# Harmonic Algorithms for Packing $d$-Dimensional Cuboids into Bins

**Eklavya Sharma**
Department of Computer Science and Automation, Indian Institute of Science, Bengaluru, India

## Abstract

We explore approximation algorithms for the $d$-dimensional geometric bin packing problem ($d$BP). Caprara [8] gave a *harmonic-based* algorithm for $d$BP having an asymptotic approximation ratio (AAR) of $T_\infty^{d-1}$ (where $T_\infty \approx 1.691$). However, their algorithm doesn't allow items to be rotated. This is in contrast to some common applications of $d$BP, like packing boxes into shipping containers. We give approximation algorithms for $d$BP when items can be orthogonally rotated about all or a subset of axes. We first give a fast and simple harmonic-based algorithm having AAR $T_\infty^d$. We next give a more sophisticated harmonic-based algorithm, which we call $\texttt{HGaP}_k$, having AAR $T_\infty^{d-1}(1+\varepsilon)$. This gives an AAR of roughly $2.860+\varepsilon$ for 3BP with rotations, which improves upon the best-known AAR of 4.5. In addition, we study the *multiple-choice* bin packing problem that generalizes the rotational case. Here we are given $n$ sets of $d$-dimensional cuboidal items and we have to choose exactly one item from each set and then pack the chosen items. Our algorithms also work for the multiple-choice bin packing problem. We also give fast and simple approximation algorithms for the multiple-choice versions of $d$D strip packing and $d$D geometric knapsack.

**2012 ACM Subject Classification** Theory of computation → Packing and covering problems

**Keywords and phrases** Geometric bin packing

**Digital Object Identifier** 10.4230/LIPIcs.FSTTCS.2021.32

**Related Version** *ArXiv*: `https://arxiv.org/abs/2011.10963`

**Acknowledgements** I want to thank my advisor, Prof. Arindam Khan, for his valuable comments, and Arka Ray for helpful suggestions.

## 1 Introduction

Packing of rectangular and cuboidal items is a fundamental problem in computer science, mathematics, and operations research. Packing problems find numerous applications in practice, e.g., packing of concrete 3D items during storage or transportation [7], cutting prescribed 2D pieces from cloth or metal sheet while minimizing the waste [16], etc. In this paper, we study packing of $d$-dimensional ($d$D) cuboidal items (for $d \geq 2$).

Let $I$ be a set of $n$ $d$D cuboidal items, where each item has length at most one in each dimension. A feasible packing of items into a $d$D cuboid is a packing where items are placed inside the cuboid parallel to the axes without any overlapping. A $d$D *unit cube* is a $d$D cuboid of length one in each dimension. In the $d$D bin packing problem ($d$BP), we have to compute a feasible packing of $I$ (without rotating the items) into the minimum number of bins that are $d$D unit cubes. Let $\mathrm{opt}_{d\mathrm{BP}}(I)$ be the minimum number of bins needed to pack $I$.

$d$BP is NP-hard, as it generalizes the classic bin packing problem [9]. Thus, we study approximation algorithms. For $d$BP, the worst-case approximation ratio usually occurs only for *small* pathological instances. Thus, the standard performance measure is the asymptotic approximation ratio (AAR). For an algorithm $\mathcal{A}$, AAR is defined as:

$$\lim_{m\to\infty} \sup_{I\in\mathtt{I}:\,\mathrm{opt}(I)=m} \frac{\mathcal{A}(I)}{\mathrm{opt}(I)},$$

where $\mathtt{I}$ is the set of all problem instances. $\mathcal{A}(I)$ and $\mathrm{opt}(I)$ are the number of bins used by $\mathcal{A}$ and the optimal algorithm, respectively, on $I$.


Coffman et al. [10] initiated the study of approximation algorithms for rectangle packing. They studied algorithms such as First-Fit Decreasing Height (FFDH) and Next-Fit Decreasing Height (NFDH). In his seminal paper, Caprara [8] devised a polynomial-time algorithm for $d$BP called $\texttt{HDH}_k$ (Harmonic Decreasing Height), where $k \in \mathbb{Z}$ is a parameter to the algorithm. $\texttt{HDH}_k$ has AAR equal to $T_k^{d-1}$, where $T_k$ is a decreasing function of $k$ and $T_\infty := \lim_{k\to\infty} T_k \approx 1.691$. $\texttt{HDH}_k$ is based on the harmonic algorithm [24] for 1BP.

A limitation of $\texttt{HDH}_k$ is that it does not allow rotation of items. This is in contrast to some real-world problems, like packing boxes into shipping containers ($d = 3$), where items can often be rotated orthogonally, i.e., 90° rotation around all or a subset of axes [1, 31]. Orientation constraints may sometimes limit the vertical orientation of a box to one dimension ("This side up") or to two (of three) dimensions (e.g., long but low and narrow box should not be placed on its smallest surface). These constraints are introduced to deter goods and packaging from being damaged and to ensure the stability of the load. One of our primary contributions is presenting variants of $\texttt{HDH}_k$ that work for generalizations of $d$BP that capture the notion of orthogonal rotation of items.

## 1.1 Prior Work

For 2BP, Bansal et al. [3] obtained AAR of $T_\infty + \varepsilon$ even for the case with rotations, using a more sophisticated algorithm that used properties of harmonic rounding. Then there has been a series of improvements [3, 19] culminating with the present best AAR of 1.406 [5], for both the cases with and without orthogonal rotations. Bansal et al. [6] showed that $d$BP is APX-hard for $d \geq 2$, and gave an asymptotic PTAS for $d$BP when all items are $d$D squares.

Closely related to $d$BP is the $d$D strip packing problem ($d$SP), where we have to pack $I$ (without rotating the items) into a $d$D cuboid (called a strip) that has length one in the first $d-1$ dimensions and the minimum possible length (called height) in the $d^{\text{th}}$ dimension.

For 2SP, an asymptotic PTAS was given by Kenyon and Rémila [22]. Jansen and van Stee [21] extended this to the case with orthogonal rotations. For 3SP, when rotations are not allowed, Bansal et al. [4] gave a harmonic-based algorithm achieving AAR of $T_\infty + \varepsilon$. Recently, this has been improved to $1.5 + \varepsilon$ [20]. Miyazawa and Wakabayashi [25] studied 3SP and 3BP when rotations are allowed, and gave algorithms with AAR 2.64 and 4.89, respectively. Epstein and van Stee [13] gave an improved AAR of 2.25 and 4.5 for 3SP and 3BP with rotations, respectively. The $\texttt{HDH}_k$ algorithm also works for $d$SP and has an AAR of $T_k^{d-1}$. For online $d$BP, there are harmonic-based $T_\infty^d$-asymptotic-approximation algorithms [12, 11], which are optimal for $O(1)$ memory algorithms.

## 1.2 Multiple-Choice Packing

We will now define the $d$D multiple-choice bin packing problem ($d$MCBP). This generalizes $d$BP and captures the notion of orthogonal rotation of items. This perspective will be helpful in designing algorithms for the rotational case. In $d$MCBP, we're given a set $\mathcal{I} = \{I_1, I_2, \ldots, I_n\}$, where for each $j$, $I_j$ is a set of items, henceforth called an *itemset*. We have to pick exactly one item from each itemset and pack those items into the minimum number of bins. See Figure 1 for an example of 2MCBP.

We can model rotations using multiple-choice packing: Given a set $I$ of items, for each item $i \in I$, create an itemset $I_i$ that contains all allowed orientations of $i$. Then the optimal solution to $\mathcal{I} := \{I_i : i \in I\}$ will tell us how to rotate and pack items in $I$.

Some algorithms for 2D bin packing with rotations assume that the bin is square [3, 19, 5]. This assumption holds without loss of generality when rotations are forbidden, because we can scale the items. But if rotations are allowed, this won't work because items $i_1$ and $i_2$

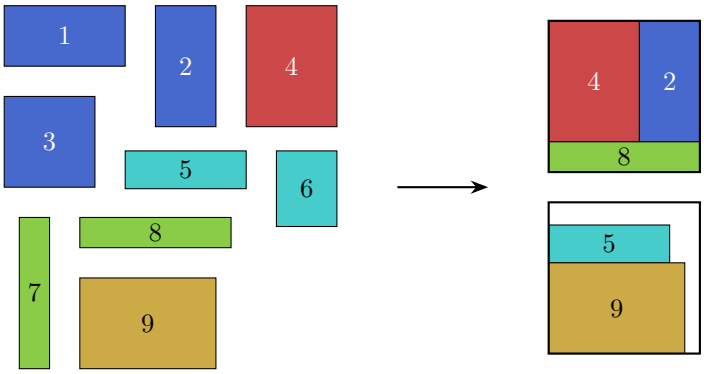


**Figure 1** 2MCBP example: packing the input $\mathcal{I} = \{\{1,2,3\},\{4\},\{5,6\},\{7,8\},\{9\}\}$ into two bins. Here items of the same color belong to the same itemset.

that are rotations of each other may stop being rotations of each other after they are scaled. Multiple-choice packing algorithms can be used in this case. For each item $i \in I$, we will create an itemset $I_i$ that contains scaled orientations of $i$.

Multiple-choice packing problems have been studied before. Lawler gave an FPTAS for the multiple-choice knapsack problem [23]. Patt-Shamir and Rawitz gave an algorithm for multiple-choice vector bin packing having AAR $O(\log d)$ and a PTAS for multiple-choice vector knapsack [27]. Similar notions have been studied in the scheduling of malleable or moldable jobs [32, 18].

## 1.3 Our Contributions

After the introduction of the harmonic algorithm for online 1BP by Lee and Lee [24], many variants have found widespread use in multidimensional packing problems (both offline and online) [8, 3, 4, 2, 12, 11, 17, 28, 29]. They are also simple, fast, and easy to implement. For example, among algorithms for 3SP, 2BP and 3BP with practical running time, harmonic-based algorithms provide the best AAR.

In our work, we extend harmonic-based algorithms to $d$MCBP. $d$MCBP subsumes the rotational case for geometric bin packing, and we believe $d$MCBP is an important natural generalization of geometric bin packing that may be of independent interest.

In Section 3, we describe ideas from $\texttt{HDH}_k$ [8] that help us devise harmonic-based algorithms for $d$MCBP. In Section 4, we show an $O(Nd + nd \log n)$-time algorithm for $d$MCBP, called $\texttt{fullh}_k$, having an AAR of $T_k^d$, where $n$ is the number of itemsets and $N$ is the total number of items across all the $n$ itemsets. $\texttt{fullh}_k$ is a fast and simple algorithm that works in two stages: In the first stage, we select the *smallest* item from each itemset (we will precisely define *smallest* in Section 4). In the second stage, we pack the selected items into bins using a variant of the $\texttt{HDH}_k$ algorithm.

In Section 5, we show an algorithm for $d$MCBP, called $\texttt{HGaP}_k$, having an AAR of $T_k^{d-1}(1+\varepsilon)$ and having a running time of $N^{O(1/\varepsilon^2)} n^{(1/\varepsilon)^{O(1/\varepsilon)}} + O(Nd + nd \log n)$. For $d \geq 3$, this matches the present best AAR for the case where rotations are forbidden. Also, for large $k$, this gives an AAR of roughly $T_\infty^2 \approx 2.860$ for 3D bin packing when orthogonal rotations are allowed, which is an improvement over the previous best AAR of 4.5 [13], an improvement after fourteen years.

Our techniques can be extended to some other packing problems, like strip packing and geometric knapsack. In Appendix C of the full version of our paper [30], we define the $d$D multiple-choice strip packing problem ($d$MCSP) and extend Caprara's $\texttt{HDH}_k$ algorithm [8] to

$d$MCSP. The algorithm has AAR $T_k^{d-1}$ and runs in time $O(Nd + nd \log n)$, where $n$ is the number of itemsets and $N$ is the total number of items across all itemsets. In Appendix D of [30], we define the $d$D multiple-choice knapsack problem ($d$MCKS), and for any $0 < \varepsilon < 1$, we show an $O(Nd+N \log N+Nn/\varepsilon+nd \log n)$-time algorithm that is $(1-\varepsilon)3^{-d}$-approximate.

## 2 Preliminaries

Let $[n] := \{1, 2, \ldots, n\}$. For a set $X$, define $\text{sum}(X) := \sum_{x \in X} x$. For an $n$-dimensional vector $\mathbf{v}$, define $\text{sum}(\mathbf{v}) := \sum_{i=1}^{n} \mathbf{v}_i$. For a set $X \subseteq I$ of items and any function $f : I \mapsto \mathbb{R}$, $f(X)$ is defined to be $\sum_{i \in X} f(i)$, unless stated otherwise.

The length of a $d$D item $i$ in the $j^{\text{th}}$ dimension is denoted by $\ell_j(i)$. Define $\text{vol}(i) := \prod_{j=1}^{d} \ell_j(i)$. For a $d$D cuboid $i$, call the first $d-1$ dimensions *base dimensions* and call the $d^{\text{th}}$ dimension *height*. For a set $I$ of items, $|I|$ is the number of items in $I$. Let $|P|$ denote the number of bins used by a packing $P$ of items into bins.

▶ **Lemma 1.** *Consider the inequality $x_1 + x_2 + \ldots + x_n \leq s$, where for each $j \in [n]$, $x_j \in \mathbb{Z}_{\geq 0}$. Let $N$ be the number of solutions to this inequality. Then $N = \binom{s+n}{n} \leq (s+1)^n$.*

**Proof.** The proof of $N = \binom{s+n}{n}$ is a standard result in combinatorics.

To prove $N \leq (s+1)^n$, note that we can choose each $x_j \in \{0, 1, \ldots, s\}$ independently. ◀

### 2.1 Multiple-Choice Packing

Let $\mathcal{I}$ be a set of itemsets. Define $\text{flat}(\mathcal{I})$ to be the union of all itemsets in $\mathcal{I}$.

Let $K$ be a set of items that contains exactly one item from each itemset in $\mathcal{I}$. Formally, for each itemset $I \in \mathcal{I}$, $|K \cap I| = 1$. Then $K$ is called an *assortment* of $\mathcal{I}$. Let $\Psi(\mathcal{I})$ denote the set of all assortments of $\mathcal{I}$. In $d$MCBP, given an input instance $\mathcal{I}$, we have to select an assortment $K \in \Psi(\mathcal{I})$ and output a bin packing of $K$, such that the number of bins used is minimized. Therefore, $\text{opt}_{d\text{MCBP}}(\mathcal{I}) = \min_{K \in \Psi(\mathcal{I})} \text{opt}_{d\text{BP}}(K)$.

## 3 Important Ideas from the $\texttt{HDH}_k$ Algorithm

In this section, we will describe some important ideas behind the $\texttt{HDH}_k$ algorithm for $d$BP by Caprara [8]. These ideas are the building blocks for our algorithms for $d$MCBP.

### 3.1 Weighting Functions

Fekete and Schepers [14] present a useful approach for obtaining lower bounds on the optimal solution to bin packing problems. Their approach is based on *weighting functions*.

▶ **Definition 2.** *$g : [0, 1] \mapsto [0, 1]$ is a weighting function iff for all $m \in \mathbb{Z}_{>0}$ and $x \in [0, 1]^m$,*

$$\sum_{i=1}^{m} x_i \leq 1 \implies \sum_{i=1}^{m} g(x_i) \leq 1$$

*(Weighting functions are also called* dual feasible functions *(DFFs)).*

▶ **Theorem 3.** *Let $I$ be a set of $d$D items that can be packed into a bin. Let $g_1, g_2, \ldots, g_d$ be weighting functions. For $i \in I$, define $g(i)$ as the item whose length is $g_j(\ell_j(i))$ in the $j^{th}$ dimension, for each $j \in [d]$. Then $\{g(i) : i \in I\}$ can be packed into a $d$D bin (without rotating the items).*

Theorem 3 is proved in Appendix E of [30].

## 3.2 The Harmonic Function

To obtain a lower-bound on $\mathrm{opt}_{d\mathrm{BP}}(I)$ using Theorem 3, Caprara [8] defined a function $f_k$. For an integer constant $k \geq 3$, $f_k : [0,1] \mapsto [0,1]$ is defined as

$$f_k(x) := \begin{cases} \frac{1}{q} & x \in \left(\frac{1}{q+1}, \frac{1}{q}\right] \text{ for } q \in [k-1] \\ \frac{k}{k-2}x & x \leq \frac{1}{k} \end{cases}.$$

$f_k$ was originally defined and studied by Lee and Lee [24] for their online algorithm for 1BP, except that they used $k/(k-1)$ instead of $k/(k-2)$. Define $\mathrm{type}_k : [0,1] \mapsto [k]$ as

$$\mathrm{type}_k(x) := \begin{cases} q & x \in \left(\frac{1}{q+1}, \frac{1}{q}\right] \text{ for } q \in [k-1] \\ k & x \leq \frac{1}{k} \end{cases}.$$

Define $T_k$ to be the smallest positive constant such that $H_k(x) := f_k(x)/T_k$ is a weighting function. We call $H_k$ the *harmonic weighting function*. We can efficiently compute $T_k$ as a function of $k$ using ideas from [24]. Table 1 lists the values of $T_k$ for the first few $k$. It can also be proven that $T_k$ is a decreasing function of $k$ and $T_\infty := \lim_{k\to\infty} T_k \approx 1.6910302$.

**Table 1** Values of $T_k$.

| $k$ | 3 | 4 | 5 | 6 | 7 | $\infty$ |
|---|---|---|---|---|---|---|
| $T_k$ | 3 | 2 | $11/6 = 1.8\overline{3}$ | $7/4 = 1.75$ | $26/15 = 1.7\overline{3}$ | $\approx 1.6910302$ |

For a $d$D cuboid $i$, define $f_k(i)$ to be the cuboid whose length is $f_k(\ell_j(i))$ in the $j^{\text{th}}$ dimension, for each $j \in [d]$. For a set $I$ of $d$D cuboids, let $f_k(I) := \{f_k(i) : i \in I\}$. Similarly define $H_k(i)$ and $H_k(I)$. Define $\mathrm{type}(i)$ to be a $d$-dimensional vector whose $j^{\text{th}}$ component is $\mathrm{type}_k(\ell_j(i))$. Note that there can be at most $k^d$ different values of $\mathrm{type}(i)$. Sometimes, for the sake of convenience, we may express $\mathrm{type}(i)$ as an integer in $[k^d]$.

▶ **Theorem 4.** *For a set of $I$ of $d$D items,* $\mathrm{vol}(f_k(I)) \leq T_k^d \,\mathrm{opt}_{d\mathrm{BP}}(I)$.

**Proof.** Let $m := \mathrm{opt}_{d\mathrm{BP}}(I)$. Let $J_j$ be the items in the $j^{\text{th}}$ bin in the optimal bin packing of $I$. By Theorem 3 and because $H_k$ is a weighting function, $H_k(J_j)$ fits in a bin. Therefore,

$$\mathrm{vol}(f_k(I)) = \sum_{j=1}^{m} T_k^d \,\mathrm{vol}(H_k(J_j)) \leq \sum_{j=1}^{m} T_k^d = T_k^d \,\mathrm{opt}_{d\mathrm{BP}}(I). \qquad \blacktriangleleft$$

## 3.3 The HDH-unit-pack$_k$ Subroutine

From the HDH$_k$ algorithm by Caprara [8], we extracted out a useful subroutine, which we call HDH-unit-pack$_k$, that satisfies the following useful property:

▶ **Property 5.** *The algorithm* $\texttt{HDH-unit-pack}_k^{[t]}(I)$ *takes a* sequence *$I$ of $d$D items such that all items have type $t$ and* $\mathrm{vol}(f_k(I - \{\mathrm{last}(I)\})) < 1$ *(here* $\mathrm{last}(I)$ *is the last item in sequence $I$). It returns a packing of $I$ into a single $d$D bin in $O(nd \log n)$ time, where $n := |I|$.*

We use HDH-unit-pack$_k$ as a black-box subroutine in our algorithms, i.e., HDH-unit-pack$_k$ can be replaced by any algorithm that satisfies Property 5. See Appendix B of [30] for a complete description of HDH-unit-pack$_k$ and proof that it satisfies Property 5.

## 4 Fast and Simple Algorithm for $d$MCBP ($\texttt{fullh}_k$)

We will now describe an algorithm for $d$BP called the *full-harmonic algorithm* ($\texttt{fullh}_k$). We will then extend it to $d$MCBP. The $\texttt{fullh}_k$ algorithm works by first partitioning the items based on their type vector (type vector is defined in Section 3.2). Then for each partition, it repeatedly picks the smallest prefix $J$ such that $\text{vol}(f_k(J)) \geq 1$ and packs $J$ into a $d$D bin using $\texttt{HDH-unit-pack}_k$. See Algorithm 1 for a more precise description of $\texttt{fullh}_k$. Note that $\texttt{fullh}_k(I)$ has a running time of $O(|I|d \log |I|)$.

**Algorithm 1** $\texttt{fullh}_k(I)$: Returns a bin packing of $d$D items $I$.

```
Let P be an empty list.
for each type t do
    I^[t] = {i ∈ I : type(i) = t}.
    while |I^[t]| > 0 do
        Find J, the smallest prefix of I^[t] such that J = I^[t] or vol(f_k(J))) ≥ 1.
        B = HDH-unit-pack_k^[t](J).          // B is a packing of J into a dD bin.
        Append B to the list P.
        Remove J from I^[t].
    end while
end for
return the list P of bins.
```

▶ **Theorem 6.** *The number of bins used by* $\texttt{fullh}_k(I)$ *is less than* $Q + \text{vol}(f_k(I))$*, where* $Q$ *is the number of distinct* types *of items (so* $Q \leq k^d$*).*

**Proof.** Let $I^{[t]}$ be the items in $I$ of type $t$. Suppose $\texttt{fullh}_k(I)$ uses $m^{[t]}$ bins to pack $I^{[t]}$. For each type $t$, the first $m^{[t]} - 1$ bins have vol $\cdot f_k$ at least 1, so $\text{vol}(f_k(I^{[t]})) > m^{[t]} - 1$. Therefore, total number of bins used is $\sum_{t=1}^{Q} m^{[t]} < \sum_{t=1}^{Q}(1 + \text{vol}(f_k(I^{[t]}))) = Q + \text{vol}(f_k(I))$. ◀

By Theorems 4 and 6, $\texttt{fullh}_k(I)$ uses less than $Q + T_k^d \, \text{opt}_{d\text{BP}}(I)$ bins.

▶ **Theorem 7.** *Let* $\mathcal{I}$ *be a dMCBP instance. Let* $\widehat{K} := \{\text{argmin}_{i \in I} \text{vol}(f_k(i)) : I \in \mathcal{I}\}$*, i.e.,* $\widehat{K}$ *is the assortment obtained by picking from each itemset the item* $i$ *having the minimum value of* $\text{vol}(f_k(i))$*. Then the number of bins used by* $\texttt{fullh}_k(\widehat{K})$ *is less than* $Q + T_k^d \, \text{opt}_{d\text{MCBP}}(\mathcal{I})$*, where* $Q$ *is the number of distinct* types *of items in* $\text{flat}(\mathcal{I})$ *(so* $Q \leq k^d$*).*

**Proof.** Let $K^*$ be the assortment in an optimal packing of $\mathcal{I}$. So, $\text{vol}(f_k(\widehat{K})) \leq \text{vol}(f_k(K^*))$. By Theorems 4 and 6, the number of bins used by $\texttt{fullh}_k(\widehat{K})$ is less than

$$Q + \text{vol}(f_k(\widehat{K})) \leq Q + \text{vol}(f_k(K^*)) \leq Q + T_k^d \, \text{opt}_{d\text{BP}}(K^*) = Q + T_k^d \, \text{opt}_{d\text{MCBP}}(\mathcal{I}). \quad \blacktriangleleft$$

We can compute $\widehat{K}$ in $O(Nd)$ time and $\texttt{fullh}_k(\widehat{K})$ in $O(nd \log n)$ time, where $N := |\text{flat}(\mathcal{I})|$, $n := |\mathcal{I}|$. So, we get an $O(Nd + nd \log n)$-time $d$MCBP algorithm having AAR $T_k^d$.

### 4.1 $d$BP with Rotations

As mentioned before, we can solve the rotational version of $d$BP by reducing it to $d$MCBP. Specifically, for each item $i$ in the $d$BP instance, we create an itemset containing all orientations of $i$, and we pack the resulting $d$MCBP instance using $\texttt{fullh}_k$. Since an item can have up to $d!$ allowed orientations, this can take up to $O(nd! + nd \log n)$ time. Hence, the running time is large when $d$ is large. However, we can do better for some special cases.

When the bin has the same length in each dimension, then for any item $i$, $\mathrm{vol}(f_k(i))$ is independent of how we orient $i$. Hence, we can orient the items $I$ arbitrarily and then pack them using $\texttt{fullh}_k$ in $O(nd \log n)$ time.

Suppose there are no orientation constraints, i.e., all $d!$ orientations of each item are allowed. Let $L_j$ be the length of the bin in the $j^{\text{th}}$ dimension, for each $j \in [d]$. To use $\texttt{fullh}_k$ to pack $I$, we need to find the best orientation for each item $i \in I$, i.e., we need to find a permutation $\pi$ for each item $i$ such that $\prod_{j=1}^{d} f_k\left(\ell_{\pi_j}(i)/L_j\right)$ is minimized. This can be formulated as a maximum-weight bipartite matching problem on a graph with $d$ vertices in each partition: for every $u \in [d]$ and $v \in [d]$, the edge $(u, v)$ has a non-negative weight of $-\log(f_k(\ell_u(i)/L_v))$. So, using the Kuhn-Munkres algorithm [26], we can find the best orientation for each item in $O(d^3)$ time. Hence, we can pack $I$ using $\texttt{fullh}_k$ in $O(nd^3 + nd \log n)$ time.

## 5 Better Algorithm for $d$MCBP ($\texttt{HGaP}_k$)

Here we describe a $T_k^{d-1}(1+\varepsilon)$-asymptotic-approximate algorithm for $d$MCBP based on $\texttt{HDH}_k$ and Lueker and Fernandez de la Vega's APTAS for 1BP [15]. We call our algorithm *Harmonic Guess-and-Pack* ($\texttt{HGaP}_k$). This improves upon $\texttt{fullh}_k$ that has AAR $T_k^d$.

▶ **Definition 8.** *For a $d$D item $i$, let $h(i) := \ell_d(i)$, $w(i) := \prod_{j=1}^{d-1} f_k(\ell_j(i))$ and $a(i) := w(i)h(i)$. Let $\texttt{round}(i)$ be a rectangle of height $h(i)$ and width $w(i)$. For a set $X$ of $d$D items, define $w(X) := \sum_{i \in X} w(i)$ and $\texttt{round}(X) := \{\texttt{round}(i) : i \in X\}$.*

For any $\varepsilon > 0$, the algorithm $\texttt{HGaP}_k(\mathcal{I}, \varepsilon)$ returns a bin packing of $\mathcal{I}$, where $\mathcal{I}$ is a set of $d$D itemsets. $\texttt{HGaP}_k$ first converts $\mathcal{I}$ to a set $\widehat{\mathcal{I}}$ of 2D itemsets. It then computes $P_{\text{best}}$, which is a *structured* bin packing of $\widehat{\mathcal{I}}$ (we formally define *structured* later). Finally, it uses the algorithm $\texttt{inflate}$ to convert $P_{\text{best}}$ into a bin packing of the $d$D itemsets $\mathcal{I}$, where $|\texttt{inflate}(P_{\text{best}})|$ is very close to $|P_{\text{best}}|$. See Algorithm 2 for a more precise description. We show that $|P_{\text{best}}| \lessapprox T_k^{d-1}(1+\varepsilon)\,\mathrm{opt}(\mathcal{I})$, which proves that $\texttt{HGaP}_k$ has an AAR of $T_k^{d-1}(1+\varepsilon)$. This approach of converting items to 2D, packing them, and then converting back to $d$D is very useful, because most of our analysis is about how to compute a structured 2D packing, and a packing of 2D items is easier to visualize and reason about than a packing of $d$D items.

**Algorithm 2** $\texttt{HGaP}_k(\mathcal{I}, \varepsilon)$: Returns a bin packing of $d$D itemsets $\mathcal{I}$, where $\varepsilon \in (0, 1)$.

```
Let δ := ε/(2 + ε).
Î = {round(I) : I ∈ I}
Initialize P_best to null.
for P ∈ guessShelves(Î, δ) do
    P̄ = chooseAndPack(Î, P, δ)
    if P̄ is not null and (P_best is null or |P̄| ≤ |P_best|) then
        P_best = P̄
    end if
end for
return inflate(P_best)
```

A 2D bin packing is called *shelf-based* if items are packed into *shelves* and the shelves are packed into bins, where a shelf is a rectangle of width 1. See Figure 2 for an example. A structured bin packing is a shelf-based bin packing where the heights of the shelves satisfy some additional properties (we describe these properties later). The algorithm $\texttt{guessShelves}$

repeatedly guesses the number and heights of shelves and computes a structured packing $P$ of those shelves into bins. Then for each packing $P$, the algorithm `chooseAndPack`$(\widehat{\mathcal{I}}, P, \delta)$ *tries to* pack an assortment of $\widehat{\mathcal{I}}$ into the shelves in $P$ plus one additional shelf. If `chooseAndPack` succeeds, call the resulting bin packing $\overline{P}$; else, `chooseAndPack` returns `null`. $P_{\text{best}}$ is the value of $\overline{P}$ with the minimum number of bins across all guesses by `guessShelves`.

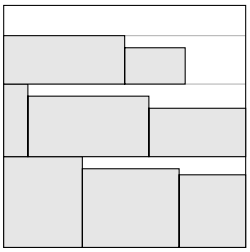

**Figure 2** An example of shelf-based packing with 3 shelves.

To prove that $\texttt{HGaP}_k$ has AAR $T_k^{d-1}(1+\varepsilon)$, we show that for some $P^* \in \texttt{guessShelves}(\widehat{\mathcal{I}}, \delta)$, we have $|P^*| \lessapprox T_k^{d-1}(1+\varepsilon)\operatorname{opt}(\mathcal{I})$ and $\texttt{chooseAndPack}(\widehat{\mathcal{I}}, P^*, \delta)$ is not `null`.

We will now precisely define *structured* packing and state the main theorems on $\texttt{HGaP}_k$.

## 5.1 Structured Packing

▶ **Definition 9** (Slicing). *Slicing a 1D item $i$ is the operation of replacing it by items $i_1$ and $i_2$ such that* $\operatorname{size}(i_1)+\operatorname{size}(i_2)=\operatorname{size}(i)$. *Slicing a rectangle $i$ using a vertical cut is the operation of replacing $i$ by two rectangles $i_1$ and $i_2$ where $h(i)=h(i_1)=h(i_2)$ and $w(i)=w(i_1)+w(i_2)$. Slicing $i$ using a horizontal cut is the operation of replacing $i$ by two rectangles $i_1$ and $i_2$ where $w(i)=w(i_1)=w(i_2)$ and $h(i)=h(i_1)+h(i_2)$.*

▶ **Definition 10** (Shelf-based $\delta$-fractional packing). *Let $\delta \in (0,1)$ be a constant. Let $K$ be a set of rectangular items. Items in $K_L := \{i \in K : h(i) > \delta\}$ are called "$\delta$-large" and items in $K_S := K - K_L$ are called "$\delta$-small". A $\delta$-fractional bin packing of $K$ is defined to be a packing of $K$ into bins where items in $K_L$ can be sliced (recursively) using vertical cuts only, and items in $K_S$ can be sliced (recursively) using both horizontal and vertical cuts.*

*A* shelf *is a rectangle of width 1 into which we can pack items such that the bottom edge of each item in the shelf touches the bottom edge of the shelf. A shelf can itself be packed into a bin. A $\delta$-fractional bin packing of $K$ is called* shelf-based *iff (all slices of) all items in $K_L$ are packed into shelves, the shelves are packed into the bins, and items in $K_S$ are packed outside the shelves (and inside the bins). Packing of items into a shelf $S$ is called* tight *iff the top edge of some item (or slice) in $S$ touches the top edge of $S$.*

▶ **Definition 11** (Structured packing). *Let $K$ be a set of rectangles and let $P$ be a packing of empty shelves into bins. Let $H$ be the set of heights of shelves in $P$ (note that $H$ is not a multiset, i.e., we only consider distinct heights of shelves). Then $P$ is called* structured *for $(K,\delta)$ iff $|H| \le \lceil 1/\delta^2 \rceil$ and each element in $H$ is the height of some $\delta$-large item in $K$.*

*A shelf-based $\delta$-fractional packing of $K$ is called* structured *iff the shelves in the packing are structured for $(K,\delta)$. Define* $\operatorname{sopt}_\delta(K)$ *to be the number of bins in the optimal structured $\delta$-fractional packing of $K$.*

$\texttt{HGaP}_k$ relies on the following key structural theorem. We formally prove it in Section 5.5 and give an outline of the proof here.

▶ **Theorem 12** (Structural theorem). *Let $I$ be a set of dD items. Let $\delta \in (0,1)$ be a constant. Then* $\operatorname{sopt}_\delta(\texttt{round}(I)) < T_k^{d-1}(1+\delta)\operatorname{opt}_{d\mathrm{BP}}(I) + \lceil 1/\delta^2 \rceil + 1 + \delta$.

**Proof outline.** Let $\widehat{I} := \texttt{round}(I)$. Let $\widehat{I}_L$ and $\widehat{I}_S$ be the $\delta$-large and $\delta$-small items in $\widehat{I}$, respectively. We give a simple greedy algorithm to pack $\widehat{I}_L$ into shelves. Let $J$ be the shelves output by this algorithm. We can treat $J$ as a 1BP instance, and $\widehat{I}_S$ as a sliceable 1D item of size $a(\widehat{I}_S)$. We prove that an optimal 1D bin packing of $J \cup \widehat{I}_S$ gives us an optimal shelf-based $\delta$-fractional packing of $\widehat{I}$.

We use linear grouping by Lueker and Fernandez de la Vega [15]. We partition $J$ into linear groups of size $\lfloor \delta \operatorname{size}(J) \rfloor + 1$ each. Let $h_j$ be the height of the first 1D item in the $j^{\text{th}}$ group. Let $J^{(\text{hi})}$ be the 1BP instance obtained by rounding up the height of each item in the $j^{\text{th}}$ group to $h_j$ for all $j$. Then $J^{(\text{hi})}$ contains at most $\lceil 1/\delta^2 \rceil$ distinct sizes, so the optimal packing of $J^{(\text{hi})} \cup \widehat{I}_S$ gives us a structured $\delta$-fractional packing of $\widehat{I}$. Therefore, $\operatorname{sopt}_\delta(\widehat{I}) \le \operatorname{opt}(J^{(\text{hi})} \cup \widehat{I}_S)$. Let $J^{(\text{lo})}$ be the 1BP instance obtained by rounding down the height of each item in the $j^{\text{th}}$ group to $h_{j+1}$ for all $j$. We prove that $J^{(\text{lo})}$ contains at most $\lceil 1/\delta^2 \rceil - 1$ distinct sizes and that $\operatorname{opt}(J^{(\text{hi})} \cup \widehat{I}_S) < \operatorname{opt}(J^{(\text{lo})} \cup \widehat{I}_S) + \delta a(\widehat{I}_L) + (1+\delta)$.

We model packing $J^{(\text{lo})} \cup \widehat{I}_S$ as a linear program, denoted by $\operatorname{LP}(\widehat{I})$, that has at most $\lceil 1/\delta^2 \rceil^{1/\delta}$ variables and $\lceil 1/\delta^2 \rceil$ non-trivial constraints. The optimum extreme point solution to $\operatorname{LP}(\widehat{I})$, therefore, has at most $\lceil 1/\delta^2 \rceil$ positive entries, so $\operatorname{opt}(J^{(\text{lo})} \cup \widehat{I}_S) \le \operatorname{opt}(\operatorname{LP}(\widehat{I})) + \lceil 1/\delta^2 \rceil$.

We use techniques from Caprara [8] to obtain a monotonic weighting function $\eta$ from the optimal solution to the dual of $\operatorname{LP}(\widehat{I})$. For each item $i \in I$, we define $p(i) := w(i)\eta(h(i))$ and prove that $p(I) \ge \operatorname{opt}(\operatorname{LP}(\widehat{I}))$. By Theorem 3, we get that $p(I) \le T_k^{d-1} \operatorname{opt}_{d\mathrm{BP}}(I)$ and $a(\widehat{I}_L) \le T_k^{d-1} \operatorname{opt}_{d\mathrm{BP}}(I)$. Combining the above facts gives us an upper-bound on $\operatorname{sopt}_\delta(\widehat{I})$ in terms of $\operatorname{opt}_{d\mathrm{BP}}(I)$. ◀

## 5.2 Subroutines

### 5.2.1 `guessShelves`

The algorithm $\texttt{guessShelves}(\widehat{\mathcal{I}}, \delta)$ takes a set $\widehat{\mathcal{I}}$ of 2D itemsets and a constant $\delta \in (0,1)$ as input. We will design `guessShelves` so that it satisfies the following theorem.

▶ **Theorem 13.** $\texttt{guessShelves}(\widehat{\mathcal{I}}, \delta)$ *returns all possible packings of empty shelves into at most $|\widehat{\mathcal{I}}|$ bins such that each packing is structured for* $(\operatorname{flat}(\widehat{\mathcal{I}}), \delta)$. $\texttt{guessShelves}(\widehat{\mathcal{I}}, \delta)$ *returns at most $T := (N^{\lceil 1/\delta^2 \rceil} + 1)(n+1)^R$ packings, where $N := |\operatorname{flat}(\widehat{\mathcal{I}})|$, $n := |\widehat{\mathcal{I}}|$, and $R := \binom{\lceil 1/\delta^2 \rceil + \lceil 1/\delta \rceil - 1}{\lceil 1/\delta \rceil - 1} \le (1 + \lceil 1/\delta^2 \rceil)^{1/\delta}$. Its running time is $O(T)$.*

`guessShelves` works by first guessing at most $\lceil 1/\delta^2 \rceil$ distinct heights of shelves. It then enumerates all configurations, i.e., different ways in which shelves can be packed into a bin. It then guesses the configurations in a bin packing of the shelves. `guessShelves` can be easily implemented using standard techniques. For the sake of completeness, we give a more precise description of `guessShelves` and prove Theorem 13 in Appendix A.2.

### 5.2.2 `chooseAndPack`

$\texttt{chooseAndPack}(\widehat{\mathcal{I}}, P, \delta)$ takes as input a set $\widehat{\mathcal{I}}$ of 2D itemsets, a constant $\delta \in (0,1)$, and a bin packing $P$ of empty shelves that is structured for $(\operatorname{flat}(\widehat{\mathcal{I}}), \delta)$. It tries to pack an assortment of $\widehat{\mathcal{I}}$ into the shelves in $P$.

`chooseAndPack` works by rounding up the width of all $\delta$-large items in $\widehat{\mathcal{I}}$ to a multiple of $1/n$. This would increase the number of shelves required by 1, so it adds another empty shelf. It then uses dynamic programming to pack an assortment into the shelves, such that

the area of the chosen $\delta$-small items is minimum. This is done by maintaining a dynamic programming table that keeps track of the number of itemsets considered so far and the remaining space in shelves of each type. If it is not possible to pack the items into the shelves, then `chooseAndPack` outputs `null`. In Appendix A.3, we give the details of this algorithm and formally prove the following theorems:

▶ **Theorem 14.** *If there exists an assortment $\widehat{K}$ of $\widehat{\mathcal{I}}$ having a structured $\delta$-fractional bin packing $P$, then* $\texttt{chooseAndPack}(\widehat{\mathcal{I}}, P, \delta)$ *does not output* `null`.

▶ **Theorem 15.** *If the output of* $\texttt{chooseAndPack}(\widehat{\mathcal{I}}, P, \delta)$ *is not* `null`*, then the output $\overline{P}$ is a shelf-based $\delta$-fractional packing of some assortment of $\widehat{\mathcal{I}}$ such that $|\overline{P}| \leq |P| + 1$ and the distinct shelf heights in $\overline{P}$ are the same as that in $P$.*

▶ **Theorem 16.** $\texttt{chooseAndPack}(\widehat{\mathcal{I}}, P, \delta)$ *runs in $O(Nn^{2\lceil 1/\delta^2 \rceil})$ time. Here $N := |\operatorname{flat}(\widehat{\mathcal{I}})|$, $n := |\widehat{\mathcal{I}}|$.*

### 5.2.3 `inflate`

For a set $I$ of $d$D items, `inflate` is an algorithm that converts a shelf-based packing of $\texttt{round}(I)$ into a packing of $I$ having roughly the same number of bins.

For a $d$D item $i$, btype($i$) (called *base type*) is defined to be a $(d-1)$-dimensional vector whose $j^{\text{th}}$ component is $\text{type}_k(\ell_j(i))$. Roughly, $\texttt{inflate}(P)$ works as follows: It first slightly modifies the packing $P$ so that items of different base types are in different shelves and $\delta$-small items are no longer sliced using horizontal cuts. Then it converts each 2D shelf to a $d$D shelf of the same height using $\texttt{HDH-unit-pack}_k$ (a $d$D shelf is a cuboid where the first $d-1$ dimensions are equal to 1).

In Appendix A.4, we formally describe `inflate` and prove the following theorem.

▶ **Theorem 17.** *Let $I$ be a set of $d$D items having $Q$ distinct base types (there can be at most $k^{d-1}$ distinct base types, so $Q \leq k^{d-1}$). Let $P$ be a shelf-based $\delta$-fractional packing of* $\texttt{round}(I)$ *where shelves have $t$ distinct heights. Then* $\texttt{inflate}(P)$ *returns a packing of $I$ into less than $|P|/(1-\delta) + t(Q-1) + 1 + \delta Q/(1-\delta)$ bins in $O(|I|d \log |I|)$ time.*

Now that we have mentioned the guarantees of all the subroutines used by $\texttt{HGaP}_k$, we can prove the correctness and running time of $\texttt{HGaP}_k$.

## 5.3 Correctness and Running Time of $\texttt{HGaP}_k$

▶ **Theorem 18.** *The number of bins used by* $\texttt{HGaP}_k(\mathcal{I}, \varepsilon)$ *to pack $\mathcal{I}$ is less than*

$$T_k^{d-1}(1+\varepsilon)\operatorname{opt}_{d\text{MCBP}}(\mathcal{I}) + \left\lceil \left(\frac{2}{\varepsilon}+1\right)^2 \right\rceil \left(Q + \frac{\varepsilon}{2}\right) + 3 + (Q+3)\frac{\varepsilon}{2}.$$

*Here $Q \leq k^{d-1}$ is the number of distinct base types in* $\operatorname{flat}(\mathcal{I})$.

**Proof.** Let $K^*$ be the assortment in an optimal bin packing of $\mathcal{I}$. Let $\widehat{K}^* = \texttt{round}(K^*)$. Let $P^*$ be the optimal structured $\delta$-fractional bin packing of $\widehat{K}^*$. Then $|P^*| = \operatorname{sopt}_\delta(\widehat{K}^*)$ by the definition of sopt. By Theorem 13, $P^* \in \texttt{guessShelves}(\widehat{\mathcal{I}}, \delta)$. Let $\overline{P}^* = \texttt{chooseAndPack}(\widehat{\mathcal{I}}, P^*, \delta)$. By Theorem 14, $\overline{P}^*$ is not `null`. By Theorem 15, $P_{\text{best}}$ is structured for $(\operatorname{flat}(\widehat{\mathcal{I}}), \delta)$ and $|P_{\text{best}}| \leq |\overline{P}^*| \leq \operatorname{sopt}_\delta(\widehat{K}^*) + 1$.

By Theorem 17, we get that

$$|\,\texttt{inflate}(P_{\text{best}})| < \frac{\operatorname{sopt}_\delta(\widehat{K}^*)}{1-\delta} + \left\lceil \frac{1}{\delta^2} \right\rceil (Q-1) + 1 + \frac{\delta Q + 1}{1-\delta}.$$

By Theorem 12 (structural theorem) and using $\operatorname{opt}_{d\text{BP}}(K^*) = \operatorname{opt}_{d\text{MCBP}}(\mathcal{I})$, we get

$$\operatorname{sopt}_\delta(\widehat{K}^*) < T_k^{d-1}(1+\delta)\operatorname{opt}_{d\text{MCBP}}(\mathcal{I}) + \lceil 1/\delta^2 \rceil + 1 + \delta.$$

Therefore, $|\,\texttt{inflate}(P_{\text{best}})|$ is less than

$$T_k^{d-1}\frac{1+\delta}{1-\delta}\operatorname{opt}_{d\text{MCBP}}(\mathcal{I}) + \left\lceil \frac{1}{\delta^2} \right\rceil \left(Q + \frac{\delta}{1-\delta}\right) + 3 + \frac{\delta(3+Q)}{1-\delta}$$
$$= T_k^{d-1}(1+\varepsilon)\operatorname{opt}_{d\text{MCBP}}(\mathcal{I}) + \left\lceil \left(\frac{2}{\varepsilon}+1\right)^2 \right\rceil \left(Q + \frac{\varepsilon}{2}\right) + 3 + (Q+3)\frac{\varepsilon}{2}. \quad \blacktriangleleft$$

▶ **Theorem 19.** *$\texttt{HGaP}_k(\mathcal{I}, \varepsilon)$ runs in time $O(N^{1+\lceil 1/\delta^2 \rceil} n^{R+2\lceil 1/\delta^2 \rceil} + Nd + nd\log n)$, where $n := |\widehat{\mathcal{I}}|$, $N := |\operatorname{flat}(\widehat{\mathcal{I}})|$, $\delta := \varepsilon/(2+\varepsilon)$ and $R := \binom{\lceil 1/\delta^2 \rceil + \lceil 1/\delta \rceil - 1}{\lceil 1/\delta \rceil - 1} \le (1 + \lceil 1/\delta^2 \rceil)^{1/\delta}$.*

**Proof.** Follows from Theorems 13, 16, and 17. ◀

Appendix A.5 gives hints on improving the running time of $\texttt{HGaP}_k$.

## 5.4 *d*BP with Rotations

We can solve the rotational version of $d$BP by reducing it to $d$MCBP and using the $\texttt{HGaP}_k$ algorithm. Since each item can have up to $d!$ orientations, the running time is polynomial in $nd!$, which is large when $d$ is large. But we can do better for some special cases.

When the bin has the same length in each dimension, then for any item $i$, $w(i) := \prod_{j=1}^{d-1} f_k(\ell_j(i))$ is invariant to permuting the first $d-1$ dimensions. In the first step of $\texttt{HGaP}_k$, we replace each $d$D item $i$ by a rectangle of width $w(i)$ and height $\ell_d(i)$. So, instead of considering all $d!$ orientations, we just need to consider at most $d$ different orientations, where each orientation has a different length in the $d^{\text{th}}$ dimension.

Suppose there are no orientation constraints, i.e., all $d!$ orientations of each item are allowed. Let $L_j$ be the length of the bin in the $j^{\text{th}}$ dimension, for each $j \in [d]$. Analogous to the trick in Section 4.1, we first fix the $d^{\text{th}}$ dimension of the item and then optimally permute the first $d-1$ dimensions using a max-weight bipartite matching algorithm. Hence, we need to consider only $d$ orientations instead of $d!$.

## 5.5 Proof of the Structural Theorem

In this section, we give a formal proof of the Structural Theorem (Theorem 12).

### 5.5.1 Predecessors and Canonical Shelving

▶ **Definition 20.** *Let $I_1$ and $I_2$ be sets of 1D items. $I_1$ is called a predecessor of $I_2$ ($I_1 \preceq I_2$) iff there exists a one-to-one mapping $\pi : I_1 \mapsto I_2$ such that $\forall i \in I_1, \operatorname{size}(i) \le \operatorname{size}(\pi(i))$.*

▶ **Observation 21.** *Let $I_1 \preceq I_2$ and $\pi$ be the corresponding mapping. We can get a packing of $I_1$ from a packing of $I_2$, by packing each $i \in I_1$ in the place of $\pi(i)$. Hence, $\operatorname{opt}(I_1) \le \operatorname{opt}(I_2)$.*

▶ **Definition 22** (Canonical shelving). *Let $I$ be a set of rectangles. Order the items in $I$ in non-increasing order of height (break ties arbitrarily but deterministically) and greedily pack them into tight shelves, slicing items using vertical cuts if necessary. The set of shelves thus obtained is called the* canonical shelving *of $I$, and is denoted by* canShelv$(I)$. *(The canonical shelving is unique because ties are broken deterministically.)*

See Figure 3 for an example of canonical shelving.

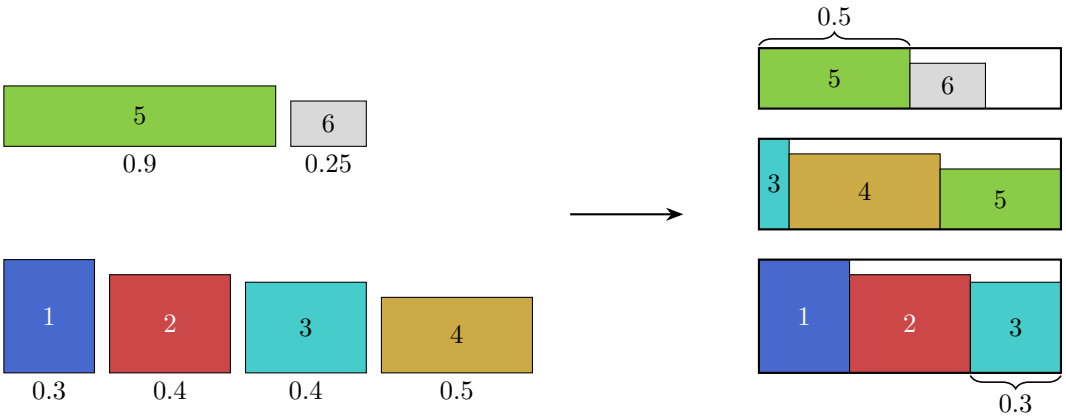


**Figure 3** Six items and their canonical shelving into three tight shelves of width 1. The items are numbered by decreasing order of height. Each item has its width mentioned below it. Item 3 was sliced into two items of widths 0.3 and 0.1. Item 5 was sliced into two items of widths 0.4 and 0.5.

Suppose a set $I$ of rectangular items is packed into a set $J$ of shelves. Then we can interpret $J$ as a 1BP instance where the height of each shelf is the size of the corresponding 1D item. We will now prove that the canonical shelving is optimal, i.e., any shelf-based bin packing of items can be obtained by first computing the canonical shelving and then packing the shelves into bins like a 1BP instance.

▶ **Lemma 23.** *If $J^* :=$ canShelv$(I)$ and $I$ can be packed into shelves $J$, then $J^* \preceq J$.*

**Proof.** We say that a shelf is full if the total width of items in a shelf is 1. Arrange the shelves $J$ in non-increasing order of height, and arrange the items $I$ in non-increasing order of height. Then try to pack $I$ into $J$ using the following greedy algorithm: For each item $i$, pack the largest possible slice of $i$ into the first non-full shelf and pack the remaining slice (if any) in the next shelf. If this greedy algorithm succeeds, then within each shelf of $J$, there is a shelf of $J^*$, so $J^* \preceq J$. We will now prove that this greedy algorithm always succeeds.

For the sake of proof by contradiction, assume that the greedy algorithm failed, i.e., for an item (or slice) $i$ there was a non-full shelf $S$ but $h(i) > h(S)$. Let $I'$ be the items (and slices) packed before $i$ and $J'$ be the shelves before $S$. Therefore, $w(I') = |J'|$.

Items in $I'$ have height at least $h(i)$, so shelves in $J'$ have height at least $h(i)$. Shelves after $J'$ have height less than $h(i)$. So, $J'$ is exactly the set of shelves of height at least $h(i)$. In the packing $P$, $I' \cup \{i\}$ can only be packed into shelves of height at least $h(i)$, so $w(I') + w(i) \leq |J'|$. This contradicts $w(I') = |J'|$. So, the greedy algorithm cannot fail. ◀

### 5.5.2 Linear Grouping

Let $I$ be a set of $d$D items. Let $\widehat{I} := \texttt{round}(I)$. Let $\delta \in (0, 1)$ be a constant. Let $\widehat{I}_L := \{i \in \widehat{I} : h(i) > \delta\}$ and $\widehat{I}_S := \widehat{I} - \widehat{I}_L$. Let $J :=$ canShelv$(\widehat{I}_L)$. Let $m := |J|$, i.e., $J$ contains $m$ shelves. We can interpret $\widehat{I}_S$ as a single sliceable 1D item of size $a(\widehat{I}_S)$.

To prove Theorem 12, we will show the existence of a structured $\delta$-fractional packing of $\widehat{I}$ into at most $T_k^{d-1}(1+\delta)\operatorname{opt}_{d\mathrm{BP}}(I) + \lceil 1/\delta^2 \rceil + 1 + \delta$ bins.

▶ **Definition 24** (Linear grouping [15]). *Arrange the 1D items $J$ in non-increasing order of size and number them from 1 to $m$. Let $q := \lfloor \delta \operatorname{size}(J) \rfloor + 1$. Let $J_1$ be the first $q$ items, $J_2$ be the next $q$ items, and so on. $J_j$ is called the $j^{th}$* linear group *of $J$. This gives us $t := \lceil m/q \rceil$ linear groups. Note that the last group, $J_t$, may have less than $q$ items.*

*Let $h_j$ be the size of the first item in $J_j$. Let $h_{t+1} := 0$. For $j \in [t-1]$, let $J_j^{(\mathrm{lo})}$ be the items obtained by decreasing the height of items in $J_j$ to $h_{j+1}$. For $j \in [t]$, let $J_j^{(\mathrm{hi})}$ be the items obtained by increasing the height of items in $J_j$ to $h_j$.*

*Let $J^{(\mathrm{lo})} := \bigcup_{j=1}^{t-1} J_j^{(\mathrm{lo})}$ and $J^{(\mathrm{hi})} := \bigcup_{j=1}^{t} J_j^{(\mathrm{hi})}$. We call $J^{(\mathrm{lo})}$ a down-rounding of $J$ and $J^{(\mathrm{hi})}$ an up-rounding of $J$.*

▶ **Lemma 25.** $t \le \lceil 1/\delta^2 \rceil$.

**Proof.** Since each shelf in $J$ has height more than $\delta$, $\operatorname{size}(J) > |J|\delta$.

$$t := \left\lceil \frac{|J|}{\lfloor \delta \operatorname{size}(J) \rfloor + 1} \right\rceil \le \left\lceil \frac{\operatorname{size}(J)/\delta}{\delta \operatorname{size}(J)} \right\rceil = \left\lceil \frac{1}{\delta^2} \right\rceil . \qquad \blacktriangleleft$$

▶ **Lemma 26.** $J^{(\mathrm{lo})} \preceq J \preceq J^{(\mathrm{hi})} \preceq J^{(\mathrm{lo})} \cup J_1^{(\mathrm{hi})}$.

**Proof.** It is trivial to see that $J^{(\mathrm{lo})} \preceq J \preceq J^{(\mathrm{hi})}$. For $j \in [t-1]$, all (1D) items in both $J_j^{(\mathrm{lo})}$ and $J_{j+1}^{(\mathrm{hi})}$ have height $h_{j+1}$, and $|J_{j+1}| \le q = |J_j|$. Therefore, $J_{j+1}^{(\mathrm{hi})} \preceq J_j^{(\mathrm{lo})}$ and hence

$$J^{(\mathrm{hi})} = J_1^{(\mathrm{hi})} \cup \bigcup_{j=1}^{t-1} J_{j+1}^{(\mathrm{hi})} \preceq J_1^{(\mathrm{hi})} \cup \bigcup_{j=1}^{t-1} J_j^{(\mathrm{lo})} = J_1^{(\mathrm{hi})} \cup J^{(\mathrm{lo})}. \qquad \blacktriangleleft$$

▶ **Lemma 27.** $\operatorname{size}(J) < 1 + a(\widehat{I}_L)$.

**Proof.** In the canonical shelving of $\widehat{I}_L$, let $S_j$ be the $j^{\text{th}}$ shelf. Let $h(S_j)$ be the height of $S_j$. Let $a(S_j)$ be the total area of the items in $S_j$. Since the shelves are tight, items in $S_j$ have height at least $h(S_{j+1})$. So, $a(S_j) \ge h(S_{j+1})$ and

$$\operatorname{size}(J) = \sum_{j=1}^{|J|} h(S_j) \le 1 + \sum_{j=1}^{|J|-1} h(S_{j+1}) \le 1 + \sum_{j=1}^{|J|-1} a(S_j) < 1 + a(\widehat{I}_L). \qquad \blacktriangleleft$$

▶ **Lemma 28.** $\operatorname{sopt}_\delta(\widehat{I}) < \operatorname{opt}(J^{(\mathrm{lo})} \cup \widehat{I}_S) + \delta a(\widehat{I}_L) + (1+\delta)$.

**Proof.** By the definition of canShelv, $\widehat{I}_L$ can be packed into $J$. By Lemma 26, $J \preceq J^{(\mathrm{hi})}$, so $\widehat{I}_L$ can be packed into $J^{(\mathrm{hi})}$. By Lemma 25, the number of distinct sizes in $J^{(\mathrm{hi})}$ is at most $\lceil 1/\delta^2 \rceil$. So, the optimal 1D bin packing of $J^{(\mathrm{hi})} \cup \widehat{I}_S$ will give us a structured $\delta$-fractional bin packing of $\widehat{I}$. Hence, $\operatorname{sopt}_\delta(\widehat{I}) \le \operatorname{opt}(J^{(\mathrm{hi})} \cup \widehat{I}_S)$. By Lemma 26 and Observation 21 we get

$$\operatorname{opt}(J^{(\mathrm{hi})} \cup \widehat{I}_S) \le \operatorname{opt}(J^{(\mathrm{lo})} \cup J_1^{(\mathrm{hi})} \cup \widehat{I}_S) \le \operatorname{opt}(J^{(\mathrm{lo})} \cup \widehat{I}_S) + \operatorname{opt}(J_1^{(\mathrm{hi})}).$$

By Lemma 27,

$$\operatorname{opt}(J_1^{(\mathrm{hi})}) \le |J_1^{(\mathrm{hi})}| \le q \le 1 + \delta \operatorname{size}(J) < 1 + \delta(1 + a(\widehat{I}_L)). \qquad \blacktriangleleft$$

### 5.5.3 LP for Packing $J^{(\text{lo})} \cup \widehat{I}_S$

We will formulate an integer linear program for bin packing $J^{(\text{lo})} \cup \widehat{I}_S$.

Let $C \in \mathbb{Z}_{\geq 0}^{t-1}$ such that $h_C := \sum_{j=1}^{t-1} C_j h_{j+1} \leq 1$. Then $C$ is called a configuration. $C$ represents a set of 1D items that can be packed into a bin and where $C_j$ items are from $J_j^{(\text{lo})}$. Let $\mathcal{C}$ be the set of all configurations. We can pack at most $\lceil 1/\delta \rceil - 1$ 1D items into a bin because $h_t > \delta$. By Lemma 1, we get $|\mathcal{C}| \leq \binom{\lceil 1/\delta \rceil - 1 + t - 1}{t-1} \leq \lceil 1/\delta^2 \rceil^{1/\delta}$.

Let $x_C$ be the number of bins packed according to configuration $C$. Bin packing $J^{(\text{lo})} \cup \widehat{I}_S$ is equivalent to finding the optimal integer solution to the following linear program, which we denote as $\text{LP}(\widehat{I})$.

$$\begin{aligned}
\min_{x \in \mathbb{R}^{|\mathcal{C}|}} \quad & \sum_{C \in \mathcal{C}} x_C \\
\text{where} \quad & \sum_{C \in \mathcal{C}} C_j x_C \geq q && \forall j \in [t-1] \\
& \sum_{C \in \mathcal{C}} (1 - h_C) x_C \geq a(\widehat{I}_S) \\
& x_C \geq 0 && \forall C \in \mathcal{C}
\end{aligned}$$

Here the first set of constraints say that for each $j \in [t-1]$, all of the $q := \lfloor \delta \, \text{size}(J) \rfloor + 1$ shelves $J_j^{(\text{lo})}$ should be covered by the configurations in $x$. The second constraint says that we should be able to pack $a(\widehat{I}_S)$ into the non-shelf space in the bins.

▶ **Lemma 29.** $\text{opt}(J^{(\text{lo})} \cup \widehat{I}_S) \leq \text{opt}(\text{LP}(\widehat{I})) + t$.

**Proof.** Let $x^*$ be an optimal extreme-point solution to $\text{LP}(\widehat{I})$. Then $x^*$ has at most $t$ non-zero entries. Let $\widehat{x}$ be a vector where $\widehat{x}_C := \lceil x^*_C \rceil$. Then $\widehat{x}$ is an integral solution to $\text{LP}(\widehat{I})$ and $\sum_C \widehat{x}_C < t + \sum_C x^*_C = \text{opt}(\text{LP}(\widehat{I})) + t$. ◀

The dual of $\text{LP}(\widehat{I})$, denoted by $\text{DLP}(\widehat{I})$, is

$$\begin{aligned}
\max_{y \in \mathbb{R}^{t-1}, z \in \mathbb{R}} \quad & a(\widehat{I}_S) z + q \sum_{j=1}^{t-1} y_j \\
\text{where} \quad & \sum_{j=1}^{t-1} C_j y_j + (1 - h_C) z \leq 1 \quad \forall C \in \mathcal{C} \\
& z \geq 0 \text{ and } y_j \geq 0 \quad \forall j \in [t-1]
\end{aligned}$$

We will now see how to obtain a monotonic weighting function $\eta : [0,1] \mapsto [0,1]$ from a feasible solution to $\text{DLP}(\widehat{I})$. To do this, we adapt techniques from Caprara's analysis of $\texttt{HDH}_k$ [8]: we first describe a transformation to convert any feasible solution of $\text{DLP}(\widehat{I})$ to a feasible solution that is *monotonic*, and then show how to obtain a weighting function from this monotonic solution. Such a weighting function will help us upper-bound $\text{opt}(\text{LP}(\widehat{I}))$ in terms of $\text{opt}_{d\text{BP}}(I)$.

▶ **Definition 30.** *Let $(y, z)$ be a feasible solution to $\text{DLP}(\widehat{I})$. Let $h_{t+1} := 0$ and for $j \in [t-1]$ let $\widehat{y}_j := \max(y_j, \widehat{y}_{j+1} + (h_{j+1} - h_{j+2})z)$. Then $(\widehat{y}, z)$ is called the monotonization of $(y, z)$.*

▶ **Lemma 31.** *Let $(y, z)$ be a feasible solution to $\text{DLP}(\widehat{I})$. Let $(\widehat{y}, z)$ be the monotonization of $(y, z)$. Then $(\widehat{y}, z)$ is a feasible solution to $\text{DLP}(\widehat{I})$.*

**Proof.** (See Appendix A.1.) ◀

Let $(y^*, z^*)$ be an optimal solution to $\mathrm{DLP}(\widehat{I})$. Let $(\widehat{y}, z^*)$ be the monotonization of $(y^*, z^*)$. Then define the function $\eta : [0,1] \mapsto [0,1]$ as

$$\eta(x) := \begin{cases} \widehat{y}_1 & \text{if } x \in [h_2, 1] \\ \widehat{y}_j & \text{if } x \in [h_{j+1}, h_j), \text{ for } 2 \le j \le t-1 \\ xz^* & \text{if } x < h_t \end{cases}.$$

▶ **Lemma 32.** *$\eta$ is a monotonic weighting function.*

**Proof.** (See Appendix A.1.) ◀

▶ **Lemma 33.** *For $i \in I$, let $p(i) := \eta(h(i))w(i)$. Then $\mathrm{opt}(\mathrm{LP}(\widehat{I})) \le p(I) \le T_k^{d-1} \mathrm{opt}_{d\mathrm{BP}}(I)$.*

**Proof.** Let $(y^*, z^*)$ be an optimal solution to $\mathrm{DLP}(\widehat{I})$. Let $(\widehat{y}, z^*)$ be its monotonization.

In the canonical shelving of $I$, suppose a rectangular item $i$ (or a slice thereof) lies in shelf $S$ where $S \in J_j$. Then $h(i) \in [h_{j+1}, h_j]$, where $h_{t+1} := 0$. This is because shelves in $J := \mathrm{canShelv}(\widehat{I})$ are tight. If $j = 1$, then $\eta(h(i)) = \widehat{y}_1 \ge y_1^*$. If $2 \le j \le t-1$, then $\eta(h(i)) \in \{\widehat{y}_{j-1}, \widehat{y}_j\} \ge \widehat{y}_j \ge y_j^*$. We know that $w(S) = 1$ for each shelf $S \in J_j$ for $j \in [t-1]$.

$$\begin{aligned} p(I) &= \sum_{j=1}^{t} \sum_{S \in J_j} \sum_{i \in S} \eta(h(i))w(i) + \sum_{i \in \widehat{I}_S} \eta(h(i))w(i) && \text{(by definition of } p\text{)} \\ &\ge \sum_{j=1}^{t-1} \sum_{S \in J_j} \sum_{i \in S} y_j^* w(i) + \sum_{i \in \widehat{I}_S} (h(i)z^*)w(i) && \text{(by definition of } \eta\text{)} \\ &= \sum_{j=1}^{t-1} y_j^* q + a(\widehat{I}_S)z^* && \text{(since } w(J_j) = q \text{ for } j \in [t-1]\text{)} \\ &= \mathrm{opt}(\mathrm{DLP}(\widehat{I})). && \text{(}(y^*, z^*) \text{ is optimal for } \mathrm{DLP}(\widehat{I})\text{)} \end{aligned}$$

By strong duality of linear programs, $\mathrm{opt}(\mathrm{LP}(\widehat{I})) = \mathrm{opt}(\mathrm{DLP}(\widehat{I})) \le p(I)$. Since $\eta$ and $H_k$ are weighting functions (by Lemma 32), we get that $p(I) \le T_k^{d-1} \mathrm{opt}_{d\mathrm{BP}}(I)$ by Theorem 3. ◀

▶ **Theorem 12** (Structural theorem). *Let $I$ be a set of $dD$ items. Let $\delta \in (0,1)$ be a constant. Then $\mathrm{sopt}_\delta(\mathtt{round}(I)) < T_k^{d-1}(1+\delta)\,\mathrm{opt}_{d\mathrm{BP}}(I) + \lceil 1/\delta^2 \rceil + 1 + \delta$.*

**Proof.**

$$a(\widehat{I}_L) \le a(\widehat{I}) = \sum_{i \in I} \left( \ell_d(i) \prod_{j=1}^{d-1} f_k(\ell_j(i)) \right) \le T_k^{d-1} \mathrm{opt}_{d\mathrm{BP}}(I). \qquad \text{(by Theorem 3)}$$

$$\begin{aligned} \mathrm{sopt}_\delta(\widehat{I}) &< \mathrm{opt}(J^{(\mathrm{lo})} \cup \widehat{I}_S) + \delta a(\widehat{I}_L) + (1+\delta) && \text{(by Lemma 28)} \\ &\le \mathrm{opt}(\mathrm{LP}(\widehat{I})) + \left\lceil \frac{1}{\delta^2} \right\rceil + \delta T_k^{d-1} \mathrm{opt}_{d\mathrm{BP}}(I) + (1+\delta) && \text{(by Lemmas 25 and 29)} \\ &\le T_k^{d-1}(1+\delta)\,\mathrm{opt}_{d\mathrm{BP}}(I) + \left\lceil \frac{1}{\delta^2} \right\rceil + 1 + \delta. && \text{(by Lemma 33)} \end{aligned}$$

◀

## References


1 Mauro Maria Baldi, Guido Perboli, and Roberto Tadei. The three-dimensional knapsack problem with balancing constraints. *Applied Mathematics and Computation*, 218(19):9802–9818, 2012. doi:10.1016/j.amc.2012.03.052.

2 János Balogh, József Békési, György Dósa, Leah Epstein, and Asaf Levin. A new and improved algorithm for online bin packing. In *European Symposium on Algorithms (ESA)*, pages 5:1–5:14, 2018. doi:10.4230/LIPIcs.ESA.2018.5.

3 Nikhil Bansal, Alberto Caprara, and Maxim Sviridenko. A new approximation method for set covering problems, with applications to multidimensional bin packing. *SIAM Journal on Computing*, 39(4):1256–1278, 2010. doi:10.1137/080736831.

4 Nikhil Bansal, Xin Han, Kazuo Iwama, Maxim Sviridenko, and Guochuan Zhang. A harmonic algorithm for the 3d strip packing problem. *SIAM Journal on Computing*, 42(2):579–592, 2013. doi:10.1137/070691607.

5 Nikhil Bansal and Arindam Khan. Improved approximation algorithm for two-dimensional bin packing. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 13–25, 2014. doi:10.1137/1.9781611973402.2.

6 Nikhil Bansal and Maxim Sviridenko. New approximability and inapproximability results for 2-dimensional bin packing. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 196–203, 2004.

7 Andreas Bortfeldt and Gerhard Wäscher. Constraints in container loading–a state-of-the-art review. *European Journal of Operational Research*, 229(1):1–20, 2013. doi:10.1016/j.ejor.2012.12.006.

8 Alberto Caprara. Packing $d$-dimensional bins in $d$ stages. *Mathematics of Operations Research*, 33:203–215, February 2008. doi:10.1287/moor.1070.0289.

9 Edward G. Coffman, János Csirik, Gábor Galambos, Silvano Martello, and Daniele Vigo. Bin packing approximation algorithms: survey and classification. In *Handbook of combinatorial optimization*, pages 455–531. Springer New York, 2013.

10 Edward G. Coffman, Michael R. Garey, David S. Johnson, and Robert E. Tarjan. Performance bounds for level-oriented two-dimensional packing algorithms. *SIAM Journal on Computing*, 9:808–826, 1980. doi:10.1137/0209062.

11 János Csirik and André van Vliet. An on-line algorithm for multidimensional bin packing. *Operations Research Letters*, 13(3):149–158, 1993. doi:10.1016/0167-6377(93)90004-Z.

12 Leah Epstein and Rob van Stee. Optimal online algorithms for multidimensional packing problems. *SIAM Journal on Computing*, 35(2):431–448, 2005. doi:10.1137/S0097539705446895.

13 Leah Epstein and Rob van Stee. This side up! *ACM Transactions on Algorithms (TALG)*, 2(2):228–243, 2006. doi:10.1145/1150334.1150339.

14 Sándor P. Fekete and Jörg Schepers. A general framework for bounds for higher-dimensional orthogonal packing problems. *Mathematical Methods of Operations Research*, 60(2):311–329, 2004. doi:10.1007/s001860400376.

15 Wenceslas Fernandez de la Vega and George S. Lueker. Bin packing can be solved within $1+\varepsilon$ in linear time. *Combinatorica*, 1(4):349–355, 1981. doi:10.1007/BF02579456.

16 Paul C. Gilmore and Ralph E. Gomory. A linear programming approach to the cutting-stock problem. *Operations Research*, 9(6):849–859, 1961. doi:10.1287/opre.9.6.849.

17 Xin Han, Francis YL Chin, Hing-Fung Ting, Guochuan Zhang, and Yong Zhang. A new upper bound 2.5545 on 2D online bin packing. *ACM Transactions on Algorithms (TALG)*, 7(4):1–18, 2011. doi:10.1145/2000807.2000818.

18 Klaus Jansen. A $(3/2+\varepsilon)$ approximation algorithm for scheduling moldable and non-moldable parallel tasks. In *Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 224–235, 2012. doi:10.1145/2312005.2312048.

19 Klaus Jansen and Lars Prädel. New approximability results for two-dimensional bin packing. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 919–936, 2013. doi:10.1007/s00453-014-9943-z.

20 Klaus Jansen and Lars Prädel. A new asymptotic approximation algorithm for 3-dimensional strip packing. In *SOFSEM*, pages 327–338, 2014. doi:10.1007/978-3-319-04298-5_29.

21 Klaus Jansen and Rob van Stee. On strip packing with rotations. In *Symposium on Theory of Computing (STOC)*, pages 755–761. ACM, 2005. doi:10.1145/1060590.1060702.

22 Claire Kenyon and Eric Rémila. Approximate strip packing. In *Foundations of Computer Science (FOCS)*, pages 31–36, 1996. doi:10.1109/SFCS.1996.548461.

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

24 C. C. Lee and D. T. Lee. A simple on-line bin-packing algorithm. *Journal of the ACM*, 32(3):562–572, July 1985. doi:10.1145/3828.3833.

25 Flavio Keidi Miyazawa and Yoshiko Wakabayashi. Three-dimensional packings with rotations. *Computers & Operations Research*, 36(10):2801–2815, 2009. doi:10.1016/j.cor.2008.12.015.

26 James Munkres. Algorithms for the assignment and transportation problems. *Journal of the Society for Industrial and Applied Mathematics*, 5(1):32–38, 1957.

27 Boaz Patt-Shamir and Dror Rawitz. Vector bin packing with multiple-choice. *Discrete Applied Mathematics*, 160(10-11):1591–1600, 2012. doi:10.1016/j.dam.2012.02.020.

28 Prakash Ramanan, Donna J Brown, Chung-Chieh Lee, and Der-Tsai Lee. On-line bin packing in linear time. *Journal of Algorithms*, 10(3):305–326, 1989. doi:10.1016/0196-6774(89)90031-X.

29 Steven S Seiden. On the online bin packing problem. *Journal of the ACM*, 49(5):640–671, 2002. doi:10.1145/585265.585269.

30 Eklavya Sharma. Harmonic algorithms for packing $d$-dimensional cuboids into bins. *ArXiv*, 2011.10963, 2020. arXiv:2011.10963.

31 Y. G. Stoyan and Andrey M. Chugay. Packing different cuboids with rotations and spheres into a cuboid. *Advances in Decision Sciences*, 2014, 2014. doi:10.1155/2014/571743.

32 Hu Zhang and Klaus Jansen. Scheduling malleable tasks. In *Handbook of Approximation Algorithms and Metaheuristics*. Chapman & Hall/CRC, 2007.


# A Details of the $\mathtt{HGaP}_k$ Algorithm

## A.1 Details of the Weighting Function from $\mathrm{DLP}(\widehat{I})$

▶ **Transformation 34.** *Let $(y, z)$ be a feasible solution to $\mathrm{DLP}(\widehat{I})$ (see Section 5.5.3 for the definition of $\mathrm{DLP}(\widehat{I})$). Let $s \in [t-1]$. Define $y_t := 0$ and $h_{t+1} := 0$. Then change $y_s$ to $\max(y_s, y_{s+1} + (h_{s+1} - h_{s+2})z)$.*

▶ **Lemma 35.** *Let $(y, z)$ be a feasible solution to $\mathrm{DLP}(\widehat{I})$ and $(\widehat{y}, z)$ be the result of applying Transformation 34 to $(y, z)$ with parameter $s \in [t-1]$. Then $(\widehat{y}, z)$ is feasible for $\mathrm{DLP}(\widehat{I})$.*

**Proof.** For a configuration $C$, let $f(C, y, z) := C^T y + (1 - h_C)z$, where $C^T y := \sum_{j=1}^{t-1} C_j y_j$. Since $(y, z)$ is feasible for $\mathrm{DLP}(\widehat{I})$, $f(C, y, z) \leq 1$. As per Transformation 34,

$$\widehat{y}_j := \begin{cases} \max(y_s, y_{s+1} + (h_{s+1} - h_{s+2})z) & j = s \\ y_j & j \neq s \end{cases}.$$

If $y_s \geq y_{s+1} + (h_{s+1} - h_{s+2})z$, then $\widehat{y} = y$, so $(\widehat{y}, z)$ would be feasible for $\mathrm{DLP}(\widehat{I})$. So now assume that $y_s < y_{s+1} + (h_{s+1} - h_{s+2})z$.

Let $C$ be a configuration. Define $C_t := 0$. Let

$$\widehat{C}_j := \begin{cases} 0 & j = s \\ C_s + C_{s+1} & j = s+1 \\ C_j & \text{otherwise} \end{cases}.$$

Then, $C^T\widehat{y} - \widehat{C}^T y = C_s\widehat{y}_s + C_{s+1}\widehat{y}_{s+1} - \widehat{C}_s y_s - \widehat{C}_{s+1}y_{s+1} = C_s(h_{s+1} - h_{s+2})z$.

Also, $h_{\widehat{C}} - h_C = \widehat{C}_s h_{s+1} + \widehat{C}_{s+1}h_{s+2} - C_s h_{s+1} - C_{s+1}h_{s+2} = -C_s(h_{s+1} - h_{s+2})$.

Since $h_{\widehat{C}} \le h_C \le 1$, $\widehat{C}$ is a configuration.

$$\begin{aligned} f(C, \widehat{y}, z) &= C^T\widehat{y} + (1 - h_C)z \\ &= (\widehat{C}^T y + C_s(h_{s+1} - h_{s+2})z) + (1 - h_{\widehat{C}} - C_s(h_{s+1} - h_{s+2}))z \\ &= f(\widehat{C}, y, z) \le 1. \end{aligned}$$

Therefore, $(\widehat{y}, z)$ is feasible for DLP$(\widehat{I})$. ◀

▶ **Lemma 31.** *Let $(y, z)$ be a feasible solution to* DLP$(\widehat{I})$*. Let $(\widehat{y}, z)$ be the monotonization of $(y, z)$. Then $(\widehat{y}, z)$ is a feasible solution to* DLP$(\widehat{I})$*.*

**Proof.** $(\widehat{y}, z)$ can be obtained by multiple applications of Transformation 34: first with $s = t-1$, then $s = t-2$, and so on till $s = 1$. By Lemma 35, $(\widehat{y}, z)$ is feasible for DLP$(\widehat{I})$. ◀

▶ **Lemma 32.** *$\eta$ is a monotonic weighting function.*

**Proof.** $\eta$ is monotonic by the definition of monotonization.

Let $X \subseteq (0, 1]$ be a finite set such that $\text{sum}(X) \le 1$. Let $X_0 := X \cap [0, h_t)$, let $X_1 := X \cap [h_2, 1]$ and for $2 \le j \le t-1$, let $X_j := X \cap [h_{j+1}, h_j)$. Let $C \in \mathbb{Z}_{\ge 0}^{t-1}$ such that $C_j := |X_j|$. Let $h_C := \sum_{j=1}^{t-1} C_j h_{j+1}$.

$$\begin{aligned} 1 \ge \text{sum}(X) &= \text{sum}(X_0) + \sum_{j=1}^{t-1} \text{sum}(X_j) \\ &\ge \text{sum}(X_0) + \sum_{j=1}^{t-1} C_j h_{j+1} \qquad \text{(for } j \ge 1\text{, each element in } X_j \text{ is at least } h_{j+1}\text{)} \\ &= \text{sum}(X_0) + h_C. \end{aligned}$$

Since $h_C \le 1 - \text{sum}(X_0) \le 1$, $C$ is a configuration. Therefore,

$$\begin{aligned} \sum_{x \in X} \eta(x) = \sum_{j=0}^{t-1} \sum_{x \in X_j} \eta(x) &= z^* \,\text{sum}(X_0) + \sum_{j=1}^{t-1} C_j \widehat{y}_j && \text{(by definition of } \eta\text{)} \\ &\le (1 - h_C)z^* + C^T\widehat{y} && (h_C \le 1 - \text{sum}(X_0)) \\ &\le 1. && (C \text{ is a configuration and } (\widehat{y}, z^*) \text{ is feasible for DLP}(\widehat{I}) \text{ by Lemma 31}) \end{aligned}$$

◀

## A.2 Guessing Shelves and Bins

We want `guessShelves`$(\widehat{\mathcal{I}}, \delta)$ to return all possible packings of empty shelves into at most $n := |\widehat{\mathcal{I}}|$ bins such that each packing is structured for $(\text{flat}(\widehat{\mathcal{I}}), \delta)$.

Let $H := \{h(i) : i \in \text{flat}(\widehat{\mathcal{I}})\}$. Let $N := |\,\text{flat}(\widehat{\mathcal{I}})|$. `guessShelves`$(\widehat{\mathcal{I}}, \delta)$ starts by picking the distinct heights of shelves by iterating over all subsets of $H$ of size at most $\lceil 1/\delta^2 \rceil$. There are at most $N^{\lceil 1/\delta^2 \rceil} + 1$ such subsets. Let $\widetilde{H} := \{h_1, h_2, \ldots, h_t\}$ be one such guess, where $t \le \lceil 1/\delta^2 \rceil$. Without loss of generality, assume $h_1 > h_2 > \ldots > h_t > \delta$.

Next, `guessShelves` needs to decide the number of shelves of each height and a packing of those shelves into bins. Let $C \in \mathbb{Z}_{\ge 0}^{t}$ such that $h_C := \sum_{j=1}^{t-1} C_j h_j \le 1$. Then $C$ is called

a configuration. $C$ represents a set of shelves that can be packed into a bin and where $C_j$ shelves have height $h_j$. Let $\mathcal{C}$ be the set of all configurations. We can pack at most $\lceil 1/\delta \rceil - 1$ items into a bin because $h_t > \delta$. By Lemma 1, we get

$$|\mathcal{C}| \leq \binom{\lceil 1/\delta \rceil - 1 + t}{t} \leq \binom{\lceil 1/\delta \rceil - 1 + \lceil 1/\delta^2 \rceil}{\lceil 1/\delta \rceil - 1} \leq \left( \left\lceil \frac{1}{\delta^2} \right\rceil + 1 \right)^{1/\delta}.$$

There can be at most $n$ bins, and `guessShelves` has to decide the configuration of each bin. By Lemma 1, the number of ways of doing this is at most $\binom{|\mathcal{C}|+n}{|\mathcal{C}|} \leq (n+1)^{|\mathcal{C}|}$. Therefore, `guessShelves` computes all configurations and then iterates over all $\binom{|\mathcal{C}|+n}{|\mathcal{C}|}$ combinations of these configs. This completes the description of `guessShelves` and proves Theorem 13.

## A.3 `chooseAndPack`

`chooseAndPack`$(\widehat{\mathcal{I}}, P, \delta)$ takes as input a set $\widehat{\mathcal{I}}$ of 2D itemsets, a packing $P$ of empty shelves into bins and constant $\delta \in (0,1)$. It tries to pack $\widehat{\mathcal{I}}$ into $P$ and one additional shelf. Before we design `chooseAndPack`, let us see how to handle a special case. $\widehat{\mathcal{I}}$ is called $\delta$-*simple* iff the width of each $\delta$-large item in flat$(\widehat{\mathcal{I}})$ is a multiple of $1/|\widehat{\mathcal{I}}|$.

Let $P$ be a bin packing of empty shelves. Let $h_1 > h_2 > \ldots > h_t$ be the distinct heights of the shelves in $P$, where $h_t > \delta$. We will use dynamic programming to either pack a simple instance $\widehat{\mathcal{I}}$ into $P$ or claim that no assortment of $\widehat{\mathcal{I}}$ can be packed into $P$. Call this algorithm `simpleChooseAndPack`$(\widehat{\mathcal{I}}, P, \delta)$.

Let $\widehat{\mathcal{I}} := \{I_1, I_2, \ldots, I_n\}$. For $j \in \{0, 1, \ldots, n\}$, define $\widehat{\mathcal{I}}_j := \{I_1, I_2, \ldots, I_j\}$, i.e., $\widehat{\mathcal{I}}_j$ contains the first $j$ itemsets from $\widehat{\mathcal{I}}$. Let $\vec{u} := [u_1, u_2, \ldots, u_t] \in \{0, 1, \ldots, n^2\}^t$ be a vector. Let $\Phi(j, \vec{u})$ be the set of all assortments of $\widehat{\mathcal{I}}_j$ that can be packed into $t$ shelves, where the $r^{\text{th}}$ shelf has height $h_r$ and width $u_r/n$. For a set $K$ of items, define smallArea$(K)$ as the total area of $\delta$-small items in $K$. Define $g(j, \vec{u}) := \min_{K \in \Phi(j,\vec{u})} \text{smallArea}(K)$. If $\Phi(j, \vec{u}) = \emptyset$, then we let $g(j, \vec{u}) = \infty$.

We will show how to compute $g(j, \vec{u})$ for all $j \in \{0, 1, \ldots, n\}$ and all $\vec{u} \in \{0, 1, \ldots, n^2\}^t$ using dynamic programming. Let there be $n_r$ shelves in $P$ having height $h_r$. Then for $j = n$ and $u_r = n_r n$, $\widehat{\mathcal{I}}$ can be packed into $P$ iff $g(j, \vec{u})$ is at most the area of non-shelf space in $P$.

Note that in any solution $K$ corresponding to $g(j, \vec{u})$, we can assume without loss of generality that the item $i$ from $K \cap I_j$ is placed in the smallest shelves possible. This is because we can always swap $i$ with the slices of items in those shelves. This observation gives us the following recurrence relation for $g(j, \vec{u})$:

$$g(j, \vec{u}) = \begin{cases} \infty & \text{if } u_j < 0 \text{ for some } j \in [t] \\ 0 & \text{if } n = 0 \text{ and } u_j \geq 0 \text{ for all } j \in [t] \\ \min_{i \in I_j} \begin{pmatrix} \text{smallArea}(\{i\}) \\ + g(j-1, \text{reduce}(\vec{u}, i)) \end{pmatrix} & \text{if } n > 0 \text{ and } u_j \geq 0 \text{ for all } j \in [t] \end{cases} \tag{1}$$

Here reduce$(\vec{u}, i)$ is a vector obtained as follows: If $i$ is $\delta$-small, then reduce$(\vec{u}, i) := \vec{u}$. Otherwise, initialize $x$ to $w(i)$. Let $p_i$ be the largest integer $r$ such that $h(i) \leq h_r$. For $r$ varying from $p_i$ to 2, subtract $\min(x, u_j)$ from $x$ and $u_j$. Then subtract $x$ from $u_1$. The new value of $\vec{u}$ is defined to be the output of reduce$(\vec{u}, i)$.

The recurrence relation allows us to compute $g(j, \vec{u})$ for all $j$ and $\vec{u}$ using dynamic programming in time $O(Nn^{2t})$ time, where $N := |\text{flat}(\widehat{\mathcal{I}})|$. With a bit more work, we can also compute the corresponding assortment $K$, if one exists. Therefore, `simpleChooseAndPack`$(\widehat{\mathcal{I}}, P, \delta)$ computes a packing of $\widehat{\mathcal{I}}$ into $P$ if one exists, or returns `null` if no assortment of $\widehat{\mathcal{I}}$ can be packed into $P$.

Now we will look at the case where $\widehat{\mathcal{I}}$ is not $\delta$-simple. Let $\widehat{\mathcal{I}}'$ be the instance obtained by rounding up the width of each $\delta$-large item in $\widehat{\mathcal{I}}$ to a multiple of $1/n$, where $n := |\widehat{\mathcal{I}}|$. Let $\overline{P}$ be the bin packing obtained by adding another bin to $P$ containing a single shelf of height $h_1$. `chooseAndPack`$(\widehat{\mathcal{I}}, P, \delta)$ computes $\widehat{\mathcal{I}}'$ and $\overline{P}$ and returns the output of `simpleChooseAndPack`$(\widehat{\mathcal{I}}', \overline{P}, \delta)$.

▶ **Theorem 15.** *If the output of* `chooseAndPack`$(\widehat{\mathcal{I}}, P, \delta)$ *is not* `null`*, then the output* $\overline{P}$ *is a shelf-based* $\delta$*-fractional packing of some assortment of* $\widehat{\mathcal{I}}$ *such that* $|\overline{P}| \leq |P| + 1$ *and the distinct shelf heights in* $\overline{P}$ *are the same as that in* $P$*.*

**Proof.** Follows from the definition of `simpleChooseAndPack`. ◀

▶ **Theorem 14.** *If there exists an assortment* $\widehat{K}$ *of* $\widehat{\mathcal{I}}$ *having a structured* $\delta$*-fractional bin packing* $P$*, then* `chooseAndPack`$(\widehat{\mathcal{I}}, P, \delta)$ *does not output* `null`*.*

**Proof.** Let $\widehat{K}'$ be the items obtained by rounding up the width of each item in $\widehat{K}$ to a multiple of $1/n$. Then $\widehat{K}'$ is an assortment of $\widehat{\mathcal{I}}'$. We will show that $\widehat{K}'$ fits into $\overline{P}$, so `simpleChooseAndPack`$(\widehat{\mathcal{I}}', \overline{P}, \delta)$ will not output `null`.

Slice each item $i \in \widehat{K}'$ into two pieces using a vertical cut such that one piece has width equal to the original width of $i$ in $\widehat{K}$, and the other piece has width less than $1/n$. This splits $\widehat{K}'$ into sets $\widehat{K}$ and $T$. $T$ contains at most $n$ items, each of width less than $1/n$. Therefore, we can pack $\widehat{K}$ into $P$ and we can pack $T$ into the newly-created shelf of height $h_1$. Therefore, $\widehat{K}'$ can be packed into $\overline{P}$, so `simpleChooseAndPack`$(\widehat{\mathcal{I}}', \overline{P}, \delta)$ won't output `null`. ◀

▶ **Theorem 16.** `chooseAndPack`$(\widehat{\mathcal{I}}, P, \delta)$ *runs in* $O(Nn^{2^{\lceil 1/\delta^2 \rceil}})$ *time. Here* $N := |\operatorname{flat}(\widehat{\mathcal{I}})|$*,* $n := |\widehat{\mathcal{I}}|$*.*

**Proof.** The running time of `chooseAndPack`$(\widehat{\mathcal{I}}, P, \delta)$ is dominated by computing $g(j, \vec{u})$ for all $j$ and $\vec{u}$, which takes $O(Nn^{2t})$ time. Since $P$ is structured for $(\widehat{\mathcal{I}}, \delta)$, the number of distinct shelves in $P$, which is $t$, is at most $\lceil 1/\delta^2 \rceil$. ◀

## A.4 `inflate`

Let $I$ be a set of $d$D items. Let $P$ be a shelf-based $\delta$-fractional bin packing of $\widehat{I} :=$ `round`$(I)$ into $m$ bins, where the shelves have $t$ distinct heights: $h_1 > \ldots > h_t > \delta$. We will design an algorithm `inflate`$(P)$ that packs $I$ into approximately $|P|$ bins. Let $\widehat{I}_L := \{i \in \widehat{I} : h(i) > \delta\}$ and $\widehat{I}_S := \widehat{I} - \widehat{I}_L$. Let there be $Q$ distinct base types in $I$ (so $Q \leq k^{d-1}$).

### A.4.1 Separating Base Types

We will now impose an additional constraint over $P$: items in each shelf must have the same btype. This will be helpful later, when we will try to compute a packing of $d$D items $I$.

Separating base types of $\widehat{I}_S$ is easy, since we can slice them in both directions. An analogy is to think of a mixture of multiple immiscible liquids settling into equilibrium.

Let there be $n_j$ shelves of height $h_j$. Let $\widehat{I}_j$ be the items packed into shelves of height $h_j$. Therefore, $w(\widehat{I}_j) \leq n_j$. Let $\widehat{I}_{j,q} \subseteq \widehat{I}_j$ be the items of base type $q \in [Q]$.

For each $q$, pack $\widehat{I}_{j,q}$ into $\lceil w(\widehat{I}_{j,q}) \rceil$ shelves of height $h_j$ (slicing items if needed). For these newly-created shelves, define the btype of the shelf to be the btype of the items in it. Let the number of newly-created shelves of height $h_j$ be $n'_j$. Then

$$n'_j = \sum_{q=1}^{Q} \lceil w(\widehat{I}_{j,q}) \rceil < \sum_{q=1}^{Q} w(\widehat{I}_{j,q}) + Q \leq n_j + Q \implies n'_j \leq n_j + Q - 1.$$

$n_j$ of these shelves can be packed into existing bins in place of the old shelves. The remaining $n'_j - n_j \leq Q - 1$ shelves can be packed on the base of new bins.

Therefore, by using at most $t(Q-1)$ new bins, we can ensure that for every shelf, all items in that shelf have the same btype. These new bins don't contain any items from $\widehat{I}_S$. Call this new bin packing $P'$. This transformation takes $O(|I|d\log|I|)$ time.

### A.4.2 Forbidding Horizontal Slicing

We will now use $P'$ to compute a shelf-based bin packing $P''$ of $\widehat{I}$ where items in $\widehat{I}$ can be sliced using vertical cuts only.

Let $\widehat{I}_{q,S}$ be the items in $\widehat{I}_S$ of base type $q$. Pack items $\widehat{I}_{q,S}$ into shelves using canShelv. Suppose canShelv used $m_q$ shelves to pack $\widehat{I}_{q,S}$. For $j \in [m_q]$, let $h_{q,j}$ be the height of the $j^{\text{th}}$ shelf. Let $H_q := \sum_{j=1}^{m_q} h_{q,j}$ and $H := \sum_{q=1}^{Q} H_q$. Since for $j \in [m_q - 1]$, all items in the $j^{\text{th}}$ shelf have height at least $h_{q,j+1}$,

$$a(\widehat{I}_{q,S}) > \sum_{j=1}^{m_q-1} h_{q,j+1} \geq H_q - h_{q,1} \geq H_q - \delta.$$

Therefore, $H < a(\widehat{I}_S) + Q\delta$. Let $\widehat{J}_S$ be the set of these newly-created shelves.

Use Next-Fit to pack $\widehat{J}_S$ into the space used by $\widehat{I}_S$ in $P'$. $\widehat{I}_S$ uses at most $m$ bins in $P'$ (recall that $m := |P|$). A height of less than $\delta$ will remain unpacked in each of those bins. The total height occupied by $\widehat{I}_S$ in $P'$ is $a(\widehat{I}_S)$. Therefore, Next-Fit will pack a height of more than $a(\widehat{I}_S) - \delta m$.

Some shelves in $\widehat{J}_S$ may still be unpacked. Their total height will be less than $H - (a(\widehat{I}_S) - \delta m) < \delta(Q+m)$. We will pack these shelves into new bins using Next-Fit. The number of new bins used is at most $\lceil \delta(Q+m)/(1-\delta) \rceil$. Call this bin packing $P''$. The number of bins in $P''$ is at most $m' := m + t(Q-1) + \lceil \delta(Q+m)/(1-\delta) \rceil$.

### A.4.3 Shelf-Based $d$D packing

We will now show how to convert the packing $P''$ of $\widehat{I}$ that uses $m'$ bins into a packing of $I$ that uses $m'$ $d$D bins.

First, we repack the items into the shelves. For each $q \in [Q]$, let $\widehat{J}_q$ be the set of shelves in $P''$ of btype $q$. Let $\widehat{I}^{[q]}$ be the items packed into $\widehat{J}_q$. Compute $\widehat{J}^*_q := \text{canShelv}(\widehat{I}^{[q]})$ and pack the shelves $\widehat{J}^*_q$ into $\widehat{J}_q$. This is possible by Lemma 23.

This repacking gives us an ordering of shelves in $\widehat{J}_q$. Number the shelves from 1 onwards. All items have at most 2 slices. If an item has 2 slices, and one slice is packed into shelf number $p$, then the other slice is packed into shelf number $p+1$. The slice in shelf $p$ is called the leading slice. Every shelf has at most one leading slice.

Let $S_j$ be the $j^{\text{th}}$ shelf of $\widehat{J}_q$. Let $R_j$ be the set of unsliced items in $S_j$ and the item whose leading slice is in $S_j$. Order the items in $R_j$ arbitrarily, except that the sliced item, if any, should be last. Then $w(R_j - \text{last}(R_j)) < 1$. So, we can use $\texttt{HDH-unit-pack}^{[q]}_k(R_j)$ to pack $R_j$ into a $(d-1)$D bin. This $(d-1)$D bin gives us a $d$D shelf whose height is the same as that of $S_j$. On repeating this process for all shelves in $\widehat{J}_q$ and for all $q \in [Q]$, we get a packing of $I$ into shelves. Since each $d$D shelf corresponds to a shelf in $P''$ of the same height, we can pack these $d$D shelves into bins in the same way as $P''$. This gives us a bin packing of $I$ into $m'$ bins.

### A.4.4 The Algorithm

Appendices A.4.1–A.4.3 describe how to convert a shelf-based $\delta$-fractional packing $P$ of $\widehat{I}$ having $t$ distinct shelf heights into a shelf-based $d$D bin packing of $I$. We call this conversion algorithm `inflate`.

It is easy to see that the time taken by `inflate` is $O(|I|d \log |I|)$.

If $P$ has $m$ bins, then the number of bins in `inflate`$(P)$ is at most

$$m + t(Q-1) + \left\lceil \frac{\delta(Q+m)}{1-\delta} \right\rceil < \frac{m}{1-\delta} + t(Q-1) + 1 + \frac{\delta Q}{1-\delta}.$$

This proves Theorem 17.

## A.5 Improving Running Time

For simplicity of presentation, we left out some opportunities for improving the running time of $\texttt{HGaP}_k$. Here we briefly describe a way of speeding up $\texttt{HGaP}_k$ which reduces its running time from $O(N^{1+\lceil 1/\delta^2 \rceil} n^{R+2\lceil 1/\delta^2 \rceil} + Nd + nd \log n)$ to $O(N^{1+\lceil 1/\delta^2 \rceil} n^{2\lceil 1/\delta^2 \rceil} + Nd + nd \log n)$. Here $N := |\operatorname{flat}(\widehat{\mathcal{I}})|$, $n := |\widehat{\mathcal{I}}|$, $\delta := \varepsilon/(2+\varepsilon)$ and $R := \binom{\lceil 1/\delta^2 \rceil + \lceil 1/\delta \rceil - 1}{\lceil 1/\delta \rceil - 1} \leq (1 + \lceil 1/\delta^2 \rceil)^{1/\delta}$.

In `guessShelves`, we guess two things simultaneously: (i) the number and heights of shelves (ii) the packing of the shelves into bins. This allows us to guess the optimal structured $\delta$-fractional packing. But we don't need that; an approximate structured packing would do.

Therefore, we only guess the number and heights of shelves. We guess at most $N^{\lceil 1/\delta^2 \rceil} + 1$ distinct heights of shelves, and by Lemma 1, we guess at most $(n+1)^{\lceil 1/\delta^2 \rceil}$ vectors of shelf-height frequencies. Then we can use Lueker and Fernandez de la Vega's $O(n \log n)$-time APTAS for 1BP [15] to pack the shelves into bins.

Also, once we guess the distinct heights of shelves, we don't need to run `chooseAndPack` afresh for every packing of empty shelves. We can reuse the dynamic programming table.

The running time is, therefore,

$$\begin{aligned} &O\left(N^{\lceil 1/\delta^2 \rceil}\left(n^{\lceil 1/\delta^2 \rceil} n \log n + N n^{2\lceil 1/\delta^2 \rceil}\right) + Nd + nd \log n\right) \\ &= O(N^{1+\lceil 1/\delta^2 \rceil} n^{2\lceil 1/\delta^2 \rceil} + Nd + nd \log n). \end{aligned}$$