Tight Approximation Algorithms For Geometric Bin Packing with Skewed Items

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

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

Abstract

In the *Two-dimensional Bin Packing (2BP)* problem, we are given a set of rectangles of height and width at most one and our goal is to find an axis-aligned nonoverlapping packing of these rectangles into the minimum number of unit square bins. The problem admits no APTAS and the current best approximation ratio is 1.406 by Bansal and Khan [SODA'14]. A well-studied variant of the problem is *Guillotine Two-dimensional Bin Packing (G2BP)*, where all rectangles must be packed in such a way that every rectangle in the packing can be obtained by recursively applying a sequence of end-to-end axis-parallel cuts, also called *guillotine cuts*. Bansal, Lodi, and Sviridenko [FOCS'05] obtained an APTAS for this problem. Let λ be the smallest constant such that for every set I of items, the number of bins in the optimal solution to G2BP for I is upper bounded by $\lambda \operatorname{opt}(I) + c$, where $\operatorname{opt}(I)$ is the number of bins in the optimal solution to 2BP for I and c is a constant. It is known that $4/3 \leq \lambda \leq 1.692$. Bansal and Khan [SODA'14] conjectured that $\lambda = 4/3$. The conjecture, if true, will imply a $(4/3+\varepsilon)$-approximation algorithm for 2BP. According to convention, for a given constant $\delta > 0$, a rectangle is *large* if both its height and width are at least δ, and otherwise it is called *skewed*. We make progress towards the conjecture by showing $\lambda = 4/3$ for *skewed instance*, i.e., when all input rectangles are skewed. Even for this case, the previous best upper bound on λ was roughly 1.692. We also give an APTAS for 2BP for skewed instance, though general 2BP does not admit an APTAS.

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

Keywords and phrases Geometric bin packing, guillotine separability, approximation algorithms

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2021.22

Category APPROX

Related Version *ArXiv*: https://arxiv.org/abs/2105.02827

1 Introduction

Two-dimensional Bin Packing (2BP) is a well-studied problem in combinatorial optimization. It finds numerous applications in logistics, databases, and cutting stock. In 2BP, we are given a set of n rectangular items and square bins of side length 1. The i^{th} item is characterized by its width $w(i) \in (0, 1]$ and height $h(i) \in (0, 1]$. Our goal is to find an axis-aligned nonoverlapping packing of these items into the minimum number of square bins of side length 1. There are two well-studied variants: (i) where the items cannot be rotated, and (ii) they can be rotated by 90 degrees.

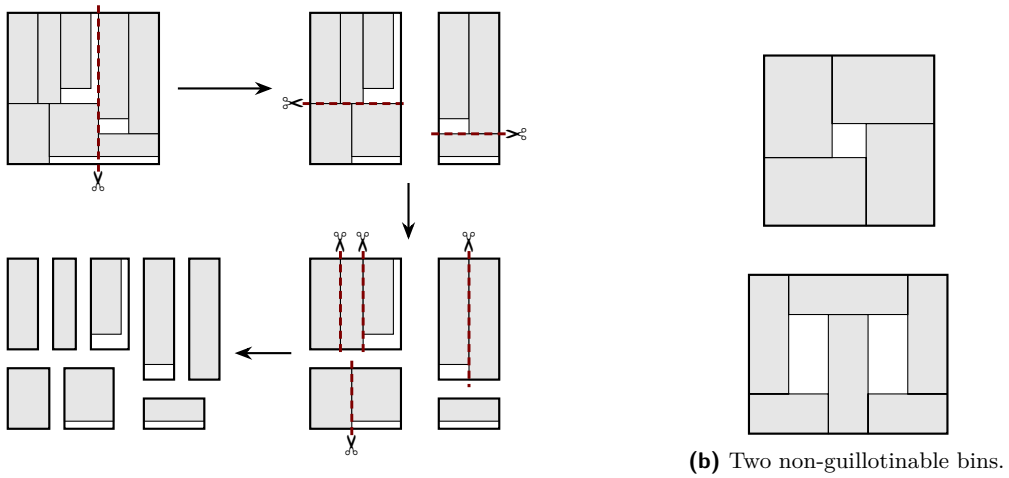
As is conventional in bin packing, we focus on asymptotic approximation algorithms. For any optimization problem, the asymptotic approximation ratio (AAR) of algorithm $\mathcal{A}$ is defined as $\lim_{m\to\infty} \sup_{I:\operatorname{opt}(I)=m}(\mathcal{A}(I)/\operatorname{opt}(I))$, where $\operatorname{opt}(I)$ is the optimal objective value and $\mathcal{A}(I)$ is the objective value of the solution output by algorithm $\mathcal{A}$, respectively, on input I. Intuitively, AAR captures the algorithm's behavior when $\operatorname{opt}(I)$ is large. We call a


© Arindam Khan and Eklavya Sharma;
licensed under Creative Commons License CC-BY 4.0

bin packing algorithm α-asymptotic-approximate iff its AAR is at most α. An Asymptotic Polynomial-Time Approximation Scheme (APTAS) is an algorithm that accepts a parameter ε and has AAR of $(1+\varepsilon)$.

2BP is a generalization of classical 1-D bin packing problem [24, 15]. However, unlike 1-D bin packing, 2BP does not admit an APTAS unless P=NP [5]. In 1982, Chung, Garey, and Johnson [13] gave an approximation algorithm with AAR 2.125 for 2BP. Caprara [9] improved the AAR to $T_\infty (\approx 1.691)$. After a series of works [4, 25, 7], the AAR was gradually improved to $1 + \ln(1.5) + \varepsilon \approx 1.405$ (for both the rotational and non-rotational versions of 2BP). The best-known lower bounds on the AAR for 2BP are 1 + 1/3792 and 1 + 1/2196 [11], for the versions with and without rotations, respectively.

In the context of geometric packing, guillotine cuts are well-studied and heavily used in practice [41]. The notions of *guillotine cuts* and *k-stage packing* were introduced by Gilmore and Gomory in their seminal paper [22] on the cutting stock problem. In k-stage packing, the items can be separated from each other using k stages of axis-parallel end-to-end cuts, also called guillotine cuts, where in each stage, either all cuts are vertical or all cuts are horizontal. In each stage, each rectangular region obtained in the previous stage is considered separately and can be cut again using guillotine cuts. Note that in the cutting process we change the orientation (vertical or horizontal) of the cuts $k-1$ times. 2-stage packing, also called *shelf packing*, has been studied extensively. A packing is called *guillotinable* iff it is a k-stage packing for some integer k. See Figure 1 for examples. Caprara et al. [10] gave an APTAS for 2-stage 2BP. Bansal et al. [8] showed an APTAS for guillotine 2BP.



(a) The items can be separated using 3 stages of guillotine cuts, so this is a guillotinable packing.

(b) Two non-guillotinable bins.

Figure 1 Examples of guillotinable and non-guillotinable packing.

The presence of an APTAS for guillotine 2BP raises an important question: can the optimal solution to guillotine 2BP be used as a good approximate solution to 2BP? Formally, let $\mathrm{opt}(I)$ and $\mathrm{opt}_g(I)$ be the minimum number of bins and the minimum number of guillotinable bins, respectively, needed to pack items I. Let λ be the smallest constant such that for some constant c and for every set I of items, we get $\mathrm{opt}_g(I) \leq \lambda\,\mathrm{opt}(I) + c$. Then λ is called the Asymptotic Price of Guillotinability (APoG). It is easy to show that

APoG $\geq 4/3$[1]. Bansal and Khan [7] conjectured that APoG $= 4/3$. If true, this would imply a $(4/3 + \varepsilon)$-asymptotic-approximation algorithm for 2BP [8]. However, the present upper bound on APoG is only T_∞ (≈ 1.691), due to Caprara's HDH algorithm [9] for 2BP, which produces a 2-stage packing.

APTASes are known for some special cases for 2BP, such as when all items are squares [5] or when all rectangles are small in both dimensions [14]. Another important class is *skewed* rectangles. We say that a rectangle is δ-large if, for some constant $\delta > 0$, its width and height are more than δ; otherwise, the rectangle is δ-skewed. We just say that a rectangle is large or skewed when δ is clear from the context. An instance of 2BP is skewed if all the rectangles in the input are skewed. Skewed instances are important in geometric packing (see Section 1.1). This special case is practically relevant [18]: e.g., in scheduling, it captures scenarios where no job can consume a significant amount of a shared resource (energy, memory space, etc.) for a significant amount of time. Even for skewed instance for 2BP, the best known AAR is 1.406 [7]. Also, for skewed instance, the best known upper bound on APoG is $T_\infty \approx 1.691$.

1.1 Related Works

Multidimensional packing problems are fundamental in combinatorial optimization [12]. Vector packing (VP) is another variant of bin packing, where the input is a set of vectors in $[0, 1]^d$ and the goal is to partition the vectors into the minimum number of parts (bins) such that in each part, the sum of vectors is at most 1 in every coordinate. The present best approximation algorithm attains an AAR of $(0.807 + \ln(d + 1))$ [6] and there is a matching $\Omega(\ln d)$-hardness [37]. Generalized multidimensional packing [33, 32] generalizes both geometric and vector packing.

In two-dimensional strip packing (2SP) [14, 40], we are given a set of rectangles and a bounded width strip. The goal is to obtain an axis-aligned nonoverlapping packing of all rectangles such that the height of the packing is minimized. The best-known approximation ratio for 2SP is $5/3 + \varepsilon$ [23] and it is NP-hard to obtain better than 3/2-approximation. However, there exist APTASes for the problem [28, 26]. In two-dimensional knapsack (2GK) [27], the rectangles have associated profits and our goal is to pack the maximum profit subset into a unit square knapsack. The present best polynomial-time (resp. pseudopolynomial-time) approximation ratio for 2GK is 1.809 [20] (resp. 4/3 [21]). These geometric packing problems have also been studied for d-dimensions ($d \geq 2$) [39].

2SP and 2GK are also well-studied under guillotine packing. Seiden and Woeginger [38] gave an APTAS for guillotine 2SP. Khan et al. [29] recently gave a pseudopolynomial-time approximation scheme for guillotine 2GK. Recently, guillotine cuts [35] have received attention due to their connection with the maximum independent set of rectangles (MISR) problem [2]. In MISR, we are given a set of possibly overlapping rectangles and the goal is to find the maximum cardinality set of rectangles so that there is no pairwise overlap. It was noted in [30, 1] that for any set of n non-overlapping axis-parallel rectangles, if there is a guillotine cutting sequence separating αn of them, then it implies a $1/\alpha$-approximation for MISR.

Skewed instance is an important special case in these problems. In some problems, such as MISR and 2GK, if all items are δ-large then we can solve them exactly in polynomial time. So, the inherent difficulty of these problems lies in skewed instances. For VP, hard instances are again skewed, e.g., Bansal, Eliáš and Khan [6] showed that hard instances for 2-D VP

[1] Consider a set I of items containing $2m$ rectangles of width 0.6 and height 0.4 and $2m$ rectangles of width 0.4 and height 0.6. Then $\mathrm{opt}(I) = m$ and $\mathrm{opt}_g(I) = \lceil 4m/3 \rceil$.

(for a class of algorithms called *rounding based algorithms*) are skewed instances, where one dimension is $1-\varepsilon$ and the other dimension is ε. Galvez el al. [18] recently studied strip packing when all items are skewed. For skewed instances, they showed $(3/2-\varepsilon)$ hardness of approximation and a matching $(3/2+\varepsilon)$-approximation algorithm. For 2GK, when the height of each item is at most ε^3, a $(1-72\varepsilon)$-approximation algorithm is known [17].

1.2 Our Contributions

We study 2BP for the special case of δ-skewed rectangles, where $\delta \in (0, 1/2]$ is a constant.

First, we make progress towards the conjecture [7] that APoG $= 4/3$. Even for skewed rectangles, we only knew $4/3 \leq \text{APoG} \leq T_\infty (\approx 1.691)$. We resolve the conjecture for skewed rectangles, by giving lower and upper bounds of roughly $4/3$ when δ is a small constant.

Specifically, we give an algorithm for 2BP, called $\texttt{skewed4Pack}_\varepsilon$, that takes a parameter $\varepsilon \in (0, 1)$ as input. For a set I of δ-skewed rectangles, we show that when δ and ε are close to 0, $\texttt{skewed4Pack}_\varepsilon(I)$ outputs a 4-stage packing of I into roughly $4\,\text{opt}(I)/3 + O(1)$ bins.

▶ **Theorem 1.** *Let I be a set of δ-skewed items, where $\delta \in (0, 1/2]$. Then $\texttt{skewed4Pack}_\varepsilon(I)$ outputs a 4-stage packing of I in time $O((1/\varepsilon)^{O(1/\varepsilon)} + n \log n)$. Furthermore, the number of bins used is at most $(4/3)(1+8\delta)(1+7\varepsilon)\,\text{opt}(I) + (8/\varepsilon^2) + 30$.*

The lower bound of $4/3$ on APoG can be extended to skewed items. We formally prove this in Appendix D. Hence, our bounds are tight for skewed items. Our result indicates that to improve the bounds on APoG in the general case, we should focus on δ-large items.

Our other main result is an APTAS for 2BP for skewed items. Formally, we give an algorithm for 2BP, called `skewedCPack`, and show that for every constant $\varepsilon \in (0, 1)$, there exists a constant $\delta \in (0, \varepsilon)$ such that the algorithm has an AAR of $1+\varepsilon$ when all items in the input are δ-skewed rectangles. `skewedCPack` can be extended to the rotational version of 2BP. The best-known AAR for 2BP is $1 + \ln(1.5) + \varepsilon$. Our result indicates that to improve upon algorithms for 2BP, one should focus on δ-large items.

In Section 3, we describe the `skewed4Pack` algorithm and prove Theorem 1. In Section 4, we describe the `skewedCPack` algorithm and prove that it has an AAR of $1+\varepsilon$.

2 Preliminaries

Let $[n] := \{1, 2, \ldots, n\}$, for $n \in \mathbb{N}$. For a rectangle i, its area $a(i) := w(i)h(i)$. For a set I of rectangles, let $a(I) := \sum_{i \in I} a(i)$. An *axis-aligned packing* of an item i in a bin is specified by a pair $(x(i), y(i))$, where $x(i), y(i) \in [0, 1]$, so that i is placed in the region $[x(i), x(i)+w(i)] \times [y(i), y(i)+h(i)]$. A packing of rectangles in a bin is called *nonoverlapping* iff for any two distinct items i and j, the rectangles $(x(i), x(i)+w(i)) \times (y(i), y(i)+h(i))$ and $(x(j), x(j)+w(j)) \times (y(j), y(j)+h(j))$ are disjoint. Equivalently, items may only intersect at their boundaries.

Next-Fit Decreasing Height (NFDH). NFDH [14] is a simple algorithm for 2SP and 2BP. We will use the following results on NFDH (cf. Appendix B in [31]).

▶ **Lemma 2.** *Let I be a set of items where each item i has $w(i) \leq \delta_W$ and $h(i) \leq \delta_H$. NFDH can pack I into a bin of width W and height H if $a(I) \leq (W - \delta_W)(H - \delta_H)$.*

▶ **Lemma 3.** *NFDH uses less than $(2a(I)+1)/(1-\delta)$ bins to pack I when $h(i) \leq \delta$ for each item i and less than $2a(I)/(1-\delta)+3$ bins when $w(i) \leq \delta$ for each item i.*

If we swap the coordinate axes in NFDH, we get the Next-Fit Decreasing Width (NFDW) algorithm. Analogs of the above results hold for NFDW.

Slicing Items. We will consider variants of 2BP where some items can be *sliced*. Formally, slicing a rectangular item i using a horizontal cut is the operation of replacing i by two items i_1 and i_2 such that $w(i) = w(i_1) = w(i_2)$ and $h(i) = h(i_1) + h(i_2)$. Slicing using vertical cut is defined analogously. Allowing some items to be sliced may reduce the number of bins required to pack them.

Variants of 2SP where items can be sliced using vertical cuts find applications in resource allocation problems [3, 16, 19]. Many packing algorithms [28, 25, 8] solve the sliceable version of the problem as a subroutine.

3 Guillotinable Packing of Skewed Rectangles

An item is called (δ_W, δ_H)-skewed iff its width is at most δ_W or its height is at most δ_H. In this section, we consider the problem of obtaining tight upper and lower bounds on APoG for (δ_W, δ_H)-skewed items. We will describe the `skewed4Pack` algorithm and prove Theorem 1.

3.1 Packing With Slicing

Before describing `skewed4Pack`, let us first look at a closely-related variant of this problem, called the *sliceable 2D bin packing problem*, denoted as S2BP. In this problem, we are given two sets of rectangular items, $\widetilde{W}$ and $\widetilde{H}$, where items in $\widetilde{W}$ have width more than $1/2$, and items in $\widetilde{H}$ have height more than $1/2$. $\widetilde{W}$ is called the set of wide items and $\widetilde{H}$ is called the set of tall items. We are allowed to *slice* items in $\widetilde{W}$ using horizontal cuts and slice items in $\widetilde{H}$ using vertical cuts, and our task is to pack $\widetilde{W} \cup \widetilde{H}$ into the minimum number of bins without rotating the items.

We first describe a 4/3-asymptotic-approximation algorithm for S2BP, called `greedyPack`, that outputs a 2-stage packing. Later, we will use `greedyPack` to design `skewed4Pack`.

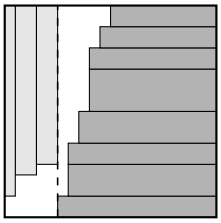
We assume that the bin is a square of side length 1. Since we can slice items, we allow items in $\widetilde{W}$ to have height more than 1 and items in $\widetilde{H}$ to have width more than 1.

For $X \subseteq \widetilde{W}$, $Y \subseteq \widetilde{H}$, let $\text{hsum}(X) := \sum_{i \in X} h(i)$; $\text{wsum}(Y) := \sum_{i \in Y} w(i)$; $\text{wmax}(X) := \max_{i \in X} w(i)$ if $X \neq \emptyset$, and 0 if $X = \emptyset$; $\text{hmax}(Y) := \max_{i \in Y} h(i)$ if $Y \neq \emptyset$, and 0 if $Y = \emptyset$.

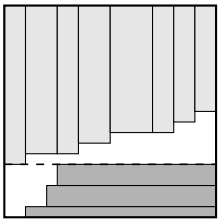
In the algorithm `greedyPack`$(\widetilde{W}, \widetilde{H})$, we first sort items $\widetilde{W}$ in decreasing order of width and sort items $\widetilde{H}$ in decreasing order of height. Suppose $\text{hsum}(\widetilde{W}) \geq \text{wsum}(\widetilde{H})$. Let X be the largest prefix of $\widetilde{W}$ of total height at most 1, i.e., if $\text{hsum}(\widetilde{W}) > 1$, then X is a prefix of $\widetilde{W}$ such that $\text{hsum}(X) = 1$ (slice items if needed), and $X = \widetilde{W}$ otherwise. Pack X into a bin such that the items touch the right edge of the bin. Then we pack the largest possible prefix of $\widetilde{H}$ into the empty rectangular region of width $1 - \text{wmax}(X)$ in the left side of the bin. We call this a type-1 bin. See Figure 2a for an example. If $\text{hsum}(\widetilde{W}) < \text{wsum}(\widetilde{H})$, we proceed analogously in a coordinate-swapped way, i.e., we first pack tall items in the bin and then pack wide items in the remaining space. Call this bin a type-2 bin. We pack the rest of the items into bins in the same way.

▷ Claim 4. `greedyPack`$(\widetilde{W}, \widetilde{H})$ outputs a 2-stage packing of $\widetilde{W} \cup \widetilde{H}$ in $O(m + |\widetilde{W}| \log |\widetilde{W}| + |\widetilde{H}| \log |\widetilde{H}|)$ time, where m is the number of bins used. It slices items in $\widetilde{W}$ by making at most $m - 1$ horizontal cuts and slices items in $\widetilde{H}$ by making at most $m - 1$ vertical cuts.

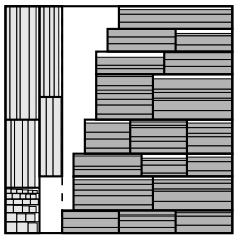
Since items in $\widetilde{W}$ have width more than $1/2$, no two items can be placed side-by-side. Hence, $\lceil \text{hsum}(\widetilde{W}) \rceil = \text{opt}(\widetilde{W}) \leq \text{opt}(\widetilde{W} \cup \widetilde{H})$. Similarly, $\lceil \text{wsum}(\widetilde{H}) \rceil \leq \text{opt}(\widetilde{W} \cup \widetilde{H})$. So, if all bins have the same type, `greedyPack` uses $\max(\lceil \text{hsum}(\widetilde{W}) \rceil, \lceil \text{wsum}(\widetilde{H}) \rceil) = \text{opt}(\widetilde{W} \cup \widetilde{H})$ bins. We will now focus on the case where some bins have type 1 and some have type 2.



(a) A type-1 bin produced by `greedyPack`. Wide items are packed on the right. Tall items are packed on the left.



(b) A type-2 bin produced by `greedyPack`. Tall items are packed above. Wide items are packed below.



(c) A type-1 bin in the packing of $\widehat{I}$ computed by `skewed4Pack` (see Section 3.2). The packing contains 5 tall containers in 2 tall shelves and 18 wide containers in 8 wide shelves.

▶ **Definition 5.** *In a type-1 bin, let X and Y be wide and tall items, respectively. The bin is called* full *iff* $\text{hsum}(X) = 1$ *and* $\text{wsum}(Y) = 1 - \text{wmax}(X)$. *Similarly define* full *for type 2.*

We first show that full bins pack items of large total area, and then show that if some bins have type 1 and some have type 2, then there are at most 2 non-full bins. This helps us upper-bound the number of bins used by $\texttt{greedyPack}(\widetilde{W}, \widetilde{H})$ in terms of $a(\widetilde{W} \cup \widetilde{H})$.

▶ **Lemma 6.** *Let there be m_1 type-1 full bins, containing items J_1. Then $m_1 \leq 4a(J_1)/3 + 1/3$.*

Proof. In the j^{th} full bin of type 1, let X_j be the items from $\widetilde{W}$ and Y_j be the items from $\widetilde{H}$. Let $\ell_j := \text{wmax}(X_j)$ if $j \leq m_1$ and $\ell_{m_1+1} := 1/2$. Since all items have their larger dimension more than $1/2$, $\ell_j \geq 1/2$ and $\text{hmax}(Y_j) > 1/2$, for any $j \in [m_1]$.

$a(X_j) \geq \ell_{j+1}$, since X_j has height 1 and width at least ℓ_{j+1}. $a(Y_j) \geq (1 - \ell_j)/2$, since Y_j has width $1 - \ell_j$ and height more than $1/2$. So, $a(J_1) = \sum_{j=1}^{m_1}(a(X_j) + a(Y_j)) \geq \sum_{j=1}^{m_1}(\ell_{j+1} + (1 - \ell_j)/2) \geq \sum_{j=1}^{m_1}((\ell_{j+1}/2) + (1/4) + (1/2) - (\ell_j/2)) \geq (3m_1 - 1/4)$.
In the above inequalities, we used $\ell_{j+1} \geq 1/2$ and $\ell_1 \leq 1$.

Therefore, $m_1 \leq 4a(J_1)/3 + 1/3$. ◀

An analog of Lemma 6 can be proven for type-2 bins. Lemma 6 implies that very few full bins can have items of total area significantly less than $3/4$.

Suppose $\texttt{greedyPack}(\widetilde{W}, \widetilde{H})$ uses m bins. After j bins are packed, let A_j be the height of the remaining items in $\widetilde{W}$ and B_j be the width of the remaining items in $\widetilde{H}$. Let t_j be the type of the j^{th} bin (1 for type-1 bin, 2 for type-2 bin). So $t_j = 1 \iff A_{j-1} \geq B_{j-1}$.

We first show that $|A_{j-1} - B_{j-1}| \leq 1 \implies |A_j - B_j| \leq 1$, i.e., once $|\,\text{hsum}(\widetilde{W}) - \text{wsum}(\widetilde{H})|$ becomes at most 1 during `greedyPack`, it continues to stay at most 1. Next, we show that $t_j \neq t_{j+1} \implies |A_{j-1} - B_{j-1}| \leq 1$, i.e., if all bins don't have the same type, then $|\,\text{hsum}(\widetilde{W}) - \text{wsum}(\widetilde{H})|$ eventually becomes at most 1 during `greedyPack`. In the first non-full bin, we use up all the wide items or all the tall items. We will show that the remaining items have total height or total width at most 1, so we have at most 2 non-full bins.

In the j^{th} bin, let a_j be the height of items from $\widetilde{W}$ and b_j be the width of items from $\widetilde{H}$. Hence, for all $j \in [m]$, $A_{j-1} = A_j + a_j$ and $B_{j-1} = B_j + b_j$.

▶ **Lemma 7.** $|A_{j-1} - B_{j-1}| \leq 1 \implies |A_j - B_j| \leq 1$.

Proof. W.l.o.g., assume $A_{j-1} \geq B_{j-1}$. So, $t_j = 1$. Suppose $a_j < b_j$. Then $a_j < 1$, so we used up $\widetilde{W}$ in the j^{th} bin. Therefore, $A_j = 0 \implies A_{j-1} = a_j < b_j \leq b_j + B_j = B_{j-1}$, which contradicts. Hence, $a_j \geq b_j$. As $0 \leq (A_{j-1} - B_{j-1}), (a_j - b_j) \leq 1$, we get $A_j - B_j = (A_{j-1} - B_{j-1}) - (a_j - b_j) \in [-1, 1]$. ◀

▶ **Lemma 8.** $t_j \neq t_{j+1} \implies |A_{j-1} - B_{j-1}| \leq 1$.

Proof. W.l.o.g., assume $t_j = 1$ and $t_{j+1} = 2$. Then

$$A_{j-1} \geq B_{j-1} \text{ and } A_j < B_j \implies B_{j-1} \leq A_{j-1} < B_{j-1} + a_j - b_j \implies A_{j-1} - B_{j-1} \in [0, 1).$$ ◀

▶ **Lemma 9.** *If all bins don't have the same type, then there can be at most 2 non-full bins.*

Proof. Let there be p full bins. Assume w.l.o.g. that in the $(p+1)^{\text{th}}$ bin, we used up all items from $\widetilde{W}$ but not $\widetilde{H}$. Hence, $A_{p+1} = 0$ and $\forall i \geq p+2$, $t_i = 2$. Since all bins don't have the same type, $\exists k \leq p+1$ such that $t_k = 1$ and $t_{k+1} = 2$. By Lemmas 7 and 8, we get $|A_{p+1} - B_{p+1}| \leq 1$, implying $B_{p+1} \leq 1$. Hence, the $(p+1)^{\text{th}}$ bin will use up all tall items, implying at most 2 non-full bins. ◀

▶ **Theorem 10.** *The number of bins m used by* `greedyPack` *is at most* $\max\left(\lceil \text{hsum}(\widetilde{W}) \rceil, \lceil \text{wsum}(\widetilde{H}) \rceil, \frac{4}{3} a(\widetilde{W} \cup \widetilde{H}) + \frac{8}{3}\right)$.

Proof. If all bins have the same type, then $m \leq \max(\lceil \text{hsum}(\widetilde{W}) \rceil, \lceil \text{wsum}(\widetilde{H}) \rceil)$.

Let there be m_1 (resp. m_2) full bins of type 1 (resp. type 2) and let J_1 (resp. J_2) be the items inside those bins. Then by Lemma 6, we get $m_1 \leq 4a(J_1)/3 + 1/3$ and $m_2 \leq 4a(J_2)/3 + 1/3$. Hence, $m_1 + m_2 \leq 4a(\widetilde{W} \cup \widetilde{H})/3 + 2/3$. If all bins don't have the same type, then by Lemma 9, there can be at most 2 non-full bins, so `greedyPack`$(\widetilde{W}, \widetilde{H})$ uses at most $4a(\widetilde{W} \cup \widetilde{H})/3 + 8/3$ bins. ◀

3.2 The `skewed4Pack` Algorithm

We now return to the 2BP problem. `skewed4Pack` is an algorithm for 2BP takes as input a set I of rectangular items and a parameter $\varepsilon \in (0, 1)$ where $\varepsilon^{-1} \in \mathbb{Z}$. It outputs a 4-stage bin packing of I. `skewed4Pack` has the following outline:

A. Use linear grouping [15, 28] to round up the width or height of each item in I. This gives us a new instance $\widehat{I}$.

B. Pack $\widehat{I}$ into $1/\varepsilon^2 + 1$ shelves, after *slicing* some items. A shelf is a rectangular region with width or height more than $1/2$ and is fully packed, i.e., the area of items in a shelf equals the area of the shelf. If we treat each shelf as an item, we get a new instance $\widetilde{I}$.

C. Compute a packing of $\widetilde{I}$ into bins, after possibly slicing some items, using `greedyPack`.

D. Repack most items of $\widehat{I}$ without slicing into the shelves. We will prove that the remaining items have very small area, so they can be packed separately using NFDH.

A. Item Classification and Rounding. Define $W := \{i \in I : h(i) \leq \delta_H\}$ and $H := I - W$. Items in W are called *wide* and items in H are called *tall*. Let $W^{(L)} := \{i \in W : w(i) > \varepsilon\}$ and $W^{(S)} := W - W^{(L)}$. Similarly, let $H^{(L)} := \{i \in H : h(i) > \varepsilon\}$ and $H^{(S)} := H - H^{(L)}$.

We will use *linear grouping* [15, 28] to round up the widths of $W^{(L)}$ and the heights of $H^{(L)}$ to get items $\widehat{W}^{(L)}$ and $\widehat{H}^{(L)}$, respectively. By Claim 27 in Appendix A, items in $\widehat{W}^{(L)}$ have at most $1/\varepsilon^2$ distinct widths and items in $\widehat{H}^{(L)}$ have at most $1/\varepsilon^2$ distinct heights.

Let $\widehat{W} := \widehat{W}^{(L)} \cup W^{(S)}$, $\widehat{H} := \widehat{H}^{(L)} \cup H^{(S)}$, $\widehat{I} := \widehat{W} \cup \widehat{H}$. For $\widehat{X} \subseteq \widehat{I}$, let $\text{fopt}(\widehat{X})$ be the minimum number of bins needed to pack $\widehat{X}$ when items in $\widehat{X} \cap \widehat{W}^{(L)}$ can be sliced using horizontal cuts, items in $\widehat{X} \cap \widehat{H}^{(L)}$ can be sliced using vertical cuts, and items in $\widehat{X} \cap (W^{(S)} \cup H^{(S)})$ can be sliced both vertically and horizontally. The following lemma follows from Lemma 28 in Appendix A.

▶ **Lemma 11.** $\text{fopt}(\widehat{I}) < (1 + \varepsilon)\,\text{opt}(I) + 2$.

B. Creating Shelves. We will use ideas from Kenyon and Rémila's 2SP algorithm [28] to pack $\widehat{I}$ into *shelves*. Roughly, we solve a linear program to compute an optimal strip packing of $\widehat{W}$, where the packing is 3-stage. The first stage of cuts gives us shelves and the second stage gives us containers. From each shelf, we trim off space that doesn't belong to any container. See Section 3.3 for details. Let $\widetilde{W}$ be the shelves thus obtained. Analogously, we can pack items $\widehat{H}$ into shelves $\widetilde{H}$. Shelves in $\widetilde{W}$ are called *wide shelves* and shelves in $\widetilde{H}$ are called *tall shelves*. Let $\widetilde{I} := \widetilde{W} \cup \widetilde{H}$. We can interpret each shelf in $\widetilde{I}$ as a rectangular item. We allow slicing $\widetilde{W}$ and $\widetilde{H}$ using horizontal cuts and vertical cuts, respectively. In Section 3.3, we prove the following facts.

▶ **Lemma 12.** *$\widetilde{I}$ has the following properties: (a) $|\widetilde{W}| \le 1 + 1/\varepsilon^2$ and $|\widetilde{H}| \le 1 + 1/\varepsilon^2$; (b) Each item in $\widetilde{W}$ has width more than $1/2$ and each item in $\widetilde{H}$ has height more than $1/2$; (c) $a(\widetilde{I}) = a(\widehat{I})$; (d) $\max(\lceil \mathrm{hsum}(\widetilde{W}) \rceil, \lceil \mathrm{wsum}(\widetilde{H}) \rceil) \le \mathrm{fopt}(\widehat{I})$.*

C. Packing Shelves into Bins. So far, we have packed $\widehat{I}$ into shelves $\widetilde{W}$ and $\widetilde{H}$. We will now use `greedyPack`$(\widetilde{W}, \widetilde{H})$ to pack the shelves into bins. By Claim 4, we get a 2-stage packing of $\widetilde{W} \cup \widetilde{H}$ into m bins, where we make at most $m-1$ horizontal cuts in $\widetilde{W}$ and at most $m-1$ vertical cuts in $\widetilde{H}$. The horizontal cuts (resp. vertical cuts) increase the number of wide shelves (resp. tall shelves) from at most $1+1/\varepsilon^2$ to at most $m+1/\varepsilon^2$. By Theorem 10, Lemma 12(d) and Lemma 12(c), we get $m \le \max\left(\lceil \mathrm{hsum}(\widetilde{W}) \rceil, \lceil \mathrm{wsum}(\widetilde{H}) \rceil, \frac{4}{3}a(\widetilde{I}) + \frac{8}{3}\right) \le \frac{4}{3}\,\mathrm{fopt}(\widehat{I}) + \frac{8}{3}$.

D. Packing Items into Containers. So far, we have a packing of shelves into m bins, where the shelves contain slices of items $\widehat{I}$. We will now repack a large subset of $\widehat{I}$ into the shelves without slicing $\widehat{I}$. See Figure 2c for an example output. We do this using a standard greedy algorithm. See Appendix B for details of the algorithm and proof of the following lemma.

▶ **Lemma 13.** *Let P be a packing of $\widetilde{I}$ into m bins, where we made at most $m-1$ horizontal cuts in wide shelves and at most $m-1$ vertical cuts in tall shelves. Then we can (without slicing) pack a large subset of $\widehat{I}$ into the shelves in P such that the unpacked items (also called* discarded items*) from $\widehat{W}$ have total area less than $\varepsilon\,\mathrm{hsum}(\widetilde{W}) + \delta_H(1+\varepsilon)(m+1/\varepsilon^2)$, and the unpacked items from $\widehat{H}$ have area less than $\varepsilon\,\mathrm{wsum}(\widetilde{H}) + \delta_W(1+\varepsilon)(m+1/\varepsilon^2)$.*

We pack wide discarded items into new bins using NFDH and pack tall discarded items into new bins using NFDW. Finally, we prove the performance guarantee of `skewed4Pack`$_\varepsilon(I)$.

▶ **Lemma 14.** *Let I be a set of (δ_W, δ_H)-skewed items. Then* `skewed4Pack`$_\varepsilon(I)$ *outputs a 4-stage packing of I in time $O((1/\varepsilon)^{O(1/\varepsilon)} + n \log n)$ and uses less than $\alpha(1+\varepsilon)\,\mathrm{opt}(I) + 2\beta$ bins, where $\Delta := \frac{1}{2}\left(\frac{\delta_H}{1-\delta_H} + \frac{\delta_W}{1-\delta_W}\right)$, $\alpha := \frac{4}{3}(1+4\Delta)(1+3\varepsilon)$, $\beta := \frac{2\Delta(1+\varepsilon)}{\varepsilon^2} + \frac{10}{3} + \frac{19\Delta}{3} + \frac{16\Delta\varepsilon}{3}$.*

Proof. The discarded items are packed using NFDH or NFDW, which output a 2-stage packing. Since `greedyPack` outputs a 2-stage packing of the shelves and the packing of items into the shelves is a 2-stage packing, the bin packing of non-discarded items is a 4-stage packing. The time taken by `skewed4Pack` is at most $O((1/\varepsilon)^{O(1/\varepsilon)} + n \log n)$.

Suppose `greedyPack` uses at most m bins. By Theorem 10, $m \le 4\,\mathrm{fopt}(\widehat{I})/3 + 8/3$. Let W^d and H^d be the items discarded from W and H, respectively. By Lemma 13 and Lemma 12(d), $a(W^d) < \varepsilon\,\mathrm{fopt}(\widehat{I}) + \delta_H(1+\varepsilon)(m+1/\varepsilon^2)$ and $a(H^d) < \varepsilon\,\mathrm{fopt}(\widehat{I}) + \delta_W(1+\varepsilon)(m+1/\varepsilon^2)$.

By Lemmas 3 and 11, the number of bins used by $\texttt{skewed4Pack}_\varepsilon(I)$ is less than

$$\begin{aligned}
&m + \frac{2a(W^d)+1}{1-\delta_H} + \frac{2a(H^d)+1}{1-\delta_W} \\
&\leq (1+4\Delta(1+\varepsilon))m + 4\varepsilon(1+\Delta)\operatorname{fopt}(\widehat{I}) + 2(1+\Delta) + 4\Delta(1+\varepsilon)/\varepsilon^2 \\
&\leq \alpha \operatorname{fopt}(\widehat{I}) + 2(\beta-1) < \alpha(1+\varepsilon)\operatorname{opt}(I) + 2\beta. \qquad \blacktriangleleft
\end{aligned}$$

Now we conclude with the proof of Theorem 1.

Proof of Theorem 1. This is a simple corollary of Lemma 14, where $\delta \leq 1/2$ gives us $\Delta \leq 2\delta$, $\alpha(1+\varepsilon) \leq (4/3)(1+8\delta)(1+7\varepsilon)$, and $\beta \leq 4/\varepsilon^2 + 15$. $\blacktriangleleft$

3.3 Creating Shelves

Here we will describe how to obtain shelves $\widetilde{W}$ and $\widetilde{H}$ from items $\widehat{W}$ and $\widehat{H}$, respectively. Let $\operatorname{opt}_{\mathrm{SP}}(\widehat{W})$ denote the optimal strip packing of $\widehat{W}$ where items in $\widehat{W}$ can be sliced using horizontal cuts. Then $\operatorname{fopt}(\widehat{W}) = \lceil \operatorname{opt}_{\mathrm{SP}}(\widehat{W}) \rceil$. Hence, we will now try to compute a near-optimal strip packing of $\widehat{W}$.

Define a horizontal configuration S as a tuple $(S_0, S_1, S_2, \ldots)$ of $1/\varepsilon^2 + 1$ non-negative integers, where $S_0 \in \{0,1\}$ and $\sum_{j=1}^{1/\varepsilon^2} S_j w_j \leq 1$. For any horizontal line at height y in a strip packing of $\widehat{W}$, the multiset of items intersecting the line corresponds to a configuration. S_0 indicates whether the line intersects items from $W^{(S)}$, and S_j is the number of items from $\widehat{W}_j^{(L)}$ that the line intersects. Let $\mathcal{S}$ be the set of all horizontal configurations. Let $N := |\mathcal{S}|$.

To obtain an optimal packing, we need to determine the height of each configuration. This can be done with the following linear program.

$$\begin{aligned}
\min_{x \in \mathbb{R}^N} \quad & \sum_{S \in \mathcal{S}} x_S & & \\
\text{where} \quad & \sum_{S \in \mathcal{S}} S_j x_S = h(\widehat{W}_j^{(L)}) & & \forall j \in [1/\varepsilon^2] \\
\text{and} \quad & \sum_{S:S_0=1} \left(1 - \sum_{j=1}^{1/\varepsilon^2} S_j w_j\right) x_S = a(W^{(S)}) & & \\
\text{and} \quad & x_S \geq 0 & & \forall S \in \mathcal{S}
\end{aligned}$$

Let x^* be an optimal extreme-point solution to the above LP. This gives us a packing where the strip is divided into rectangular regions called *shelves* that are stacked on top of each other. Each shelf has a configuration S associated with it and has height $h(S) := x^*_S$ and contains S_j *containers* of width w_j. Containers of width w_j only contain items from $\widehat{W}_j^{(L)}$, and we call them *type-j* containers. If $S_0 = 1$, S also contains a container of width $1 - \sum_{j=1}^{1/\varepsilon^2} S_j w_j$ that contains small items. We call this container a *type*-0 container. Each container is fully filled with items. Let $w(S)$ denote the width of shelf S, i.e., the sum of widths of all containers in S. Note that if $S_0 = 1$, then $w(S) = 1$. Otherwise, $w(S) = \sum_{j=1}^{1/\varepsilon^2} S_j w_j$.

▶ **Lemma 15.** *x^* contains at most $1/\varepsilon^2 + 1$ positive entries.*

Proof sketch. Follows by applying Rank Lemma[2] to the linear program. $\blacktriangleleft$

[2] Rank Lemma: the number of non-zero variables in an extreme-point solution to a linear program is at most the number of non-trivial constraints [34, Lemma 2.1.4].

▶ **Lemma 16.** $x^*_S > 0 \implies w(S) > 1/2$.

Proof. Suppose $w(S) \le 1/2$. Then we could have split S into two parts by making a horizontal cut in the middle and packed the parts side-by-side, reducing the height of the strip by $x^*_S/2$. But that would contradict the fact that x^* is optimal. ◀

Treat each shelf S as an item of width $w(S)$ and height $h(S)$. Allow each such item to be sliced using horizontal cuts. This gives us a new set $\widetilde{W}$ of items such that $\widehat{W}$ can be packed inside $\widetilde{W}$. By applying an analogous approach to $\widehat{H}$, we get a new set $\widetilde{H}$ of items. Let $\widetilde{I} := \widetilde{W} \cup \widetilde{H}$. We call the shelves of $\widetilde{W}$ *wide shelves* and the shelves of $\widetilde{H}$ *tall shelves*. The containers in wide shelves are called *wide containers* and the containers in tall shelves are called *tall containers*.

▶ **Lemma 12.** *$\widetilde{I}$ has the following properties: (a) $|\widetilde{W}| \le 1 + 1/\varepsilon^2$ and $|\widetilde{H}| \le 1 + 1/\varepsilon^2$; (b) Each item in $\widetilde{W}$ has width more than $1/2$ and each item in $\widetilde{H}$ has height more than $1/2$; (c) $a(\widetilde{I}) = a(\widehat{I})$; (d) $\max(\lceil \mathrm{hsum}(\widetilde{W}) \rceil, \lceil \mathrm{wsum}(\widetilde{H}) \rceil) \le \mathrm{fopt}(\widehat{I})$.*

Proof. Lemma 15 implies (a) and Lemma 16 implies (b). $a(\widehat{I}) = a(\widetilde{I})$ as the shelves are tightly packed. Since x^* is an optimal solution to the linear program, $\lceil \mathrm{hsum}(\widetilde{W}) \rceil = \lceil \sum_{S \in \mathcal{S}} x^*_S \rceil = \lceil \mathrm{opt}_{\mathrm{SP}}(\widehat{W}) \rceil = \mathrm{fopt}(\widehat{W}) \le \mathrm{fopt}(\widehat{I})$. Similarly, $\lceil \mathrm{wsum}(\widetilde{H}) \rceil = \mathrm{fopt}(\widehat{H}) \le \mathrm{fopt}(\widehat{I})$. ◀

4 Almost-Optimal Bin Packing of Skewed Rectangles

In this section, we will describe the algorithm `skewedCPack`. `skewedCPack` takes as input a set I of items and a parameter $\varepsilon \in (0, 1/2]$, where $\varepsilon^{-1} \in \mathbb{Z}$. We will prove that `skewedCPack` has AAR $1 + 20\varepsilon$ when δ is sufficiently small. `skewedCPack` works roughly as follows:

1. Invoke the subroutine `round`(I) (cf. Section 4.1). `round`(I) returns a pair $(\widetilde{I}, I_{\mathrm{med}})$. Here I_{med}, called the set of *medium items*, has low total area, so we can pack it in a small number of bins. $\widetilde{I}$, called the set of *rounded items*, is obtained by rounding up the width or height of each item in $I - I_{\mathrm{med}}$, so that $\widetilde{I}$ has properties that help us pack it easily.
2. Compute the optimal *fractional compartmental* bin packing of $\widetilde{I}$ (we will define *fractional* and *compartmental* later).
3. Use this packing of $\widetilde{I}$ to obtain a packing of I that uses slightly more number of bins.

To bound the AAR of `skewedCPack`, we will prove a structural theorem (Section 4.2), which says that the optimal fractional compartmental packing of $\widetilde{I}$ uses close to $\mathrm{opt}(I)$ bins.

4.1 Classifying and Rounding Items

We now describe the algorithm `round` and show that its output satisfies important properties.

First, we will find a set $I_{\mathrm{med}} \subseteq I$ and positive constants ε_1 and ε_2 such that $a(I_{\mathrm{med}}) \le \varepsilon a(I)$, $\varepsilon_2 \ll \varepsilon_1$, and $I - I_{\mathrm{med}}$ is $(\varepsilon_2, \varepsilon_1]$-free, i.e., no item in $I - I_{\mathrm{med}}$ has its width or height in the interval $(\varepsilon_2, \varepsilon_1]$. Then we can remove I_{med} from I and pack it separately into a small number of bins using NFDH. We will see that the $(\varepsilon_2, \varepsilon_1]$-freeness of $I - I_{\mathrm{med}}$ will help us pack $I - I_{\mathrm{med}}$ efficiently. Specifically, we require $\varepsilon_1 \le \varepsilon$, $\varepsilon_1^{-1} \in \mathbb{Z}$, and $\varepsilon_2 = f(\varepsilon_1)$, where $f(x) := \varepsilon x / \left(104(1 + 1/(\varepsilon x))^{2/x-2}\right)$. We explain this choice of f in Section 4.3.4. Intuitively, such an f ensures that $\varepsilon_2 \ll \varepsilon_1$ and $\varepsilon_2^{-1} \in \mathbb{Z}$. For `skewedCPack` to work, we require $\delta \le \varepsilon_2$. Finding such an I_{med} and ε_1 is a standard technique [25, 7], so we defer the details to Appendix C.1.

Next, we classify the items in $I - I_{\mathrm{med}}$ into three disjoint classes:

- Wide items: $W := \{i \in I : w(i) > \varepsilon_1 \text{ and } h(i) \le \varepsilon_2\}$.
- Tall items: $H := \{i \in I : w(i) \le \varepsilon_2 \text{ and } h(i) > \varepsilon_1\}$.
- Small items: $S := \{i \in I : w(i) \le \varepsilon_2 \text{ and } h(i) \le \varepsilon_2\}$.

We will now use *linear grouping* [15, 28] to round up the widths of items W and the heights of items H to get items $\widetilde{W}$ and $\widetilde{H}$, respectively. By Claim 27 in Appendix A, items in $\widetilde{W}$ have at most $1/(\varepsilon\varepsilon_1)$ distinct widths and items in $\widetilde{H}$ have at most $1/(\varepsilon\varepsilon_1)$ distinct heights. Let $\widetilde{I} := \widetilde{W} \cup \widetilde{H} \cup S$.

▶ **Definition 17** (Fractional packing)**.** *Suppose we are allowed to slice wide items in $\widetilde{I}$ using horizontal cuts, slice tall items in $\widetilde{I}$ using vertical cuts and slice small items in $\widetilde{I}$ using both horizontal and vertical cuts. For any $\widetilde{X} \subseteq \widetilde{I}$, a bin packing of the slices of $\widetilde{X}$ is called a* fractional packing *of $\widetilde{X}$. The optimal fractional packing of $\widetilde{X}$ is denoted by* $\text{fopt}(\widetilde{X})$.

▶ **Lemma 18.** $\text{fopt}(\widetilde{I}) < (1+\varepsilon)\,\text{opt}(I) + 2$.

Proof. Directly follows from Lemma 28 in Appendix A. ◀

4.2 Structural Theorem

We will now define compartmental packing and prove the structural theorem, which says that the number of bins in the optimal fractional compartmental packing of $\widetilde{I}$ is roughly equal to $\text{fopt}(\widetilde{I})$.

We first show how to *discretize* a packing, i.e., we show that given a fractional packing of items in a bin, we can remove a small fraction of tall and small items and shift the remaining items leftwards so that the left and right edges of each wide item belong to a constant-sized set $\mathcal{T}$, where $|\mathcal{T}| \leq (1+1/\varepsilon\varepsilon_1)^{2/\varepsilon_1 - 2}$. Next, we define *compartmental* packing and show how to convert a discretized packing to a compartmental packing.

For any rectangle i packed in a bin, let $x_1(i)$ and $x_2(i)$ denote the x-coordinates of its left and right edges, respectively, and let $y_1(i)$ and $y_2(i)$ denote the y-coordinates of its bottom and top edges, respectively. Let R be the set of distinct widths of items in $\widetilde{W}$. Given the way we rounded items, $|R| \leq 1/\varepsilon\varepsilon_1$. Recall that $\varepsilon_1 \leq \varepsilon \leq 1/2$ and $\varepsilon_1^{-1}, \varepsilon^{-1} \in \mathbb{Z}$.

▶ **Theorem 19.** *Given a fractional packing of items $\widetilde{J} \subseteq \widetilde{I}$ into a bin, we can remove tall and small items of total area less than ε and shift some of the remaining items to the left such that for every wide item i, we get $x_1(i), x_2(i) \in \mathcal{T}$.*

Proof. For wide items u and v in the bin, we say that $u \prec v$ iff the right edge of u is to the left of the left edge of v. Formally $u \prec v \iff x_2(u) \leq x_1(v)$. We call u a *predecessor* of v. A sequence $[i_1, i_2, \ldots, i_k]$ such that $i_1 \prec i_2 \prec \ldots \prec i_k$ is called a *chain* ending at i_k. For a wide item i, define $\text{level}(i)$ as the number of items in the longest chain ending at i. Formally, $\text{level}(i) := 1$ if i has no predecessors, and $(1 + \max_{j \prec i} \text{level}(j))$ otherwise. Let W_j be the items at level j, i.e., $W_j := \{i : \text{level}(i) = j\}$. Note that the level of an item can be at most $1/\varepsilon_1 - 1$, since each wide item has width more than ε_1.

We will describe an algorithm for discretization. But first, we need to introduce two recursively-defined set families $(S_1, S_2, \ldots)$ and $(T_0, T_1, \ldots)$. Let $T_0 := \{0\}$ and $t_0 := 1$. For any $j > 0$, define $t_j := (1 + 1/\varepsilon\varepsilon_1)^{2j}$, $\delta_j := \varepsilon\varepsilon_1/t_{j-1}$, $S_j := T_{j-1} \cup \{k\delta_j : k \in \mathbb{Z}, 0 \leq k < 1/\delta_j\}$, $T_j := \{x + y : x \in S_j, y \in R \cup \{0\}\}$. Note that $\forall j > 0$, we have $T_{j-1} \subseteq S_j \subseteq T_j$ and $\delta_j^{-1} \in \mathbb{Z}$. Define $\mathcal{T} := T_{1/\varepsilon_1 - 1}$.

Our discretization algorithm proceeds in stages, where in the j^{th} stage, we apply two transformations to the items in the bin, called *strip-removal* and *compaction*.

Strip-removal: For each $x \in T_{j-1}$, consider a strip of width δ_j and height 1 in the bin whose left edge has coordinate x. Discard the slices of tall and small items inside the strips.

Compaction: Move all tall and small items as much towards the left as possible (imagine a gravitational force acting leftwards on the tall and small items) while keeping the wide items fixed. Then move each wide item $i \in W_j$ leftwards till $x_1(i) \in S_j$.

Observe that the algorithm maintains the following invariant: *after k stages, for each $j \in [k]$, each item $i \in W_j$ has $x_1(i) \in S_j$ (and hence $x_2(i) \in T_j$).* This ensures that after the algorithm ends, $x_1(i), x_2(i) \in \mathcal{T}$. All that remains to prove is that the total area of items discarded during strip-removal is at most ε and that compaction is always possible.

▶ **Lemma 20.** *For all $j \geq 0$, $|T_j| \leq t_j$.*

Proof by induction. $|T_0| = t_0 = 1$, so the base case holds. Now assume $|T_{j-1}| \leq t_{j-1}$. Then

$$|T_j| \leq (|R|+1)|S_j| \leq \left(\frac{1}{\varepsilon\varepsilon_1}+1\right)\left(|T_{j-1}|+\frac{1}{\delta_j}\right) \leq \left(\frac{1}{\varepsilon\varepsilon_1}+1\right)^2 t_{j-1} = t_j. \qquad \blacktriangleleft$$

Therefore, $|\mathcal{T}| \leq t_{1/\varepsilon_1 - 1} = (1 + 1/\varepsilon\varepsilon_1)^{2/\varepsilon_1 - 2}$.

▶ **Lemma 21.** *Discarded items (across all stages) have total area less than ε.*

Proof. In the j^{th} stage, we create $|T_{j-1}|$ strips, and each strip has total area at most δ_j. Therefore, the area discarded in the j^{th} stage is at most $|T_{j-1}|\delta_j \leq t_{j-1}\delta_j = \varepsilon\varepsilon_1$. Since there can be at most $1/\varepsilon_1 - 1$ stages, we discard a total area of less than ε across all stages. ◀

▶ **Lemma 22.** *Compaction always succeeds, i.e., in the j^{th} stage, while moving item $i \in W_j$ leftwards, no other item will block its movement.*

Proof. Let $i \in W_j$. Let z be the x-coordinate of the left edge of the strip immediately to the left of item i, i.e., $z := \max(\{x \in T_{j-1} : x \leq x_1(i)\})$. For any wide item i', we have $x_2(i') \leq x_1(i) \iff i' \prec i \iff \text{level}(i') \leq j-1$. By our invariant, we get $\text{level}(i') \leq j-1 \implies x_2(i') \in T_{j-1} \implies x_2(i') \leq z$. Therefore, for every wide item i', $x_2(i') \notin (z, x_1(i)]$.

In the j^{th} strip-removal, we cleared the strip $[z, z+\delta_j] \times [0,1]$. If $x_1(i) \in [z, z+\delta_j]$, then i can freely move to z, and $z \in T_{j-1} \subseteq S_j$. Since no wide item has its right edge in $(z, x_1(i)]$, if $x_1(i) > z + \delta_j$, all the tall and small items whose left edge lies in $[z+\delta_j, x_1(i)]$ will move leftwards by at least δ_j during compaction. Hence, there would be an empty space of width at least δ_j to the left of item i. Therefore, we can move i leftwards to make $x_1(i)$ a multiple of δ_j, and then $x_1(i)$ would belong to S_j. ◀

Hence, compaction always succeeds and we get $x_1(i), x_2(i) \in \mathcal{T}$ for each wide item i. ◀

▶ **Definition 23** (Compartmental packing)**.** *Consider a packing of some items into a bin. A* compartment *C is defined as a rectangular region in the bin satisfying the following properties:*

- *$x_1(C), x_2(C) \in \mathcal{T}$.*
- *$y_1(C), y_2(C)$ are multiples of $\varepsilon_{\text{cont}} := \varepsilon\varepsilon_1/6|\mathcal{T}|$.*
- *C does not contain both wide items and tall items.*
- *If C contains tall items, then $x_1(C)$ and $x_2(C)$ are consecutive values in $\mathcal{T}$.*

If a compartment contains a wide item, it is called a wide compartment*. Otherwise it is called a* tall compartment*. A packing of items into a bin is called* compartmental *iff there is a set of non-overlapping* compartments *in the bin such that each wide or tall item lies completely inside some compartment, and there are at most $n_W := 3(1/\varepsilon_1 - 1)|\mathcal{T}| + 1$ wide compartments and at most $n_H := (1/\varepsilon_1 - 1)|\mathcal{T}|$ tall compartments in the bin. A packing of items into multiple bins is called compartmental iff each bin is compartmental.*

Note that small items can be packed both inside and outside compartments.

The following lemma states that a discretized packing can be converted to a compartmental packing. It can be proved using standard techniques (e.g., Section 3.2.3 in [36]). See Appendix G.2 in [31] for a formal proof.

▶ **Lemma 24.** *If $x_1(i), x_2(i) \in \mathcal{T}$ for each wide item i in a bin, then by removing wide and small items of area less than ε, we can get a compartmental packing of the remaining items.*

▶ **Theorem 25.** *For a set $\widetilde{I}$ of δ-skewed rounded items, define* $\mathrm{fcopt}(\widetilde{I})$ *as the number of bins in the optimal fractional compartmental packing*[3] *of $\widetilde{I}$. Then* $\mathrm{fcopt}(\widetilde{I}) < (1+4\varepsilon)\,\mathrm{fopt}(\widetilde{I}) + 2$.

Proof. Consider a fractional packing of $\widetilde{I}$ into $m := \mathrm{fopt}(\widetilde{I})$ bins. From each bin, we can discard items of area at most 2ε and get a compartmental packing of the remaining items by Theorem 19 and Lemma 24.

Let X be the set of wide and small discarded items and let Y be the set of tall discarded items. For each item $i \in X$, if $w(i) \leq 1/2$, slice it using a horizontal cut in the middle and place the pieces horizontally next to each other to get a new item of width $2w(i)$ and height $h(i)/2$. Repeat until $w(i) > 1/2$. Now pack the items in bins by stacking them one-over-the-other so that for each item $i \in X$, $x_1(i) = 0$. This will require less than $2a(X) + 1$ bins, and the packing will be compartmental.

Similarly, we can get a compartmental packing of Y into $2a(Y)+1$ bins. Since $a(X \cup Y) < 2\varepsilon m$, we will require less than $4\varepsilon m + 2$ bins. Therefore, the total number of compartmental bins used to pack $\widetilde{I}$ is less than $(1+4\varepsilon)m + 2$. ◀

4.3 Packing Algorithm

We now describe the `skewedCPack` algorithm for packing a set I of n δ-skewed items.

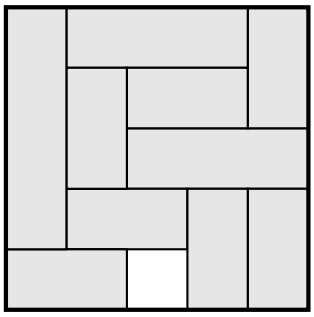
1. **Classifying and Rounding Items** (see Section 4.1): Compute $(\widetilde{I}, I_{\mathrm{med}}) := \texttt{round}(I)$. Recall that I_{med}, called the set of *medium items*, has low total area, and $\widetilde{I}$, called the set of *rounded items*, is obtained by rounding up the width or height of each item in $I - I_{\mathrm{med}}$.
2. **Enumerating Packing of Compartments**: Compute all possible packings of empty compartments into at most n bins.
3. **Fractionally Packing Items into Compartments**: For each packing P of empty compartments, fractionally pack $\widetilde{I}$ into P using a linear program.
4. **Converting a Fractional Packing to a Non-Fractional Packing**: Discard a small set $D \subseteq \widetilde{I}$ of items and use an extreme-point solution to the linear program to non-fractionally pack $\widetilde{I} - D$ into P.
5. Pack $I_{\mathrm{med}} \cup D$ into bins using NFDH.

See Figure 3 for a visual overview of `skewedCPack`. We describe steps 2, 3 and 4 in Sections 4.3.1–4.3.3, respectively. In Section 4.3.4, we bound the AAR of `skewedCPack`.

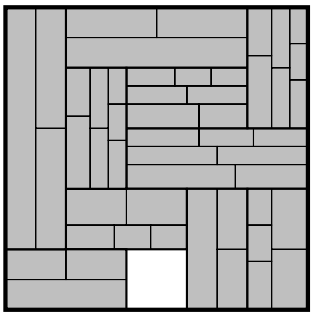
4.3.1 Enumerating Packing of Compartments

We will now describe a subroutine, called $\texttt{iterPackings}(\widetilde{I})$, that outputs all packings of empty compartments into at least $\lceil a(\widetilde{I}) \rceil$ bins and at most n bins. A packing of empty compartments in a bin is called a *configuration*. We will first enumerate all configurations and then output multisets of configurations of cardinality ranging from $\lceil a(\widetilde{I}) \rceil$ to n.

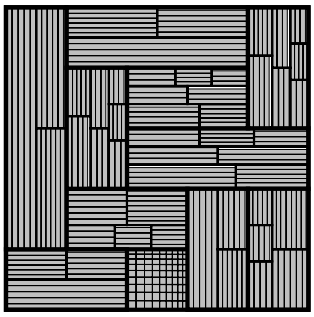
[3] A *fractional compartmental* packing of $\widetilde{I}$ is a fractional packing of $\widetilde{I}$ that is also compartmental.



(a) Guess the packing of empty compartments in each bin (Section 4.3.1).



(b) Fractionally pack wide and tall items into compartments. This partitions each compartment into *containers* (Section 4.3.2).



(c) Pack the items non-fractionally (Section 4.3.3).

Figure 3 Major steps of `skewedCPack` after `rounding` I.

There can be at most $n_W := 3(1/\varepsilon_1 - 1)|\mathcal{T}| + 1$ wide compartments in a bin. Each wide compartment can have $(1/\varepsilon_{\text{cont}})^2$ y-coordinates of the top and bottom edges and at most $|\mathcal{T}|^2/2$ x-coordinates of the left and right edges, where $\varepsilon_{\text{cont}} := \varepsilon\varepsilon_1/6|\mathcal{T}|$. The rest of the space is for tall compartments. Therefore, the number of configurations is at most

$$n_C := \left((1/\varepsilon_{\text{cont}})^2|\mathcal{T}|^2/2\right)^{n_W} \leq \left(\frac{3|\mathcal{T}|^2}{\varepsilon\varepsilon_1}\right)^{6|\mathcal{T}|/\varepsilon_1} \leq \left(1+\frac{1}{\varepsilon\varepsilon_1}\right)^{\left(1+\frac{1}{\varepsilon\varepsilon_1}\right)^{2/\varepsilon_1+1}}.$$

Since each configuration can have at most n bins, the number of combinations of configurations is at most $(n+1)^{n_C}$. Therefore, we can output all possible bin packings of empty compartments in $O(n^{n_C})$ time. This completes the description of `iterPackings`.

4.3.2 Fractionally Packing Items into Compartments

For each bin packing P of empty compartments, we will try to fractionally pack the items into the bins. To do this, we will create a feasibility linear program, called $\text{FP}(\widetilde{I}, P)$, that is feasible iff wide and tall items in $\widetilde{I}$ can be packed into the compartments in P. If $\text{FP}(\widetilde{I}, P)$ is feasible, then small items can also be fractionally packed since P contains at least $a(\widetilde{I})$ bins.

Let $w'_1, w'_2, \ldots, w'_p$ be the distinct widths of wide compartments in P. Let U_j be the set of wide compartments in P having width w'_j. Let $h(U_j)$ be the sum of heights of the compartments in U_j. By Definition 23, we know that $p \leq |\mathcal{T}|^2/2$. Let $w_1, w_2, \ldots, w_r$ be the distinct widths of items in $\widetilde{W}$ (recall that $\widetilde{W}$ is the set of wide items in $\widetilde{I}$). Let $\widetilde{W}_j$ be the items in $\widetilde{W}$ having width w_j. Let $h(\widetilde{W}_j)$ be the sum of heights of all items in $\widetilde{W}_j$. By Claim 27, we get $r \leq 1/\varepsilon\varepsilon_1$.

Let $C := [C_0, C_1, \ldots, C_r]$ be a vector, where $C_0 \in [p]$ and $C_j \in \mathbb{Z}_{\geq 0}$ for $j \in [r]$. C is called a *wide configuration* iff $w(C) := \sum_{j=1}^{r} C_j w_j \leq w'_{C_0}$. Intuitively, a wide configuration C represents a set of wide items that can be placed side-by-side into a compartment of width w'_{C_0}. Let $\mathcal{C}$ be the set of all wide configurations. Then $|\mathcal{C}| \leq p/\varepsilon_1^r$, which is a constant. Let $\mathcal{C}_j := \{C \in \mathcal{C} : C_0 = j\}$.

To pack $\widetilde{W}$ into wide compartments, we must determine the height of each configuration. Let $x \in \mathbb{R}_{\geq 0}^{|\mathcal{C}|}$ be a vector where x_C denotes the height of configuration C. Then $\widetilde{W}$ can be packed into wide compartments according to x iff x is a feasible solution to the following

feasibility linear program, named $\text{FP}_W(\widetilde{I}, P)$:

$$\begin{aligned}
\sum_{C \in \mathcal{C}} C_j x_C &\geq h(\widetilde{W}_j) \quad \forall j \in [r] && (\widetilde{W}_j \text{ should be covered}) \\
\sum_{C \in \mathcal{C} \text{ and } C_0 = j} x_C &\leq h(U_j) \quad \forall j \in [p] && (\mathcal{C}_j \text{ should fit in } U_j) \\
x_C &\geq 0 \quad \forall C \in \mathcal{C}
\end{aligned}$$

Let x^* be an extreme point solution to $\text{FP}_W(\widetilde{I}, P)$ (if $\text{FP}_W(\widetilde{I}, P)$ is feasible). By Rank Lemma[4], at most $p + r$ entries of x^* are non-zero. Since the number of variables and constraints is constant, x^* can be computed in constant time.

Let $\widetilde{H}$ be the set of tall items in $\widetilde{I}$. Items in $\widetilde{H}$ have at most $1/\varepsilon\varepsilon_1$ distinct heights. Let there be q distinct heights of tall compartments in P. By Definition 23, we get $q \leq 1/\varepsilon_{\text{cont}} = 6|\mathcal{T}|/\varepsilon\varepsilon_1$. We can similarly define *tall configurations* and define a feasibility linear program for tall items, named $\text{FP}_H(\widetilde{I}, P)$. $\widetilde{H}$ can be packed into tall compartments in P iff $\text{FP}_H(\widetilde{I}, P)$ is feasible. Let y^* be an extreme point solution to $\text{FP}_H(\widetilde{I}, P)$. Then y^* can be computed in constant time and y^* has at most $q + 1/\varepsilon\varepsilon_1$ positive entries.

Hence, $\widetilde{I}$ can be packed into P iff $\text{FP}(\widetilde{I}, P) := \text{FP}_W(\widetilde{I}, P) \wedge \text{FP}_H(\widetilde{I}, P)$ is feasible. The solution (x^*, y^*) shows us how to split each compartment into *shelves*, where each shelf corresponds to a configuration C and the shelf can be split into C_j *containers* of width w_j and one container of width $w'_{C_0} - w(C)$. Let there be m bins in P. After splitting the configurations across compartments, we get at most $p + q + 2/\varepsilon\varepsilon_1 + m(n_W + n_H)$ shelves.

4.3.3 Converting a Fractional Packing to a Non-Fractional Packing

Consider a packing P of empty compartments into m bins. Let x^* and y^* be extreme-point solutions to $\text{FP}_W(\widetilde{I}, P)$ and $\text{FP}_H(\widetilde{I}, P)$, respectively (assuming $\widetilde{I}$ can fit into P). Then (x^*, y^*) gives us a fractional compartmental packing of $\widetilde{I}$ into m bins. We now show how to convert this to a non-fractional compartmental packing by removing some items from $\widetilde{I}$.

Formally, we give an algorithm called `greedyCPack`$(\widetilde{I}, P, x^*, y^*)$. It returns a pair (Q, D), where Q is a (non-fractional) compartmental bin packing of items $\widetilde{I} - D$, where the compartments in the bins are as per P. D is called the set of discarded items.

`greedyCPack` is based on standard techniques. We prove in Appendix C.2 that

$$a(D) < \frac{52|\mathcal{T}|\varepsilon_2}{\varepsilon_1} m + 4\varepsilon_2 \left(\frac{|\mathcal{T}|^2}{2} + \frac{6|\mathcal{T}|}{\varepsilon\varepsilon_1} + \frac{2}{\varepsilon\varepsilon_1} \right). \tag{1}$$

We give an outline of `greedyCPack` here and defer the details to Appendix C.2.

1. For each j, iteratively assign wide items from $\widetilde{W}_j$ to a container of width w_j. When the total height of assigned items exceeds the height of the container, discard the last-assigned item and switch to a new container and repeat.
2. Similarly assign tall items to tall containers.
3. Identify rectangular regions where we can pack small items:
   - For each configuration C, there is a free region of width $w'_{C_0} - w(C)$ and height x^*_C in a wide compartment. Similarly, we get free regions in tall compartments.

[4] Rank Lemma: the number of non-zero variables in an extreme-point solution to a linear program is at most the number of non-trivial constraints [34, Lemma 2.1.4].

- In each bin, the number of compartments is constant, so the space outside compartments be divided into a constant number of rectangular regions (see Lemma 29).

Pack most of the small items into these free regions using NFDH. Discard the rest.

4.3.4 Summary

See Appendix C.3 for a precise description of `skewedCPack`.

Recall the function f from Section 4.1. Since $\varepsilon_2 := f(\varepsilon_1)$, we get

$$\varepsilon_2 = f(\varepsilon_1) = \frac{\varepsilon\varepsilon_1}{104(1+1/\varepsilon\varepsilon_1)^{2/\varepsilon_1-2}} \le \frac{\varepsilon\varepsilon_1}{104|\mathcal{T}|}. \tag{2}$$

The last inequality follows from the fact that $|\mathcal{T}| \le (1+1/\varepsilon\varepsilon_1)^{2/\varepsilon_1-2}$.

▶ **Theorem 26.** *The number of bins used by* $\texttt{skewedCPack}_\varepsilon(\widetilde{I})$ *is less than*

$$(1+20\varepsilon)\operatorname{opt}(I) + \frac{1}{13}\left(1+\frac{1}{\varepsilon\varepsilon_1}\right)^{2/\varepsilon_1-2} + 23.$$

Proof. In an optimal fractional compartmental bin packing of $\widetilde{I}$, let P^* be the corresponding packing of empty compartments into bins. Hence, P^* contains $m := \operatorname{fcopt}(\widetilde{I})$ bins. Since $\texttt{iterPackings}(\widetilde{I})$ iterates over all bin packings of compartments, $P^* \in \texttt{iterPackings}(\widetilde{I})$. Since wide and tall items in $\widetilde{I}$ can be packed into the compartments of P^*, we get that x^* and y^* are not `null`. By Lemma 3, the number of bins used by NFDH to pack $I_{\text{med}} \cup D$ is less than $2a(I_{\text{med}} \cup D)/(1-\delta) + 3 + 1/(1-\delta)$. Therefore, the number of bins used by $\texttt{skewedCPack}(I)$ is less than

$$\begin{aligned}
&m + \frac{2a(I_{\text{med}} \cup D)}{1-\delta} + 3 + \frac{1}{1-\delta} \\
&< m + \frac{2\varepsilon}{1-\delta}a(I) + \frac{2\varepsilon_2}{1-\delta}\left(\frac{52|\mathcal{T}|}{\varepsilon_1}m + 4\left(\frac{|\mathcal{T}|^2}{2} + \frac{6|\mathcal{T}|+2}{\varepsilon\varepsilon_1}\right)\right) + 3 + \frac{1}{1-\delta} \\
&\qquad\qquad \text{(by Equation (1) and } a(I_{\text{med}}) \le \varepsilon a(I)\text{)} \\
&= \left(1 + \frac{104\varepsilon_2|\mathcal{T}|}{\varepsilon_1(1-\delta)}\right)m + \frac{2\varepsilon}{1-\delta}a(I) + 3 + \frac{1}{1-\delta} + \frac{8\varepsilon_2}{1-\delta}\left(\frac{|\mathcal{T}|^2}{2} + \frac{6|\mathcal{T}|+2}{\varepsilon\varepsilon_1}\right) \\
&= \left(1+\frac{\varepsilon}{1-\delta}\right)m + \frac{2\varepsilon}{1-\delta}a(I) + 3 + \frac{1}{13(1-\delta)}\left(\frac{\varepsilon\varepsilon_1|\mathcal{T}|}{2} + 19 + \frac{2}{|\mathcal{T}|}\right). \\
&\qquad\qquad \text{(by Equation (2))}
\end{aligned}$$

By Theorem 25 and Lemma 18, we get

$$m = \operatorname{fcopt}(\widetilde{I}) < (1+4\varepsilon)\operatorname{fopt}(\widetilde{I}) + 2 < (1+4\varepsilon)(1+\varepsilon)\operatorname{opt}(I) + 4 + 8\varepsilon.$$

Therefore, the number of bins used by $\texttt{skewedCPack}(I)$ is less than

$$\begin{aligned}
&\left((1+4\varepsilon)(1+\varepsilon)\left(1+\frac{\varepsilon}{1-\delta}\right) + \frac{2\varepsilon}{1-\delta}\right)\operatorname{opt}(I) \\
&\qquad + (4+8\varepsilon)\left(1+\frac{\varepsilon}{1-\delta}\right) + 3 + \frac{1}{13(1-\delta)}\left(\frac{\varepsilon\varepsilon_1|\mathcal{T}|}{2} + 19 + \frac{2}{|\mathcal{T}|}\right) \\
&\le (1+20\varepsilon)\operatorname{opt}(I) + \frac{1}{13}\left(1+\frac{1}{\varepsilon\varepsilon_1}\right)^{2/\varepsilon_1-2} + 23. \qquad \text{(since } \delta \le \varepsilon_1 \le \varepsilon \le 1/2\text{)}
\end{aligned}$$

◀

References

1 Fidaa Abed, Parinya Chalermsook, José Correa, Andreas Karrenbauer, Pablo Pérez-Lantero, José A Soto, and Andreas Wiese. On guillotine cutting sequences. In *International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX)*, pages 1–19, 2015. doi:10.4230/LIPIcs.APPROX-RANDOM.2015.1.

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. doi:10.1145/3326122.

3 Soroush Alamdari, Therese Biedl, Timothy M Chan, Elyot Grant, Krishnam Raju Jampani, Srinivasan Keshav, Anna Lubiw, and Vinayak Pathak. Smart-grid electricity allocation via strip packing with slicing. In *Workshop on Algorithms and Data Structures (WADS)*, pages 25–36. Springer, 2013. doi:10.1007/978-3-642-40104-6_3.

4 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.

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. doi:10.1287/moor.1050.0168.

6 Nikhil Bansal, Marek Eliáš, and Arindam Khan. Improved approximation for vector bin packing. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1561–1579. SIAM, 2016. doi:10.1137/1.9781611974331.ch106.

7 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.

8 Nikhil Bansal, Andrea Lodi, and Maxim Sviridenko. A tale of two dimensional bin packing. In *Symposium on Foundations of Computer Science (FOCS)*, pages 657–666. IEEE, 2005. doi:10.1109/SFCS.2005.10.

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

10 Alberto Caprara, Andrea Lodi, and Michele Monaci. Fast approximation schemes for two-stage, two-dimensional bin packing. *Mathematics of Operations Research*, 30(1):150–172, 2005. doi:10.1287/moor.1040.0112.

11 Miroslav Chlebík and Janka Chlebíková. Hardness of approximation for orthogonal rectangle packing and covering problems. *Journal of Discrete Algorithms*, 7(3):291–305, 2009. doi:10.1016/j.jda.2009.02.002.

12 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. doi:10.1016/j.cosrev.2016.12.001.

13 Fan RK Chung, Michael R Garey, and David S Johnson. On packing two-dimensional bins. *SIAM Journal on Algebraic Discrete Methods*, 3(1):66–76, 1982. doi:10.1137/0603007.

14 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.

15 W 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 Max A. Deppert, Klaus Jansen, Arindam Khan, Malin Rau, and Malte Tutas. Peak demand minimization via sliced strip packing, 2021. arXiv:2105.07219.

17 Aleksei V Fishkin, Olga Gerber, and Klaus Jansen. On efficient weighted rectangle packing with large resources. In *International Symposium on Algorithms and Computation (ISAAC)*, pages 1039–1050. Springer, 2005. doi:10.1007/11602613_103.

18 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 *International*

Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX), 2020. doi:10.4230/LIPIcs.APPROX/RANDOM.2020.44.

19 Waldo Gálvez, Fabrizio Grandoni, Afrouz Jabal Ameli, and Kamyar Khodamoradi. Approximation algorithms for demand strip packing, 2021. arXiv:2105.08577.

20 Waldo Gálvez, Fabrizio Grandoni, Sandy Heydrich, Salvatore Ingala, Arindam Khan, and Andreas Wiese. Approximating geometric knapsack via L-packings. In *Symposium on Foundations of Computer Science (FOCS)*, pages 260–271. IEEE, 2017. doi:10.1109/FOCS.2017.32.

21 Waldo Gálvez, Fabrizio Grandoni, Arindam Khan, Diego Ramirez-Romero, and Andreas Wiese. Improved approximation algorithms for 2-dimensional knapsack: Packing into multiple L-shapes, spirals and more. In *International Symposium on Computational Geometry (SoCG)*, volume 189, pages 39:1–39:17, 2021. doi:10.4230/LIPIcs.SoCG.2021.39.

22 Paul C Gilmore and Ralph E Gomory. Multistage cutting stock problems of two and more dimensions. *Operations Research*, 13(1):94–120, 1965. doi:10.1287/opre.13.1.94.

23 Rolf Harren, Klaus Jansen, Lars Prädel, and Rob Van Stee. A $(5/3+\varepsilon)$-approximation for strip packing. In *Workshop on Algorithms and Data Structures (WADS)*, pages 475–487. Springer, 2011. doi:10.1007/978-3-642-22300-6_40.

24 Rebecca Hoberg and Thomas Rothvoss. A logarithmic additive integrality gap for bin packing. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2616–2625, 2017. doi:10.1137/1.9781611974782.172.

25 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.

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

27 Klaus Jansen and Guochuan Zhang. On rectangle packing: maximizing benefits. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, volume 4, pages 204–213, 2004.

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

29 Arindam Khan, Arnab Maiti, Amatya Sharma, and Andreas Wiese. On guillotine separable packings for the two-dimensional geometric knapsack problem. In *International Symposium on Computational Geometry (SoCG)*, volume 189, pages 48:1–48:17, 2021. doi:10.4230/LIPIcs.SoCG.2021.48.

30 Arindam Khan and Madhusudhan Reddy Pittu. On guillotine separability of squares and rectangles. In *International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX)*, 2020. doi:10.4230/LIPIcs.APPROX/RANDOM.2020.47.

31 Arindam Khan and Eklavya Sharma. Tight approximation algorithms for geometric bin packing with skewed items. *ArXiv*, 2105.02827, 2021. arXiv:2105.02827.

32 Arindam Khan, Eklavya Sharma, and K. V. N. Sreenivas. Approximation algorithms for generalized multidimensional knapsack. *ArXiv*, 2102.05854, 2021. arXiv:2102.05854.

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

34 Lap Chi Lau, Ramamoorthi Ravi, and Mohit Singh. *Iterative Methods in Combinatorial Optimization*, volume 46. Cambridge University Press, 2011.

35 János Pach and Gábor Tardos. Cutting glass. In *International Symposium on Computational Geometry (SoCG)*, pages 360–369, 2000. doi:10.1145/336154.336223.

36 Lars Dennis Prädel. *Approximation Algorithms for Geometric Packing Problems*. PhD thesis, Kiel University, 2012. URL: https://macau.uni-kiel.de/servlets/MCRFileNodeServlet/dissertation_derivate_00004634/dissertation-praedel.pdf?AC=N.

37 Sai Sandeep. Almost optimal inapproximability of multidimensional packing problems. *ArXiv*, 2101.02854, 2021. arXiv:2101.02854.

38 Steven S. Seiden and Gerhard J. Woeginger. The two-dimensional cutting stock problem revisited. *Mathematical Programming*, 102(3):519–530, 2005. doi:10.1007/s10107-004-0548-1.

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

40 A. Steinberg. A strip-packing algorithm with absolute performance bound 2. *SIAM Journal on Computing*, 26(2):401–409, 1997. doi:10.1137/S0097539793255801.

41 Paul E. Sweeney and Elizabeth Ridenour Paternoster. Cutting and packing problems: A categorized, application-orientated research bibliography. *Journal of the Operational Research Society*, 43(7):691–706, 1992. doi:10.1057/jors.1992.101.

A Linear Grouping

In this section, we describe the *linear grouping* technique [15, 28] for wide and tall items.

Let ε and ε_1 be constants in $(0, 1)$. Let W be a set of items where each item has width more than ε_1. We will describe an algorithm, called `lingroupWide` that takes W, ε and ε_1 as input and returns the set $\widehat{W}$ as output, where $\widehat{W}$ is obtained by increasing the width of each item in W. `lingroupWide`$(W, \varepsilon, \varepsilon_1)$ first arranges the items W in decreasing order of width and stacks them one-over-the-other (i.e., the widest item in W is at the bottom). Let h_L be the height of the stack. Let $y(i)$ be the y-coordinate of the bottom edge of item i. Split the stack into sections of height $\varepsilon\varepsilon_1 h_L$ each. For $j \in [1/\varepsilon\varepsilon_1]$, let w_j be the width of the widest item intersecting the j^{th} section from the bottom, i.e.,

$$w_j := \max(\{w(i) : i \in W \text{ and } (y(i), y(i) + h(i)) \cap ((j-1)\varepsilon\varepsilon_1 h_L, j\varepsilon\varepsilon_1 h_L) \neq \emptyset\}).$$

Round up the width of each item i to the smallest w_j that is at least $w(i)$ (see Figure 4). Let W_j be the items whose width got rounded to w_j and let $\widehat{W}_j$ be the resulting rounded items. (There may be ties, i.e., there may exist $j_1 < j_2$ such that $w_{j_1} = w_{j_2}$. In that case, define $W_{j_2} := \widehat{W}_{j_2} = \emptyset$. This ensures that all W_j are disjoint.) Finally, define $\widehat{W} := \bigcup_j \widehat{W}_j$.

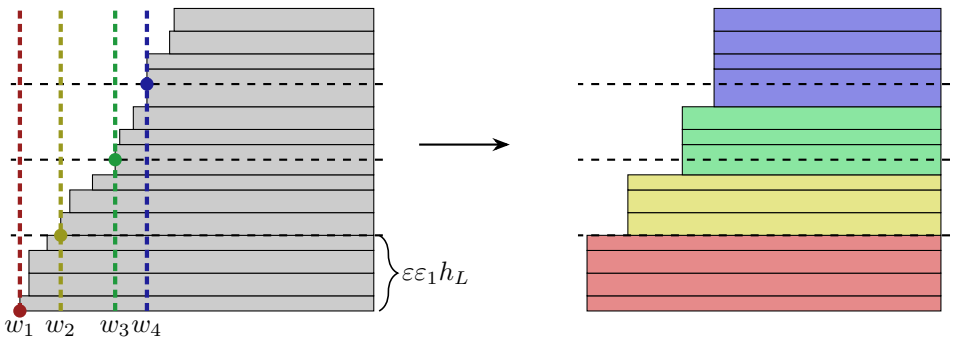


Figure 4 Example invocation of `lingroupWide` for $\varepsilon = \varepsilon_1 = 1/2$.

We can similarly define the algorithm `lingroupTall`. Let H be a set of items where each item has height more than ε_1. `lingroupTall` that takes H, ε and ε_1 as input and returns $\widehat{H}$, where $\widehat{H}$ is obtained by increasing the height of each item in H.

▷ Claim 27. Items in `lingroupWide`$(W, \varepsilon, \varepsilon_1)$ have at most $1/(\varepsilon\varepsilon_1)$ distinct widths. Items in `lingroupTall`$(H, \varepsilon, \varepsilon_1)$ have at most $1/(\varepsilon\varepsilon_1)$ distinct heights.

▶ **Lemma 28.** *Let W, H and S be sets of items, where items in W have width more than ε_1 and items in H have height more than ε_1. Let $\widehat{W} := \texttt{lingroupWide}(W, \varepsilon, \varepsilon_1)$ and $\widehat{H} := \texttt{lingroupTall}(H, \varepsilon, \varepsilon_1)$. If we allow slicing items in $\widehat{W}$ and $\widehat{H}$ using horizontal and vertical cuts, respectively, then we can pack $\widehat{W} \cup \widehat{H} \cup S$ into less than $(1+\varepsilon)\operatorname{opt}(W \cup H \cup S)+2$ bins.*

Proof. (cf. Appendix C in [31]) ◀

B `skewed4Pack`: Packing Items into Containers

▶ **Lemma 13.** *Let P be a packing of $\widetilde{I}$ into m bins, where we made at most $m-1$ horizontal cuts in wide shelves and at most $m-1$ vertical cuts in tall shelves. Then we can (without slicing) pack a large subset of $\widehat{I}$ into the shelves in P such that the unpacked items (also called* discarded items*) from $\widehat{W}$ have total area less than $\varepsilon \operatorname{hsum}(\widetilde{W}) + \delta_H(1+\varepsilon)(m+1/\varepsilon^2)$, and the unpacked items from $\widehat{H}$ have area less than $\varepsilon \operatorname{wsum}(\widetilde{H}) + \delta_W(1+\varepsilon)(m+1/\varepsilon^2)$.*

Proof. For each $j \in [1/\varepsilon^2]$, number the type-j wide containers arbitrarily, and number the items in $\widehat{W}_j^{(L)}$ arbitrarily. Now greedily assign items from $\widehat{W}_j^{(L)}$ to the first container C until the total height of the items exceeds $h(C)$. Then move to the next container and repeat. As per the constraints of the linear program, all items in $\widehat{W}_j^{(L)}$ will get assigned to some type-j wide container. Similarly, number the type-0 wide containers arbitrarily and number the items in $W^{(S)}$ arbitrarily. Greedily assign items from $W^{(S)}$ to the first container C until the total area of the items exceeds $a(C)$. Then move to the next container and repeat. As per the constraints of the linear program, all items in $W^{(S)}$ will get assigned to some type-0 wide container. Similarly, assign all items from $\widehat{H}$ to tall containers.

Let C be a type-j wide container and $\widehat{J}$ be the items assigned to it. If we discard the last item from $\widehat{J}$, then the items can be packed into C. The area of the discarded item is at most $w(C)\delta_H$. Let C be a type-0 wide container and $\widehat{J}$ be the items assigned to it. Arrange the items in $\widehat{J}$ in decreasing order of height and pack the largest prefix $\widehat{J}' \subseteq \widehat{J}$ into C using NFDW (Next-Fit Decreasing Width).

Discard the items $\widehat{J} - \widehat{J}'$. By Lemma 2, $a(\widehat{J} - \widehat{J}') < \varepsilon h(C) + \delta_H w(C) + \varepsilon\delta_H$. Therefore, for a wide shelf S, the total area of discarded items is less than $\varepsilon h(S) + \delta_H(1+\varepsilon)$.

After slicing the shelves in $\widetilde{I}$ to get P, we get at most $m + 1/\varepsilon^2$ wide shelves and at most $m + 1/\varepsilon^2$ tall shelves. Therefore, the total area of discarded items from W is less than $\varepsilon \operatorname{hsum}(\widetilde{W}) + \delta_H(1+\varepsilon)(m+1/\varepsilon^2)$, and the total area of discarded items from H is less than $\varepsilon \operatorname{wsum}(\widetilde{H}) + \delta_W(1+\varepsilon)(m+1/\varepsilon^2)$. ◀

C Details of `skewedCPack`

C.1 Removing Medium Items

Let $T := \lceil 2/\varepsilon \rceil$. Let $\mu_0 = \varepsilon$. For $t \in [T]$, define $\mu_t := f(\mu_{t-1})$ and define

$$J_t := \{i \in I : w(i) \in (\mu_t, \mu_{t-1}] \text{ or } h(i) \in (\mu_t, \mu_{t-1}]\}.$$

Let $r := \operatorname{argmin}_{t=1}^{T} a(J_t)$, $I_{\text{med}} := J_r$, $\varepsilon_1 := \mu_{r-1}$. Each item belongs to at most 2 sets J_t, so

$$a(I_{\text{med}}) = \min_{t=1}^{T} a(J_t) \le \frac{1}{T}\sum_{t=1}^{T} a(J_t) \le \frac{2}{\lceil 2/\varepsilon \rceil} a(I) \le \varepsilon a(I).$$

C.2 Converting a Fractional Packing to a Non-Fractional Packing

▶ **Lemma 29.** *Let there be a set I of rectangles packed inside a bin. Then there is a polynomial-time algorithm that can decompose the empty space in the bin into at most $3|I|+1$ rectangles by making horizontal cuts only.*

Proof. Extend the top and bottom edge of each rectangle leftwards and rightwards till they hit another rectangle or an edge of the bin. This decomposes the empty region into rectangles R. See Figure 5.

For each rectangle $i \in I$, the top edge of i is the bottom edge of a rectangle in R, the bottom edge of i is the bottom edge of two rectangles in R. Apart from possibly the rectangle in R whose bottom edge is at the bottom of the bin, the bottom edge of every rectangle in R is either the bottom or top edge of a rectangle in I. Therefore, $|R| \leq 3|I|+1$. ◀

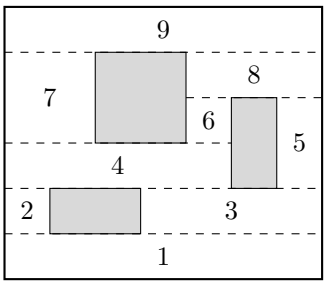


Figure 5 Horizontal cuts partition empty space around the 3 items into 9 rectangular regions.

Let $(Q, D) := \texttt{greedyCPack}(\widetilde{I}, P, x^*, y^*)$, where P is a packing of empty compartments into m bins. We will describe `greedyCPack` and show that

$$a(D) < \frac{52|\mathcal{T}|\varepsilon_2}{\varepsilon_1}m + 4\varepsilon_2\left(\frac{|\mathcal{T}|^2}{2} + \frac{6|\mathcal{T}|}{\varepsilon\varepsilon_1} + \frac{2}{\varepsilon\varepsilon_1}\right).$$

For a configuration C in a wide compartment, there is a container of width $w'_{C_0} - w(C)$ available for packing small items. Hence, there are $p + q + 2/\varepsilon\varepsilon_1 + m(n_W + n_H)$ containers available inside compartments for packing small items. By Lemma 29, we can partition the space outside compartments into at most $m(3(n_W + n_H) + 1)$ containers. Therefore, the total number of containers available for packing small items is at most

$$m_S := (p + q + 2/\varepsilon\varepsilon_1) + m(4(n_W + n_H) + 1) \leq \left(\frac{|\mathcal{T}|^2}{2} + \frac{6|\mathcal{T}|}{\varepsilon\varepsilon_1} + \frac{2}{\varepsilon\varepsilon_1}\right) + \frac{16|\mathcal{T}|}{\varepsilon_1}m.$$

Greedily assign small items to small containers, i.e., keep assigning small items to a container till the area of items assigned to it is at least the area of the container, and then resume from the next container. Each small item will get assigned to some container. For each container C, pack the largest possible prefix of the assigned items using the Next-Fit Decreasing Height (NFDH) algorithm. By Lemma 2, the area of unpacked items would be less than $\varepsilon_2 + \delta + \varepsilon_2\delta$. Summing over all containers, we get that the unpacked area is less than $(\varepsilon_2 + \delta + \varepsilon_2\delta)m_S \leq 3\varepsilon_2 m_S$.

For each j, greedily assign wide items from $\widetilde{W}_j$ to containers of width w_j, i.e., keep assigning items till the height of items exceeds the height of the container. Each wide item will get assigned to some container. Then discard the last item from each container. For each shelf in a wide compartment having configuration C, the total area of items we discard is at most $\delta w(C)$. Similarly, we can discard tall items of area at most $\delta h(C)$ from each shelf in a tall compartment having configuration C.

Hence, across all configurations, we discard wide and tall items of area at most

$$\delta((p+q+2/\varepsilon\varepsilon_1)+m(n_W+n_H)) \le \delta\left(\frac{|\mathcal{T}|^2}{2}+\frac{6|\mathcal{T}|}{\varepsilon\varepsilon_1}+\frac{2}{\varepsilon\varepsilon_1}\right)+\frac{4\delta|\mathcal{T}|}{\varepsilon_1}m.$$

Therefore, for $(Q, D) := \texttt{greedyCPack}(\widetilde{I}, P, x^*, y^*)$, we get

$$a(D) < \frac{52|\mathcal{T}|\varepsilon_2}{\varepsilon_1}m + 4\varepsilon_2\left(\frac{|\mathcal{T}|^2}{2}+\frac{6|\mathcal{T}|}{\varepsilon\varepsilon_1}+\frac{2}{\varepsilon\varepsilon_1}\right).$$

C.3 Pseudocode for `skewedCPack`

Algorithm 1 $\texttt{skewedCPack}_\varepsilon(I)$: Packs a set I of δ-skewed rectangular items into bins without rotating the items.

```
(Ĩ, I_med) = round_ε(I).
Initialize Q_best to null.
for P ∈ iterPackings(Ĩ) do                    // iterPackings is defined in Section 4.3.1.
    x* = opt(FP_W(Ĩ, P)).                     // FP_W and FP_H are defined in Section 4.3.2.
    // If FP_W(Ĩ, P) is feasible, x* is an extreme-point solution to FP_W(Ĩ, P).
    // If FP_W(Ĩ, P) is infeasible, x* is null.
    y* = opt(FP_H(Ĩ, P)).
    if x* ≠ null and y* ≠ null then           // if Ĩ can be packed into P
        (Q, D) = greedyCPack(Ĩ, P, x*, y*).   // greedyCPack is defined in Section 4.3.3.
        Q_D = NFDH(D ∪ I_med).
        if Q ∪ Q_D uses less bins than Q_best then
            Q_best = Q ∪ Q_D.
        end if
    end if
end for
return Q_best
```

D Lower Bound on APoG

In this section, we prove a lower bound of roughly 4/3 on the APoG for skewed rectangles.

▶ **Lemma 30.** *Let m and k be positive integers and $\varepsilon \in (0, 1)$. Let J be a set of items packed into a bin, where each item has the longer dimension equal to $(1+\varepsilon)/2$ and the shorter dimension equal to $(1-\varepsilon)/2k$. If the bin is guillotine-separable, then $a(J) \le 3/4+\varepsilon/2-\varepsilon^2/4$.*

Proof sketch. For an item packed in the bin, if the height is $(1-\varepsilon)/2k$, call it a wide item, and if the width is $(1-\varepsilon)/2k$, call it a tall item. Let W be the set of wide items in J.

We can rearrange the items in the bin so that all wide items touch the left edge of the bin and all tall items touch the bottom edge of the bin. See Appendix E in [31] for a formal proof and Figure 6 for an example.

Therefore, the square region of side length $(1-\varepsilon)/2$ at the top-right corner of the bin is empty. Hence, the area occupied in each bin is at most $3/4+\varepsilon/2-\varepsilon^2/4$. ◀

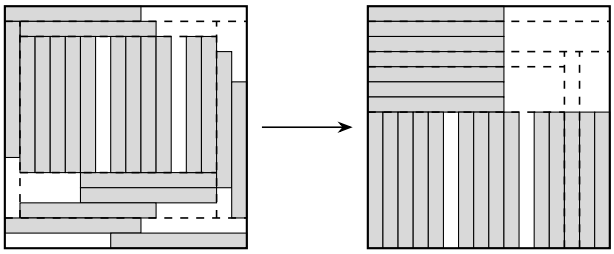


Figure 6 Structuring a guillotine-separable packing.

▶ **Theorem 31.** *Let m and k be positive integers and $\varepsilon \in (0,1)$. Let I be a set of $4mk$ items, where $2mk$ items have width $(1+\varepsilon)/2$ and height $(1-\varepsilon)/2k$, and $2mk$ items have height $(1+\varepsilon)/2$ and width $(1-\varepsilon)/2k$. Let $\mathrm{opt}(I)$ be the number of bins in the optimal packing of I and $\mathrm{opt}_g(I)$ be the number of bins in the optimal guillotinable packing of I. Then*

$$\frac{\mathrm{opt}_g(I)}{\mathrm{opt}(I)} \geq \frac{4}{3}(1-\varepsilon).$$

This holds true even if items in I are allowed to be rotated.

Proof. For an item $i \in I$, if $h(i) = (1-\varepsilon)/2k$, call it a wide item, and if $w(i) = (1-\varepsilon)/2k$, call it a tall item. Let W be the set of wide items and H be the set of tall items.

Partition W into groups of k elements. In each group, stack items one-over-the-other. This gives us $2m$ containers of width $(1+\varepsilon)/2$ and height $(1-\varepsilon)/2$. Similarly, get $2m$ containers of height $(1+\varepsilon)/2$ and height $(1-\varepsilon)/2$ by stacking items from H side-by-side. We can pack 4 containers in one bin, so I can be packed into m bins. See Figure 7 for an example. Therefore, $\mathrm{opt}(I) \leq m$.

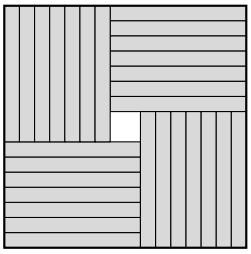


Figure 7 Packing $4k$ items in one bin. Here $k = 7$.

We will now show a lower-bound on $\mathrm{opt}_g(I)$. In any guillotine-separable packing of I, the area occupied by each bin is at most $3/4 + \varepsilon/2 - \varepsilon^2/4$ (by Lemma 30). Note that $a(I) = m(1-\varepsilon^2)$. Therefore,

$$\mathrm{opt}_g(I) \geq \frac{m(1-\varepsilon^2)}{3/4 + \varepsilon/2 - \varepsilon^2/4}$$
$$\implies \frac{\mathrm{opt}_g(I)}{\mathrm{opt}(I)} \geq \frac{4}{3} \times \frac{1-\varepsilon^2}{1 + 2\varepsilon/3 - \varepsilon^2/3} = \frac{4}{3} \times \frac{1-\varepsilon}{1-\varepsilon/3} \geq \frac{4}{3}(1-\varepsilon). \qquad \blacktriangleleft$$