set

$$I_p \;:=\; [\delta_0, 2\delta_0] \;\cap\; \bigcup_i \left[\frac{p-\varepsilon}{i}, \frac{p}{i}\right]$$

where the union is over all integers $i$ between $\frac{1-\varepsilon}{2\delta_0}$ and $\frac{2}{\delta_0}$. Our goal is to find a small set $\Delta$ of size $\widetilde{O}(\frac{\delta_0}{\varepsilon})$ that hits $I_p$ for all $p \in P$.

Now, each set $I_p$ is a union of $\Theta(\frac{1}{\delta_0})$ disjoint intervals of length $\Theta(\frac{\varepsilon}{1/\delta_0}) = \Theta(\varepsilon\delta_0)$ and thus has measure $\Theta(\varepsilon)$. Thus, a uniformly distributed $\delta \in [\delta_0, 2\delta_0]$ lies in $I_p$ with probability $\Theta(\frac{\varepsilon}{\delta_0})$. For a simple randomized construction, we can just choose $O(\frac{\delta_0}{\varepsilon} \log |P|)$ values uniformly from $[\delta_0, 2\delta_0]$ and obtain a set $\Delta$ that hits every $I_p$ with high probability $1 - O(\frac{1}{|P|^c})$ for any constant $c > 1$. This yields a Monte Carlo algorithm, but it can be converted to Las Vegas, since we can verify correctness of $\Delta$ by generating and sorting all $\Delta$-multiples in $[1, 2]$ in $\widetilde{O}(|\Delta|\frac{1}{\delta_0}) = \widetilde{O}(\frac{1}{\varepsilon})$ time. ◀

## A.3 Putting the Refined Second Method Together

Applying Lemma 5 together with Lemma 6 now gives the following new result:

▶ **Lemma 7.** *Let $f_1, \ldots, f_m$ be monotone step functions with ranges contained in $\{-\infty, 0\} \cup [1, B]$. 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 $1 + O(\varepsilon)$ and complexity $\widetilde{O}(\frac{1}{\varepsilon})$ in $\widetilde{O}(\frac{1}{\varepsilon}\sqrt{B}m)$ expected time, assuming $B = \widetilde{O}(\frac{1}{\varepsilon})$.*

**Proof.** For a given $b \in [1, B]$, we first describe how to compute an approximation of $\min\{f_1 \oplus \cdots \oplus f_m, b\}$ with additive error $O(\varepsilon b)$ and complexity $O(\frac{1}{\varepsilon})$:

- Construct the set $\Delta$ of size $\widetilde{O}(\frac{\delta_0}{\varepsilon})$ from Lemma 6 in $\widetilde{O}(\frac{1}{\varepsilon})$ expected time for some parameter $\delta_0 > \varepsilon$ to be specified later. Generate and sort all $\Delta$-multiples in $[1, 2]$ in $\widetilde{O}(|\Delta|\frac{1}{\delta_0}) = \widetilde{O}(\frac{1}{\varepsilon})$ time. For each $p_i$, round it down to a $\Delta$-multiple with additive error $O(\varepsilon)$ (e.g., by binary search) and change $f_i$ accordingly. This multiplies the approximation factor by $1 + O(\varepsilon)$, and thus increases the additive error by at most $O(\varepsilon b)$. Now apply Lemma 5. The additive error is $O(|\Delta|\delta_0) = O(\frac{\delta_0^2}{\varepsilon}) = O(\varepsilon b)$ by choosing $\delta_0 := \varepsilon\sqrt{b}$. The running time is $\widetilde{O}(\frac{1}{\delta_0}bm) = \widetilde{O}(\frac{1}{\varepsilon}\sqrt{b}m)$. Note that the complexity of the resulting function can be reduced to $O(\frac{1}{\varepsilon})$ by rounding down to multiples of $\varepsilon b$.

To finish, we apply the above procedure to every $b \in [1, B]$ that is a power of 2, and then return the upper envelope of the resulting $O(\log B)$ functions. This gives a $(1 + O(\varepsilon))$-factor approximation of $\min\{f_1 \oplus \cdots \oplus f_m, B\}$. The running time increases by a factor of $O(\log B)$. ◀

As $m = \widetilde{O}(\frac{1}{\varepsilon})$ and $B = O(\frac{1}{\varepsilon})$ in our application, the above lemma immediately gives an alternative algorithm with near $\widetilde{O}(n + (\frac{1}{\varepsilon})^{5/2})$ running time. Notice that this alternative is based solely on the second incremental method, without combining with the first divide-and-conquer method. Naturally, it suggests the possibility that a combination might lead to a further improvement...

## A.4 Putting Everything Together

To this end, we first show that if the size of $\Delta$ could be reduced (from $O(\frac{\delta_0}{\varepsilon})$ to, say, $O(\frac{\delta_0}{r\varepsilon})$) for some particular choice of $\delta_0$, then Lemma 7 could be improved (from $\widetilde{O}(\frac{1}{\varepsilon}\sqrt{B}m)$ time to $\widetilde{O}(\frac{1}{r^{1/4}\varepsilon}\sqrt{B}m)$), by bootstrapping:

▶ **Lemma 8.** *Let $f_1, \dots, f_m$ be monotone step functions with ranges contained in $\{-\infty, 0\} \cup [1, B]$. Let $\Delta \subset [\delta_0, 2\delta_0]$ be a set of size $O(\frac{\delta_0}{r\varepsilon})$ for some $r \in [1, B^2]$ where $\delta_0 := r^{1/4}\varepsilon\sqrt{B}$. If every $f_i$ is $p_i$-uniform and pseudo-concave for some $p_i \in [1, 2]$ which is a $\Delta$-multiple, then we can compute a monotone step function that approximates $\min\{f_1 \oplus \cdots \oplus f_m, B\}$ with factor $1 + O(\varepsilon)$ and complexity $\widetilde{O}(\frac{1}{\varepsilon})$ in $\widetilde{O}(\frac{1}{r^{1/4}\varepsilon}\sqrt{B}m)$ expected time, assuming $B = \widetilde{O}(\frac{1}{\varepsilon})$.*

**Proof.**

1. First compute an approximation of $\min\{f_1 \oplus \cdots \oplus f_m, B/s\}$ with factor $1 + O(\varepsilon)$ and complexity $\widetilde{O}(\frac{1}{\varepsilon})$ by Lemma 7 in $\widetilde{O}(\frac{1}{\varepsilon}\sqrt{B/s}\, m)$ time, for some parameter $s \geq 1$ to be specified later.
2. Next compute an approximation of $\min\{f_1 \oplus \cdots \oplus f_m, B\}$ with additive error $O(\varepsilon B/s)$. This can be done by applying Lemma 5. The additive error is $O(|\Delta|\delta_0) = O(\frac{\delta_0^2}{r\varepsilon}) = O(\varepsilon B/s)$ by choosing $\delta_0 := \varepsilon\sqrt{(r/s)B}$. The running time is $\widetilde{O}(\frac{1}{\delta_0}Bm) = \widetilde{O}(\frac{1}{\varepsilon}\sqrt{(s/r)B}m)$.

To finish, we return the upper envelope of the two resulting functions. This gives a $(1+O(\varepsilon))$-factor approximation of $\min\{f_1 \oplus \cdots \oplus f_m, B\}$ (since in the case when the function value exceeds $B/s$, the $O(\varepsilon B/s)$ additive error in the second function implies $1+O(\varepsilon)$ approximation factor). Note that the complexity of the resulting function can be reduced to $\widetilde{O}(\frac{1}{\varepsilon})$ by rounding down to powers of $1 + \varepsilon$, which multiplies the approximation factor by $1 + O(\varepsilon)$.

The total running time

$$\widetilde{O}\left(\frac{1}{\varepsilon}\sqrt{B/s}\, m + \frac{1}{\varepsilon}\sqrt{(s/r)B}m\right)$$

is $\widetilde{O}(\frac{1}{r^{1/4}\varepsilon}\sqrt{B}m)$ by setting $s := \sqrt{r}$. ◀

To reduce the size of $\Delta$, we combine the above with the first divide-and-conquer method from Lemma 2(ii), which finally leads to our new improved result after fine-tuning the choice of parameters:

▶ **Theorem 9.** *Let $f_1, \dots, f_m$ be monotone step functions with ranges contained in $\{-\infty, 0\} \cup [A, B]$. If every $f_i$ is $p_i$-uniform and pseudo-concave with $p_i \in [1, 2]$, then we can compute a monotone step function that approximates $\min\{f_1 \oplus \cdots \oplus f_m, B\}$ with factor $1 + O(\varepsilon)$ and complexity $\widetilde{O}(\frac{1}{\varepsilon})$ in $\widetilde{O}((\frac{1}{\varepsilon})^{12/5}/2^{\Omega(\sqrt{\log(1/\varepsilon)})})$ expected time if $m, B = \widetilde{O}(\frac{1}{\varepsilon})$.*

**Proof.** Construct the set $\Delta$ of size $\widetilde{O}(\frac{\delta_0}{\varepsilon})$ from Lemma 6 with $\delta_0 := r^{1/4}\varepsilon\sqrt{B}$ for some parameter $r$ to be specified later. Generate and sort all $\Delta$-multiples in $[1, 2]$ in $\widetilde{O}(|\Delta|\frac{1}{\delta_0}) = \widetilde{O}(\frac{1}{\varepsilon})$ time. For each $p_i$, round it down to a $\Delta$-multiple with additive error $O(\varepsilon)$ and change $f_i$ accordingly. This multiplies the approximation factor by $1 + O(\varepsilon)$.

Divide $\Delta$ into $r$ subsets $\Delta_1, \dots, \Delta_r$ each of size $\widetilde{O}(\frac{\delta_0}{r\varepsilon})$. For each subset $\Delta_j$, approximate the $\oplus$ of all not-yet-considered functions $f_i$ with $p_i$ being a $\Delta_j$-multiple, by Lemma 8. Finally, return an approximation of the $\oplus$ of the resulting $r$ functions by using Lemma 2(i). The total time is

$$\widetilde{O}\left(\frac{1}{r^{1/4}\varepsilon}\sqrt{B}m + r\left(\frac{1}{\varepsilon}\right)^2/\,2^{\Omega(\sqrt{\log(1/\varepsilon)})}\right) = \widetilde{O}\left(\frac{1}{r^{1/4}}\left(\frac{1}{\varepsilon}\right)^{5/2} + r\left(\frac{1}{\varepsilon}\right)^2/\,2^{\Omega(\sqrt{\log(1/\varepsilon)})}\right). \tag{1}$$

Setting $r = \left\lceil (\frac{1}{\varepsilon})^{2/5} 2^{c\sqrt{\log(1/\varepsilon)}} \right\rceil$ and $\delta_0 = \varepsilon^{2/5} 2^{(c/3)\sqrt{\log(1/\varepsilon)}}$ for a sufficiently small constant $c$ yields the theorem. ◀

▶ **Corollary 10.** *There is a $(1+\varepsilon)$-approximation algorithm for 0-1 knapsack with expected running time $O(n \log \frac{1}{\varepsilon} + (\frac{1}{\varepsilon})^{12/5}/2^{\Omega(\sqrt{\log(1/\varepsilon)})})$.*

**Proof.** In the case when all $p_i \in [1,2]$, we apply the theorem with $m = \widetilde{O}(\frac{1}{\varepsilon})$ and $B = O(\frac{1}{\varepsilon})$. In addition, we take the greedy approximation and return the upper envelope of the two resulting functions. As noted earlier, the general case reduces to the $p_i \in [1,2]$ case, by merging $O(\log \frac{n}{\varepsilon})$ functions via Lemma 2(i), taking additional $(\frac{1}{\varepsilon})^2/2^{\Omega(\sqrt{\log(1/\varepsilon)})}$ time. Initial sorting takes $O(n \log n)$ time. (As before, we may assume $n \le (\frac{1}{\varepsilon})^{O(1)}$, for otherwise we can switch to Lawler's algorithm.) ◀

## A.5 Derandomization

Our algorithm is randomized because of Lemma 6. In the proof of Lemma 6, instead of random sampling, we can run the standard greedy algorithm for hitting set, with $O(\frac{\delta_0}{\varepsilon} \log |P|)$ iterations. We gather all the intervals of $I_p$ over all $p \in P$. In each iteration, we find a *deepest* point $\lambda$, i.e., a point that hits the most intervals, and delete the intervals in all the sets $I_p$ that are hit by $\lambda$. Initially, the total number of intervals is $O(\frac{1}{\delta_0}|P|) = O(\frac{1}{\delta_0 \varepsilon})$, and this bounds the total number of deletions as well. It is not difficult to design a data structure that supports deletions, searching for the deepest point, and searching for the intervals hit by a given point, all in logarithmic time per operation. Thus, the total time is $\widetilde{O}(\frac{1}{\delta_0 \varepsilon})$, which is at most $\widetilde{O}((\frac{1}{\varepsilon})^2)$.

This adds an $\widetilde{O}((\frac{1}{\varepsilon})^2)$ term to the time bounds of Lemmas 7 and 8, and an $\widetilde{O}(r(\frac{1}{\varepsilon})^2)$ to (1), which slightly affects the final bound in the extra superpolylogarithmic factors. We can fix this by modifying the proof of Lemma 7: if $b \ge (\frac{1}{\varepsilon})^{0.1}$, we proceed as before and notice that the construction time for $\Delta$ is $\widetilde{O}(\frac{1}{\delta_0 \varepsilon}) \le O(\frac{1}{\varepsilon^{2-\Omega(1)}})$; but if $b < (\frac{1}{\varepsilon})^{0.1}$, we can switch to using a singleton set $\Delta = \{\varepsilon\}$ with $\delta_0 = \varepsilon$, which leads to running time $\widetilde{O}(\frac{1}{\varepsilon} bm) \le \widetilde{O}((\frac{1}{\varepsilon})^{1.1} m)$. All this adds an extra $\widetilde{O}((\frac{1}{\varepsilon})^{1.1} m + r \cdot (\frac{1}{\varepsilon})^{2-\Omega(1)})$ term to (1), which does not affect the final bound.

▶ **Corollary 11.** *The algorithm in Corollary 10 can be made deterministic.*

As before, the algorithm can be modified to compute not just an approximation of the optimal value but also a corresponding subset of items.

# B Variants for Small $n$

We note two further results which are better when $n$ is small relative to $\frac{1}{\varepsilon}$.

▶ **Corollary 12.** *There is a $(1+\varepsilon)$-approximation algorithm for 0-1 knapsack with expected running time $\widetilde{O}(\frac{1}{\varepsilon} n^{3/2})$.*

**Proof.** In the case when all $p_i \in [1,2]$, an $\widetilde{O}(\frac{1}{\varepsilon} n^{3/2})$ time bound follows directly from Lemma 7, since the number of distinct $p_i$ values is $m \le n$, and a trivial upper bound for the maximum optimal value is $B \le 2n$.

As noted earlier, the general case reduces to the $p_i \in [1,2]$ case, by merging $O(\log \frac{n}{\varepsilon})$ functions via Lemma 2(i), taking additional $(\frac{1}{\varepsilon})^2/2^{\Omega(\sqrt{\log(1/\varepsilon)})}$ time. To avoid the merging cost, we need to bypass Lemma 2(i). First, we can easily generalize Lemmas 5 and 7 to compute $f \oplus f_1 \oplus \cdots \oplus f_m$ for an arbitrary monotone step function $f$ with complexity $\widetilde{O}(\frac{1}{\varepsilon})$. We can then apply Lemma 7 iteratively, with each iteration handling all $p_i \in [2^j, 2^{j+1}]$ (which can be rescaled to $[1,2]$), in increasing order of $j$. The approximation factor increases to $(1+\varepsilon)^{O(\log \frac{B}{A})} = 1 + O(\varepsilon \log \frac{B}{A})$. We can adjust $\varepsilon$ by a logarithmic factor. ◀

▶ **Corollary 13.** *There is a $(1+\varepsilon)$-approximation algorithm for 0-1 knapsack with running time $O(((\frac{1}{\varepsilon})^{4/3}n + (\frac{1}{\varepsilon})^2)/2^{\Omega(\sqrt{\log(1/\varepsilon)})})$.*

**Proof.** Divide the input items into $r$ subsets of $\frac{n}{r}$ items each. For each subset, apply the method from Corollary 12. Finally, return an approximation of the $\oplus$ of the resulting $r$ functions by using Lemma 2(i). The total time is

$$\widetilde{O}\left(r\,\frac{1}{\varepsilon}\left(\frac{n}{r}\right)^{3/2} + r\left(\frac{1}{\varepsilon}\right)^2/2^{\Omega(\sqrt{\log(1/\varepsilon)})}\right).$$

Setting $r = \left\lceil \varepsilon^{2/3} n 2^{c\sqrt{\log(1/\varepsilon)}} \right\rceil$ for a sufficiently small constant $c$ yields the corollary. This algorithm can be made deterministic as in Section A.5. The derandomization adds an extra $\widetilde{O}((\frac{1}{\varepsilon})^{1.1}m + r \cdot (\frac{1}{\varepsilon})^{2-\Omega(1)})$ term, which does not affect the final bound. ◀

Corollary 12 gives the current best bound for $n \ll (\frac{1}{\varepsilon})^{2/3}$, and Corollary 13 gives the current best bound for $(\frac{1}{\varepsilon})^{2/3} \ll n \ll (\frac{1}{\varepsilon})^{16/15}$. For example, when $n = \frac{1}{\varepsilon}$, Corollary 13 gives $\widetilde{O}((\frac{1}{\varepsilon})^{7/3})$, which is a little better than $\widetilde{O}((\frac{1}{\varepsilon})^{12/5})$. This case is of interest, since for certain related problems such as subset-sum or unbounded knapsack (but unfortunately not for the general 0-1 knapsack problem), there are efficient preprocessing algorithms that can reduce the input size $n$ to $\widetilde{O}(\frac{1}{\varepsilon})$.