No Polynomial Kernels for Knapsack

Klaus Heeger
Department of Industrial Engineering and Management, Ben-Gurion University of the Negev, Beer-Sheva, Israel

Danny Hermelin
Department of Industrial Engineering and Management, Ben-Gurion University of the Negev, Beer-Sheva, Israel

Matthias Mnich
Institute for Algorithms and Complexity, Hamburg University of Technology, Hamburg, Germany

Dvir Shabtay
Department of Industrial Engineering and Management, Ben-Gurion University of the Negev, Beer-Sheva, Israel

Abstract

This paper focuses on kernelization algorithms for the fundamental Knapsack problem. A kernelization algorithm (or kernel) is a polynomial-time reduction from a problem onto itself, where the output size is bounded by a function of some problem-specific parameter. Such algorithms provide a theoretical model for data reduction and preprocessing and are central in the area of parameterized complexity. In this way, a kernel for Knapsack for some parameter k reduces any instance of Knapsack to an equivalent instance of size at most $f(k)$ in polynomial time, for some computable function f. When $f(k) = k^{O(1)}$ then we call such a reduction a polynomial kernel.

Our study focuses on two natural parameters for Knapsack: The number $w_\#$ of different item weights, and the number $p_\#$ of different item profits. Our main technical contribution is a proof showing that Knapsack does not admit a polynomial kernel for any of these two parameters under standard complexity-theoretic assumptions. Our proof discovers an elaborate application of the standard kernelization lower bound framework, and develops along the way novel ideas that should be useful for other problems as well. We complement our lower bounds by showing that Knapsack admits a polynomial kernel for the combined parameter $w_\# \cdot p_\#$.

2012 ACM Subject Classification Theory of computation → Parameterized complexity and exact algorithms; Mathematics of computing → Combinatorial optimization

Keywords and phrases Knapsack, polynomial kernels, compositions, number of different weights, number of different profits

Digital Object Identifier 10.4230/LIPIcs.ICALP.2024.83

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: https://arxiv.org/abs/2308.12593 [28]

Funding Supported by the ISF, grant No. 1070/20.
Matthias Mnich: Supported by DFG grant MN 59/4-1.

1 Introduction

This paper proves a new complexity-theoretic barrier for the classic Knapsack problem. Namely, we prove that Knapsack has no polynomial kernels when parameterized by either the number $w_\#$ of different weights, or the number $p_\#$ of different profits. Our results hold under the standard complexity-theoretic assumption NP $\not\subseteq$ coNP/poly. We also show that when both $w_\#$ and $p_\#$ are taken as a combined parameter, Knapsack does admit a polynomial kernel. Below we give a brief review of recent algorithmic progress for the

Knapsack problem, as well as a quick survey through kernelization and parameterized complexity. We then describe how our results fit into the current state of the art, and give an overview of the main techniques used for obtaining our new hardness result for Knapsack.

The Knapsack problem. Knapsack (also known as 0/1 Knapsack) is one the most fundamental and well studied problems in combinatorial optimization and theoretical computer science. In its most basic form, it is defined as follows:

Knapsack
Input: A set $\mathcal{X} = \{x_1, \ldots, x_n\}$ of n items, a weight function $w : \mathcal{X} \to \mathbb{N}$, a profit function $p : \mathcal{X} \to \mathbb{N}$, and two integers W and P.
Question: Is there a subset $\mathcal{S} \subseteq \mathcal{X}$ with total weight $w(\mathcal{S}) = \sum_{x \in \mathcal{S}} w(x) \leq W$ and total profit $p(\mathcal{S}) = \sum_{x \in \mathcal{S}} p(x) \geq P$?

Knapsack enjoys a key status in algorithmic design due to numerous reasons. First, it has many natural applications, in various practical areas such as resource allocation and scheduling. Second, it has immense educational value: Karp's NP-hardness proof (from his seminal paper [35]) is the first example of a reduction to a problem involving numbers, while the $O(Wn)$-time (or $O(Pn)$-time) algorithm by Bellman [4] from 1957 is one of the first dynamic programming algorithms, and it is still taught in most undergraduate algorithms courses to this day. And third, Knapsack has deep connections to other areas of computation: For example, one of the earliest cryptosystems by Merkle and Hellman [41] was based on Knapsack, and this was later extended to a host of other *Knapsack-type* cryptosystems [7, 14, 30, 42].

Knapsack is also important since it is both a generalization and a special case of a few other classic problems. For instance, it is a special case of the fundamental scheduling problem of minimizing the weighted number of tardy jobs on a single machine, the so-called $1||\sum w_j U_j$ problem [43, p. 19]. The variant of Knapsack where $W = P$ and $w(x) = p(x)$ for each item $x \in \mathcal{X}$ is precisely the Subset Sum problem:

Subset Sum
Input: A set $\mathcal{A} = \{a_1, \ldots, a_n\}$ of n non-negative integers and a target integer B.
Question: Is there a subset $\mathcal{A}^* \subseteq \mathcal{A}$ with $\sum_{a \in \mathcal{A}^*} a = B$?

Entire books [37, 40] are dedicated to algorithmics for Knapsack. Quite surprisingly, major algorithmic advances are still being discovered in recent years. These typically involve improvements on Bellman's classic $O(\min\{W, P\} \cdot n)$-time algorithm in cases where the maximum weight $w_{\max} = \max_x w(x)$ or the maximum profit $p_{\max} = \max_x p(x)$ (or both) are relatively small. Furthermore, they may result in faster approximation schemes. Currently, the fastest known approximation scheme runs – after a series of improvements in recent years [9, 12, 32] – in $\widetilde{O}(n + (1/\varepsilon)^{2.2})$ time [19]. Pisinger [44] was the first to present such an improvement with his $O(w_{\max} \cdot p_{\max} \cdot n)$-time algorithm. Later followed a series of papers with further improvements [2, 3, 36], culminating in an $\widetilde{O}(\min\{w_{\max}^3, p_{\max}^3\} + n)$-time[1] algorithm by Polak et al. [45], and an $\widetilde{O}(\min\{w_{\max}^{2/3} \cdot p_{\max},\ w_{\max} \cdot p_{\max}^{2/3}\} \cdot n)$-time algorithm by Bringmann and Cassis [10]. Very recently, Chen et al. [13] gave an $O(n + w_{\max}^{2.4})$-time algorithm. Subsequently, Bringmann [8] as well as Jin [33] announced $\widetilde{O}(n + w_{\max}^2)$-time algorithms (and also $\widetilde{O}(n + p_{\max}^2)$-time algorithms) for Knapsack. Cygan et al. [17]

[1] We use $\widetilde{O}(\cdot)$ to suppress polylogarithmic factors.

and independently Künnemann et al. [38] showed that there are no $O((W+n)^{2-\varepsilon})$-time algorithms for Knapsack, for any $\varepsilon > 0$, unless $(\min, +)$-convolutions can be solved in truly subquadratic time. An easy modification of their argument shows also that there are also no $O((P+n)^{2-\varepsilon})$-time algorithms under the same hypothesis. However, interestingly enough, these two lower bounds do not hold simultaneously together as shown by Bringmann and Cassis [9], who showed that Knapsack can be solved in $O((W+P)^{1.5}+n)$ time.

Parameters $w_\#$ and $p_\#$. As discussed above, the time complexity of Knapsack with respect to $w_{\max}$ and $p_{\max}$ is well understood. Thus, it makes sense to consider other natural parameters. Let $w_\# = |\{w(x_i) : i \in \{1,\dots,n\}\}|$ denote the number of *different* weights in a given Knapsack instance, and let $p_\# = |\{p(x_i) : i \in \{1,\dots,n\}|$ denote the number of different profits. These parameters can be expected to be small in several natural applications (for example, the first step of many approximation schemes is to round the profits of the items such that only $f(\varepsilon)$ many different profits remain; see, e.g., the approximation scheme by Deng et al. [19]) and have been initially studied for Subset Sum (i.e., when $w(x_i) = p(x_i)$ for all items x_i) by Fellows et al. [23] in the context of the more general *small number of numbers* paradigm. Clearly, $w_\# \leq w_{\max}$ and $p_\# \leq p_{\max}$, and so designing algorithms which are efficient in terms of $w_\#$ or $p_\#$ is more challenging than for $w_{\max}$ and $p_{\max}$. In particular, since Knapsack is NP-hard and $w_\#, p_\# \leq n$, we cannot expect algorithms with running times of the form $(w_\# \cdot n)^{O(1)}$, $(p_\# \cdot n)^{O(1)}$, or even $(w_\# \cdot p_\# \cdot n)^{O(1)}$.

Since polynomial-time algorithms with respect to $w_\#$ and $p_\#$ are unlikely to exist, it is natural to consider these two parameters in the context of parameterized complexity [16]. In this context, it is not difficult to show that Knapsack can be formulated as an Integer Linear Program (ILP) with either $O(w_\#)$ or $O(p_\#)$ variables [22]. Using one of several solvers for ILP with few variables, such as Lenstra's famous algorithm [39, 34], gives us algorithms with running times of $2^{\widetilde{O}(w_\#)} \cdot |I|$ and $2^{\widetilde{O}(p_\#)} \cdot |I|$ for Knapsack, where $|I|$ is the total encoding length of the input (see Hermelin et al. [29] for algorithms with similar running times for the more general $1||\sum w_j U_j$ problem). Such algorithms are known as *fixed-parameter algorithms* (FPT) in the terminology of parameterized complexity.

Note that both of these algorithms cannot be significantly improved assuming the Exponential Time Hypothesis (ETH). Indeed, it is known that assuming ETH, there are no $2^{o(n)}$-time algorithms for Subset Sum [11, 31]. As both $w_\#$ and $p_\#$ are bounded by n, and Subset Sum is a special case of Knapsack, this implies that, most likely, there are no $2^{o(w_\#)} \cdot n$-time or $2^{o(p_\#)} \cdot n$-time algorithms for Knapsack. Using the Strong Exponential Time Hypothesis (SETH) instead of ETH, Abboud et al. [1] showed a stronger lower bound for Subset Sum, excluding $B^{1-\varepsilon} \cdot 2^{o(n)}$-time algorithms for any $\varepsilon > 0$ (which excludes $(W+P)^{1-\varepsilon} \cdot 2^{o(w_\#+p_\#)}$-time algorithms for Knapsack). Once fixed-parameter algorithms for a particular problem – in this case, Knapsack – have been devised, the next step is to understand the kernelization complexity of the problem at hand.

Kernelization. One of the most fundamental and important techniques in parameterized complexity is kernelization [24]:

▶ **Definition 1.** *A* kernelization *algorithm (or* kernel*) for a parameterized problem* Π *is an algorithm that receives as input an instance* I *of* Π *with parameter* k *and outputs in polynomial time an instance* J *of* Π *with parameter* ℓ *such that*

- (I,k) *is a "yes"-instance for* Π *if and only if* (J,ℓ) *is a "yes"-instance for* Π*, and*
- $|J| + \ell \leq f(k)$ *for some computable function* f*.*

The expression $f(k)$ *is referred to as the* size *of the kernel.*

Thus, a kernel is a self-reduction from a problem onto itself that produces an equivalent instance with size bounded by the input parameter. In this way, kernelization may be thought of as preprocessing that aims to simplify or "kernelize" a problem instance by reducing its size while preserving some solution. Following this line of thought, problems that admit kernels of small sizes may be thought of as problems that allow efficient and effective preprocessing. For this reason, research into kernelization algorithms has seen a significant surge in recent years, and has become one of the central topics in parameterized complexity (see e.g. the monograph by Fomin et al. [24] and the numerous references within).

In the context of Knapsack, a kernelization algorithm for say parameter $w_\#$ transforms any Knapsack instance in polynomial time into an equivalent instance where the total encoding length of item weights and profits are bounded by $f(w_\#)$, for some function f. Observe that the $2^{\widetilde{O}(w_\#)} \cdot n$-time algorithm for Knapsack implies a kernel of size $2^{\widetilde{O}(w_\#)}$ [22]: Indeed, let $|I|$ denote the total encoding length of item weights and profits in a given Knapsack instance I with n items and observe that $n \leq |I|$. The kernelization algorithm can first check whether $|I| = 2^{\widetilde{O}(w_\#)}$. If this is the case, then the instance already has size bounded by $2^{\widetilde{O}(w_\#)}$, and otherwise it is solvable in polynomial time by the $2^{\widetilde{O}(w_\#)} \cdot n = 2^{\widetilde{O}(w_\#)} \cdot |I|$-time algorithm. A similar argument shows that Knapsack has a kernel of size $2^{\widetilde{O}(p_\#)}$.

The obvious question to ask is whether we can obtain smaller kernels with respect to either $w_\#$ or $p_\#$. Here, the gold standard in parameterized complexity are *polynomial kernels*, kernels with size $f(k) = k^{O(1)}$. By now we know of countless fixed-parameter tractable (NP-hard) problems that also admit polynomial kernels [24]; yet, at the same time, for many other problems polynomial kernels were shown to be unlikely to exist. Thus, the central question this paper addresses is:

Does Knapsack admit a polynomial kernel with respect to either $w_\#$ or $p_\#$?

Note that there are results that give encouraging indications to the question above. Etscheid et al. [22] show that Subset Sum admits a polynomial kernel with respect to $a_\#$, the number of different numbers in the instance (i.e, $a_\# = |\{a_i : i \in \{1, \ldots, n\}\}|$). Can this result be generalized to hold also for the Knapsack problem?

1.1 Our results

Our main technical result of the paper is a negative answer to the question above. We use the by now standard framework for excluding polynomial kernels [5] based on the assumption NP $\not\subseteq$ coNP/poly (whose negation implies that the polynomial hierarchy collapses) to show the following:

▶ **Theorem 2.** *Assuming NP $\not\subseteq$ coNP/poly, there is no polynomial kernel for* Knapsack *parameterized by the number $w_\#$ of different weights, nor by the number $p_\#$ of different profits.*

The proof of the theorem above is obtained through a rather involved construction of what is known as a composition algorithm (see Definition 4). This algorithm composes several instances of a specialized variant of Subset Sum into a single Knapsack instance where either $w_\#$ or $p_\#$ is kept relatively small. We give an overview of this algorithm in Section 1.2.

Complementing the negative result of Theorem 2, we show that when both $w_\#$ and $p_\#$ are taken as a combined parameter, the polynomial kernel for Subset Sum [22] can be extended to a polynomial kernel for Knapsack.

▶ **Theorem 3.** Knapsack *parameterized by $w_\# \cdot p_\#$ admits a polynomial kernel.*

Thus, the lower bounds for $w_\#$ and $p_\#$ cannot be combined. This is somewhat reminiscent to the situation mentioned above, where Knapsack is unlikely to have algorithms with subquadratic running times in either $(W+n)$ or $(P+n)$ [17], but admits an algorithm with subquadratic running time in $(W+P+n)$ [9].

1.2 Technical overview

We prove Theorem 2 for the parameter $w_\#$; the statement for parameter $p_\#$ then follows by a reduction from Polak et al. [45, Chapter 4]. In broad terms, our proof follows the standard framework for showing kernelization lower bounds that has been developed over the years [5, 6, 18, 21]. In this framework, to exclude a polynomial kernel for a given problem Π_1 parameterized by some parameter k, one shows what is called an *OR-composition algorithm* (see Definition 4 for a formal definition) from some known NP-hard problem Π_2 to Π_1. This algorithm takes as input t instances of Π_2 of size n each, and converts these instances in polynomial time to an equivalent instance of Π_1 such that $k = (n \lg t)^{O(1)}$. Here, equivalence means that at least one of the t input instances is a "yes"-instance for Π_2 if and only if the output instance of the composition is a "yes"-instance for Π_1. Such a composition algorithm then directly implies that Π_2 has no polynomial kernel with respect to k under the assumption NP $\not\subseteq$ coNP/poly [5, 6].

We show an OR-composition algorithm from a restricted version of Subset Sum which we call Restricted Subset Sum. In this restricted version, any instance of size n is restricted to include only numbers from a known set $\widetilde{\mathcal{A}}_n$ of size $O(n^3)$. This allows us to bound the number of different numbers in any t instances of Restricted Subset Sum of the same size. Our composition algorithm then proceeds as follows: Given any t instances $\mathcal{A}_0, \ldots, \mathcal{A}_{t-1}$ of Restricted Subset Sum, the algorithm converts any integer $a \in \mathcal{A}_i$, $i \in \{0, \ldots, t-1\}$, to a Knapsack item x_a (which we refer to as an *encoding item*) with weight $w(x_a) = a$. We then assign a higher profit to items corresponding to Restricted Subset Sum instances of higher index, meaning that it is always less profitable to choose an item corresponding to some integer $a \in \mathcal{A}_i$ than choosing an item corresponding to some integer $a \in \mathcal{A}_{i_0}$ for $i_0 > i$. This allows us to encode any solution $\mathcal{A}_i^* \subseteq \mathcal{A}_i$ to the i'th Restricted Subset Sum instance by a solution $\mathcal{X}(\mathcal{A}_i^*)$ to our Knapsack instance which includes all items x_a for $a \in \mathcal{A}_i^*$, and all items x_a for $a \in \mathcal{A}_{i_0}$ for $i_0 > i$. Thus, if we knew index i in advance, our composition would be complete.

However, we do not have a priori knowledge of index i. To circumvent this, we use *index items* that encode the selection of $i \in \{0, \ldots, t-1\}$. These are $2 \lg t$ items that encode the selection of t binary values $i(0), \ldots, i(\lg t - 1)$ that correspond to the base-2 representation of i, i.e., $i = \sum_k i(k) \cdot 2^k$. This is somewhat akin to the *"colors and IDs"* technique for composition algorithms which was introduced by Dom et al. [20]. The goal of these index items is to ensure that choosing a solution $\mathcal{X}(\mathcal{A}_i^*)$, for any value of i, allows adding a subset of index items to the knapsack such that any such selection has the same weight and profit. However, just adding the $O(\lg t)$ index items is not sufficient: As the composed instance shall achieve the same profit for any solution $\mathcal{X}(\mathcal{A}_i^*)$, we need to compensate for this difference in profit between $\mathcal{X}(\mathcal{A}_i^*)$ and $\mathcal{A}_{t-1}$. The difference in profit between solution $\mathcal{X}(\mathcal{A}_i^*)$ which includes all items from instances $\mathcal{A}_{i_0}$ for $i_0 > i$, and solution $\mathcal{X}(\mathcal{A}_{t-1}^*)$ which includes only items corresponding to instance $\mathcal{A}_{t-1}$, is quadratic in i. Hence, we cannot compensate for this difference using only the $O(\lg t)$ index items.

We therefore introduce additional $O(\lg^2 t)$ items which we refer to as *quadratization items*. These encode the selection of binary pair values $i(k)i(\ell)$ in $i^2 = \sum_k \sum_\ell i(k)i(\ell) \cdot 2^{k+\ell}$, and allow us to encode the quadratic compensation mentioned above. Unfortunately, this introduces

additional technical difficulties, such as ensuring compatibility between the selection of the index and quadratization items. Meanwhile, we still need to maintain the compatibility between the encoding and index items as well. These difficulties are overcome using various applications of a basic algebraic lemma that we prove in Section 2. In the end, we obtain an instance with $w_\# = O(n^3 \cdot \lg^2 t)$ many different weights, which by previously known results (Theorem 5) implies that a polynomial kernel would imply NP $\not\subseteq$ coNP/poly. The full details of the entire composition are given in Section 3.

In Section 4, we derive a polynomial kernel by first modeling Knapsack by an ILP with $w_\# \cdot p_\#$ many variables, and then reducing the size of the numbers occurring in the ILP by a well-known result by Frank and Tardos [26].

Theorems whose proof is omitted (and can be found in the full version [28]) are marked by $\star$.

2 Preliminaries

We next quickly review the kernelization lower bounds framework, introduced by Bodlaender et al. [5] and further developed by [6, 18, 21], that will be used for proving Theorem 2. At the heart of the framework lies the notion of a composition.

▶ **Definition 4.** *A composition algorithm from a problem Π_1 to a parameterized problem Π_2 is an algorithm that receives as input t instances $I_0, \ldots, I_{t-1}$ of size n of Π_1, and computes in polynomial time an instance (J, k) of Π_2 such that*

- *J is a "yes"-instance of Π_2 if and only if I_i is a "yes"-instance of Π_1 for some $i \in \{0, \ldots, t-1\}$,*
- *and $k \leq (n + \lg t)^{O(1)}$.*

The main connection between composition algorithms and the exclusion of polynomial kernels is given in the theorem below, whose proof relies on a complexity-theoretic lemma by Fortnow and Santhanam [25].

▶ **Theorem 5** ([5, 6])**.** *Let Π_1 be an NP-hard problem, and Π_2 be a parameterized problem with a composition algorithm from Π_1 to Π_2. Then Π_2 does not admit a polynomial kernel, assuming NP $\not\subseteq$ coNP/poly.*

Note that NP $\not\subseteq$ coNP/poly is a widely believed assumption in complexity theory, and if it were false then the polynomial hierarchy would collapse to its third level [47].

As a final point, we will also make use of the following basic algebraic lemma throughout our proof. Below we provide a proof for the sake of completeness.

▶ **Lemma 6.** *Let $b > 1$ and k be positive integers. Then there exists a unique integer solution to the equation*

$$x_0 b^0 + x_1 b^1 + \cdots + x_{k-1} b^{k-1} = \sum_{i=0}^{k-1} b^i$$

constrained by $x_i \in \{0, 1, \ldots, b\}$ for each $i \in \{0, \ldots, k-1\}$. Namely, the solution is

$$x_0 = x_1 = \cdots = x_{k-1} = 1 \ .$$

Proof. We show the statement by induction on k. For $k = 1$, the statement is obvious. So fix $k > 1$. Note that $\sum_{i=0}^{k-1} b^i = b^{k-1} + \sum_{i=0}^{k-2} b^i = b^{k-1} + \frac{b^{k-1}}{b-1} < 2b^{k-1}$, so we have that $x_{k-1} \in \{0, 1\}$. Further, we have $\sum_{i=0}^{k-2} x_i b^i \leq \sum_{i=0}^{k-2} b \cdot b^i = \sum_{i=1}^{k-1} b^i < \sum_{i=0}^{k-1} b^i$, implying that $x_k > 0$. Thus, we have $x_{k-1} = 1$ and $x_i = 1$ for $i \in \{0, \ldots, k-2\}$ follows by induction. ◀

Further, for any function $f : S \to \mathbb{R}$, we define $f(S') := \sum_{s \in S'} f(s)$ for any $S' \subseteq S$.

3 No Polynomial Kernel for Parameter $w_{\#}$

In the following, we present the proof of Theorem 2 for parameter $w_{\#}$, the total number of different weights in a Knapsack instance. As mentioned above, in our proof we construct a composition from a restricted version of Subset Sum to Knapsack parameterized by $w_{\#}$. The proof is divided into four parts: In the first part, we introduce the Restricted Subset Sum problem, and prove that it is NP-hard. In the second part we begin to describe our composition by showing how to encode t instances $\mathcal{A}_0, \ldots, \mathcal{A}_{t-1}$ of Restricted Subset Sum into a single set $\mathcal{X}$ of *encoding items*. In the third part we describe the set $\mathcal{Y}$ of *quadratization items*, and the set $\mathcal{Z}$ of *index items*, which together form the instance selection gadget of the composition. The final part is devoted to finishing details, and to proving the correctness of the composition.

3.1 Restricted Subset Sum

We start with a restricted version of Subset Sum which allows us to bound the number of different numbers appearing in any set of t Subset Sum instances. Namely, in our restricted version of Subset Sum, any instance of size n contains numbers from a restricted set of $O(n^3)$ numbers. Apart from this, its useful properties are that the target value only depends on the number of input numbers, and that any solution must have the same cardinality. For this, we define the set of possible numbers which may be contained in a Restricted Subset Sum instance of size n as

$$\widetilde{\mathcal{A}}_n := \left\{(3n+1)^{j_1} + (3n+1)^{j_2} + (3n+1)^{j_3} \ : \ j_1, j_2, j_3 \in \{1, \ldots, 3n\}\right\} .$$

Note that $\widetilde{\mathcal{A}}_n$ contains $27n^3 = O(n^3)$ integers. Furthermore, we also define a global target for all instances of size n by $B_n := \sum_{j=1}^{3n}(3n+1)^j$.

Restricted Subset Sum
Input: A set $\mathcal{A} = \{a_1, \ldots, a_{3n}\}$ of $3n$ integers from $\widetilde{\mathcal{A}}_n$ with $\sum_{j=1}^{3n} a_j = 3B_n$.
Question: Is there a subset $\mathcal{A}^* \subseteq \mathcal{A}$ with $|\mathcal{A}^*| = n$ such that $\sum_{a \in \mathcal{A}^*} a = B_n$?

▶ **Lemma 7.** Restricted Subset Sum *is NP-complete.*

Proof. We present a reduction from a variant of Restricted Exact Cover by 3-Sets where each element appears in exactly three sets. Recall that in Restricted Exact Cover by 3-Sets, the input consists of a set $\mathcal{T}$ consisting of 3-element subsets of $\{1, \ldots, 3n\}$ such that each $j \in \{1, \ldots, 3n\}$ appears in exactly three sets from $\mathcal{T}$, and the question is whether there exists a subset $\mathcal{T}' \subseteq \mathcal{T}$ such that each element from $\{1, \ldots, 3n\}$ appears in exactly one set from $\mathcal{T}'$. This variant of Restricted Exact Cover by 3-Sets is well-known to be NP-hard [27].

Our reduction is almost identical to the original hardness reduction for Subset Sum by Karp [35], only that we reduce from Restricted Exact Cover by 3-Sets instead of the more general Exact Cover: Let $\mathcal{T}$ be an instance of Restricted Exact Cover by 3-Sets. For each 3-element set $T \in \mathcal{T}$, a number $a_T := \sum_{j \in T}(3n+1)^j$ is added to the Restricted Subset Sum instance. In this way, the constructed instance of Restricted Subset Sum is $\mathcal{A} := \{a_T : T \in \mathcal{T}\}$. Note that $a_T \in \widetilde{\mathcal{A}}_n$ for every $T \in \mathcal{T}$. Further, since each $j \in \{1, \ldots, 3n\}$ appears in exactly three sets from $\mathcal{T}$, we have $\sum_{T \in \mathcal{T}} a_T = 3\sum_{j=1}^{3n}(3n+1)^j = 3B_n$. Below we prove the correctness of this construction.

Suppose there is a solution $\mathcal{T}' \subseteq \mathcal{T}$ for the Restricted Exact Cover by 3-Sets instance. Then $|\mathcal{T}'| = n$ and we have $\sum_{T \in \mathcal{T}} a_T = \sum_{\ell=1}^{3n}(3n+1)^\ell = B_n$. Conversely, suppose there is a solution $\mathcal{A}^* \subseteq \mathcal{A}$ with $|\mathcal{A}^*| = n$ for the Restricted Subset Sum instance. Let

$\mathcal{T}^* := \{T : a_T \in \mathcal{A}^*\}$. Since each term $(3n+1)^j$ appears only $3 < 3n+1$ times, Lemma 6 implies that the only way for some numbers from $\mathcal{A}$ to add up to $B_n = \sum_{j=1}^{3n}(3n+1)^j$ is that each term $(3n+1)^j$ appears in exactly one $a_T \in \mathcal{A}^*$. In other words, for each $j \in \{1,\dots,3n\}$, there is exactly one $T \in \mathcal{T}^*$ with $j \in T$. ◀

3.2 Encoding Gadget

In the following, we show how to encode t instances of Restricted Subset Sum into a single Knapsack instance. Throughout the remainder of the section, we use $\mathcal{A}_0,\dots,\mathcal{A}_{t-1}$ to denote the t input instances of Restricted Subset Sum to our composition, where $|\mathcal{A}_i| = 3n$ for each $i \in \{0,\dots,t-1\}$. By copying instances, we may assume without loss of generality that t is a power of 2, i.e., $t = 2^s$ for some $s \in \mathbb{N}$. Furthermore, we let $\mathcal{A}_i = \{a_1^i,\dots,a_{3n}^i\}$ denote the i'th instance for each $i \in \{0,\dots,t-1\}$. By the definition of Restricted Subset Sum, we have $a_j^i \in \widetilde{\mathcal{A}}_n$ for each each $i \in \{0,\dots,t-1\}$ and $j \in \{1,\dots,3n\}$, and $\sum_j a_j^i = 3B_n$ for each $i \in \{0,\dots,t-1\}$.

Recall that Subset Sum can be seen as a special case of Knapsack where the profit of each item equals its weight. This yields an easy reduction from each single Restricted Subset Sum instance $\mathcal{A}_i$ to Knapsack. We apply this reduction with a slight modification: To capture the condition that each solution of Restricted Subset Sum shall contain exactly n numbers, we add a large number $X = 3tnB_n$ to each number in a Restricted Subset Sum instance, and set $B := B_n + n \cdot X$. We also increase the profit of each item corresponding to the i'th Restricted Subset Sum instance by adding $i \cdot 3B$ to its profit. More precisely, for each $i \in \{0,\dots,t-1\}$ and $j \in \{1,\dots,3n\}$, we construct an *encoding item* x_j^i with

- $w(x_j^i) = X + a_j^i$, and
- $p(x_j^i) = X + a_j^i + i \cdot 3B$.

Intuitvely, adding the constant X to the weights ensures that all encoding items have roughly the same size, so for any weight budget W^*, the set of maximum profit will always have size $\lfloor W^*/X \rfloor$ or $\lfloor W^*/X \rfloor - 1$. Adding the constant X to the profits of x_j^i ensures that for any two sets of encoding items with different cardinalities, the larger one will always have the larger profit. Further, adding $i \cdot 3B$ to the profits will ensure that it will be always more profitable to pick an item $x_{j_1}^{i_1}$ over item $x_{j_0}^{i_0}$ for any $i_1 > i_0$. We use $\mathcal{X} = \{x_j^i : i \in \{0,\dots,t-1\}, j \in \{1,\dots,3n\}\}$ to denote the set of all encoding items.

Let $\mathcal{X}_i$ denote the set of encoding items corresponding to instance $\mathcal{A}_i$ of Restricted Subset Sum, i.e., $\mathcal{X}_i = \{x_j^i : j \in \{1,\dots,3n\}\}$. Note that $w(\mathcal{X}_i) = 3B$ and $p(\mathcal{X}_i) = 3B + 9nB \cdot i$. Now suppose $\mathcal{A}_i$ has a solution $\mathcal{A}_i^* \subset \mathcal{A}_i$. We would like to encode this solution using the following set of encoding items

$$\mathcal{X}(\mathcal{A}_i^*) = \mathcal{X}_i^* \cup \mathcal{X}_{i+1} \cup \mathcal{X}_{i+2} \cup \dots \cup \mathcal{X}_{t-1},$$

where $\mathcal{X}_i^* := \{x_i^j : a_i^j \in \mathcal{A}_i^*\}$ is the set of items corresponding to $\mathcal{A}_i^*$. An easy calculation shows that if the elements of $\mathcal{A}_i^*$ sum up to B (i.e., $\mathcal{A}_i^*$ is indeed a solution), then $w(\mathcal{X}(\mathcal{A}_i^*)) = (3t - 3i - 2) \cdot B$ and $p(\mathcal{X}(\mathcal{A}_i^*)) = (3t - 3i - 2) \cdot B + (\binom{t}{2} - \binom{i+1}{2} + \frac{i}{3}) \cdot 9nB$. The next lemma shows that the converse is also true; namely, that if there is a subset of encoding items with the above weight and profit, then there must be a solution to the i'th Restricted Subset Sum instance.

▶ **Lemma 8** (⋆). *Let $i \in \{0,\dots,t-1\}$. There exists a subset $\mathcal{X}^* \subseteq \mathcal{X}$ of encoding items with total weight*

$$w(\mathcal{X}^*) \leq (3t - 3i - 2) \cdot B$$

and total profit

$$p(\mathcal{X}^*) \geq (3t - 3i - 2) \cdot B \; + \; \left(\binom{t}{2} - \binom{i+1}{2} + \frac{i}{3} \right) \cdot 9nB$$

for our Knapsack *instance if and only if there exist a solution* $\mathcal{A}_i^* \subseteq \mathcal{A}_i$ *to the* i*'th* Restricted Subset Sum *instance.*

Lemma 8 implies that if we knew which Restricted Subset Sum instance $\mathcal{A}_i$ has a solution, then we could easily set the weight and profit of our composed Knapsack instance to encode this solution. However, we do not have prior information about index i. Furthermore, observe that as the value of i increases, both the profit and weight of the required solution decrease. Since we do not know the value of i, it would be beneficial to balance all possible choices of i in terms of weight and profit.

Thus, the remaining construction focuses on ensuring that solutions to the Knapsack instance corresponding to different $\mathcal{A}_i$'s all have the same weight and profit. In particular, the construction guarantees that any choice of i can obtain a profit of $(3t-2) \cdot B + 9 \cdot \binom{t}{2} \cdot nB$, in addition to some large constant. Considering the profit guaranteed by solutions of Lemma 8, we need to compensate for the loss of the quadratic term

$$\left(\binom{i+1}{2} - \frac{i}{3} \right) \cdot 9nB \; . \tag{1}$$

We call the term above the *compensation term of* i. It will play an important role in the remainder of our construction.

3.3 Instance Selection Gadget

We next add additional items to our Knapsack instance that will serve as an instance selection gadget. This gadget selects an instance of Restricted Subset Sum for which presumably there is a solution. The gadget consists of two types of items: The *index items* which encode an index of an Restricted Subset Sum instance $i \in \{0, \ldots, t-1\}$, and *quadratization items* that help to encode the compensation term of i given in Equation 1.

Quadratization items. The main idea behind the quadratization items is as follows: Any integer $i \in \{0, \ldots, t-1\}$ can be written as the sum

$$i = \sum_{k=0}^{\lg t - 1} i(k) \cdot 2^k,$$

for some binary values $i(0), \ldots, i(\lg t - 1) \in \{0, 1\}$. Thus, using these same $\lg t$ binary values, we can write the compensation term of i as

$$\begin{aligned}
\left(\binom{i+1}{2} - \frac{i}{3} \right) \cdot 9nB \quad &= \quad \left(0.5 \cdot i^2 + \frac{1}{6} \cdot i \right) \cdot 9nB \\
&= \quad \left(0.5 \cdot \sum_{k=0}^{\lg t - 1} \sum_{\ell=0}^{\lg t - 1} i(k) \cdot i(\ell) \cdot 2^{k+\ell} + \frac{1}{6} \sum_{k=0}^{\lg t - 1} i(k) \cdot 2^k \right) \cdot 9nB \\
&= \quad \left(9 \cdot \sum_{\substack{i(k)=1, \\ i(\ell)=1, \\ k<\ell}} 2^{k+\ell} \; + \; 4.5 \cdot \sum_{i(k)=1} 2^{k+k} \; + \; 1.5 \cdot \sum_{i(k)=1} 2^k \right) \cdot nB \; .
\end{aligned}$$

(2)

Thus, we construct $3 \cdot \binom{\lg t}{2} + \lg t$ different quadratization items, each modeling the contribution of all possible values of $i(k)$ and $i(\ell)$, $k \leq \ell \in \{0, \ldots, \lg t - 1\}$, in the last equality of Equation 2 above.

Let $Y = t^2 \cdot 3nB$, and observe that Y is larger than the total profit of all encoding items. Furthermore, let $f : \{0, \ldots, \lg t - 1\}^2 \to \{0, \ldots, \lg^2 t - 1\}$ be any bijective function. For each pair of indices k and ℓ with $0 \leq k < \ell \leq \lg t - 1$, we add three quadratization items $y^{1,0}_{k,\ell}$, $y^{0,1}_{k,\ell}$, and $y^{1,1}_{k,\ell}$ with the following weight and profit:

- $w(y^{1,0}_{k,\ell}) = p(y^{1,0}_{k,\ell}) = 3^{f(k,\ell)} \cdot Y$.
- $w(y^{0,1}_{k,\ell}) = p(y^{0,1}_{k,\ell}) = 3^{f(\ell,k)} \cdot Y$.
- $w(y^{1,1}_{k,\ell}) = (3^{f(k,\ell)} + 3^{f(\ell,k)}) \cdot Y$ and $p(y^{1,1}_{k,\ell}) = (3^{f(k,\ell)} + 3^{f(\ell,k)}) \cdot Y + 2^{k+\ell} \cdot 9nB$.

Furthermore, for each $k \in \{0, \ldots, \lg t - 1\}$, we add a single quadratization item $y^{1,1}_{k,k}$ with:

- $w(y^{1,1}_{k,k}) = 3^{f(k,k)} \cdot Y$ and $p(y^{1,1}_{k,k}) = 3^{f(k,k)} \cdot Y + 2^{k+k} \cdot 4.5nB + 2^k \cdot 1.5nB$.

We use $\mathcal{Y} = \{y^{1,0}_{k,\ell}, y^{0,1}_{k,\ell}, y^{1,1}_{k,\ell} : 0 \leq k < \ell \leq \lg t - 1\} \cup \{y^{1,1}_{k,k} : 0 \leq k \leq \lg t - 1\}$ to denote the set of all quadratization items.

The role of the additional terms that depend on Y will become clearer when we introduce the index items. But for now, one can observe that the smaller terms used in the profits of $y^{1,1}_{k,\ell}$ and $y^{1,1}_{k,k}$ allow us to encode the compensation term of i. In particular, an easy calculation using Equation 2 gives us the following useful lemma:

▶ **Lemma 9.** *Let $i \in \{0, \ldots, t-1\}$, and let $i(0), \ldots, i(t-1) \in \{0, 1\}$ be binary values such that $i = \sum_k i(k) \cdot 2^k$. Moreover, let $\mathcal{Y}_i$ denote the set of quadratization items defined by*

$$\mathcal{Y}_i = \{y^{i(k),i(\ell)}_{k,\ell} : 0 \leq k \leq \ell \leq \lg t - 1\},$$

where $y^{i(k),i(\ell)}_{k,\ell}$ is the empty item (i.e., an item with weight and profit 0) if $i(k) = i(\ell) = 0$. Then

$$p(\mathcal{Y}_i) \quad = \quad w(\mathcal{Y}_i) + \left(\binom{i+1}{2} - \frac{i}{3}\right) \cdot 9nB \ .$$

Index items. The index items ensure that only quadratization items that correspond to subsets $\mathcal{Y}_i$ as in Lemma 9 above can be picked into any solution of our Knapsack instance. In particular, the index items will encode the selection of an index $i \in \{0, \ldots, \lg t - 1\}$ that will be compatible with the selection of a subset $\mathcal{Y}_i$ of quadratization items.

Let $Z = \lg^2 t Y^2 \cdot 3^{\lg^2 t}$, and observe that Z is larger than the profit of all encoding and quadratization items in total. For each $k \in \{0, \ldots, \lg t - 1\}$, we construct two *index items* z^0_k and z^1_k corresponding to selecting either $i(k) = 0$ or $i(k) = 1$ in the binary representation $i(0), \ldots, i(\lg t - 1)$ of i. The weight and profit of these two items are defined by:

- $w(z^0_k) = p(z^0_k) = 2^k \cdot Z + \sum_{\ell=0}^{\lg t - 1} 3^{f(k,\ell)} \cdot Y$.
- $w(z^1_k) = p(z^1_k) = 2^k \cdot Z + 2^k \cdot 3B$.

We use $\mathcal{Z} = \{z^0_k, z^1_k : 0 \leq k \leq \lg t - 1\}$ to denote the set of all index items.

Let $i \in \{0, \ldots, t-1\}$, and let $i(0), \ldots, i(t-1) \in \{0, 1\}$ be binary values such that $i = \sum_k i(k) \cdot 2^k$. Observe that the set of index items $\mathcal{Z}_i$ defined by

$$\mathcal{Z}_i = \{z^{i(k)}_k : 0 \leq k \leq \lg t - 1\}$$

naturally corresponds to index i. In the lemma below, we show that due to our selection of the large value Z, any set of items of sufficiently small weight and sufficiently large profit contains a subset of index items that corresponds precisely to some index $i \in \{0, \ldots, t-1\}$. We let w_Z denote the weight function $w_Z(x) = \lfloor w(x)/Z \rfloor$, and p_Z denote the profit function $p_Z(x) = \lfloor p(x)/Z \rfloor$.

▶ **Lemma 10** (⋆). *Let $\mathcal{S} \subseteq \mathcal{X} \cup \mathcal{Y} \cup \mathcal{Z}$ be a set of items with $w_Z(\mathcal{S}) \leq t-1$ and $p_Z(\mathcal{S}) \geq t-1$. Then there exists some $i \in \{0, \ldots, t-1\}$ for which $\mathcal{S} \cap \mathcal{Z} = \mathcal{Z}_i$.*

Now let us address the terms that depend on Y in the profit and weight of the instance selection items. Define T to be the constant $T := \sum_{k=0}^{\lg^2 t-1} 3^k$. Now consider some set $\mathcal{Z}_i$ of index items corresponding to index $i \in \{0, \ldots, t-1\}$. Let w_Y denote the weight function $w_Y(x) = \lfloor (w(x) - Z \cdot w_Z(x))/Y \rfloor$, and similarly define the profit function p_Y as $p_Y(x) = \lfloor (p(x) - Z \cdot p_Z(x))/Y \rfloor$. Then one can observe that, by construction of the weights and profits above, we have that both $w_Y(\mathcal{Z}_i) + w_Y(\mathcal{Y}_i)$ and $p_Y(\mathcal{Z}_i) + p_Y(\mathcal{Y}_i)$ equal T. Moreover, any other set of items with w_Y-weight at most T has lesser profit. This ensures a compatible selection of the index items and the quadratization items, formally proven in the lemma below.

▶ **Lemma 11.** *Let $\mathcal{S} \subseteq \mathcal{X} \cup \mathcal{Y} \cup \mathcal{Z}$ be a set of items with weight $w_Z(\mathcal{S}) \leq (t-1)$, $w_Y(\mathcal{S}) \leq T$, $p_Z(\mathcal{S}) \geq (t-1)$, $p_Y(\mathcal{S}) \geq T$, and there is no set $\mathcal{S}' \subseteq \mathcal{X} \cup \mathcal{Y} \cup \mathcal{Z}$ with $w(\mathcal{S}') \leq w(\mathcal{S})$ and $p(\mathcal{S}') > p(\mathcal{S})$. Then $\mathcal{S} \cap \mathcal{Y} = \mathcal{Y}_i$ and $\mathcal{S} \cap \mathcal{Z} = \mathcal{Z}_i$ for some $i \in \{0, \ldots, t-1\}$.*

Proof. Let $\mathcal{Z}^* = \mathcal{S} \cap \mathcal{Z}$ and $\mathcal{Y}^* = \mathcal{S} \cap \mathcal{Y}$. As $w_Z(\mathcal{S}) \leq (t-1)$ and $p_Z(\mathcal{S}) \geq (t-1)$, by Lemma 10 we have that $\mathcal{Z}^* = \mathcal{Z}_i$ for some $i \in \{0, \ldots, t-1\}$. Thus, to complete the proof, we focus on showing that $\mathcal{Y}^* = \mathcal{Y}_i$. Let $i(0), \ldots, i(\lg t - 1) \in \{0, 1\}$ be such that $i = \sum_k i(k) \cdot 2^k$.

Observe that by the construction of the weights and profits of the index items, we have $w_Y(\mathcal{Z}_i) = p_Y(\mathcal{Z}_i) = \sum_{\alpha(k)=0} \sum_\ell 3^{f(k,\ell)}$. From this, one can see that both $w_Y(\mathcal{Z}_i) + w_Y(\mathcal{Y}^*)$ and $w_Y(\mathcal{Z}_i) + p_Y(\mathcal{Y}^*)$ equal

$$\Big(\sum_{\substack{i(k)=0,\\ 0\leq \ell \leq \lg t-1}} 3^{f(k,\ell)} \Big) + \Big(\sum_{y^{1,0}_{k,\ell} \in \mathcal{Y}_i} 3^{f(k,\ell)} + \sum_{y^{0,1}_{k,\ell} \in \mathcal{Y}_i} 3^{f(\ell,k)} + \sum_{y^{1,1}_{k,\ell} \in \mathcal{Y}_i} (3^{f(k,\ell)} + 3^{f(\ell,k)}) + \sum_{y^{1,1}_{k,\ell} \in \mathcal{Y}_i} 3^{f(k,k)} \Big) =$$
$$\Big(\sum_{\substack{i(k)=0,\\ 0\leq \ell \leq \lg t-1}} 3^{f(k,\ell)} \Big) + \Big(\sum_{\substack{i(k)=1,\\ i(\ell)=0,\\ k<\ell}} 3^{f(k,\ell)} + \sum_{\substack{i(k)=1,\\ i(\ell)=0,\\ k>\ell}} 3^{f(k,\ell)} + \sum_{\substack{i(k)=1,\\ i(\ell)=1,\\ k\neq\ell}} 3^{f(k,\ell)} + \sum_{i(k)=1} 3^{f(k,k)} \Big) \ .$$

Note that as $w_Y(\mathcal{Z}_i) + w_Y(\mathcal{Y}^*) = w_Y(\mathcal{S}) \leq T$, the sum above is bounded from above by T. Moreover, by Lemma 6, the only way this reaches the bound with equality is if we have

$$w_Y(\mathcal{Y}^*) = \sum_{\substack{i(k)=1,\\ i(\ell)=0,\\ k<\ell}} 3^{f(k,\ell)} + \sum_{\substack{i(k)=1,\\ i(\ell)=0,\\ k>\ell}} 3^{f(k,\ell)} + \sum_{\substack{i(k)=1,\\ i(\ell)=1,\\ k\neq\ell}} 3^{f(k,\ell)} + \sum_{i(k)=1} 3^{f(k,k)} = \sum_{\substack{i(k)=0,\\ 0\leq \ell \leq \lg t-1}} 3^{f(k,\ell)} \ ,$$

which then gives us

$$w_Y(\mathcal{Z}_i) + w_Y(\mathcal{Y}^*) = \sum_{\substack{i(k)=0,\\ 0\leq \ell \leq \lg t-1}} 3^{f(k,\ell)} + \sum_{\substack{i(k)=1,\\ 0\leq \ell \leq \lg t-1}} 3^{f(k,\ell)} = \sum_{k=0}^{\lg^2 t-1} 3^k = T \ .$$

(Here, the penultimate equality follows because $f(\cdot,\cdot)$ is bijective.)

Note that by construction and Lemma 6, the only sets of quadratization items $\mathcal{Y}^*$ with $w_Y(\mathcal{Y}^*) = p_Y(\mathcal{Y}^*) = \sum_{\alpha(k)=1} \sum_\ell 3^{f(k,\ell)}$ are either $\mathcal{Y}_i$, or any set of quadratization items obtained from $\mathcal{Y}_i$ by replacing some $y^{1,1}_{k,\ell} \in \mathcal{Y}_i$ with $y^{1,0}_{k,\ell}$ and $y^{0,1}_{k,\ell}$. If $\mathcal{S}$ contained $y^{1,0}_{k,\ell}$ and $y^{0,1}_{\ell,k}$ for some $0 \leq k < \ell \leq \lg t - 1$, then $\mathcal{S}' := (\mathcal{S} \setminus \{y^{1,0}_{k,\ell}, y^{0,1}_{k,\ell}\}) \cup \{y^{1,1}_{k,\ell}\}$ satisfies $w(\mathcal{S}') = w(\mathcal{S})$ and $p(\mathcal{S}') > p(\mathcal{S})$, a contradiction to the definition of $\mathcal{S}$. Thus, we conclude that $\mathcal{Y}^* = \mathcal{Y}_i$ and the lemma is proven. ◀

Table 1 The weights and profits of the items used in the proof of Theorem 2 for parameter $w_{\#}$. The three large constants used in the proof are $X = 3tn \cdot B_n$, $Y = t^2 \cdot 3nB$, and $Z = \lg^2 t Y^2 \cdot 3^{\lg^2 t}$.

Item	Weight	Profit	Index Range
z_k^1	$2^k \cdot Z + 2^k \cdot 3B$	$2^k \cdot Z + 2^k \cdot 3B$	$0 \leq k \leq \lg t - 1$
z_k^0	$2^k \cdot Z + \sum_\ell 3^{f(k,\ell)} \cdot Y$	$2^k \cdot Z + \sum_\ell 3^{f(k,\ell)} \cdot Y$	$0 \leq k \leq \lg t - 1$
$y_{k,\ell}^{1,0}$	$3^{f(k,\ell)} \cdot Y$	$3^{f(k,\ell)} \cdot Y$	$0 \leq k < \ell \leq \lg t - 1$
$y_{k,\ell}^{0,1}$	$3^{f(\ell,k)} \cdot Y$	$3^{f(\ell,k)} \cdot Y$	$0 \leq k < \ell \leq \lg t - 1$
$y_{k,\ell}^{1,1}$	$(3^{f(k,\ell)} + 3^{f(\ell,k)}) \cdot Y$	$(3^{f(k,\ell)} + 3^{f(\ell,k)}) \cdot Y + 2^{k+\ell} \cdot 9nB$	$0 \leq k < \ell \leq \lg t - 1$
$y_{k,k}^{1,1}$	$3^{f(k,k)} \cdot Y$	$3^{f(k,k)} \cdot Y + 2^{k+k} \cdot 4.5nB + 2^k \cdot 1.5nB$	$0 \leq k \leq \lg t - 1$
x_j^i	$X + a_j^i$	$X + a_j^i + i \cdot 3B$	$0 \leq i \leq t-1$, $1 \leq j \leq 3n$

3.4 Correctness

Our entire Knapsack instance consists of all items $\mathcal{X} \cup \mathcal{Y} \cup \mathcal{Z}$. An overview of the weight and profit of each item can be found in Table 1. We set the weight W of the Knapsack instance to

$$W := (t-1) \cdot Z \ + \ T \cdot Y \ + \ (3t-2) \cdot B$$

and the desired profit P to

$$\begin{aligned} P &:= W \ + \ \binom{t}{2} \cdot 9nB \\ &= (t-1) \cdot Z \ + \ T \cdot Y \ + \ (3t-2) \cdot B \ + \ \binom{t}{2} \cdot 9nB \ . \end{aligned}$$

In the next two lemmas below we prove the correctness of our constructed composition.

▶ **Lemma 12.** *If $\mathcal{A}_i$ is a "yes"-instance of* Restricted Subset Sum *for some $i \in \{0, \ldots, t-1\}$ then there exists a subset of items $\mathcal{S} \subseteq \mathcal{X} \cup \mathcal{Y} \cup \mathcal{Z}$ with $w(\mathcal{S}) \leq W$ and $p(\mathcal{S}) \geq P$.*

Proof. Suppose $\mathcal{A}_i$ is a "yes"-instance of Restricted Subset Sum for some $i \in \{0, \ldots, t-1\}$, and let $i(0), \ldots, i(\lg t - 1) \in \{0, 1\}$ be such that $i = \sum_k i(k) \cdot 2^k$. Then $w(\mathcal{X}(\mathcal{A}_i)) = (3t - 3i - 2) \cdot B$ as discussed in Section 3.2. Furthermore, $w(\mathcal{Y}_i) = w_Y(\mathcal{Y}_i) \cdot Y = (T - w_Y(\mathcal{Z}_i)) \cdot Y$ as is shown in the proof of Lemma 11. Finally, we have $w_Z(\mathcal{Z}_i) = (t-1) \cdot Z$, and

$$w(\mathcal{Z}_i) \ = \ w_Z(\mathcal{Z}_i) + w_Y(\mathcal{Z}_i) + \sum_{i(k)=1} 2^k \cdot 3B \ = \ (t-1) \cdot Z + w_Y(\mathcal{Z}_i) \cdot Y + i \cdot 3B \ .$$

So altogether we have

$$\begin{aligned} w(\mathcal{X}(\mathcal{A}_i) \cup \mathcal{Y}_i \cup \mathcal{Z}_i) \ &= \ (3t - 3i - 2) \cdot B \ + \ (T - w_Y(\mathcal{Z}_i)) \cdot Y \\ &\quad + (t-1) \cdot Z \ + \ w_Y(\mathcal{Z}_i) \cdot Y \ + \ i \cdot 3B \\ &= (t-1) \cdot Z \ + \ T \cdot Y \ + \ (3t-2) \cdot B \ = W \ . \end{aligned}$$

Let us next calculate the profit of $\mathcal{X}(\mathcal{A}_i) \cup \mathcal{Y}_i \cup \mathcal{Z}_i$. Recall that $p(\mathcal{X}(\mathcal{A}_i)) = w(\mathcal{X}(\mathcal{A}_i)) + (\binom{t}{2} - \binom{i+1}{2} + \frac{i}{3}) \cdot 9nB$ as discussed in Section 3.2. By Lemma 9 we have $p(\mathcal{Y}_i) = w(\mathcal{Y}_i) + (\binom{i+1}{2} - \frac{i}{3}) \cdot 9nB$, and by construction we have $p(\mathcal{Z}_i) = w(\mathcal{Z}_i)$. Thus, altogether we have

$$\begin{aligned} p(\mathcal{X}(\mathcal{A}_i) \cup \mathcal{Y}_i \cup \mathcal{Z}_i) &= w(\mathcal{X}(\mathcal{A}_i)) + (\binom{t}{2} - \binom{i+1}{2} + \frac{i}{3}) \cdot 9nB \\ &\quad + w(\mathcal{Y}_i) + (\binom{i+1}{2} - \frac{i}{3}) \cdot 9nB \ + \ w(\mathcal{Z}_i) \\ &= w(\mathcal{X}(\mathcal{A}_i) \cup \mathcal{Y}_i \cup \mathcal{Z}_i) + \binom{t}{2} \cdot 9nB = P \ . \end{aligned}$$

Thus the set of items $\mathcal{S} = \mathcal{X}(\mathcal{A}_i) \cup \mathcal{Y}_i \cup \mathcal{Z}_i$ is a solution for our Knapsack instance, and the lemma is proven. ◀

▶ **Lemma 13.** *If there exists a subset of items $\mathcal{S} \subseteq \mathcal{X} \cup \mathcal{Y} \cup \mathcal{Z}$ with $w(\mathcal{S}) \leq W$ and $p(\mathcal{S}) \geq P$ then there is some $i \in \{0, \ldots, t-1\}$ for which $\mathcal{A}_i$ is a "yes"-instance of* Restricted Subset Sum.

Proof. Let $\mathcal{S}$ be a solution with $w(\mathcal{S}) \leq W$ and $p(\mathcal{S}) \geq P$. Let $\mathcal{X}^* = \mathcal{S} \cap \mathcal{X}$, $\mathcal{Y}^* = \mathcal{S} \cap \mathcal{Y}$, and $\mathcal{Z}^* = \mathcal{S} \cap \mathcal{Z}$. Then as $w(\mathcal{S}) \leq W < t \cdot Z$ and $p(\mathcal{S}) \leq P < t \cdot Z$, we have by Lemma 10 that $\mathcal{Z}^* = \mathcal{Z}_i$ for some $i \in \{0, \ldots, t-1\}$. Assume, without loss of generality, that $\mathcal{S}$ is of maximal profit among all solutions with weight at most $w(\mathcal{S})$ (i.e., there is no set $\mathcal{S}'$ with $w(\mathcal{S}') \leq w(\mathcal{S})$ and $p(\mathcal{S}') > p(\mathcal{S})$. Then, by Lemma 11, we have $\mathcal{Y}^* = \mathcal{Y}_i$. As shown in the proof of Lemma 12, we have $w(\mathcal{Z}_i) + w(\mathcal{Y}_i) = (t-1) \cdot Z + T \cdot Y + i \cdot 3B$. Thus,

$$w(\mathcal{X}^*) \leq W - w(\mathcal{Z}_i) - w(\mathcal{Y}_i) = (3t - 3i - 2) \cdot B \ .$$

Moreover, by Lemma 9 we have $p(\mathcal{Y}_i) = w(\mathcal{Y}_i) + (\binom{i+1}{2} - \frac{i}{3}) \cdot 9nB$, and by construction we have $p(\mathcal{Z}_i) = w(\mathcal{Z}_i)$. Thus,

$$\begin{aligned} p(\mathcal{X}^*) &\geq P - p(\mathcal{Z}_i) - p(\mathcal{Y}_i) \\ &= P - w(\mathcal{Z}_i) - w(\mathcal{Y}_i) - \left(\binom{i+1}{2} - \frac{i}{3}\right) \cdot 9nB \\ &= P - (t-1) \cdot Z - T \cdot Y - i \cdot 3B - \left(\binom{i+1}{2} - \frac{i}{3}\right) \cdot 9nB \\ &= (3t - 3i - 2) \cdot B + \left(\binom{t}{2} - \binom{i+1}{2} + \frac{i}{3}\right) \cdot 9nB \ . \end{aligned}$$

It therefore follows by Lemma 8 that instance $\mathcal{A}_i$ is indeed a "yes"-instance of Restricted Subset Sum, and the lemma follows. ◀

Proof of Theorem 2. We presented above an algorithm that composes any t instances $\mathcal{A}_0, \ldots, \mathcal{A}_{t-1}$ of Restricted Subset Sum into a single instance of Knapsack in polynomial-time. By Lemmas 12 and 13, the constructed Knapsack instance is a "yes"-instance if and only if $\mathcal{A}_i$ is "yes"-instance of Restricted Subset Sum for some $i \in \{0, \ldots, t-1\}$. Observe that total number of different weights in our constructed Knapsack instance is

$$w_\# \leq |\widetilde{\mathcal{A}}_n| + |\mathcal{Y}| + |\mathcal{Z}| = O(n^3 + \lg^2 t) \ .$$

Thus our algorithm fulfills all requirements of a composition algorithm, as given in Definition 4. The proof for $w_\#$ then follows by a direct application of Theorem 5. The statement for $p_\#$ follows by applying the reduction from Polak et al. [45, Chapter 4] which reduces an instance with $w_\# = k$ different item weights to an instance with $p_\# = k$ different item profits. ◀

4 Polynomial Kernel for Parameter $w_\# \cdot p_\#$

In this section we present a polynomial kernel for Knapsack parameterized by $w_\# + p_\#$, thereby proving Theorem 3. Our kernel is a direct generalization of the polynomial kernel for Subset Sum parameterized by $a_\#$ of Etscheid et al. [22].

The presented kernel utilizes two classic results: First, we use the fact that integer programming is fixed-parameter tractable with respect to the number of variables (this was first shown by Lenstra [39], and the currently best known running time with respect to the number of variable is due to Reis and Rothvoss [46]):

▶ **Theorem 14** ([46]). Integer Linear Programming *with input size s (i.e., the number of bits needed to encode the instance) and n variables can be solved in $2^{O(n \lg \lg n)} \cdot s^{O(1)}$ time.*

Second, we use the following theorem by Frank and Tardos [26]:

▶ **Theorem 15** ([26]). *There is an algorithm that, given a vector $w \in \mathbb{Q}^r$ and a natural number N, computes in polynomial time a vector $\overline{w} \in \mathbb{Q}^r$ with $\|w\|_\infty \leq 2^{4r^3} \cdot N^{r^2+2r}$ and $\operatorname{sign}(w \cdot b) = \operatorname{sign}(\overline{w} \cdot b)$ for every $b \in \mathbb{Z}^r$ with $\|b\|_1 \leq N$.*

Proof of Theorem 3. Let $w_1, \ldots, w_{w_\#}$ be the different weights and $p_1, \ldots, p_{p_\#}$ be the different profits in a given Knapsack instance with n items. We denote by $n_{i,j}$ for $i \in \{1, \ldots, w_\#\}$ and $j \in \{1, \ldots, p_\#\}$ the number of items with weight w_i and profit p_j. First note that the following is an ILP formulation of Knapsack with $w_\# \cdot p_\#$ many variables $x_{i,j}$, one for each $i \in \{1, \ldots, w_\#\}$ and $j \in \{1, \ldots, p_\#\}$, and two inequalities:

$$\begin{aligned} \sum_{i=1}^{w_\#} \sum_{j=1}^{p_\#} x_{i,j} \cdot w_i &\leq W \\ \sum_{i=1}^{w_\#} \sum_{j=1}^{p_\#} x_{i,j} \cdot p_j &\geq P \\ x_{i,j} &\in \{0, 1, \ldots, n_{i,j}\}. \end{aligned} \tag{3}$$

By Theorem 14, if $w_\# \cdot p_\# \cdot \lg\lg(w_\# \cdot p_\#) \leq \lg n$ (recall that n denotes the total number of items of the Knapsack instance), then the instance can be solved in polynomial time using ILP (3). Thus, devising a polynomial kernel in this case is trivial. So assume that $w_\# \cdot p_\# \cdot \lg(w_\# \cdot p_\#) > \lg n$. To reduce the encoding length of ILP (3), we apply Theorem 15 to the first two inequalities of ILP (3) as follows: let w^* be a $w_\# \cdot p_\#$-dimensional vector whose $(p_\# \cdot (i-1) + j)$'th component equals w_i, for each $i \in \{1, \ldots, w_\#\}$ and $j \in \{1, \ldots, p_\#\}$. We apply Theorem 15 to the $(w_\# \cdot p_\# + 1)$-dimensional vector $w := (w^*, -W)$ and $N := n+1$, resulting in a vector $(\overline{w}^*, -\overline{W})$. Similarly, let p^* be a $w_\# \cdot p_\#$-dimensional vector whose $(p_\# \cdot (i-1) + j)$'th component equals p_j, for $i \in \{1, \ldots, w_\#\}$ and $j \in \{1, \ldots, p_\#\}$. We apply Theorem 15 to the vector $w := (-p^*, P)$ and $N := n+1$, resulting in a vector $(-\overline{p}^*, \overline{P})$. By Theorem 15, ILP (3) is equivalent to the following ILP (4):

$$\begin{aligned} \overline{w}^* \cdot x &\leq \overline{W} \\ \overline{p}^* \cdot x &\geq \overline{P} \\ x_{i,j} &\in \{0, 1, \ldots, n_{i,j}\} \end{aligned} \tag{4}$$

By Theorem 15, each number from ILP (4) has encoding length $O(r^3 + r^2 \lg n)$ for $r = w_\# \cdot p_\#$. Since $\lg n \leq r \cdot \lg r$, it follows that the size of ILP (4) is $O(r^4 \cdot \lg r)$. As Knapsack is NP-complete, and Integer Linear Programming is in NP (see e.g. [15]), we can reduce ILP (4) in polynomial time to an instance of Knapsack. The resulting Knapsack instance is equivalent to the original instance, and has size polynomial in $r = w_\# \cdot p_\#$. ◀

Using the reduction from Unbounded Knapsack (i.e., the variation of Knapsack where a solution is a multiset of items instead of a regular set) to Knapsack (essentially also used e.g. by Etscheid et al. [22, Theorem 12]), one can also make the reduction from ILP (4) to Knapsack explicit, resulting in a kernel of size $\widetilde{O}((w_{\#} \cdot p_{\#})^5)$; see the full version [28] for details.

References

1 Amir Abboud, Karl Bringmann, Danny Hermelin, and Dvir Shabtay. Seth-based lower bounds for subset sum and bicriteria path. *ACM Trans. Algorithms*, 18(1):6:1–6:22, 2022. doi:10.1145/3450524.

2 Kyriakos Axiotis and Christos Tzamos. Capacitated dynamic programming: Faster knapsack and graph algorithms. In *Proc. of the 46th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 19:1–19:13, 2019.

3 MohammadHossein Bateni, MohammadTaghi Hajiaghayi, Saeed Seddighin, and Clifford Stein. Fast algorithms for knapsack via convolution and prediction. In *Proc. of the 50th ACM Symposium on the Theory Of Computing (STOC)*, pages 1269–1282, 2018.

4 Richard E. Bellman. *Dynamic programming*. Princeton University Press, 1957.

5 Hans L. Bodlaender, Rodney G. Downey, Michael R. Fellows, and Danny Hermelin. On problems without polynomial kernels. *Journal of Computer and System Sciences*, 75(8):423–434, 2009. doi:10.1016/J.JCSS.2009.04.001.

6 Hans L. Bodlaender, Bart M. P. Jansen, and Stefan Kratsch. Kernelization lower bounds by cross-composition. *SIAM Journal on Discrete Mathematics*, 28(1):277–305, 2014. doi:10.1137/120880240.

7 Ernest F. Brickell and Andrew M. Odlyzko. Cryptanalysis: A survey of recent results. *Proceedings of the IEEE*, 76(5):578–593, 1988.

8 Karl Bringmann. Knapsack with small items in near-quadratic time. In *Proc. of the 56th ACM Symposium on the Theory of Computing (STOC)*, 2024. To appear.

9 Karl Bringmann and Alejandro Cassis. Faster knapsack algorithms via bounded monotone min-plus-convolution. In *Proc. of the 49th International Colloquium on Automata, Languages, and Programming, (ICALP)*, pages 31:1–31:21, 2022.

10 Karl Bringmann and Alejandro Cassis. Faster 0-1-knapsack via near-convex min-plus-convolution. In *Proc. of the 31st Annual European Symposium on Algorithms (ESA)*, pages 24:1–24:16, 2023. doi:10.4230/LIPICS.ESA.2023.24.

11 Harry Buhrman, Bruno Loff, and Leen Torenvliet. Hardness of approximation for knapsack problems. *Theory of Computing Systems*, 56(2):372–393, 2015.

12 Timothy M. Chan. Approximation schemes for 0-1 knapsack. In *1st Symposium on Simplicity in Algorithms (SOSA)*, pages 5:1–5:12, 2018. doi:10.4230/OASICS.SOSA.2018.5.

13 Lin Chen, Jiayi Lian, Yuchen Mao, and Guochuan Zhang. Faster algorithms for bounded knapsack and bounded subset sum via fine-grained proximity results. In *Proc. of the 2024 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 4828–4848, 2024. doi:10.1137/1.9781611977912.171.

14 Benny Chor and Ronald R. Rivest. A knapsack-type public key cryptosystem based on arithmetic in finite fields. *IEEE Transactions on Information Theory*, 34(5):901–909, 1988.

15 William J. Cook, A. M. H. Gerards, Alexander Schrijver, and Éva Tardos. Sensitivity theorems in integer linear programming. *Mathematical Programming*, 34(3):251–264, 1986. doi:10.1007/BF01582230.

16 Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. doi:10.1007/978-3-319-21275-3.

17 Marek Cygan, Marcin Mucha, Karol Węgrzycki, and Michał Włodarczyk. On problems equivalent to (min, +)-convolution. *ACM Transactions on Algorithms*, 15(1):14:1–14:25, 2019.

18 Holger Dell and Dieter van Melkebeek. Satisfiability allows no nontrivial sparsification unless the polynomial-time hierarchy collapses. *Journal of the ACM*, 61(4):23:1–23:27, 2014.

19 Mingyang Deng, Ce Jin, and Xiao Mao. Approximating knapsack and partition via dense subset sums. In *Proc. of the 2023 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2961–2979, 2023. doi:10.1137/1.9781611977554.CH113.

20 Michael Dom, Daniel Lokshtanov, and Saket Saurabh. Kernelization lower bounds through colors and ids. *ACM Transactions on Algorithms*, 11(2):13:1–13:20, 2014.

21 Andrew Drucker. New limits to classical and quantum instance compression. *SIAM Journal on Computing*, 44(5):1443–1479, 2015.

22 Michael Etscheid, Stefan Kratsch, Matthias Mnich, and Heiko Röglin. Polynomial kernels for weighted problems. *Journal of Computer and System Sciences*, 84:1–10, 2017. doi:10.1016/J.JCSS.2016.06.004.

23 Michael R. Fellows, Serge Gaspers, and Frances A. Rosamond. Parameterizing by the number of numbers. *Theory of Computing Systems*, 50(4):675–693, 2012.

24 Fedor V. Fomin, Daniel Lokshtanov, Saket Saurabh, and Meirav Zehavi. *Kernelization: Theory of Parameterized Preprocessing*. Cambridge University Press, 2019.

25 Lance Fortnow and Rahul Santhanam. Infeasibility of instance compression and succinct PCPs for NP. *Journal of Computer and System Sciences*, 77(1):91–106, 2011.

26 András Frank and Éva Tardos. An application of simultaneous diophantine approximation in combinatorial optimization. *Combinatorica*, 7(1):49–65, 1987. doi:10.1007/BF02579200.

27 Teofilo F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38:293–306, 1985.

28 Klaus Heeger, Danny Hermelin, Matthias Mnich, and Dvir Shabtay. No polynomial kernels for knapsack. *CoRR*, 2023. doi:10.48550/arXiv.2308.12593.

29 Danny Hermelin, Shlomo Karhi, Michael L. Pinedo, and Dvir Shabtay. New algorithms for minimizing the weighted number of tardy jobs on a single machine. *Annals of Operations Research*, 298(1):271–287, 2021.

30 Russell Impagliazzo and Moni Naor. Efficient cryptographic schemes provably as secure as subset sum. *Journal of Cryptology*, 9(4):199–216, 1996.

31 Klaus Jansen, Felix Land, and Kati Land. Bounding the running time of algorithms for scheduling and packing problems. *SIAM Journal on Discrete Mathematics*, 30(1):343–366, 2016.

32 Ce Jin. An improved FPTAS for 0-1 knapsack. In *Proc. of the 46th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 76:1–76:14, 2019. doi:10.4230/LIPICS.ICALP.2019.76.

33 Ce Jin. 0-1 knapsack in nearly quadratic time. In *Proc. of the 56th ACM Symposium on the Theory of Computing (STOC)*, 2024. To appear.

34 Ravi Kannan. Minkowski's convex body theorem and integer programming. *Mathematics of Operations Research*, 12(3):415–440, 1987. doi:10.1287/MOOR.12.3.415.

35 Richard M. Karp. Reducibility among combinatorial problems. In *Proc. of a symposium on the Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

36 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, 2004.

37 Hans Kellerer, Ulrich Pferschy, and David Pisinger. *Knapsack problems*. Springer, 2004.

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

39 Hendrik W. Lenstra Jr. Integer programming with a fixed number of variables. *Mathematics of Operations Research*, 8(4):538–548, 1983. doi:10.1287/MOOR.8.4.538.

40 Silvano Martello and Paolo Toth. *Knapsack problems: algorithms and computer implementations*. John Wiley & Sons, Inc., 1990.

41 Ralph Merkle and Martin Hellman. Hiding information and signatures in trapdoor knapsacks. *IEEE Transactions on Information Theory*, 24(5):525–530, 1978.

42 Andrew M. Odlyzko. The rise and fall of knapsack cryptosystems. *Cryptology and Computational Number Theory*, 42:75–88, 1990.

43 Michael Pinedo. *Scheduling: Theory, Algorithms, and Systems*. Springer, 2016.

44 David Pisinger. Linear time algorithms for knapsack problems with bounded weights. *Journal of Algorithms*, 32:1–14, 1999.

45 Adam Polak, Lars Rohwedder, and Karol Węgrzycki. Knapsack and subset sum with small items. In *Proc. of the 48th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 106:1–106:19, 2021.

46 Victor Reis and Thomas Rothvoss. The subspace flatness conjecture and faster integer programming. In *Proc. of the 64th IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 974–988, 2023. doi:10.1109/FOCS57990.2023.00060.

47 Chee-Keng Yap. Some consequences of non-uniform conditions on uniform classes. *Theoretical Computer Science*, 26:287–300, 1983. doi:10.1016/0304-3975(83)90020-8.