A PTAS for Packing Hypercubes into a Knapsack

Klaus Jansen
Universität Kiel, Germany

Arindam Khan
Department of Computer Science and Automation, Indian Institute of Science, Bengaluru, India

Marvin Lira
Universität Kiel, Germany

K. V. N. Sreenivas
Department of Computer Science and Automation, Indian Institute of Science, Bengaluru, India

Abstract

We study the d-dimensional hypercube knapsack problem (d-D Hc-Knapsack) where we are given a set of d-dimensional hypercubes with associated profits, and a knapsack which is a unit d-dimensional hypercube. The goal is to find an axis-aligned non-overlapping packing of a subset of hypercubes such that the profit of the packed hypercubes is maximized. For this problem, Harren (ICALP'06) gave an algorithm with an approximation ratio of $(1 + 1/2^d + \varepsilon)$. For $d = 2$, Jansen and Solis-Oba (IPCO'08) showed that the problem admits a polynomial-time approximation scheme (PTAS); Heydrich and Wiese (SODA'17) further improved the running time and gave an efficient polynomial-time approximation scheme (EPTAS). Both the results use structural properties of 2-D packing, which do not generalize to higher dimensions. For $d > 2$, it remains open to obtain a PTAS, and in fact, there has been no improvement since Harren's result.

We settle the problem by providing a PTAS. Our main technical contribution is a structural lemma which shows that any packing of hypercubes can be converted into another *structured* packing such that a high profitable subset of hypercubes is packed into a constant number of *special* hypercuboids, called $\mathcal{V}$-Boxes and $\mathcal{N}$-Boxes. As a side result, we give an almost optimal algorithm for a variant of the strip packing problem in higher dimensions. This might have applications for other multidimensional geometric packing problems.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms

Keywords and phrases Multidimensional knapsack, geometric packing, cube packing, strip packing

Digital Object Identifier 10.4230/LIPIcs.ICALP.2022.78

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: https://arxiv.org/abs/2202.11902

Funding *Klaus Jansen*: Research supported by German Research Foundation (DFG), project JA 612/25-1.
Arindam Khan: Research partly supported by Pratiksha Trust Young Investigator Award, Google India Research Award, and Google ExploreCS Award.
Marvin Lira: Research supported by German Research Foundation (DFG), project JA 612/25-1.

Acknowledgements We thank Roberto Solis-Oba and three anonymous reviewers for their comments and suggestions on this paper.

1 Introduction

Multidimensional geometric packing problems are well-studied natural generalizations of the classical knapsack and bin packing problems. In the d-dimensional geometric knapsack problem (d-D Gen-Knapsack), where d is a fixed constant parameter, we are given a set of n items $\mathcal{I} := \{1, 2, \ldots, n\}$. Each item $i \in [n]$ is a d-dimensional (d-D) hypercuboid with

side length $s_k(i) \in (0, 1]$ along the k^{th} dimension and profit $p(i) \in \mathbb{Q}_{>0}$. The goal is to pack a subset of hypercuboids into a d-D unit knapsack (i.e., $[0, 1]^d$) such that the profit is maximized. We need the packing of the hypercuboids to be axis-aligned and non-overlapping. In this paper, we study d-dimensional hypercube knapsack (d-D Hc-Knapsack), a special case of d-D Gen-Knapsack, where all the items are hypercubes, i.e., for all $i \in [n]$, the i^{th} hypercuboid is a hypercube of side length $s(i)$.

d-D Gen-Knapsack generalizes the classical (1-D) knapsack problem [35] and thus is NP-hard. It finds numerous applications in scheduling, ad placement, and cutting stock [18]. For 2-D Gen-Knapsack, Jansen and Zhang [27] gave a $(2 + \varepsilon)$-approximation algorithm. Gálvez et al. [18] gave a 1.89-approximation and $(3/2 + \varepsilon)$-approximation for the cardinality version (i.e., all items have the same profit). Adamaszek and Wiese [3] gave a QPTAS when the input size is quasi-polynomially bounded. Gálvez et al. [19] later gave a pseudo-polynomial time $(4/3 + \varepsilon)$-approximation. For 3-D Gen-Knapsack, the present best approximation ratio is $7 + \varepsilon$ [14]. For $d \geq 4$, Sharma [41] has given a $(1 + \varepsilon)3^d$-approximation algorithm. Interestingly for $d \geq 2$, unlike d-D Gen-Knapsack, d-D Hc-Knapsack is not a generalization of 1-D knapsack. Leung et al. [36] showed 2-D Hc-Knapsack is strongly NP-hard, using a reduction from the 3-partition problem. The NP-hardness status of d-D Hc-Knapsack for $d > 2$, was open for a long time. Recently, Lu et al. [37, 38] settled the status by showing that d-D Hc-Knapsack is also NP-hard for $d > 2$.

A related problem is d-D Hc-Binpacking where we are given d-D hypercubes and the goal is to pack them into the minimum number of unit hypercubes. Back in 2006, Bansal et al. [5] gave an APTAS for this problem. Their algorithm starts by classifying the input set into small, medium, and large items. They first pack the large items ($O(1)$ in number) near-optimally by brute force and then pack the small items using Next Fit Decreasing Height (NFDH) [11] in the gaps left in between the large items. The remaining unpacked (small and medium) items are packed into additional bins using NFDH. However, this constructive approach can not be used to devise a PTAS for d-D Hc-Knapsack. Specifically, after packing the large items, the total space left to pack the small items can be very small and this space can be fragmented among several voids in between the large items. This renders packing the small items difficult, especially if the small items occupy a small volume in the optimal solution and carry a lot of profit. This turns out to be a bottleneck case. Otherwise, if the volume of small items in an optimal solution is significant, or if their profit is not significant, or if the empty space in the optimal solution is significant, we can easily devise a PTAS. In fact, the algorithm in [5] can be adapted to devise PTASes for the special cases of d-D Hc-Knapsack: (i) cardinality case, and (ii) when each item has the profit:volume ratio in the range $[1, r]$ for a fixed constant r. However, the case of arbitrary profits remains very difficult to handle.

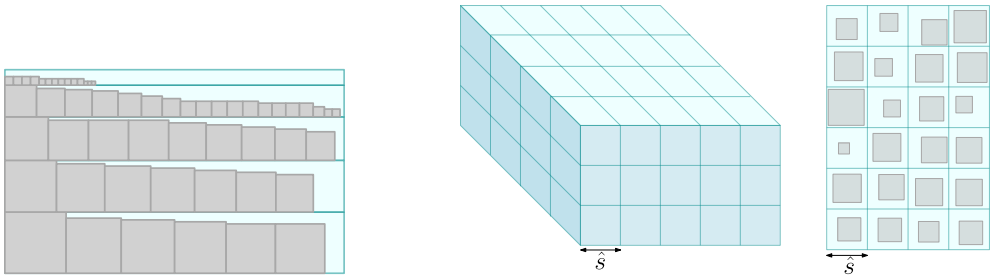
For d-D Hc-Knapsack, Harren [21] gave a $(1 + 1/2^d + \varepsilon)$-approximation algorithm, by removing a least profitable large item and using the empty space to pack the small items. Interestingly, the approximation ratio gets smaller as d grows. He also studied d-D Hc-StripPacking, where, given a set of hypercubes and a strip with a $(d - 1)$-dimensional base and unbounded height, the goal is to pack all the hypercubes while minimizing the height. He gave an APTAS for the special case of d-D Hc-StripPacking when the ratio between the shortest and longest sides of the base is bounded by a constant.

Jansen and Solis-Oba [26] gave a PTAS for 2-D Hc-Knapsack, by overcoming the above-mentioned bottleneck. They consider an optimal packing and categorize the packed items into *small, medium,* and *large* items. Then, by extending the edges of the large items, the remaining space is divided into $O(1)$ number of rectilinear regions, where each region

is classified as either *large* or *elongated* or *small*, based on the largest item intersecting or completely contained in it. Then, to make sure that a region has no items partially intersecting it, they repack all the items as follows. The *large* region is rectangular and its dimensions are much larger compared to the largest item intersecting it. Here NFDH can repack a high profitable subset of the items. An *elongated* region is again rectangular and it has one dimension much longer than the other. Here an algorithm for strip packing [28] is used to repack the items. Finally, the *small* regions are handled by applying the above transformations recursively as they can have complicated shapes (they may not be rectangular). Recently, Heydrich and Wiese [23] gave an EPTAS, effectively achieving the best-possible approximation. However, a PTAS for d-D Hc-Knapsack for $d > 2$ remains elusive as it is difficult to find such transformations for higher dimensions. Even for 3-D, the best-known approximation ratio remains $9/8$ [21]. In 2-D, many structural theorems [4, 18] show that a near-optimal solution exists where all items are packed into $O(1)$-number of rectangular regions where items are either packed as a stack or packed using NFDH. But these results do not generalize to higher dimensions. E.g., while extending the approach of *large* and *elongated* blocks of [26] to $d > 2$, we may not obtain a near-optimal structure packed in d-D hypercuboids; rather, we might obtain complicated rectilinear regions. However, prior to our work, there exist no algorithms which considered packing in such rectilinear regions.

1.1 Our Contributions

For the d-D Hc-Knapsack problem, we design a PTAS, thus settling the problem. Our main structural result intuitively says that any packing, by incurring a loss of only ε-fraction of the profit, can be transformed into $O(1)$ number of special hypercuboidal regions such that each region either contains a single large item, or contains items very small compared to its dimensions ($\mathcal{V}$-Box, see Figure 1a), or contains items placed along a multidimensional grid ($\mathcal{N}$-Box, see Figure 1b). We then provide an algorithm that guesses the arrangement of these hypercuboids and uses results of [26, 35] to find a near-optimal packing of items in these special boxes.



(a) An $\mathcal{V}$-Box is just a huge cuboid when compared to its size parameter $\hat{s}$. Each item has side length at most $\hat{s}$ and the item set assigned to it can be packed by NFDH.

(b) An $\mathcal{N}$-Box is a multidimensional grid. An item set assigned to an $\mathcal{N}$-Box can be packed by placing each item in a cell.

Figure 1 $\mathcal{V}$-Box and $\mathcal{N}$-Box.

For our structural theorem, we start with an optimal solution. First, we divide the items in the optimal packing into large, medium, and small items such that the medium items have a tiny profit and can be discarded. Then, by extending the facets of the large items, we create a non-uniform grid with $O(1)$ hypercuboidal cells. However, some items may intersect the facets of these cells. We consider the facets of all the cells and identify (based on the intersecting items) each of the facets as either *good* or *bad*. Intuitively, if we only have good facets then we can repack items so that no items intersect the facets. To get rid of the bad

facets, we merge some cells so that they form composite cells, which we call *collections* (See Figure 2). Now we need to repack the items in the *collections* which may have complex rectilinear shapes. If the collection has $k \in \{0, 1, \ldots, d\}$ long dimensions then we call it a k-elongated collection. Note that we can have $d + 1$ types of collections.

Our structural theorem can be viewed as a generalization of [26, 21]; however, due to the complicated shape of collections, we had to overcome several technical obstacles in the process. For simplicity, we explain the intuition for $d = 3$, where we obtain a packing of items into four types of rectilinear regions (collections) as in Figure 2.

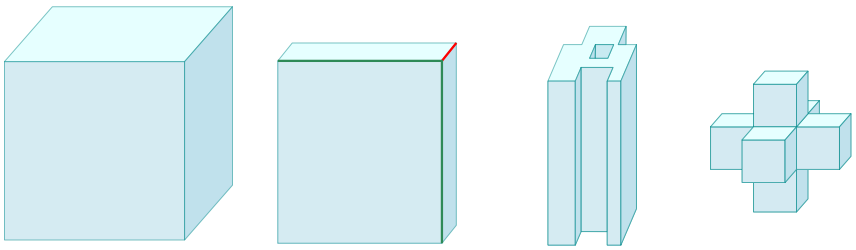


Figure 2 Repacking items inside the left-most collection is easy (using NFDH). The right-most collections are handled recursively. However, repacking the items in the other two collections is challenging (we needed a novel strip packing algorithm).

The first collection (3-elongated) is a $\mathcal{V}$-Box, i.e., all its three dimensions are long compared to the largest item packed in it. The simple NFDH algorithm suffices to repack the items inside it while losing only a few items with a small profit. Although repacking in large collections can be easily handled by NFDH, there exist no previous algorithms to repack items in the remaining types of collections. Now for $k \in [d-1]$, collections are long in at least one dimension and thus can be viewed as a type of strip (though the base may not be rectangular). Harren [21] showed how to repack items almost optimally in a d-dimensional strip with a rectangular base with a bounded aspect ratio. However, their approach doesn't trivially extend to the 1- and 2-elongated collections in Figure 2, as the base can be non-rectangular or it may not satisfy the property of bounded aspect ratio even though it is rectangular. In this paper, we circumvent this bottleneck by designing a PTAS (under resource augmentation) for d-dimensional strip packing to pack items on such complicated bases. Now we explain how we use the strip packing algorithm to pack these collections.

In the second collection (2-elongated), one dimension (marked in red) is reasonably short compared to the largest item intersecting it and the other dimensions (in green) are long. We first divide the items in the collection into large, medium, and small items and ensure that the total volume of medium items is marginal. We then consider the large items and round their sizes to up to $O(1)$ types using linear grouping [28]. The number of (rounded) large items that can fit on along the short dimension is $O(1)$. So, similar to [28], we solve an LP that represents the fractional strip packing with (rounded) large items where the items are allowed to be sliced along the long dimensions. We then convert this into an integral packing of large items using resource augmentation along the long dimensions. Then we pack as many small items as possible in the gaps left. The rest of the small items, together with the medium items, are packed on the top.

The third collection (1-elongated), though complicated, is ensured (by our construction) to have a base with a bounded aspect ratio[1]. Moreover, it can be viewed as a union of $O(1)$ number of smaller disjoint rectangles (which we call *base cells*). We again classify each item

[1] More accurately, the minimal cuboid, which completely contains the base, has bounded-aspect ratio.

in this collection as large, medium, or small. As above, we round the large items to $O(1)$ types. Note that the number of large items that can fit on the base of the strip is only a constant (due to the bounded aspect ratio). We pack them integrally by first solving the fractional LP where items are allowed to be sliced along the long dimension. Our main novelty, in this case, is the way in which we pack the small items on the top of the strip. Unlike [21], we can't use NFDH directly to pack the small items as the base isn't rectangular. Instead, we first merge a few base cells to form a larger base cell which is rectangular (we show that this is indeed possible) and is large enough to accommodate the largest item (and thus any item). Now, having at least one large base cell, we distribute the small and medium items among all the base cells so that when they are packed using NFDH in strips on these base cells, the maximum height is minimized. Observe that if we pack the items on a single base cell, we may not efficiently use the entire area of the base and this can lead to a very tall strip. Thus this process of distributing the items among all the base cells is important.

We use our near-optimal strip packing algorithm on these 1- or 2-elongated collections to repack the items while losing only a small profit. Interestingly, the strip packing algorithm for both kinds of bases (in general, for any k-elongated collection where $k \in [d-1]$) is the same; the only change is the number of short dimensions of the base that is given as a parameter to the algorithm. We provide the algorithm for d-dimensional strip packing in the full version [24].

Finally, for 0-elongated collections, we apply the above process recursively. Using a shifting argumentation, we show that $O_\varepsilon(1)$ steps of recursion is sufficient. We also show that the resulting packing is a packing into $O(1)$ number of $\mathcal{V}$-Boxes and $\mathcal{N}$-Boxes. With more technical ingenuity, these ideas can be extended to the case of $d > 3$.

We believe that our techniques will be helpful for other multidimensional geometric packing problems involving rectilinear regions that are not hypercuboids.

1.2 Related Work

For geometric bin packing, Caprara gave a 1.691^{d-1} asymptotic approximation [9]. For $d = 2$, it admits no APTAS [5] and there is a 1.406-approximation algorithm [7]. There are several other variants of bin packing such as vector packing [6, 40], strip packing [22, 16, 25], sliced packing [13, 17], mixed packing [34], etc. The knapsack problem is also studied under several generalizations such as vector knapsack [15], fair knapsack [39], and mixed knapsack [33]. Packing problems are also well-studied under guillotine cut constraints [8, 30, 31, 1, 32]. Guillotine cut also has interesting connections with the maximum independent set of rectangles problem [2, 20]. We refer the readers to [10, 29] for a survey on multidimensional packing.

1.3 Overview of the Paper

In Section 2, we present some preliminaries. Section 3 contains the result for d-D Hc-Knapsack. The result for strip packing, which will be used as a subroutine to obtain our main structural result is given in Section 3.1.2. Due to space limitations, many proofs had to be moved to the full version [24].

2 Notations and Preliminaries

Let $[n] := \{1, 2, \ldots, n\}$. For any set of items $\mathcal{I}$, we define $\hat{s}(\mathcal{I}) := \max_{i \in \mathcal{I}} s(i)$. For any k-dimensional region R, we denote its volume by $\mathrm{VOL}_k(R)$. We extend this notation to an item i or a set of items $\mathcal{I}$, i.e., $\mathrm{VOL}_d(i) = (s(i))^d$, and $\mathrm{VOL}_d(\mathcal{I}) = \sum_{i \in \mathcal{I}} \mathrm{VOL}_d(i)$. Also, the profit of a packing is the sum of the profits of the packed items $\mathcal{Q}$: $p(\mathcal{Q}) := \sum_{i \in \mathcal{Q}} p(i)$.

Consider a set of points X in $[0,1]^d$ where each point $x \in X$ can be represented as $(x_1, x_2, \ldots, x_d)$. For any set of dimensions $\mathcal{D} \subseteq [d]$, we define the projection of X onto $\mathcal{D}$ as the set $\left\{ \left(x_{d_1}, \ldots, x_{d_{|\mathcal{D}|}}\right) \mid (x_1, \ldots, x_k) \in X \right\}$, where $\left\{ d_1, \ldots, d_{|\mathcal{D}|} \right\} = \mathcal{D}$ and $d_i < d_{i+1}$ for all $i \in [|\mathcal{D}| - 1]$.

We only consider axis-aligned packing of hypercuboids and hypercubes. Thus, a d-D hypercuboid C is given by the position of its lower corner $(x_1, \ldots, x_d) \in \mathbb{R}^d$ and side lengths $\ell_1, \ldots, \ell_d \in \mathbb{R}_+$. Then, C is the cartesian product of the intervals $\prod_{i=1}^{d} [x_i, x_i + \ell_i]$, and $\mathrm{VOL}_d(C) := \prod_{i=1}^{d} \ell_i$ and its surface area is defined as the sum of the volumes of its $(d-1)$-dimensional facets, i.e., $\mathrm{SURF}_d(C) := 2\sum_{i=1}^{d} \prod_{j=1, j\neq i}^{d} \ell_j = 2\sum_{i=1}^{d} \mathrm{VOL}_d(C)/\ell_i$. A packing is a subset $\mathcal{Q} \subseteq \mathcal{I}$ of items with positions $\mathrm{pos} : \mathcal{Q} \longrightarrow [0,1]^d$. It is valid, if each item $i \in \mathcal{Q}$ with the lower corner positioned at $\mathrm{pos}(i)$ is completely included in the unit hypercube $[0,1]^d$ and each pair of items $i, j \in \mathcal{Q}$ is positioned non-overlapping.

Consider a set of items lying in d dimensional space. For any region in the d dimensional space, we define the profit of that region as the profit of the set of items *completely* contained in that region.

We will use the higher dimensional variant of the well-known Next Fit Decreasing Height (NFDH) algorithm extensively. For the sake of completeness, we will provide a brief description of the algorithm here.

NFDH is a two dimensional packing algorithm introduced in [11] for packing a given set of rectangles in a bigger rectangular region $\mathcal{R}$. Informally, it works as follows. First, it sorts the given set of rectangles in decreasing order of heights (the vertical dimension). Then, it starts off by packing the rectangles on the base of $\mathcal{R}$ side-by-side as much as possible. Then, this level is closed, i.e., the base of $\mathcal{R}$ is shifted vertically to the top of the first rectangle packed. In the next step, the remaining rectangles are packed on the new base to whatever extent is possible. This process continues.

One can easily extend the NFDH algorithm to pack hypercuboids into a bigger hypercuboidal region. Let $k \in \mathbb{N}_{\geq 2}$. For any k-dimensional hypercuboid x, let $x^{(i)}$ denote the i-dimensional hypercuboid obtained by considering only the first i dimensions of x. We can extend this notation to sets: Let J be a set of k-dimensional hypercuboids. Then $J^{(i)} = \{x^{(i)} | x \in J\}$.

We define the NFDH algorithm in k dimensions (k-NFDH) recursively as follows. If $k = 2$, then the algorithm is simply NFDH as described above. Suppose $k > 2$. Let S be a set of k-dimensional hypercuboids to be packed into a k-dimensional hypercuboidal region $\mathcal{R}$. We first sort the rectangles in decreasing order of their lengths in the k^{th} dimension. Then we pick the largest possible prefix P of S such that $P^{(k-1)}$ can be completely packed on the base of $\mathcal{R}$ using $(k-1)$-NFDH. We pack P on the base of $\mathcal{R}$, and shift the base to the top of the tallest (longest in the k^{th} dimension) item in P. We repeat this process with $S \backslash P$.

The pseudocode of k-NFDH and some illustrations have been provided in the full version [24]. We will abbreviate k-NFDH by just NFDH. The value of k will be clear from the context. Harren [21] gave a surface area based efficiency guarantee of NFDH algorithm for hypercubes.

▶ **Lemma 1** ([21]). *Consider a set S of hypercubes with side lengths at most δ and a hypercuboid C. The NFDH algorithm either packs all items from S into C or the total volume left free inside of C is at most $\delta\, \mathrm{SURF}_d(C)/2$.*

Now we define $\mathcal{V}$-Box and $\mathcal{N}$-Box.

▶ **Definition 2** ($\mathcal{V}$-Box and $\mathcal{N}$-Box). *Let B be a d-dimensional hypercuboid, $\mathcal{Q}$ be a set of items packed in it, and let $\hat{s}$ be an upper bound on the side lengths of the items in $\mathcal{Q}$. We say that B is a $\mathcal{V}$-Box if its side lengths can be written as $n_1\hat{s}, n_2\hat{s}, \ldots, n_d\hat{s}$ where $n_1, n_2, \ldots, n_d \in \mathbb{N}_+$*

and the volume of the packed items is at most $\mathrm{VOL}_d(B) - \hat{s}\frac{\mathrm{SURF}_d(B)}{2}$. *We say that* B *is an* $\mathcal{N}$*-Box if its side lengths can be written as* $n_1\hat{s}, n_2\hat{s}, \ldots, n_d\hat{s}$ *where* $n_1, n_2, \ldots, n_d \in \mathbb{N}_+$ *and the number of packed items is at most* $\prod_{i=1}^{d} n_i$.

As the side lengths are integral multiples of $\hat{s}$, the number of distinct $\mathcal{V}$-Boxes and $\mathcal{N}$-Boxes in the near-optimal structure will be polynomially bounded. The following corollary follows from Lemma 1.

▶ **Corollary 3.** *Let* B *be a* $\mathcal{V}$*-Box with the item set* $\mathcal{Q}$ *packed in it. Then these items can be repacked into* B *using NFDH.*

Proof. We can either pack all the items in $\mathcal{I}$ into B using NFDH, or, by Lemma 1, the free volume inside of B is at most $\hat{s}\,\mathrm{SURF}_d(B)/2$. The latter possibility however implies that we packed items with a volume of at least $\mathrm{VOL}_d(B) - \hat{s}\,\mathrm{SURF}_d(B)/2$, which is an upper bound on $\mathrm{VOL}_d(\mathcal{I})$ by the definition of a $\mathcal{V}$-Box. ◀

▶ **Observation 4.** *Let* B *be an* $\mathcal{N}$*-Box with the item set* $\mathcal{Q}$ *packed in it. As* $|\mathcal{Q}| \leq \prod_{i=1}^{d} n_i$, *we can divide* B *into* $\prod_{i=1}^{d} n_i$ *cells such that each cell has length* $\hat{s}$ *in each of the* d *dimensions and we can place each item in* $\mathcal{Q}$ *in one cell.*

In this paper, we will often require that a set of hypercuboidal regions forms a grid.

▶ **Definition 5.** *A* d-dimensional grid *(see Figure 3(a)) C is a subset of the set of hypercuboids* $\left\{ \prod_{i=1}^{d} [g_{i,j_i-1}\ ,\ g_{i,j_i}] \;\middle|\; 1 \leq j_i \leq n_i \text{ for all } i \in [d] \right\}$, *where each* $n_i \in \mathbb{N}_+$ *denotes the number of grid layers in the* i^{th} *dimension, and* $g_{i,0} < \cdots < g_{i,n_i}$ *for all* $i \in [d]$ *are the grid boundaries. Each hypercuboid in C is called a* cell.

Each grid C has $|C| \leq \prod_{i=1}^{d} n_i$ cells. A grid C can also be refined by splitting some cells.

▶ **Lemma 6.** *Let* C *be a* d*-D grid with* $n_1, \ldots, n_d \in \mathbb{N}_+$ *layers and boundaries* $g_{i,j} \in \mathbb{R}$ *for* $i \in [d]$ *and* $j \in [n_i]$. *Let* R *be a set of* d*-D hypercuboids. Then the region of the grid which is not intersected by* R *is a new grid with* $n'_i \leq n_i + 2|R|$ *layers in each dimension* $i \in [d]$.

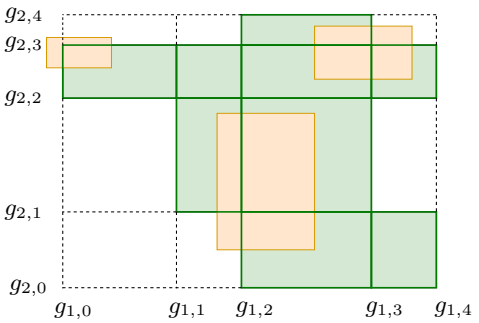
Proof. Each hypercuboid in R is the cartesian product of d intervals $\prod_{i=1}^{d} [a_i, b_i]$. The grid C can now be refined by adding the boundaries a_i and b_i in each dimension i. This increases the number of layers in each dimension by at most $2|R|$. After this refinement, each cell is either completely contained in a hypercuboid in R, and hence discarded from C, or does not intersect R. This can be seen in Figure 3(b). ◀

3 Knapsack

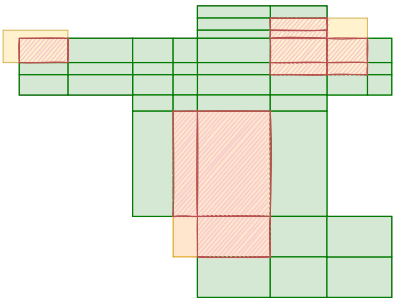
In this section, we will devise a PTAS for the d-D Hc-Knapsack problem.

3.1 Structure of a Nearly Optimal Solution

First, we prove that given an optimal packing $\mathrm{OPT}_{\text{knapsack}}(\mathcal{I})$ of the input set $\mathcal{I}$, there exists another packing containing a subset of the items packed in a simple structure which can be searched for, in polynomial time. For this, we modify the optimal packing to obtain a near-optimal packing in which all the items are packed into either $\mathcal{V}$-Boxes and $\mathcal{N}$-Boxes except for a constant number of large items. The total number of these boxes is $\mathcal{O}(1)$ and their sizes come from a set whose cardinality is polynomial in $|\mathcal{I}|$. Hence, this near-optimal packing can be searched for, in polynomial time.



(a) 3 orange rectangles partially covering a grid with 4 layers in each dimension.The dotted lines indicate the grid boundaries and the grid cells are indicated by the green rectangles.



(b) Refined grid with ≤ 10 layers in each dimension. The new grid cells are indicated in green. The shaded cells are those discarded from the refined grid as they overlap with the items.

Figure 3 Splitting the empty space inside of a grid.

▶ **Theorem 7.** *For each $0 < \varepsilon < 1/2^{d+2}$, there is a packing with the following properties:*

(i) *It consists of $\mathcal{N}$-Boxes and $\mathcal{V}$-Boxes whose total number is bounded by a constant $C_{\text{boxes}}(d, \varepsilon)$, which depends only on ε and d.*

(ii) *The number of items in the packing that are not packed in these boxes is bounded by a constant $C_{\text{large}}(d, \varepsilon)$, which depends only on ε and d.*

(iii) *The profit of the packing is at least $\left(1 - 2^{d+2}\varepsilon\right) \text{OPT}_{\text{knapsack}}(\mathcal{I})$.*

To prove the theorem, first, in Section 3.1.1, we consider a packing in an arbitrary grid (note that the knapsack can be viewed as a grid containing a single cell) and merge some of its cells into collections based on the packing of the items into the grid. The reason for choosing an arbitrary grid is that, as we will see, we recursively divide the knapsack into grids (which can have arbitrary shapes) and consider each of them separately. Then in Section 3.1.3, we show how to repack the items in a collection using NFDH or by a strip packing algorithm described in Section 3.1.2.

3.1.1 Partitioning a Grid into Collections

Consider a grid and a set of items $\mathcal{J}$ packed into it. Assume that the grid has at most N_{layer} layers in each dimension. Thus, it consists of at most $N := N_{\text{layer}}^d$ cells. Our goal is to repack (a subset of) $\mathcal{J}$ to obtain a simpler structure, by losing only a small profit. The overall grid can have a complex structure, so we consider each cell (a hypercuboid) and repack the items in the cell. However, it might be problematic to repack the items that intersect a cell only partially. Consider one such item and one of the cells it intersects. There must be a facet of this cell that cuts into this item. If this facet is orthogonal to a long dimension of the cell, then we can remove a strip of small profit and pack these intersecting items in that strip. However, an issue arises when this facet is orthogonal to a short dimension. Therefore, we merge the two cells that share this facet. Doing this iteratively, we merge some cells into a *collection of cells* or simply *collection*. See left of Figure 4.

For a cell a, we will denote the side length of the largest item of $\mathcal{J}$ partially or fully packed in it by $\hat{s}(a)$ and we denote the side lengths of the cell by $a_1, \ldots, a_d$. Furthermore, we sort the dimensions using a stable sorting algorithm [2], such that the side lengths of a

[2] A stable sorting makes the order unambiguous. This simplifies some proofs that compare the orders for different collections.

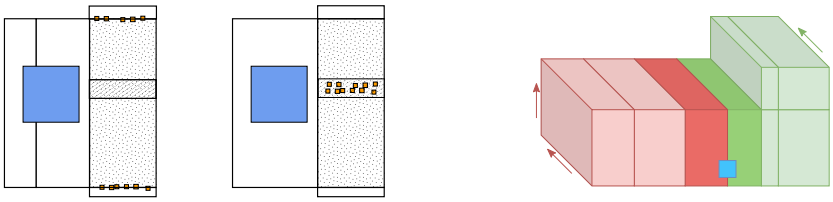


Figure 4 *(i)* The left figure shows five cells of a grid in the two-dimensional case. The orange items are small compared to the height of one of the cells (shaded with dots) they intersect. So, we transfer them to the interior of that cell by removing a least profitable strip (shaded in stripes). However, the blue item cannot fit in any of the cells it intersects, so we merge both these cells. *(ii)* The right figure demonstrates merging. An item (blue) intersects a bad facet between a 2-elongated collection (red) and a 1-elongated collection (green). After merging, we obtain a single 1-elongated collection.

in those dimensions are in non-increasing order. We will refer to that order as σ_a and thus, $a_{\sigma_a(i)}$ is the i^{th} largest side length of a. We start by defining the values which we will use to distinguish between long and short sides of a cell as

$$\alpha_d := \frac{2d}{\varepsilon}, \text{ and } \alpha_k := \frac{2}{\varepsilon}\left(C_{\text{configs}}\left(d, k, \alpha_{k+1}, N, \varepsilon\right) + 3\right) \text{ for all } k \in [d-1] \tag{1}$$

The parameter $C_{\text{configs}}\left(d, k, \alpha_{k+1}, N, \varepsilon\right)$ is a constant depending on $d, k, \alpha_{k+1}, N, \varepsilon$. Its exact value is derived in the full version [24], but for our purposes, it suffices to note that for all $k \in [d-1]$, $C_{\text{configs}}\left(d, k, \alpha_{k+1}, N, \varepsilon\right) \geq 1$ and its value gets larger when k gets smaller. Hence, $1 < 2d/\varepsilon = \alpha_d < \ldots < \alpha_1$.

Now we can define a cell a to be k-elongated for $0 \leq k \leq d$ iff k of its side lengths are long compared to some size parameter s.

▶ **Definition 8.** *A cell a is k-elongated for $k \in [0, d]$ and size parameter $s > 0$ iff*

(i) $a_{\sigma_a(i)} > \alpha_k s$ *for all* $1 \leq i \leq k$ *and*

(ii) $a_{\sigma_a(i)} \leq \alpha_i s$ *for all* $k < i \leq d$.

If a is a k-elongated cell, then for any $i \in [k]$ we call $\sigma_a(i)$ to be a *long* dimension of a and the other dimensions to be *short*. Note that for each size parameter, each cell is k-elongated for exactly one value of k. A facet of a k-elongated cell is called *good* if it is orthogonal to a long dimension and *bad* otherwise. We will see that the items that intersect only the good facets of a cell don't cause much of an issue; the complicated machinery that we devise is to deal with the items intersecting the bad facets.

For a group of cells to be merged into a collection, they should satisfy some additional properties. We define a *k-elongated collection* C as a set of cells that are k-elongated using the size of the largest item completely or partially packed in that collection as a common size parameter and are aligned in their long dimensions. More formally,

▶ **Definition 9.** *A set of cells C is a k-elongated collection if*

(i) *each cell $a \in C$ is k-elongated for the size parameter $\hat{s}(C) := \max_{c \in C}(\hat{s}(c))$,*

(ii) $\sigma_a(i) = \sigma_b(i)$ *for all* $1 \leq i \leq k$ *and all* $a, b \in C$ *and*

(iii) *the projection of a and b onto the dimension $\sigma_a(i)$ is the same for all* $1 \leq i \leq k$ *and all* $a, b \in C$.

(iv) *The set of cells form a path-connected region.*

By properties (i) and (ii) of a k-elongated collection, we can define the k dimensions that are long for any cell of the collection to be the long dimensions of the collection itself. Property (iii) strictly limits the arrangement of the cells in long dimensions. Thus, a larger k makes

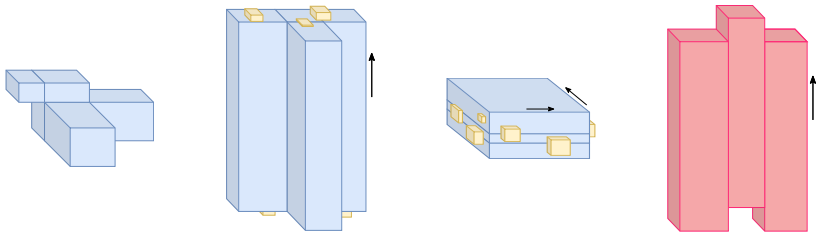


Figure 5 Different collections in 3-D with some items intersecting good facets. The arrow marks indicate the long dimensions. From left to right: 0-, 1-, 2-elongated collections, and a non-example of a collection (violates property (iii).)

the shape of the whole collection less complex (See Figure 5). A common facet between two cells $a \in C$ and $b \in D$ of two different collections C and D is called an outer facet for each of those collections. For each dimension, each cell has two facets that are orthogonal to this dimension. The facet with the lower coordinate in this dimension is called the bottom facet and the other one is called the top facet. For each long dimension of a collection the *bottom facet of the collection* is formed by the union of the bottom facets of its cells; the *top facet of the collection* is formed by the top facets of its cells.

Initially, we consider each cell of our grid as a collection containing only itself. Whenever there is an item intersecting an outer facet between two collections C and D, we will merge those collections if that facet is bad (i.e. orthogonal to a short dimension) for both of them. Such a case is visualized on the right of Figure 4. The next lemma proves that merging collections in this case will create a new collection.

▶ **Lemma 10.** *Let C be a k_C-elongated collection and let D be a k_D-elongated collection. Let $a \in C$ and $b \in D$ be two cells that have a common facet which is bad for both of them. Then $C \cup D$ is a $\min(k_C, k_D)$-elongated collection.*

Proof. Note that both k_C, k_D are strictly less than d since a, b share a common facet f which is bad for both. W.l.o.g., we will assume that $\hat{s}(C) \geq \hat{s}(D)$. Let f has side lengths $f_1, \ldots, f_d$ where f_i denotes the length in the i^{th} dimension (for simplicity, we assume f is a d-dimensional hypercuboid with a side of zero length). Note that the facet f has the same side lengths as a and b in any dimension except the dimension in which it has zero length. This implies that a and b are of almost the same shape. In the dimension orthogonal to f, both a and b have a rather short length, as f is a bad facet for both collections. Using these arguments, we can derive the following properties:

- **(i)** $a_{\sigma_f(i)} = f_{\sigma_f(i)} = b_{\sigma_f(i)}$ for all $1 \leq i \leq d-1$ (Since f is common to both a, b)
- **(ii)** $\sigma_f(i) = \sigma_a(i)$ for all $1 \leq i \leq k_C$ (Since f is bad for a which is a k_C-elongated cell)
- **(iii)** $\sigma_f(i) \in \{\sigma_a(i), \sigma_a(i+1)\}$ for all $k_C < i < d$ (explained below)
- **(iv)** $\sigma_f(i) = \sigma_b(i)$ for all $1 \leq i \leq k_D$
- **(v)** $\sigma_f(i) \in \{\sigma_b(i), \sigma_b(i+1)\}$ for all $k_D < i < d$

The property (iii) above is due to the fact that a and f have the same lengths in all dimensions except one, say $d_\perp$. So, the order of dimensions $\sigma_a(1), \sigma_a(2), \ldots, \sigma_a(d)$ can be obtained by inserting $d_\perp$ in the list $\sigma_f(1), \sigma_f(2), \ldots, \sigma_f(d-1)$ at the appropriate index but we know that this index lies after k_C since $d_\perp$ is short for a.

If we assume $k_C > k_D$, we obtain the inequality $f_{\sigma_f(k_C)} = a_{\sigma_a(k_C)} > \alpha_{k_C}\hat{s}(C)$ by (i), (ii) and Definition 8 of the k_C-elongated cell a. We also obtain the contradictory inequality $f_{\sigma_f(k_C)} = b_{\sigma_f(k_C)} \leq b_{\sigma_b(k_C)} \leq \alpha_{k_C}\hat{s}(D) \leq \alpha_{k_C}\hat{s}(C)$ by (i), (iv) and (v) and Definition 8 of the k_D-elongated cell b. Thus, our choice $\hat{s}(C) \geq \hat{s}(D)$ determines that $k_C \leq k_D$.

As $\hat{s}(C \cup D) = \hat{s}(C)$ and C is a k_C-elongated collection, we know that $C \cup D$ fulfills property (i) of Definition 9 for a k_C-elongated collection for every $a' \in C$. Let $b' \in D$. We have to show that b' is a k_C-elongated cell using the size parameter $\hat{s}(C)$ as well. For every $1 \leq i \leq k_C$ we can use Definition 9 of the k_D-elongated collection D and (i), (iv) and (ii) to prove $b'_{\sigma_{b'}(i)} = b_{\sigma_b(i)} = f_{\sigma_f(i)} = a_{\sigma_a(i)} > \alpha_{k_C}\hat{s}(C)$. For every $k_C < i \leq k_D$ we can again use Definition 9 of the collection D and properties (i), (iv) and (iii) to prove $b'_{\sigma_{b'}(i)} = b_{\sigma_b(i)} = f_{\sigma_f(i)} \leq a_{\sigma_a(i)} \leq \alpha_i\hat{s}(C)$. For every $k_D < i \leq d$ we already know that $b'_{\sigma_{b'}(i)} \leq \alpha_i\hat{s}(D) \leq \alpha_i\hat{s}(C)$ because of Definition 8 for the k_D-elongated cell b'. Thus, b' is a k_C-elongated cell using the size parameter $\hat{s}(C)$.

Now we only need to prove properties (ii) and (iii) of Definition 9 for every pair of cells in $C \cup D$ to show that it is a collection. For a pair $a', a'' \in C$ or $b', b'' \in D$ this is trivial, because C and D are k_C- or k_D-elongated collections and $k_C \leq k_D$. For the pair a and b, those properties are again trivial, because both have all their long sides on the common facet f. For any $a' \in C$ and $b' \in D$ the properties follow as the pairs a' and a, a and b, b and b' satisfy these properties. ◀

After this merging procedure, we will assign each item $x \in \mathcal{J}$ to a collection C as follows: If x is completely packed in a collection, then we assign it to that collection itself. If it is partially contained, then we assign it to one of the collections (breaking ties arbitrarily) with which it intersects only via the good outer facets. The next lemma shows that this assignment is indeed possible.

▶ **Lemma 11.** *Let G be the set of all collections that were derived after the merging procedure. Let $x \in \mathcal{J}$ be an item that is packed in the grid but isn't completely packed in any collection in G. Then there exists a collection $C \in G$ in which x is partially packed but x does not intersect any of its bad outer facets.*

Proof. First, we define an order on the set of the collections G. We associate each k_C-elongated collection $C \in G$ with the tuple of the long sides $\left(a_{\sigma_a(1)}, \ldots, a_{\sigma_a(k_C)}\right)$ for some cell $a \in C$ in a descending order. By property (iii) in the definition of a k_C-elongated collection, the tuple is independent of the choice of the cell. We now define the strict lexicographic order $\prec$ on these tuples. In other words, for a k_C-elongated collection C, a k_D-elongated collection D and cells $a \in C$ and $b \in D$, we have $C \prec D$ iff

- **(i)** $k_C < k_D$ and $a_{\sigma_a(i)} = b_{\sigma_b(i)}$ for all $1 \leq i \leq k_C$, or
- **(ii)** there is some $1 \leq k \leq \min(k_C, k_D)$ such that $a_{\sigma_a(i)} = b_{\sigma_b(i)}$ for all $i \in [k-1]$ and $a_{\sigma_a(k)} < b_{\sigma_b(k)}$.

Let $H \subseteq G$ be the set of collections where x is partially contained in. Let $C \in H$ be a maximal collection in H according to $\prec$ and let it be k_C-elongated. Now we want to prove that x does not intersect any bad outer facet of C. Assume there is a bad outer facet f of C which is intersected by x. This facet f belongs to some cells $a \in C$ and $b \in D$ where D is a different collection than C. The facet f has to be a good facet of D, because it is bad for C and the collections C and D were not merged. Let $d_\perp$ be the dimension which is orthogonal to f. Like in the proof of Lemma 10, we know that $a_i = b_i$ for all $i \in \{1, \ldots, d\} \setminus \{d_\perp\}$ because of their common facet f. As f is bad for C this contains all the sides of a that are long using the size parameter $\hat{s}(C)$. If a side of a is long using the size parameter $\hat{s}(C)$ and larger than $b_{d_\perp}$, then this side is also long using the size parameter $\hat{s}(D)$, because $b_{d_\perp}$ is already considered as long using this size parameter. Thus, we have two cases: In the first case, each side of a which is long using the size parameter $\hat{s}(C)$ is larger than $b_{d_\perp}$. Then each long side of a is also a long side of b while b has at least one additional long side $b_{d_\perp}$. Thus, we have $C \prec D$ by (i). In the second case there are only $k < k_C$ sides of a long using

the size parameter $\hat{s}(C)$ and larger than $b_{d_\perp}$. Then the k longest sides of a and b have the same size and for the $(k+1)^{\text{th}}$ longest sides we have $a_{\sigma_a(k+1)} < b_{d_\perp} = b_{\sigma_b(k+1)}$. Thus, we have $C \prec D$ by (ii). In both cases, the maximality of C is violated, and therefore no bad outer facet of C can be intersected by x. ◀

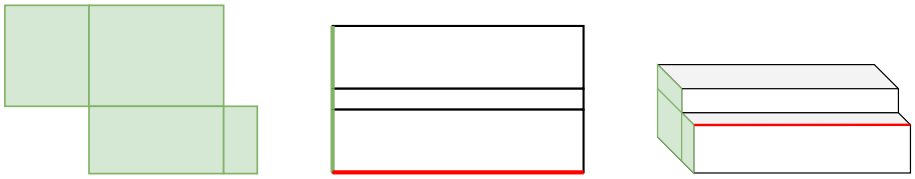
3.1.2 Strip Packing with Resource Augmentation

Before we proceed to repack the items inside of those collections, we will have a closer look at an arbitrary k-elongated collection for some $k \in [d-1]$. This constraint on k ensures that the collection is large in at least one dimension and short in at least one dimension. Thus our collection is some kind of a strip (may not be hypercuboidal). Previously, we reordered the dimensions for a collection to have the k long dimensions first; however, for the strip packing problem we assume the opposite, i.e., we suppose that the collection is short in the first $(d-k)$ dimensions and long in the last k dimensions. Each cell of this collection is the cartesian product of some intervals and property (iii) of Definition 9 ensures that the last k intervals are the same for any cell in this collection. By splitting the cartesian product for a cell after the first $(d-k)$ (short) intervals and before the last interval, we can represent the cell as $a \times B \times [h_1, h_2]$ where a is a $(d-k)$-dimensional hypercuboid that depends on the selected cell and both the $(k-1)$-dimensional hypercuboid B and the values h_1 and h_2 are common to all cells of the collection. This set of hypercuboids $\{a\}$ forms a $(d-k)$-dimensional grid A that is the projection of our collection onto the first $(d-k)$ dimensions. With these definitions, our collection can be viewed as a d-dimensional strip with base $(\cup_{a\in A} a) \times B$ and height $(h_2 - h_1)$. We would like to repack this collection using a strip packing algorithm. For this purpose, we devise an algorithm to pack hypercubes on a base (which can be extended by a slight amount in the long dimensions) with the properties described above. With this motivation, we formally define the strip packing problem and state the main result in the next few paragraphs.

Let $k \in [d-1]$. Let $\varepsilon > 0$ be some accuracy parameter and $N \in \mathbb{N}_+, \alpha \geq 1$ be some constants depending on d, ε. The input consists of a set of items $\mathcal{I}$. In the entire subsection, we abbreviate $\hat{s}(\mathcal{I})$ by $\hat{s}$.

The $(d-1)$-dimensional base on which we need to pack the input set $\mathcal{I}$ is the cartesian product $A \times B$ where A is some $(d-k)$ dimensional grid of at most N cells and B is some $(k-1)$ dimensional hypercuboid. The side length of B in the i^{th} dimension ($i \in [k-1]$) is denoted by b_i. Each side length of each cell in A has to be at most $\alpha\hat{s}$ and each b_i has to be at least $N\alpha^{d-k}\hat{s}$. As it can be seen, compared to $\hat{s}$, all the edge lengths in A are short and all the edge lengths in B are long. Hence, we refer to the dimensions of the coordinate system parallel to the edges of A as *short dimensions* and to the dimensions parallel to the edges of B as *long dimensions*. In the 3-dimensional setting, we have two possible values for k. If $k = 1$, then B vanishes and our base is just a grid of rectangles A. Such a base can be seen in Figure 6a. If $k = 2$, then B is an interval and A is just a set of non-overlapping intervals because it is 1-dimensional. In this case, the base can be visualized like in Figure 6b as a set of flat but wide rectangles, that are positioned on top of each other. So for the 3-dimensional case, we either have a complex structure through A or long sides through B but never both. In higher dimensions however, we can have both. An example for this is the base of a 4-dimensional strip in Figure 6c. By scaling every dimension equally, it is assumed that the volume of the grid A is normalized: $\mathrm{VOL}_{d-k}(A) := \sum_{a\in A} \mathrm{VOL}_{d-k}(a) = 1$. We also assume that any item in $\mathcal{I}$ can be packed on the base $A \times B$. The goal is to find a non-overlapping packing of the items in $\mathcal{I}$ into the region given by the set product of

A, B and h, that is $\mathrm{CONT}(A, B, h) := (\bigcup_{a \in A} a) \times B \times [0, h]$ where h is called the height of the packing. The optimal (i.e. minimal) height h for which it is possible to pack $\mathcal{I}$ in $\mathrm{CONT}(A, B, h)$ is denoted by $\mathrm{OPT}_{\mathrm{strip}}(\mathcal{I}, A, B)$.



(a) $d = 3, k = 1$. **(b)** $d = 3, k = 2$. **(c)** $d = 4, k = 2$.

Figure 6 The first two figures show two possible bases for a 3-dimensional strip. In the first figure, A is shown in green and we have no B. In the second figure, the grid A (in green) is 1-dimensional and thus simple, and B is shown in red (the long side). The last figure shows a possible 3-dimensional base with $k = 2$ of a 4-dimensional strip with A shown in green (the short sides) and B shown in red (the long side).

We only consider instances where a solution exists. Since we assume that $\mathrm{VOL}_{d-k}(A) = 1$, we have that $\hat{s} \leq 1$ as a larger item would not fit into the strip. Note that for each $i \in [k-1]$, $b_i \geq N\alpha^{d-k}\hat{s} \geq N(\alpha\hat{s})^{d-k} \geq \sum_{a \in A} \mathrm{VOL}_{d-k}(a) = \mathrm{VOL}_{d-k}(A) = 1$. In this section, we describe an algorithm based on Harren's [21] multidimensional generalization of the algorithm by Kenyon and Rémila [28] for 2-dimensional strip packing.

The differences between our algorithm and Harren's algorithm[21] can be summed up as follows: Due to the complex nature of A, we have multiple hypercuboids, each having a bounded aspect ratio. This changes the analysis of the algorithm. It also changes the details like packing the medium and small items on the top, because we have to split one strip into multiple smaller strips (having hypercuboidal bases) and distribute the items among them. The addition of very long sides in form of B, like in Figure 6b, breaks the bounded aspect ratio property in a more crucial way. Harren's algorithm first packs the large items on the base and then extends this packing along the height of the strip. This relies on the fact that, due to bounded aspect ratio, not many large items can be packed on the base, and thus we have to consider only a constant number of so-called *configurations*. If we have additional long sides in the base in form of B, we don't have such a bound on the number of large items that can fit in the base. For example, in Figure 6b, an item can be large with respect to A (shown in green), but the number of such items that can fit on the base $A \times B$ can't be bounded by a constant since the side shown in red can be arbitrarily long.

We work around this problem as follows: First, we consider $(d-k)$ dimensional packings, i.e., *configurations* of the items, on A. Then we not only extend these configurations along the height of the strip but also along each of the $(k-1)$ dimensions of B. For example, in Figure 6b, we first create one-dimensional packings on A and then extend each of these packings both in the rightward direction and along the height of the strip.

▶ **Theorem 12.** *Let $\mathcal{I}$, A and B be the input for the d-dimensional strip packing problem defined above and let $\varepsilon > 0$ be some additional accuracy parameter. Then there is an algorithm which packs all items of $\mathcal{I}$ into the region $\mathrm{CONT}(A, B + \hat{s}, h)$ where $B + \hat{s}$ is the hypercuboid B after increasing each side length by $\hat{s}$ and the height h is given by*

$$h \leq (1 + \mathcal{O}(\varepsilon)) \, \mathrm{OPT}_{\mathrm{strip}}(\mathcal{I}, A, B) + \mathcal{O}(\hat{s})$$

The constant omitted in the expression $\mathcal{O}(\varepsilon)$ is a function of k. The constant omitted in the expression $\mathcal{O}(\hat{s})$ is a function of $d, k, \alpha, N, \varepsilon$.

Due to space limitations, the detailed proof of Theorem 12 is given in the full version [24]. Our algorithm first classifies the input items into large, medium, and small items. The side lengths of large items are rounded using linear grouping [12] such that there is only a constant number of different sizes. When we try to pack a subset of large items by arranging them in the first $(d-k)$ dimensions while giving them the same position in the last k dimensions, we only have to consider a constant number of those subsets. This is because the volume of the strip in the first $(d-k)$ dimensions is small and thus only a small subset of large items is packable this way and items of the same size can be considered to be identical. For each configuration, we take a packing that only uses the first $(d-k)$ dimensions, and extend it to use also the last k dimensions by placing multiple items of the same size next to each other in those long dimensions. This creates an $\mathcal{N}$-Box from each item in each configuration. An LP is used to determine how large a configuration is extended in those k dimensions. The packings for the different configurations are treated as layers stacked on top of each other.

After the large items are packed, we use the gaps between the large items to create $\mathcal{V}$-Boxes to hold some of the small items. The remaining small items and medium items are placed in additional $\mathcal{V}$-Boxes on top of this packing. Again this needed several technical adaptations, as unlike [21], we have multiple different bases and we need to distribute the remaining items in a balanced way such that the total height is minimized. See the full version [24] for the details.

As we mentioned earlier, we use this theorem to repack a k-elongated collection Q ($k \in [d-1]$). So, we use α_{k+1} as α since we know that for every cell in $\mathcal{Q}$, the first k dimensions have length at least $\alpha_k \hat{s}(\mathcal{Q}) \geq N\alpha_{k+1}^{d-k}\hat{s}(\mathcal{Q})$ (see Remark 15 and definition of α_k) and each of the last $(d-k)$ dimensions have length at most $\alpha_{k+1}\hat{s}(\mathcal{Q})$. Now let us note a few useful remarks about the above theorem applied to our collection Q. The proofs of these remarks follow from a few lemmas in the full version [24].

▶ Remark 13. A more exact bound for the height of the packing obtained in Theorem 12 is

$$\left(1+\left(2+\max\left(3,2^{k-1}\right)\right)\varepsilon\right)\mathrm{OPT}_{\mathrm{strip}}\left(Q,A,B\right)+2\left(C_{\mathrm{configs}}\left(d,k,\alpha_{k+1},N,\varepsilon\right)+3\right)\hat{s}\left(\mathcal{Q}\right)$$

Here C_{configs} is only dependent on its parameters.

▶ Remark 14. The algorithm of Theorem 12 packs all items in $\mathcal{N}$-Boxes and $\mathcal{V}$-Boxes. The number of $\mathcal{N}$-Boxes is bounded by $C_{\mathrm{configs}}\left(d,k,\alpha_{k+1},N,\varepsilon\right)/C_{\rho}\left(d,k,\alpha_{k+1},N,\varepsilon\right)^{d-k}$ and the number of $\mathcal{V}$-Boxes by $C_{\mathrm{configs}}\left(d,k,\alpha_{k+1},N,\varepsilon\right)N2^{d-k}/C_{\rho}\left(d,k,\alpha_{k+1},N,\varepsilon\right)^{(d-k)^2}+N$. Here C_{configs} and C_{ρ} are only dependent on their parameters.

▶ Remark 15. The value $C_{\mathrm{configs}}\left(d,k,\alpha_{k+1},N,\varepsilon\right)$ from Remarks 13 and 14 has a lower bound of $C_{\mathrm{configs}}\left(d,k,\alpha_{k+1},N,\varepsilon\right) > N\alpha_{k+1}^{d-k} \geq \alpha_{k+1}$.

3.1.3 Repacking a Collection

Depending on the type of the collection, we can simplify the packing of the items assigned to each collection. Let $\mathcal{C}$ be a k-elongated collection and let $\mathcal{Q}$ be the set of items assigned to $\mathcal{C}$. Let $\hat{s} := \hat{s}\left(\mathcal{Q}\right)$. We consider the cases $k = d, 1 \leq k < d$, and $k = 0$ separately.

First of all, if $\mathcal{C}$ is a d-elongated collection, then it consists of only a single cell as it has no bad facets and hence can not be merged with any other cell. Thus we can repack (after removing only a small profit subset) it efficiently using NFDH because it is a hypercuboidal region that is large in all dimensions. The following lemma states this. The proof can be found in the full version [24].

▶ **Lemma 16.** *If $\mathcal{C}$ is a d-elongated collection, then it can be transformed into a single $\mathcal{V}$-Box with a loss of profit at most $6d\varepsilon p\left(\mathcal{Q}\right)$.*

For a k-elongated collection with $1 \leq k < d$, we can apply the strip packing algorithm of Theorem 12 as shown in the next lemma.

▶ **Lemma 17.** *A k-elongated collection with $1 \leq k < d$ can be transformed into a constant number of $\mathcal{V}$-Boxes and $\mathcal{N}$-Boxes by losing profit at most $\left(k + 7 + \max\left(6, 2^k\right)\right) \varepsilon p\left(\mathcal{Q}\right)$ where $\mathcal{Q}$ is the set of items assigned to it.*

Proof. We begin by shrinking the collection to compensate for the enlargement that the strip packing algorithm will cause. Define one of the long dimensions of the collection to be its height and let h be the length in this dimension. Let $c := 4 + \max\left(3, 2^{k-1}\right)$. We assumed $\varepsilon < 1/2^{(d+2)} < 1/(2c)$. Divide the strip into $\lfloor 1/\left(c\varepsilon\right) \rfloor$ slices along the height. After that each slice has a height of at least

$$\begin{aligned}
c\varepsilon h &= \left(4 + \max\left(3, 2^{k-1}\right)\right) \varepsilon h \\
&\geq \left(2 + \max\left(3, 2^{k-1}\right)\right) \varepsilon h + 2\varepsilon\alpha_k \hat{s} && \text{(since } h \geq \alpha_k \hat{s}\text{)} \\
&= \left(2 + \max\left(3, 2^{k-1}\right)\right) \varepsilon h + 4\left(C_{\text{configs}}\left(d, k, \alpha_{k+1}, N, \varepsilon\right) + 3\right) \hat{s} && \text{(by definition of } \alpha_k\text{)} \\
&\geq \left(2 + \max\left(3, 2^{k-1}\right)\right) \varepsilon h + 2\left(C_{\text{configs}}\left(d, k, \alpha_{k+1}, N, \varepsilon\right) + 3\right) \hat{s} + 6\hat{s}.
\end{aligned}$$

By clearing the slice with the lowest profit of items completely contained in it, we leave a gap with a height of $\left(2 + \max\left(3, 2^{k-1}\right)\right) \varepsilon h + 2\left(C_{\text{configs}}\left(d, k, \alpha_{k+1}, N, \varepsilon\right) + 3\right) \hat{s} + 4\hat{s}$. This causes a loss of profit at most $\frac{1}{\lfloor 1/(c\varepsilon) \rfloor} p\left(\mathcal{Q}\right) \leq \frac{1}{1/(c\varepsilon) - 1} p\left(\mathcal{Q}\right) \leq \frac{1}{1/(c\varepsilon) - 1/(2c\varepsilon)} p\left(\mathcal{Q}\right) = 2c\varepsilon p\left(\mathcal{Q}\right)$. The items intersecting the bottom or top facet of the collection fit in a gap of height $2\hat{s}$. We reduce the height of the collection to

$$h' := h - \left(2 + \max\left(3, 2^{k-1}\right)\right) \varepsilon h - 2\left(C_{\text{configs}}\left(d, k, \alpha_{k+1}, N, \varepsilon\right) + 3\right) \hat{s}$$

by shifting the items intersecting the (removed) region at the top into a gap of height $\left(2 + \max\left(3, 2^{k-1}\right)\right) \varepsilon h + 2\left(C_{\text{configs}}\left(d, k, \alpha_{k+1}, N, \varepsilon\right) + 3\right) \hat{s} + \hat{s}$.

For each other large dimension we create $\lceil 1/\varepsilon \rceil$ slices and clear the slice with the lowest profit, losing at most $\varepsilon p\left(\mathcal{Q}\right)$ and create a gap of width at least $\frac{1}{\lceil 1/\varepsilon \rceil} \alpha_k \hat{s} - 2\hat{s} \geq \frac{1}{1/\varepsilon + 1} \alpha_k \hat{s} - 2\hat{s} \geq \frac{\varepsilon}{2} \alpha_k \hat{s} - 2\hat{s} \geq 4\hat{s}$. Now we can shift the items intersecting the orthogonal facets into that gap and reduce the length of the collection in this direction by $\hat{s}$. After this shifting process, we will be left with a subset of items $\mathcal{Q}' \subseteq \mathcal{Q}$.

To be able to use the strip packing algorithm from Theorem 12, the base of our strip has to be in a special representation. For this, we create a $(d-k)$-dimensional grid A by projecting the cells of our collection to their $(d-k)$ short dimensions. We define B as a tuple holding the lengths in the $(k-1)$ long dimensions of the collection that are not the height. Now our base can be represented as the cartesian product $A \times B$. As a last preparation step, we scale the whole collection and the items in it by $f := 1/\operatorname{VOL}_{d-k}\left(A\right)^{1/(d-k)}$ in each dimension to normalize the volume of A to 1. Note that each value in the tuple B is at least $\alpha_k \hat{s} f \geq N \alpha_{k+1}^{d-k} \hat{s} f$ by the definition of α_k and Remark 15. Thus, we can use the strip packing algorithm Theorem 12 using the bound on the height by Remark 13 to compute a packing of height at most

$$\begin{aligned}
&\left(1 + \left(2 + \max\left(3, 2^{k-1}\right)\right) \varepsilon\right) \operatorname{OPT}_{\text{strip}}\left(Q', A, B\right) + 2\left(C_{\text{configs}}\left(d, k, \alpha_{k+1}, N, \varepsilon\right) + 3\right) \hat{s} f \\
&\leq \left(1 + \left(2 + \max\left(3, 2^{k-1}\right)\right) \varepsilon\right) h' f + 2\left(C_{\text{configs}}\left(d, k, \alpha_{k+1}, N, \varepsilon\right) + 3\right) \hat{s} f \\
&\leq h' f + \left(2 + \max\left(3, 2^{k-1}\right)\right) \varepsilon h f + 2\left(C_{\text{configs}}\left(d, k, \alpha_{k+1}, N, \varepsilon\right) + 3\right) \hat{s} f \\
&= h f.
\end{aligned}$$

After scaling back, the packing fits into our collection. ◀

Repacking 0-elongated Collections. For a 0-elongated collection, we first distinguish the items into large, medium, and small items such that the profit of medium items is very small so that they can be discarded. We then further distinguish between the cases when (i) there is a large item of very small profit or (ii) every large item has a significant profit. In the first case, we remove the large item to make enough space to repack the small items. In the second case, we use the fact that the number of large items can only be $O(1)$. So, we partition the grid into smaller grids and solve these recursively. We finally prove that we only require $O(1)$ number of recursive steps that we require are only a constant in number.

From now on, lets assume that $\mathcal{C}$ is a 0-elongated collection. To partition $\mathcal{Q}$ into large, medium, and small items, we define the following values. $\rho'_0 := 1, \rho'_{i+1} := \left(\frac{(\rho'_i)^d}{4dN\alpha_1^d}\right)^{d+1}$ and $\rho_i := \rho'_i \hat{s}$ for all $i \geq 0$. Let $\eta \in \mathbb{N}_+$ be minimal such that $p(\{ x \in \mathcal{Q} \mid \rho_\eta < s(x) < \rho_{\eta-1} \}) \leq \varepsilon p(\mathcal{Q})$. Note that $\eta \leq 1/\varepsilon$. We now partition $\mathcal{Q}$ into sets $\mathcal{S} := \{ x \in \mathcal{Q} \mid s(x) \leq \rho_\eta \}$, $\mathcal{M} := \{ x \in \mathcal{Q} \mid \rho_\eta < s(x) < \rho_{\eta-1} \}$ and $\mathcal{L} := \{ x \in \mathcal{Q} \mid \rho_{\eta-1} \leq s(x) \}$. We call $\mathcal{S}$ (resp. $\mathcal{M}, \mathcal{L}$) to be the set of small (resp. medium, large) items. Note that since $\eta \geq 1$ and $\rho_0 = \hat{s}$, the set of large items can't be empty. This simple observation will be useful later.

If there is a large item with a small profit, then after discarding that large item, the entire empty space can be divided into a constant number of $\mathcal{V}$-Boxes. It can then be shown that these $\mathcal{V}$-Boxes have enough volume to pack all the small items. This is formalized in the following lemma. The proof can be found in the full version [24].

▶ **Lemma 18.** *Suppose $\mathcal{C}$ is a 0-elongated collection. If an item in $\mathcal{L}$ has profit at most $\varepsilon p(\mathcal{Q})$, then the collection $\mathcal{C}$ can be transformed into a packing of profit at least $(1-2\varepsilon)\, p(\mathcal{Q})$ containing a constant number of large items and a constant number of $\mathcal{V}$-Boxes.*

If all the large items have a good profit (i.e., a profit of more than $\varepsilon p(\mathcal{Q})$), we cannot discard any of them to create space. Therefore, it is hard to repack the items directly so instead we will take a recursive approach using a shifting argumentation.

▶ **Lemma 19.** *Suppose $\mathcal{C}$ is a 0-elongated collection. If every item in $\mathcal{L}$ has a profit of at least $\varepsilon p(\mathcal{Q})$, then $\mathcal{C}$ can be transformed into a constant number of large items, $\mathcal{N}$-Boxes and $\mathcal{V}$-Boxes while losing profit at most $\max\left(6d+1, d+13, d+7+2^{d-1}\right) \varepsilon p(\mathcal{Q})$.*

Proof. In this case, we have at most $1/\varepsilon$ large items. We can split the area of the collection, which is not covered by the large items into a grid of at most $2/\varepsilon$ additional layers in each dimension and recursively start from Section 3.1.1 with the classification of cells and merging into collections. Each of these collections is a k-elongated collection ($k > 0$) or a 0-elongated collection. The former collections can be repacked using Lemma 16 or Lemma 17. Let's look at the 0-elongated collections. Note that the total profit of these collections is at most $(1-\varepsilon)p(\mathcal{Q})$ since we recurse only if there is a large item in $\mathcal{Q}$ of profit at least $\varepsilon p(\mathcal{Q})$. For a 0-elongated collection in this recursive step, we either use Lemma 18 if there exists a large item of small profit or, we continue the recursion. Note, that if we reach a depth of $\lceil \log_{1-\varepsilon}(\varepsilon) \rceil$ in this recursion, then we can just discard the remaining items and stop. The reason for this is that we packed in each recursive step at least one large item with a non-negligible profit, so at this maximal recursion depth, all the items over all collections have a profit at most $(1-\varepsilon)^{\lceil \log_{1-\varepsilon}(\varepsilon) \rceil}\, p(\mathcal{Q}) \leq \varepsilon p(\mathcal{Q})$.

During this process, the initial region was split into different collections, where each of those was assigned a partition of the items $\mathcal{Q}' \subseteq \mathcal{Q}$. At this point, no profit was lost except for the items that are lost at the maximal recursion depth with profit at most $\varepsilon p(\mathcal{Q})$. When repacking each collection by transforming them into large items, $\mathcal{N}$-Boxes and $\mathcal{V}$-Boxes, we

lose again some profit, which depends on the method we used for repacking. We lose at most $6d\varepsilon p(\mathcal{Q}')$ through Lemma 16, at most $\left(d+6+\max\left(6,2^{d-1}\right)\right)\varepsilon p(\mathcal{Q}')$ through Lemma 17, at most $2\varepsilon p(\mathcal{Q}')$ through Lemma 18. As those subsets for the collections are distinct the loss in this repacking step is bounded by $\max\left(6d, d+12, d+6+2^{d-1}, 2\right)\varepsilon p(\mathcal{Q})$. Noting that $2 < \max\left(6d, d+12, d+6+2^{d-1}\right)$ and adding the factor of $\varepsilon p(\mathcal{Q})$ that we may lose during the recursion proves the claim. ◀

The last thing to do is to prove, that the number of $\mathcal{N}$-Boxes and $\mathcal{V}$-Boxes that are created is constant. As those boxes are created out of the cells of the grid in Section 3.1.1, we start by bounding the size of this grid during the recursive algorithm. The following lemma follows from the fact that in each of the recursive step (having depth at most $\lceil \log_{1-\varepsilon} \varepsilon \rceil$) the number of layers increases by at most $2/\varepsilon$ in each of the dimensions. The proof can be found in the full version [24].

▶ **Lemma 20.** *Consider any grid $\mathcal{G}$ obtained during the recursive algorithm starting from the knapsack with an optimal packing as a grid with a single cell. Then the number of layers in each dimension in $\mathcal{G}$ is upper bounded by $C_{\text{layer}}(\varepsilon) := 2\left\lceil \log_{1-\varepsilon}(\varepsilon)\right\rceil/\varepsilon+1$.*

▶ Remark 21. Due to the above lemma, the bound on the number of layers also gives an upper bound $C_{\text{N}}(d,\varepsilon) := (C_{\text{layer}}(\varepsilon))^d$ on the number of cells in any grid that we consider in Section 3.1.1.

Using these results, we obtain the following lemma which bounds the number of $\mathcal{N}$-Boxes and $\mathcal{V}$-Boxes. The proof can be found in the full version [24].

▶ **Lemma 22.** *The numbers of large items, and the total number of $\mathcal{N}$-Boxes and $\mathcal{V}$-Boxes generated by the transformation of an optimal packing can be bounded by C_{large} and C_{boxes}, respectively. C_{large} and C_{boxes} are constants that depend only on d, ε.*

Now we can prove Theorem 7.

Proof of Theorem 7. Consider an optimal packing and interpret the initial knapsack as a cell of a grid with one layer in each dimension. Then we start with the procedure explained in Section 3.1.1 to classify the unit hypercube either as d-elongated or 0-elongated. By Lemmas 16, 18, and 19, we can simplify the structure of the packing and lose a profit of at most $\max(6d+1, d+13, d+8+2^{d-1})\varepsilon p(\mathcal{I}) \leq 2^{d+2}\varepsilon p(\mathcal{I})$ as we have $d \geq 2$. Lemma 22 bounds the number of $\mathcal{V}$-Boxes and $\mathcal{N}$-Boxes and the large items packed outside them. ◀

3.2 Algorithm

Using the results of Section 3.1, we can construct a PTAS for d-D Hc-Knapsack.

▶ **Theorem 23.** *Let $d \geq 2$ and $\varepsilon > 0$. There is an algorithm which returns for each instance of the d-dimensional hypercube knapsack packing problem given by a set of items a packing with profit at least $(1-\varepsilon)\,\text{OPT}_{\text{knapsack}}(\mathcal{I})$ with a running time which is polynomial in $|\mathcal{I}|$.*

Proof. Using Theorem 7 with an accuracy of $2^{-d-3}\varepsilon$ we know that there is a packing with a simple structure and profit at least $(1-\varepsilon/2)\,\text{OPT}_{\text{knapsack}}(\mathcal{I})$. Let $\mathcal{B}$ be the set of $\mathcal{N}$-Boxes and $\mathcal{V}$-Boxes in this structure, $\mathcal{L} \subseteq \mathcal{I}$ the set of items packed outside of those boxes and $\mathcal{S} \subseteq \mathcal{I}\backslash\mathcal{L}$ the set of items packed inside of them.

As the number of items in $\mathcal{L}$ is constant, there are at most $|\mathcal{I}|^{C_{\text{large}}(d,\varepsilon)}$ choices for this subset $\mathcal{L} \subseteq \mathcal{I}$. There are at most $C_{\text{boxes}}(d,\varepsilon)+1$ choices for the number of boxes in $\mathcal{B}$ and for each box there are at most:

- 2 choices whether it is a $\mathcal{V}$-Box or an $\mathcal{N}$-Box,
- $|\mathcal{I}|$ choices for the size parameter $\hat{s}$ of the box,
- $|\mathcal{I}|^d$ choices for the side lengths of the box.

All of these choices are polynomial in the number of items $|\mathcal{I}|$. Thus, by iterating over all possible choices, we can assume at this point that we know the set of items $\mathcal{L}$ and the set $\mathcal{B}$ of $\mathcal{V}$-Boxes and $\mathcal{N}$-Boxes of that nearly optimal solution. As both $|\mathcal{L}|$ and $|\mathcal{B}|$ are bounded by a constant, we can find a packing of those items and boxes in the unit hypercube in constant time. The last step left to do is to pack items from $\mathcal{I}\backslash\mathcal{L}$ in the boxes with a nearly optimal profit. We can do this by solving a special variant of the Generalized Assignment Problem (GAP) [42]. In GAP, we are given a set of one-dimensional knapsacks with a capacity each and a set of items that may have a possibly different size and profit for each knapsack. The goal in this problem is to find a feasible packing with maximal profit. We will consider our $O(1)$ number of $\mathcal{V}$-Boxes and $\mathcal{N}$-Boxes to be knapsacks. Consider a $\mathcal{V}$-Box B_V with size parameter $\hat{s}$. We set its capacity to be $\mathrm{VOL}_d(B) - \hat{s}(\mathrm{SURF}_d(B)/2)$. For any item i, we set its size with respect to B_V as $\mathrm{VOL}_d(i)$ if $s(i) \leq \hat{s}$ and ∞ otherwise. On the other hand, consider an $\mathcal{N}$-Box B_N with size parameter $\hat{s}$ and $\{n_i\}_{i\in d}$ denoting the number of cells in each dimension. We set its capacity to be $n_1 n_2 \dots n_d$. For any item i, we set its size with respect to B_N as 1 if $s(i) \leq \hat{s}$ and ∞ otherwise. The profit of any item i with respect to any box is just set as $p(i)$. This boils down to a variant of GAP with $O(1)$ number of knapsacks, which admits a PTAS [18]. Thus by solving this instance, we get a subset $\mathcal{S}' \subseteq \mathcal{I}\backslash\mathcal{L}$ with profit at least $p(\mathcal{S}') \geq (1-\varepsilon/2)\,p(\mathcal{S})$. Using Observation 4 and Corollary 3 we can ensure to pack those items inside the boxes. The total profit packed is $p(\mathcal{S}') + p(\mathcal{L}) \geq \left(1-\frac{\varepsilon}{2}\right)p(\mathcal{S}) + p(\mathcal{L}) > \left(1-\frac{\varepsilon}{2}\right)(p(\mathcal{S}) + p(\mathcal{L})) \geq \left(1-\frac{\varepsilon}{2}\right)^2 \mathrm{OPT}_{\text{knapsack}}(\mathcal{I}) \geq (1-\varepsilon)\,\mathrm{OPT}_{\text{knapsack}}(\mathcal{I})$. ◀

4 Conclusion

We have designed a PTAS for a variant of the knapsack problem (d-D Hc-Knapsack), where the items are d-dimensional hypercubes with arbitrary profits. En route, we have also developed a near-optimal algorithm for a variant of the d-dimensional hypercube strip packing problem, where the base need not be hypercuboidal, but we are allowed to extend some of the sides of the base (the long dimensions) by a small amount. Extending the techniques of [23], we believe that it might be possible to design an EPTAS for d-D Hc-Knapsack.

The knapsack problem for squares, cubes, and hypercubes is thus almost settled. Whether there exists a PTAS for the knapsack variant where items are rectangles is still open. It is also interesting to improve the current best approximation ratios ($(5+\varepsilon)$ with rotations and $(7+\varepsilon)$ without rotations [14]) for the knapsack problem where items are cuboids (3D) because of its practical relevance. Another interesting task of theoretical importance would be to improve the current best approximation ratio (which is $(3^d+\varepsilon)$ due to [41]) for the knapsack problem where items are d-dimensional hypercuboids.

References

1 Fidaa Abed, Parinya Chalermsook, José R. Correa, Andreas Karrenbauer, Pablo Pérez-Lantero, José A. Soto, and Andreas Wiese. On guillotine cutting sequences. In *APPROX*, pages 1–19, 2015.

2 Anna Adamaszek, Sariel Har-Peled, and Andreas Wiese. Approximation schemes for independent set and sparse subsets of polygons. *Journal of the ACM*, 66(4):29:1–29:40, 2019.

3 Anna Adamaszek and Andreas Wiese. A quasi-PTAS for the two-dimensional geometric knapsack problem. In *SODA*, pages 1491–1505, 2015.

4 Nikhil Bansal, Alberto Caprara, Klaus Jansen, Lars Prädel, and Maxim Sviridenko. A structural lemma in 2-dimensional packing, and its implications on approximability. In *ISAAC*, pages 77–86, 2009.

5 Nikhil Bansal, José R Correa, Claire Kenyon, and Maxim Sviridenko. Bin packing in multiple dimensions: inapproximability results and approximation schemes. *Mathematics of operations research*, 31(1):31–49, 2006.

6 Nikhil Bansal, Marek Eliáš, and Arindam Khan. Improved approximation for vector bin packing. In *SODA*, pages 1561–1579. SIAM, 2016.

7 Nikhil Bansal and Arindam Khan. Improved approximation algorithm for two-dimensional bin packing. In *SODA*, pages 13–25, 2014.

8 Nikhil Bansal, Andrea Lodi, and Maxim Sviridenko. A tale of two dimensional bin packing. In *FOCS*, pages 657–666, 2005.

9 Alberto Caprara. Packing d-dimensional bins in d stages. *Mathematics of Operations Research*, 33(1):203–215, 2008.

10 Henrik I. Christensen, Arindam Khan, Sebastian Pokutta, and Prasad Tetali. Approximation and online algorithms for multidimensional bin packing: A survey. *Computer Science Review*, 24:63–79, 2017.

11 Edward G Coffman, Jr, Michael R Garey, David S Johnson, and Robert Endre Tarjan. Performance bounds for level-oriented two-dimensional packing algorithms. *SIAM Journal on Computing*, 9(4):808–826, 1980.

12 W. Fernandez de la Vega and G. S. Lueker. Bin packing can be solved within $1 + \varepsilon$ in linear time. *Combinatorica*, 1:349–355, 1981.

13 Max A Deppert, Klaus Jansen, Arindam Khan, Malin Rau, and Malte Tutas. Peak demand minimization via sliced strip packing. In *APPROX/RANDOM*, volume 207, pages 21:1–21:24, 2021.

14 Florian Diedrich, Rolf Harren, Klaus Jansen, Ralf Thöle, and Henning Thomas. Approximation algorithms for 3d orthogonal knapsack. *Journal of Computer Science and Technology*, 23(5):749, 2008.

15 Alan M Frieze, Michael RB Clarke, et al. Approximation algorithms for the m-dimensional 0-1 knapsack problem: worst-case and probabilistic analyses. *European Journal of Operational Research*, 15(1):100–109, 1984.

16 Waldo Gálvez, Fabrizio Grandoni, Afrouz Jabal Ameli, Klaus Jansen, Arindam Khan, and Malin Rau. A tight $(3/2+\varepsilon)$ approximation for skewed strip packing. In *APPROX/RANDOM*, volume 176, pages 44:1–44:18, 2020.

17 Waldo Gálvez, Fabrizio Grandoni, Afrouz Jabal Ameli, and Kamyar Khodamoradi. Approximation algorithms for demand strip packing. In Mary Wootters and Laura Sanità, editors, *APPROX/RANDOM*, volume 207 of *LIPIcs*, pages 20:1–20:24, 2021.

18 Waldo Gálvez, Fabrizio Grandoni, Sandy Heydrich, Salvatore Ingala, Arindam Khan, and Andreas Wiese. Approximating geometric knapsack via l-packings. In *FOCS*, pages 260–271, 2017.

19 Waldo Gálvez, Fabrizio Grandoni, Arindam Khan, Diego Ramírez-Romero, and Andreas Wiese. Improved approximation algorithms for 2-dimensional knapsack: Packing into multiple l-shapes, spirals, and more. In *SoCG*, volume 189 of *LIPIcs*, pages 39:1–39:17, 2021.

20 Waldo Gálvez, Arindam Khan, Mathieu Mari, Tobias Mömke, Madhusudhan Reddy Pittu, and Andreas Wiese. A $(2+\varepsilon)$-approximation algorithm for maximum independent set of rectangles. *CoRR*, abs/2106.00623, 2021. `arXiv:2106.00623`.

21 Rolf Harren. Approximation algorithms for orthogonal packing problems for hypercubes. *Theoretical Computer Science*, 410(44):4504–4532, 2009.

22 Rolf Harren, Klaus Jansen, Lars Prädel, and Rob Van Stee. A $(5/3+\varepsilon)$-approximation for strip packing. *Computational Geometry*, 47(2):248–267, 2014.

23 Sandy Heydrich and Andreas Wiese. Faster approximation schemes for the two-dimensional knapsack problem. In *SODA*, pages 79–98, 2017.

24 Klaus Jansen, Arindam Khan, Marvin Lira, and K. V. N. Sreenivas. A ptas for packing hypercubes into a knapsack, 2022. doi:10.48550/ARXIV.2202.11902.

25 Klaus Jansen and Malin Rau. Closing the gap for pseudo-polynomial strip packing. In *ESA*, volume 144 of *LIPIcs*, pages 62:1–62:14, 2019.

26 Klaus Jansen and Roberto Solis-Oba. Packing squares with profits. *SIAM Journal on Discrete Mathematics*, 26:263–279, January 2012. doi:10.1137/080717110.

27 Klaus Jansen and Guochuan Zhang. Maximizing the total profit of rectangles packed into a rectangle. *Algorithmica*, 47(3):323–342, 2007.

28 Claire Kenyon and Eric Rémila. A near-optimal solution to a two-dimensional cutting stock problem. *Mathematics of Operations Research*, 25(4):645–656, 2000.

29 Arindam Khan. *Approximation algorithms for multidimensional bin packing*. PhD thesis, Georgia Institute of Technology, 2015.

30 Arindam Khan, Arnab Maiti, Amatya Sharma, and Andreas Wiese. On guillotine separable packings for the two-dimensional geometric knapsack problem. In *SoCG*, volume 189, pages 48:1–48:17, 2021.

31 Arindam Khan and Madhusudhan Reddy Pittu. On guillotine separability of squares and rectangles. In *APPROX/RANDOM*, pages 47:1–47:22, 2020.

32 Arindam Khan and Eklavya Sharma. Tight Approximation Algorithms For Geometric Bin Packing with Skewed Items. In *APPROX/RANDOM*, pages 22:1–22:23, 2021.

33 Arindam Khan, Eklavya Sharma, and K. V. N. Sreenivas. Approximation algorithms for generalized multidimensional knapsack. *CoRR*, abs/2102.05854, 2021.

34 Arindam Khan, Eklavya Sharma, and K. V. N. Sreenivas. Geometry meets vectors: Approximation algorithms for multidimensional packing. *CoRR*, abs/2106.13951, 2021. arXiv:2106.13951.

35 E. L. Lawler. Fast approximation algorithms for knapsack problems. *Mathematics of Operations Research*, 4(4):339–356, 1979.

36 Joseph Y. T. Leung, Tommy W. Tam, C. S. Wong, Gilbert H. Young, and Francis Y. L. Chin. Packing squares into a square. *Journal of Parallel and Distributed Computing*, 10(3):271–275, 1990.

37 Yiping Lu, Danny Z Chen, and Jianzhong Cha. Packing cubes into a cube in ($d > 3$)-dimensions. In *COCOON*, pages 264–276. Springer, 2015.

38 Yiping Lu, Danny Z Chen, and Jianzhong Cha. Packing cubes into a cube is np-complete in the strong sense. *Journal of Combinatorial Optimization*, 29(1):197–215, 2015.

39 Deval Patel, Arindam Khan, and Anand Louis. Group fairness for knapsack problems. In *AAMAS*, pages 1001–1009, 2021.

40 Sai Sandeep. Almost optimal inapproximability of multidimensional packing problems. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 245–256, 2022. doi:10.1109/FOCS52979.2021.00033.

41 Eklavya Sharma. Harmonic Algorithms for Packing d-Dimensional Cuboids into Bins. In *FSTTCS*, pages 32:1–32:22, 2021.

42 David B Shmoys and Éva Tardos. An approximation algorithm for the generalized assignment problem. *Mathematical programming*, 62(1):461–474, 1993.