Tight Approximation Algorithms for 2D Guillotine Strip Packing

ARINDAM KHAN, Indian Institute of Science, Bangalore, India
ADITYA LONKAR, Georgia Institute of Technology, Atlanta, Georgia, USA
ARNAB MAITI, University of Washington, Seattle, Washington, USA
AMATYA SHARMA, University of Michigan, Ann Arbor, Michigan, USA
ANDREAS WIESE, Technical University of Munich, Munich, Germany

In the Strip Packing (SP) problem, we are given a vertical half-strip $[0, W] \times [0, \infty)$ and a set of n axis-aligned rectangles of width at most W. The goal is to find a non-overlapping packing of all rectangles into the strip such that the height of the packing is minimized. A well-studied and frequently used practical constraint is to allow only those packings that are guillotine separable, i.e., every rectangle in the packing can be obtained by recursively applying a sequence of edge-to-edge axis-parallel cuts (guillotine cuts) that do not intersect any item of the solution. In this article, we study approximation algorithms for the Guillotine Strip Packing (GSP) problem, i.e., the SP problem where we require additionally that the packing needs to be guillotine separable. This problem generalizes the classical Bin Packing problem and also makespan minimization on identical machines, and thus it is already strongly NP-hard. Moreover, due to a reduction from the Partition problem, it is NP-hard to obtain a polynomial-time $(3/2 - \varepsilon)$-approximation algorithm for GSP for any $\varepsilon > 0$ (exactly as SP). We provide a matching polynomial time $(3/2 + \varepsilon)$-approximation algorithm for GSP. Furthermore, we present a pseudo-polynomial time $(1 + \varepsilon)$-approximation algorithm for GSP. This is surprising as it is NP-hard to obtain a $(5/4 - \varepsilon)$-approximation algorithm for (general) SP in pseudo-polynomial time. Thus, our results essentially settle the approximability of GSP for both the polynomial and the pseudo-polynomial settings.

CCS Concepts: • **Theory of computation** → **Packing and covering problems**; *Computational geometry*;

Additional Key Words and Phrases: Approximation Algorithms, Two-Dimensional Packing, Rectangle Packing, Guillotine Cuts, Computational Geometry

ACM Reference format:
Arindam Khan, Aditya Lonkar, Arnab Maiti, Amatya Sharma, and Andreas Wiese. 2025. Tight Approximation Algorithms for 2D Guillotine Strip Packing. *ACM Trans. Algor.* 21, 3, Article 28 (July 2025), 30 pages.
https://doi.org/10.1145/3736723

A. Khan's research is supported in part by the Google India Research Award, the SERB Core Research Grant (CRG/2022/001176) on "Optimization under Intractability and Uncertainty," the Ittiam Systems CSR grant, and the Walmart Center for Tech Excellence at IISc (CSR Grant WMGT-23-0001)."

Authors' Contact Information: Arindam Khan, Indian Institute of Science, Bangalore, India; e-mail: arindamkhan@iisc.ac.in; Aditya Lonkar (corresponding author), Georgia Institute of Technology, Atlanta, Georgia, USA; e-mail: alonkar9@gatech.edu; Arnab Maiti, University of Washington, Seattle, Washington, USA; e-mail: arnabm2@uw.edu; Amatya Sharma, University of Michigan, Ann Arbor, Michigan, USA; e-mail: amatya@umich.edu; Andreas Wiese, Technical University of Munich, Munich, Germany; e-mail: andreas.wiese@tum.de.

1 Introduction

2D packing problems form a fundamental research area in combinatorial optimization, computational geometry, and approximation algorithms. They find numerous practical applications in logistics [10], cutting stock [27], VLSI design [30], smart-grids [22], and so on. The **Strip Packing (SP)** problem, a generalization of the classical Bin Packing problem and also the makespan minimization problem on identical machines, is one of the central problems in this area. We are given an axis-aligned vertical half-strip $[0, W] \times [0, \infty)$ and a set of n axis-aligned rectangles (also called *items*) $I := \{I(1), I(2), \ldots, I(n)\}$, where for each rectangle $I(i)$ we are given an integral width $w_i \leq W$, and an integral height h_i; we assume the rectangles to be open sets. For simplicity, from now on, we refer to each item $I(i)$ by just i. The goal is to pack all items such that the maximum height of the top edge of a packed item is minimized. The packing needs to be *non-overlapping*, i.e., such a packing into a strip of height H maps each rectangle $i \in I$ to a new translated open rectangle $R(i) := (left(i), right(i)) \times (bottom(i), top(i))$ where $right(i) = left(i) + w_i$, $top(i) = bottom(i) + h_i$, $left(i) \geq 0$, $bottom(i) \geq 0$, $right(i) \leq W$, $top(i) \leq H$ and for any $i, j \in I$, we must have $R(i) \cap R(j) = \emptyset$. We assume that items are not allowed to be rotated.

The best-known polynomial time approximation algorithm for SP has an approximation ratio of $(5/3 + \varepsilon)$ for any constant $\varepsilon > 0$ [28] and a straight-forward reduction from Partition shows that it is NP-hard to approximate the problem with a ratio of $(3/2 - \varepsilon)$ for any $\varepsilon > 0$. Maybe surprisingly, one can approximate SP better in pseudo-polynomial time: There is a pseudo-polynomial time $(5/4 + \varepsilon)$-approximation algorithm [31] and it is NP-hard to obtain a $(5/4 - \varepsilon)$-approximation algorithm with this running time [29]. Hence, it remains open to close the gap between $(5/3 + \varepsilon)$ and $(3/2 - \varepsilon)$ for polynomial time algorithms, and even in pseudo-polynomial time, there can be no $(1 + \varepsilon)$-approximation for the problem for arbitrarily small $\varepsilon > 0$.

SP is particularly motivated by applications in which we want to cut out rectangular pieces of a sheet or stock unit of raw material, i.e., metal, glass, wood, or cloth, and we want to minimize the amount of wasted material. For cutting out these pieces in practice, axis-parallel end-to-end cuts, called *guillotine cuts*, are popular due to their simplicity of operation [47]. In this context, we look for solutions to cut out the individual objects by a recursive application of guillotine cuts that do not intersect any item of the solution. Starting from the classical work by Christofides and Whitlock [11] in the 1970s, settings with such guillotine cuts are widely studied in the literature [7, 12, 13, 16–18, 38, 48]. In particular, this motivates studying geometric packing problems with the additional constraint that the placed objects need to be separable by a sequence of guillotine cuts (see Figure 1). A related notion is k-stage packing, originally introduced by Gilmore and Gomory [27]. Here, each stage consists of either vertical or horizontal guillotine cuts (but not both). In each stage, each of the pieces obtained in the previous stage is considered separately and can be cut again by using either horizontal or vertical guillotine cuts. In k-stage packing, the number of cuts to obtain each rectangle from the initial packing is at most k, plus an additional cut to trim (i.e., separate the rectangles themselves from a waste area). Intuitively, this means that in the cutting process, we change the orientation of the cuts $k - 1$ times.

Therefore, in this article, we study the **Guillotine Strip Packing (GSP)** problem. The input is the same as for SP, but we require additionally that the items in the solution can be separated by a sequence of guillotine cuts, and we say then that they are *guillotine separable*. Like general SP without requiring the items to be guillotine separable, GSP generalizes Bin Packing (when all items have the same height) and makespan minimization on identical machines (when all items have the same width). Thus, it is strongly NP-hard, and the same reduction from Partition mentioned above yields a lower bound of $(3/2 - \varepsilon)$ for polynomial time algorithms (see Appendix Section A.5 for more details). For asymptotic approximation, GSP is well understood. Kenyon and Rémila [33]

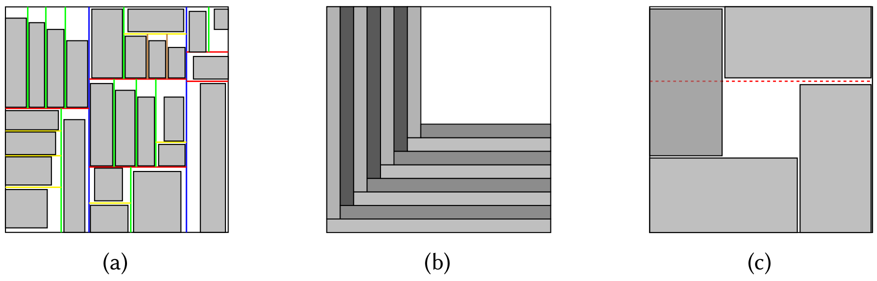


Fig. 1. Packing (a) is a 5-stage guillotine separable packing, packing (b) is a $n-1$-stage guillotine separable packing, and packing (c) is not guillotine separable as any end-to-end cut in the strip intersects a rectangle.

gave an **Asymptotic Polynomial Time Approximation Scheme (APTAS)** for (general) SP. Their algorithm produces a 5-stage packing (hence, guillotine separable), and thus yields an APTAS for GSP as well. Later, Seiden et al. [44] settled the asymptotic approximation status of GSP under k-stage packing. They gave an APTAS for GSP using 4-stage guillotine cuts, and showed $k = 2$ stages cannot guarantee any bounded asymptotic performance ratio, and $k = 3$ stages lead to asymptotic performance ratios close to 1.691. However, in the non-asymptotic setting, the approximation ratio of GSP is not yet settled. Steinberg's algorithm [46] yields a 2-approximation algorithm for GSP and this is the best known polynomial time approximation algorithm for the problem.

In this article, we present approximation algorithms for GSP that have strictly better approximation ratios than the best known algorithms for SP, and in the setting of pseudo-polynomial time algorithms we even beat the lower bound that holds for SP. Moreover, we show that all our approximation ratios are essentially the best possible.

1.1 Our Contribution

We present a polynomial time $(3/2 + \varepsilon)$-approximation algorithm for GSP. Due to the mentioned lower bound of $(3/2 - \varepsilon)$, our approximation ratio is essentially tight. Also, we present a pseudo-polynomial time $(1 + \varepsilon)$-approximation algorithm, which is also essentially tight since GSP is strongly NP-hard.

We first prove that there exists a structured solution with height at most $(1 + \varepsilon)$OPT (OPT denotes the height of the optimal solution) in which the strip is divided into $O(1)$ rectangular boxes inside which the items are *nicely packed*, e.g., horizontal items are stacked on top of each other, vertical items are placed side by side, and small items are packed greedily with the **Next Fit Decreasing Height (NFDH)** algorithm [14] (see Figure 2(a) and also Figure 4). This result starkly contrasts SP (i.e., where we do not require the items to be guillotine separable): For that problem, it is already unlikely that we can prove that there always exists such a packing with a height of less than $5/4 \cdot$ OPT. If we could prove this, we could approximate the problem in pseudo-polynomial time with a better ratio than 5/4, which is NP-hard [29].

To construct our structured packing, we start with an optimal packing and use the techniques in [35] to obtain a packing in which each item is *nicely packed* in one of a constant number of boxes and L-shaped *compartments*. We increase the height of our packing by $\varepsilon \cdot$ OPT in order to round the heights of the packed items and get some leeway within the packing. Then, we rearrange the items placed inside the L-shaped compartments. Here, we crucially exploit that the items in the initial packing are guillotine separable. In particular, this property allows us to identify certain sets of items that we can swap, e.g., items on the left and the right of a vertical guillotine cut to simplify the packing and reduce the number of boxes to $O(1)$. Then, using standard techniques, we compute

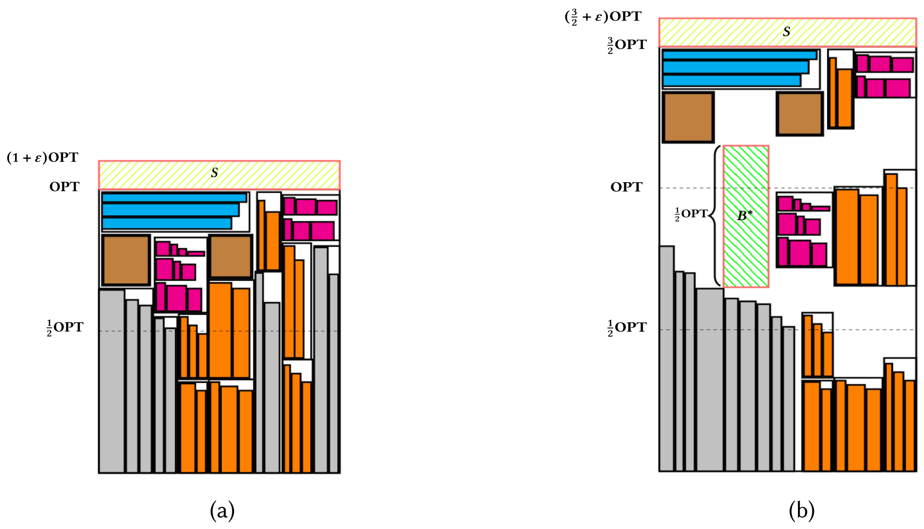


Fig. 2. (a) A guillotine separable structured packing (for the PPTAS) where all the items are packed nicely in containers. The tall items (dark gray) are stacked next to each other just like the vertical items (orange); the horizontal items (blue) are stacked on top of each other, the small items (pink) are packed according to NFDH, and the large containers contain single large items (brown). (b) A guillotine separable structured packing for the polynomial time $(3/2 + \varepsilon)$-approximation, where the packing from (a) is rearranged such that the tall items are *bottom-left-flushed* and there is an extra empty box B^* to accommodate for errors in packing the vertical items. The yellow rectangular strip S on top of both of the packings is used for packing the medium and leftover horizontal and small items.

a solution with this structure in pseudo-polynomial time and hence with a packing height of at most $(1 + \varepsilon)$OPT (see Figure 2(a)).

Note that we do not obtain a $(1+\varepsilon)$-approximation algorithm in polynomial time in this way. The reason is that when we pack the items into the rectangular boxes, we need to solve a generalization of PARTITION: There can be several boxes in which vertical items are placed side by side, and we need the widths of the items in each box to sum up to at most the width of the box. If there is only a single item that we cannot place, then we would need to place it on top of the packing, which can increase our packing height by up to OPT.

For our polynomial time $(3/2+\varepsilon)$-approximation algorithm, we, therefore, need to be particularly careful with the items whose height is larger than OPT/2, which we call the *tall items*. We prove a different structural result which is the main technical contribution of this article: We show that there is always a $(3/2 + \varepsilon)$-approximate packing in which the tall items are packed together in a *bottom-left-flushed* way, i.e., they are ordered non-increasingly by height and stacked next to each other with their bottom edges touching the base of the strip. All remaining items are nicely packed into $O_\varepsilon(1)$ boxes, and there is also an empty strip of height OPT/2 and width $\Omega_\varepsilon(W)$; see Figure 2(b). Thus, it is very easy to pack the tall items correctly according to this packing. We pack the remaining items with standard techniques into the boxes. In particular, the mentioned empty strip allows us to make slight mistakes while we pack the vertical items that are not tall; without this, we would still need to solve a generalization of PARTITION.

In order to obtain our structural packing for our polynomial time $(3/2 + \varepsilon)$-approximation algorithm, we build on the packing for the pseudo-polynomial time $(1 + \varepsilon)$-approximation. Using the fact that it is guillotine separable, we rearrange its items further. First, we move the items such that all tall items are at the bottom. To achieve this, we again argue that we can swap certain sets of items, guided by the guillotine cuts. Then, we shift certain items up by OPT/2, which leaves empty space between the shifted and the not-shifted items, see Figure 2(b). Inside this empty space, we place the empty box of height OPT/2. Also, we use this empty space in order to be able to reorder the tall items on the bottom by their respective heights. During these changes, we ensure carefully that the resulting packing stays guillotine separable.

Now, we briefly compare our approach with previous results in GSP and SP. The (previous best) polynomial time 2-approximation algorithm for GSP by Steinberg [46] uses a simple recursive constructive approach to pack items. The algorithm uses the total area of items as a lower bound on OPT (divided by the width of the half-strip W) and, for many instances, the height of their packing can be a factor 2 apart. Thus to go beyond 2-approximation, we go beyond the area-argument and exploit the structure of the optimal solution. Even for SP, the (best known) polynomial time $(5/3 + \varepsilon)$ approximation algorithm by Harren et al. [28] uses a similar idea. Their algorithm initially approximately guesses the optimal height OPT and then packs almost all rectangles, except a small subset of tall rectangles (which can even have height close to OPT). To repack these remaining rectangles, they create a hole of size OPT/3 of small width in the initial packing and pack the rest of the unpacked rectangles therein. Therefore, their approach inherently reaches a barrier of 5/3 approximation and it is unclear how to go beyond 5/3 by their approach. We crucially exploit several properties of guillotine packings to show the existence of a refined structured packing where we can pack everything into a height of at most $(3/2 + \varepsilon)$OPT.

We remark here that the most important guillotine geometric packing problems are **Guillotine 2D Bin Packing (2GBP)**, **Guillotine 2D Geometric Knapsack (2GGK)**, and GSP. Our results essentially resolve the status of one of them, i.e., GSP in the polynomial and pseudo-polynomial time regimes. Further, it is possible that also for (general) SP there always exists a structured packing of height at most $(3/2 + \varepsilon)$OPT, similar to our packing. This would yield an essentially tight polynomial time $(3/2 + \varepsilon)$-approximation for SP and thus solve the long-standing open problem to find the best possible polynomial time approximation ratio for SP. We leave this as an open question.

1.2 Other Related Work

In the 1980s, Baker et al. [3] initiated the study of approximation algorithms for SP, by giving a 3-approximation algorithm. After a sequence of improved approximations [14, 45], Steinberg [46] and Schiermeyer [43] independently gave 2-approximation algorithms. For asymptotic approximation, Kenyon and Rémila [33] settled SP by providing an APTAS.

SP has rich connections with important geometric packing problems [34] such as **2D Bin Packing (2BP)** [5, 37], **2D Geometric Knapsack (2GK)** [21, 32], dynamic storage allocation [8], **Maximum Independent Set of Rectangles (MISR)** [2, 25], sliced packing [15, 20], and so on. We refer the readers to the survey by Christensen et al. [10] for more details.

In 2BP, we are given a set of rectangles and square bins, and the goal is to find an axis-aligned non-overlapping packing of all items into a minimum number of bins. The problem admits no APTAS [4], and the present best approximation ratio is 1.406 [5]. In 2GK, we are given a set of rectangular items and a square knapsack. Each item has an associated profit, and the goal is to pack a subset of items in the knapsack such that the profit is maximized. The present best polynomial time approximation ratio is 1.89 [21]. There is a pseudo-polynomial time $(4/3 + \varepsilon)$-approximation [23] for 2GK. In MISR, we are given a set of (possibly overlapping) rectangles we need to find the maximum cardinality non-overlapping set of rectangles. Recently, Mitchell [39] gave the first constant factor approximation algorithm for the problem. Then Gálvez et al. [24] obtained a $(2 + \varepsilon)$-approximation algorithm for MISR. Their algorithms are based on a recursive geometric decomposition of the

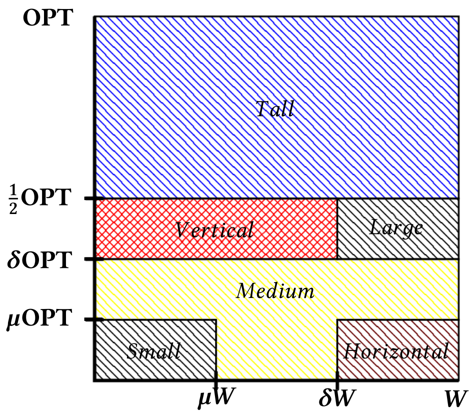


Fig. 3. Item classification: the x-axis represents width and the y-axis represents height.

plane, which can be viewed as a generalization of guillotine cuts, more precisely, to cuts with $O(1)$ bends. Pach and Tardos [41] even conjectured that for any set of n non-overlapping axis-parallel rectangles, there is a guillotine cutting sequence separating $\Omega(n)$ of them. Recently, the conjecture was proven to be true for squares [1, 36].

2BP and 2GK are also well-studied in the guillotine setting [42]. Caprara et al. [9] gave an APTAS for 2-stage SP and 2-stage BP. Later, Bansal et al. [6] showed an APTAS for 2GBP. Bansal et al. [5] conjectured that the worst-case ratio between the best 2GBP and the best general 2BP is 4/3. If true, this would imply a $(\frac{4}{3} + \varepsilon)$-approximation algorithm for 2BP. For 2GGK, Khan et al. [35] recently gave a **Pseudo-polynomial Time Approximation Scheme (PPTAS)**.

2 PPTAS

In this section, we present our PPTAS for GSP.

Let $\varepsilon > 0$ and assume without loss of generality that $1/\varepsilon \in \mathbb{N}$. We denote by OPT the height of an optimal solution. Note that, in polynomial time, we can guess OPT to within an $(1 + \varepsilon)$-approximation. In fact, in pseudo-polynomial time we can even guess OPT *exactly*. We classify the input items into a few groups according to their heights and widths. For two constants $1 \geq \delta > \mu > 0$ to be defined later, we classify each item $i \in I$ as:

- *tall* if $h_i > \text{OPT}/2$;
- *large* if $w_i > \delta W$ and $\text{OPT}/2 \geq h_i > \delta\text{OPT}$;
- *horizontal* if $w_i > \delta W$ and $h_i \leq \mu\text{OPT}$;
- *vertical* $w_i \leq \delta W$ and $\text{OPT}/2 \geq h_i > \delta\text{OPT}$;
- *medium if*
  - either $\delta\text{OPT} \geq h_i > \mu\text{OPT}$;
  - or $\delta W \geq w_i > \mu W$ and $h_i \leq \mu\text{OPT}$;
- *small* if $w_i \leq \mu W$ and $h_i \leq \mu\text{OPT}$.

See Figure 3 for a picture of item classification. Let $I_{tall}, I_{large}, I_{hor}, I_{ver}, I_{medium}, I_{small}$ be the set of tall, large, horizontal, medium, and small rectangles in I, respectively.

Using the following lemma, one can appropriately choose μ, δ such that the medium items occupy a marginal area. This effectively allows us to ignore them in our main argumentation.

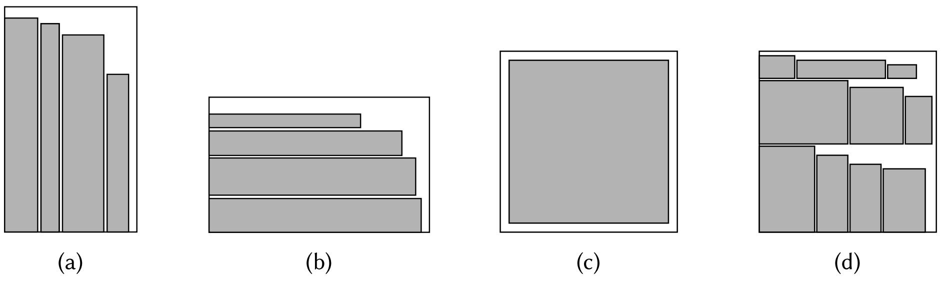


Fig. 4. Nice packing of vertical, horizontal, large, and small items in their respective containers.

Lemma 2.1 [40]. *Let $\varepsilon > 0$ and $f(\cdot)$ be any positive increasing function such that $f(x) < x$ for all $x \in (0,1]$. Then we can efficiently find $\delta, \mu \in \Omega_\varepsilon(1)$, with $\varepsilon \geq f(\varepsilon) \geq \delta \geq f(\delta) \geq \mu$ so that the total area of medium rectangles is at most $\varepsilon(\mathrm{OPT} \cdot W)$.*

We will specify how we choose the function $f(x)$ later in Section 4. In our PPTAS, we will use a packing that is defined solely via boxes.

Definition 2.2. A *box* B is an axis-aligned open rectangle that satisfies $B \subseteq [0, W] \times [0, \infty)$. We denote by $h(B)$ and $w(B)$ the height and the width of B, respectively.

Inside each box B, we will place the items *nicely,* meaning that they are either stacked horizontally or vertically, or B contains a single large item, or only small items, or only medium items. This is useful since in the first two cases, it is trivial to place a given set of items into B, and in the last two cases, it will turn out that it suffices to pack the items greedily using the NFDH algorithm [14] and Steinberg's algorithm [46], respectively. There will be one box with height at most $2\varepsilon \cdot \mathrm{OPT}$ that contains all medium items.

Definition 2.3 (Nice Packing). Let B be a box and let $I_B \subseteq I$ be a set of items that are placed non-overlappingly inside B. We say that the packing of I_B in B is *nice* if the items in I_B are guillotine separable and additionally

— I_B contains only one item, or
— $I_B \subseteq I_{hor}$ and the items in I_B are stacked on top of each other inside B, or
— $I_B \subseteq I_{tall} \cup I_{ver}$ and the items in I_B are placed side by side inside B, or
— $I_B \subseteq I_{medium}$, or
— $I_B \subseteq I_{small}$ and for each item $i \in I_B$ it holds that $w_i \leq \varepsilon \cdot w(B)$ and $h_i \leq \varepsilon \cdot h(B)$.

We will use the term *container* to refer to a box B that contains a nice packing of some set of items I_B. See Figure 4 for nice packings in different types of containers. We say that a set of boxes $\mathcal{B}$ is *guillotine separable* if there exists a sequence of guillotine cuts that separates them and that does not intersect any box in $\mathcal{B}$.

Definition 2.4 (Guillotine Cut Stages). A packing is said to be a *k-stage* packing if the number of (guillotine) cuts to obtain each packed rectangle from the initial packing is at most k (plus an extra trimming step to separate the rectangle itself from a waste area).

The cuts in different stages alternate between vertical and horizontal cuts. See Figure 1 for an example.

We now state the structural lemma for the PPTAS. Intuitively, it states that there exists a $(1+\varepsilon)$-approximate solution in which the input items are placed into $O_\varepsilon(1)$ boxes such that within each box the packing is nice. We remark that we will crucially use that in the optimal packing, the items in I are guillotine separable. In fact, if one could prove that there exists such a packing with $O_\varepsilon(1)$ boxes and a height of $\alpha \cdot \mathrm{OPT}$ for some $\alpha < \frac{5}{4}$ also in the non-guillotine case (where neither the optimal solution nor the computed solution needs to be guillotine separable), then one would obtain a pseudo-polynomial time $(\alpha + \varepsilon)$-approximation algorithm also in this case, by using straightforward adaptations of the algorithms in, e.g., [22, 31, 40] or our PPTAS. However, this is not possible for $\alpha < \frac{5}{4}$, unless P = NP [29].

Lemma 2.5 (Structural Lemma 1). *Assume that μ is sufficiently small compared to δ. Then there exists a set $\mathcal{B}$ of $O_\varepsilon(1)$ pairwise non-overlapping and guillotine separable boxes all placed inside* $[0, W] \times [0, (1 + 16\varepsilon)\mathrm{OPT})$ *and a partition $I = \bigcup_{B \in \mathcal{B}} I_B$ such that for each $B \in \mathcal{B}$ the items in I_B can be placed nicely into B.*

We choose our function f due to Lemma 2.1 such that μ is sufficiently small compared to δ, as required by Lemma 2.5. We will prove Lemma 2.5 in the next subsection. In its packing, let $\mathcal{B}_{hor}, \mathcal{B}_{ver}, \mathcal{B}_{tall}, \mathcal{B}_{large}, \mathcal{B}_{small}$, and $\mathcal{B}_{med}$ denote the set of boxes for the horizontal, vertical, tall, large, small, and medium items, respectively. Let $\mathcal{B}_{tall+ver} := \mathcal{B}_{tall} \cup \mathcal{B}_{ver}$.

2.1 Proof of Structural Lemma 1

In this section, we prove Lemma 2.5. Our strategy is to start with a structural lemma from [35] that guarantees the existence of a structured packing of all items in $I_{hard} := I_{tall} \cup I_{large} \cup I_{hor} \cup I_{ver}$. This packing uses boxes and **L**-compartments. Note that, for now, we ignore the items I_{small}. We will show how to pack them later.

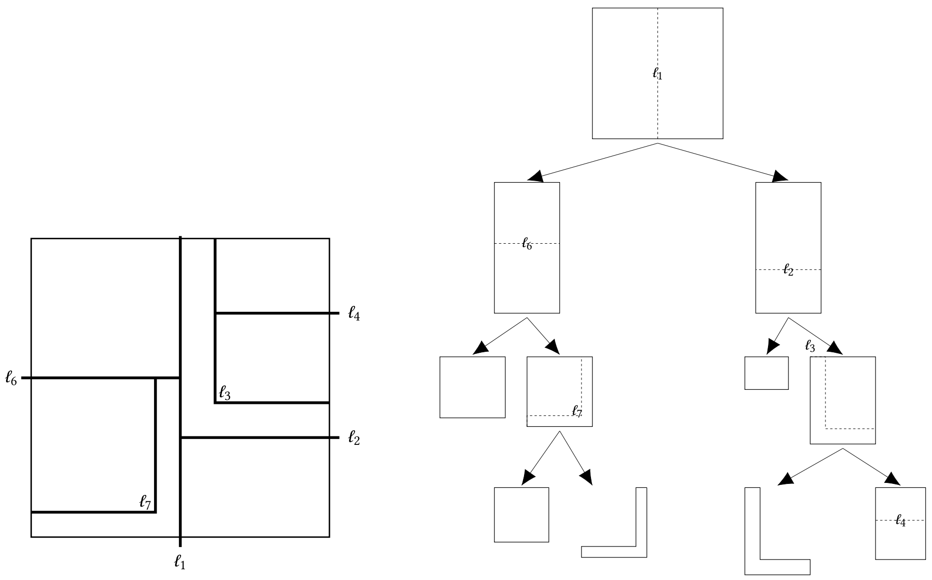
*Definition 2.6 (**L**-Compartment).* An **L**-compartment L is an open sub-region of $[0, W] \times [0, \infty)$ bounded by a simple rectilinear polygon with six edges $e_0, e_1, \ldots, e_5$ (one can arbitrarily choose an edge to be e_0 and then take the edges in the polygon in a counterclockwise manner) such that for each pair of horizontal (resp. vertical) edges e_i, e_{6-i} with $i \in \{1, 2\}$ there exists a vertical (resp. horizontal) line segment ℓ_i of length less than $\delta \cdot \frac{\mathrm{OPT}}{2}$ (resp. $\delta \cdot \frac{W}{2}$) such that both e_i and e_{6-i} intersect ℓ_i but no other edges intersect ℓ_i.

Let e_i (resp., e_j) be the largest horizontal (resp. vertical) edge in an **L**-compartment L, for some $i, j \in [6]$. Then the largest rectangle R_h (resp., R_v) that is contained inside L and contains e_i (resp., e_j) is called the horizontal (resp. vertical) arm of L. Note that $L = R_h \cup R_v$, the width of R_v is at most $\delta \cdot \frac{W}{2}$, and the height of R_h is at most $\delta \cdot \frac{\mathrm{OPT}}{2}$. Hence, for an **L**-compartment, no item $i \in I_{hor}$ can be packed in its vertical arm and similarly, no item $i \in I_{ver} \cup I_{tall}$ can be packed in its horizontal arm.

The next lemma follows immediately from a structural insight in [35] for the guillotine 2D knapsack problem. It partitions the region $[0, W] \times [0, \mathrm{OPT}]$ into non-overlapping boxes and **L**-compartments that admit a *pseudo-guillotine cutting sequence*. This is a sequence of cuts in which each cut is either a (normal) guillotine cut, or a special cut that cuts out an **L**-compartment L from the current rectangular piece R in the cutting sequence, such that $R \setminus L$ is a rectangle, see Figure 5. Thus, intuitively L lies at the boundary of R.

Lemma 2.7 [35]. *There exists a partition of* $[0, W] \times [0, \mathrm{OPT}]$ *into a set $\mathcal{B}_1$ of $O_\varepsilon(1)$ boxes and a set $\mathcal{L}$ of $O_\varepsilon(1)$ **L**-compartments such that*

- *the boxes and **L**-compartments in $\mathcal{B}_1 \cup \mathcal{L}$ are pairwise non-overlapping,*
- *$\mathcal{B}_1 \cup \mathcal{L}$ admits a pseudo-guillotine cutting sequence,*
- *the items in I_{hard} can be packed into $\mathcal{B}_1 \cup \mathcal{L}$ such that for each $B \in \mathcal{B}_1$ it either contains only items $i \in I_{tall} \cup I_{large} \cup I_{ver}$ or it contains only items $i \in I_{hor}$.*



(a) (b)

Fig. 5. (a) A pseudo-guillotine cutting sequence. The first cut is l_1, and then the resulting right piece is further subdivided by ℓ_2, ℓ_3, and ℓ_4. Similarly, ℓ_6, ℓ_7 subdivide the left piece. Note that ℓ_3 and ℓ_7 are not guillotine cuts, but they cut out the corresponding **L**-compartments. (b) Step-by-step pseudo-guillotine cutting sequence corresponding to (a). A dashed line at each level indicates a partition of a rectangle into two regions (two boxes, or one box and one **L**).

Our strategy is to take the packing due to Lemma 2.7 and transform it step by step until we obtain a packing that corresponds to Lemma 2.5. First, we round the heights of the tall, large, and vertical items such that they are integral multiples of $\delta^2\mathrm{OPT}$. Formally, for each item $i \in I_{tall} \cup I_{large} \cup I_{ver}$ we define a rounded height $h'_i := \left\lceil \frac{h_i}{\delta^2\mathrm{OPT}} \right\rceil \delta^2\mathrm{OPT}$. Let I'_{hard} denote the resulting set of items, i.e., I'_{hard} contains all items in $I_{tall} \cup I_{large} \cup I_{ver} \cup I_{hor}$ and furthermore, for each $i \in I_{tall} \cup I_{large} \cup I_{ver}$ the set I'_{hard} contains an item i with height h'_i and width w_i. By a shifting argument, we will show that we can still pack I'_{hard} into $O_\varepsilon(1)$ guillotine separable boxes and **L**-compartments if we can increase the height of the packing by a factor $1+\varepsilon$ which also does not violate guillotine separability. Then, we increase the height of the packing by an additional $\varepsilon \cdot \mathrm{OPT}$. Using this additional space, we shift the items inside each **L**-compartment L such that we can separate the vertical items from the horizontal items (see Figure 6). Due to this separation, we can partition L into $O_\varepsilon(1)$ boxes such that each box contains only horizontal or only vertical and tall items. Note, however, that they might not be packed nicely inside these boxes. We will refer to these boxes as **H**-compartments and **V**-compartments. Here, **H**-compartments are boxes with height at most $\delta \cdot \frac{\mathrm{OPT}}{2}$ and **V**-compartments are boxes with width at most $\delta \cdot \frac{W}{2}$.

Lemma 2.8. *There exists a partition of* $[0, W] \times [0, (1+2\varepsilon)\mathrm{OPT}]$ *into a set* $\mathcal{B}_2$ *of* $O_\varepsilon(1)$ *boxes such that*

— *the boxes in $\mathcal{B}_2$ are pairwise non-overlapping and admit a guillotine cutting sequence,*
— *the items in I'_{hard} can be packed into $\mathcal{B}_2$ such that they are guillotine separable and each box $B \in \mathcal{B}_2$ either contains only items from $I_{tall} \cup I_{large} \cup I_{ver}$, or contains only items from I_{hor},*
— *any item $i \in I_{tall} \cup I_{large} \cup I_{ver}$ has height $h'_i = k_i\delta^2\mathrm{OPT}$ for integer k_i, $k_i \leq 1/\delta^2 + 1$.*

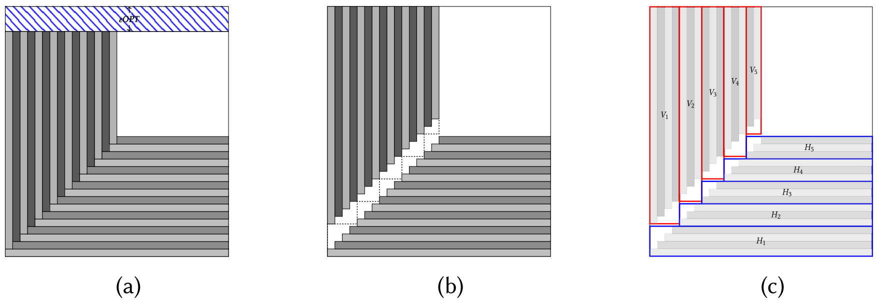


Fig. 6. Using an extra εOPT height, we convert a packing of items I in an $\mathbf{L}$-compartment into another packing such that the items in I are packed in boxes $C' = \mathbf{V} \cup \mathbf{H}$, which are guillotine separable and $|C'| = O_\varepsilon(1)$, where $\mathbf{V} = \cup_{i=1}^{i=5} V_i$ and $\mathbf{H} = \cup_{i=1}^{i=5} H_i$.

Proof. The proof consists of two steps. The first step is to process the individual $\mathbf{L}$-compartments (from Lemma 2.7) to obtain $O_\varepsilon(1)$ $\mathbf{V}$-compartments. Intuitively we do this by shifting the vertical arm of the $\mathbf{L}$ by $O(\varepsilon)$OPT amount vertically and allowing division into a constant number of new $\mathbf{V}$-compartments. In the second step, we obtain the packing of items in $O_\varepsilon(1)$ $\mathbf{H}$-compartments. Note that, from the definition, a $\mathbf{H}$-compartment cannot contain items in $I_{tall} \cup I_{ver}$ and a $\mathbf{V}$-compartment cannot contain items in I_{hor}.

We now explain the process in detail by considering an $\mathbf{L}$-compartment L defined by a simple rectilinear polygon with six edges $e_0, e_1, \ldots, e_5$ as given in Definition 2.6. Let the vertical and horizontal arms be L_V and L_H, respectively. Without loss of generality, we can assume that the horizontal arm lies on the bottom right of the vertical one (other cases can be handled analogously). Let p_1 be the bottommost point where L_V and L_H intersect. Let e_5 (resp. e_4) be the shorter vertical (resp. horizontal) edge of L_V (resp. L_H). Let the height and width of L be $h(L)$, $w(L)$, respectively.

We first shift each item in L_V vertically by an $\varepsilon \cdot h(L)$ amount; see Figure 6. We then create the boundary curve with $\frac{2}{\varepsilon}$ bends. We begin at p_1, i.e., the leftmost bottommost point of L. Using a ray shooting argument, continue drawing in the horizontal direction till the ray hits a horizontal item. Then we bend the ray and move it upwards till it hits one of the vertical items. Recursively repeating this process till the ray hits one of the bounding edges e_5 or e_4. Let this point be p_2. The trace of the ray defines the boundary curve $C_{H,V}$ between L_H and L_V, starting at p_1 and ending at p_2. Now we argue that the number of bends in the given boundary curve $C_{H,V}$ is at most $\frac{4}{\varepsilon}$. It is clear that the number of bends in $C_{H,V}$ will be twice as much as the number of distinct vertical paths in the curve $C_{H,V}$. Since each vertical item is shifted vertically by an $\varepsilon h(L)$ height initially, each such vertical path will be at least $\varepsilon h(L)$ in length. This establishes that the number of such distinct vertical paths can be at most $\frac{1}{\varepsilon}$ which bounds the number of bends by at most $2 \cdot \frac{1}{\varepsilon} + 2 \leq \frac{4}{\varepsilon}$ bends.

We then further create the vertical compartments by first extending the projections from bends of boundary curve $C_{H,V}$ in the vertical direction. If the vertical projection does not intersect with any of the vertical items, this can be considered a guillotine cut and used to separate the items. If any such vertical projection intersects any item in L_V, we cannot divide the vertical arm using this line. Instead, we exploit the fact that the vertical arm has the first stage of guillotine cuts separating them since the $\mathbf{L}$ considered is pseudo-guillotine separable. Now, we instead consider the two nearest consecutive guillotine cuts, one on the immediate left and the other on the immediate right of the vertical projection. The arrangements in previous steps and pseudo-guillotine separability ensure that any two consecutive guillotine cuts in the vertical arm are at the same level and form a box. This divides the vertical arm into one additional vertical box. Thus, at the end of this procedure, we will have at most $\frac{24}{\varepsilon}$ vertical compartments.

Similarly, we extend the horizontal projections from bends of boundary curve $C_{H,V}$ dividing the horizontal arm L_H into $O_\varepsilon(1)$ $\mathbf{H}$-compartments.

We do the procedure for all the $\mathbf{L}$-compartments which are at most $8/\delta$ many. The total height added considering $\varepsilon \cdot h(L)$ vertical shifts corresponding to each L which is an $\mathbf{L}$-compartment, can be at most $\varepsilon \cdot \mathrm{OPT}$ since the cumulative height of $\mathbf{L}$-compartments on the top of each other can at most be OPT. This gives us $O_\varepsilon(1)$ new $\mathbf{H}$ and $\mathbf{V}$-compartments. Since the items in $I_{tall} \cup I_{large} \cup I_{ver}$ have height at least $\delta \cdot \mathrm{OPT}$, there can be at most $1/\delta$ items in $I_{tall} \cup I_{large} \cup I_{ver}$ stacked on top of each other. Hence, rounding the heights of these items to multiples of $\delta^2\mathrm{OPT}$ can only increase the height of the packing by at most $\delta^2\mathrm{OPT}$ per item. Since there are at most $1/\delta$ items in $I_{tall} \cup I_{large} \cup I_{ver}$ stacked on top of each other, the overall increase in the height of the packing can be at most $\delta^2\mathrm{OPT} \cdot (1/\delta) \leq \delta \cdot \mathrm{OPT} \leq \varepsilon \cdot \mathrm{OPT}$. Hence, the total increase in the height of the packing is at most $2\varepsilon \cdot \mathrm{OPT}$. □

Let $\mathcal{B}_2$ be the set of boxes due to Lemma 2.8. Consider a box $B \in \mathcal{B}_2$ and let $I'_{hard}(B)$ denote the items from I'_{hard} that are placed inside B in the packing due to Lemma 2.8. Our goal is to partition B into $O_\varepsilon(1)$ smaller containers such that the items in $I'_{hard}(B)$ are packed nicely into these smaller boxes. If B contains horizontal items, then this can be done using standard techniques, e.g., by 1D resource augmentation (only in height) in [35]. This resource augmentation procedure maintains guillotine separability (see Appendix Section A.5).

Lemma 2.9 [35]. *Given a box $B \in \mathcal{B}_2$ such that B contains a set of items $I'_{hard}(B) \subseteq I_{hor}$. There exists a partition of B into $O_{\varepsilon'}(1)$ containers $\mathcal{B}'$ and one additional box B' of height at most $\varepsilon' h(B)$ and width $w(B)$ such that the containers $\mathcal{B}'$ are guillotine separable and the containers $\mathcal{B}' \cup \{B'\}$ contain $I'_{hard}(B)$.*

We apply Lemma 2.9 to each box $B \in \mathcal{B}_2$ that contains a horizontal item. Consider the items that are contained in their respective boxes B'. We choose $\varepsilon' = \varepsilon$ and then their total area is at most $\varepsilon \cdot \mathrm{OPT} \cdot W$ and therefore, all such items can be packed in a box of height at most $2\varepsilon \cdot \mathrm{OPT}$ and width W using Steinberg's algorithm [46]. But since this will possibly not result in a nice packing, we apply resource augmentation (only along height) again to ensure that we get a nice packing of such horizontal items in $O_\varepsilon(1)$ containers which can all be packed in a box of height at most $3\varepsilon \cdot \mathrm{OPT}$ and width W (see Appendix Section A.5).

Consider now a box $B \in \mathcal{B}_2$ that contains at least one item from $I_{tall} \cup I_{large} \cup I_{ver}$. Let $I'_{hard}(B) \subseteq I_{tall} \cup I_{large} \cup I_{ver}$ denote the items packed inside B. We argue that we can rearrange the items in $I'_{hard}(B)$ such that they are nicely placed inside $O_\varepsilon(1)$ containers. In this step, we crucially use the fact that the items in $I'_{hard}(B)$ are guillotine separable.

Consider the guillotine cutting sequence for $I'_{hard}(B)$. It is useful to think of these cuts as being organized in *stages:* In the first stage, we do vertical cuts (possibly zero cuts). In the following stage, we take each resulting piece and apply horizontal cuts. In the next stage, we again take each resulting piece and apply vertical cuts, and so on. Since the heights of the items in $I'_{hard}(B)$ are rounded to multiples of $\delta^2\mathrm{OPT}$ we can assume without loss of generality that the y-coordinates of the horizontal cuts are all integral multiples of $\delta^2\mathrm{OPT}$ (possibly moving the items a little bit). Assume here for the sake of simplicity that $t = 1/\delta^2$ is an integer. Because of the rounding of heights of the items in $I'_{hard}(B)$, there are at most $(1/\delta^2 - 1)$ y-coordinates for making a horizontal cut. For a horizontal stage of cuts, for a rectangular piece, we define a *configuration vector* $(x_1, ., x_{t-1})$: For each $i \in [t-1]$ if there is a horizontal cut in the piece at $y = \delta^2\mathrm{OPT} \cdot i$, then $x_i = 1$, otherwise $x_i = 0$. Consider $y = 0$ to be the bottom of the rectangular piece. Therefore, in each horizontal stage,

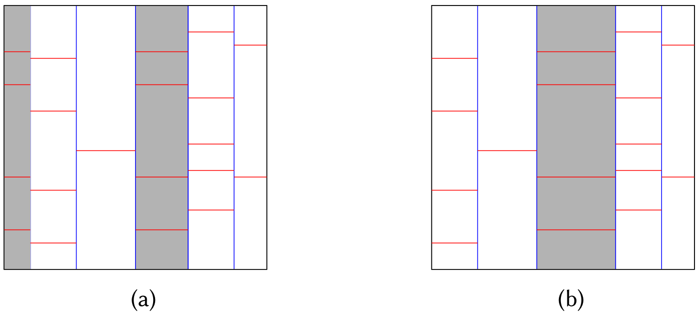


Fig. 7. (a) Two stages of guillotine cuts for a box containing vertical rectangles. (b) Since rounded heights of vertical rectangles are integral multiples of δ^2, merge configurations with the same set of horizontal cuts to get $O_\delta(1)$ configurations.

for each piece, there are at most $K := (2^{(1/\delta^2)})$ possible configurations. Consider the first stage (which has vertical cuts). If there are more than K vertical cuts then in two of the resulting pieces, in the second stage the same configuration of horizontal cuts is applied (see Figure 7). We reorder the resulting pieces and their items such that pieces with the same configuration of horizontal cuts are placed consecutively. Therefore, in the first stage we need only K vertical cuts and we can have at most $(\frac{1}{\delta} \cdot 2^{(1/\delta^2)})$ resulting pieces. We apply the same transformation to each stage with vertical cuts. Now observe that there can be at most $O(1/\delta)$ stages since there are at most $1/\delta$ possible vertical items stacked on top of the other and thus at most $1/\delta$ stages with horizontal cuts. Therefore, after our transformations, we apply only $(\frac{1}{\delta} \cdot 2^{(1/\delta^2)})^{\frac{1}{\delta}}$ cuts in total, in all stages in all resulting pieces. Thus, we obtain $O_\varepsilon(1)$ boxes at the end, in which the items are nicely packed. This leads to the following lemma.

Lemma 2.10. *Given a box $B \in \mathcal{B}_2$ such that B contains a set of items $I'_{hard}(B) \subseteq I_{tall} \cup I_{large} \cup I_{ver}$. There exists a partition of B into $O_\varepsilon(1)$ containers $\mathcal{B}'$ such that the containers $\mathcal{B}'$ are guillotine separable and contain the items $I'_{hard}(B)$.*

We apply Lemma 2.10 to each box $B \in \mathcal{B}_2$ that contains an item from $I_{tall} \cup I_{large} \cup I_{ver}$. Thus, we obtain a packing of I'_{hard} into a set of $O_\varepsilon(1)$ guillotine separable containers in which these items are nicely placed; we denote these containers by $\mathcal{B}_{hard}$. This yields directly a packing for the (original) items I_{hard}. Finally, we partition the empty space of the resulting packing into more boxes and one additional box that we place on top of the current packing. We pack the items in I_{small} inside all these boxes. We might not be able to use some parts of the empty space, e.g., if two boxes are closer than μW to each other horizontally; however, if μ is sufficiently small compared to the number of boxes, this space is small and compensated by the additional box.

Lemma 2.11. *Assume that μ is sufficiently small compared to δ. There exists a set of $O_\varepsilon(1)$ boxes $\mathcal{B}_{small}$, all contained in $[0, W] \times [0, (1 + 14\varepsilon)\mathrm{OPT}]$, such that the boxes in $\mathcal{B}_{hard} \cup \mathcal{B}_{small}$ are non-overlapping and guillotine separable and the items in I_{small} can be placed nicely into the boxes $\mathcal{B}_{small}$.*

Proof. Consider a vertical container B in which the items are nicely packed. Now we rearrange items such that items are placed side by side in a non-increasing order of their heights from left to right with no gap between the adjacent items and the items touch the bottom edge of B. Now group the items in B based on their heights. For all $i \in [(1-\delta)/\delta^2]$, group i contains the items with heights in the range $(\delta\mathrm{OPT} + (i-1) \cdot \mathrm{OPT} \cdot \delta^2, \delta\mathrm{OPT} + i \cdot \mathrm{OPT} \cdot \delta^2]$. Now for each group i, consider the minimal rectangular region containing all the items in group i and make it a container. Repeat a similar process for the horizontal containers too. Now the ratio of the area of the rectangles in a container to the area of the container is at least $1 - \delta$.

Now, we form a non-uniform grid by extending the edges of the containers until it hits a container or the edges of the half-strip $[0, W] \times [0, \infty)$. Note that the empty grid cells are guillotine separable as the containers when considered as pseudo-items are guillotine separable and the guillotine cuts coincide with one of the edges of the containers. Now we can choose μ, δ appropriately such that the total area of empty grid cells with height less than $\varepsilon'\mathrm{OPT}$ or width less than $\varepsilon' W$ is $\delta^2 \cdot \mathrm{OPT} \cdot W$ where $\varepsilon' = \mu/\varepsilon$. Note that the total number of containers that have height more than $\varepsilon'\mathrm{OPT}$ and width more than $\varepsilon' W$ is $O_\varepsilon(1)$. Now we can pack small rectangles using NFDH in the small containers. Then the total area of the small items not packed nicely is at most $3\varepsilon \cdot \mathrm{OPT} \cdot W$. Hence, using NFDH we can pack these remaining small rectangles of area in another container B_{small} of height at most $9\varepsilon \cdot \mathrm{OPT}$ and width W. Thus we have at most $O_\varepsilon(1)$ small containers which have a nice packing. Thus, to calculate the extra height on top of OPT, we have

(1) a box B_{small} of height $9\varepsilon \cdot \mathrm{OPT}$ and width W,
(2) additional containers in $\mathcal{B}_{hor}$ due to resource augmentation, which can be packed in a box of height $3\varepsilon \cdot \mathrm{OPT}$ and width W,
(3) an increase of $2\varepsilon \cdot \mathrm{OPT}$ to account for rounding the items in $I_{tall} \cup I_{large} \cup I_{ver}$ and shifting items in each **L**-compartment vertically upward according to Lemma 2.8. □

2.2 Algorithm

We describe now our algorithm that computes a packing of height at most $(1 + O(\varepsilon))\mathrm{OPT}$. First, we guess OPT and observe that there are at most $n \cdot h_{\max}$ possibilities, where $h_{\max} := \max_{i \in I} h_i$. Then, we guess the set of containers $\mathcal{B}$ due to Lemma 2.5 and their placement inside $[0, W] \times [0, (1 + O(\varepsilon))\mathrm{OPT})$. For each container $B \in \mathcal{B}$ we guess which case of Definition 2.3 applies to B, i.e., whether I_B contains only one item, $I_B \subseteq I_{hor}$, $I_B \subseteq I_{tall} \cup I_{ver}$, $I_B \subseteq I_{medium}$, or $I_B \subseteq I_{small}$. For each box $B \in \mathcal{B}$ for which I_B contains only one item $i \in I$, we guess i. Observe that for the remaining containers, this yields independent subproblems for the sets I_{hor}, $I_{tall} \cup I_{ver}$, I_{medium}, and I_{small}.

We solve these subproblems via similar routines as in [22, 31, 40]. For the sets I_{hor} and $I_{tall} \cup I_{ver}$ we pack their respective items into their containers using a standard pseudo-polynomial time dynamic program; we denote these containers by $\mathcal{B}_{hor}$ and $\mathcal{B}_{tall+ver}$, respectively. We crucially use that $|\mathcal{B}_{hor}| \le O_\varepsilon(1)$ and $|\mathcal{B}_{tall+ver}| \le O_\varepsilon(1)$.

From the proof of Lemma 2.11, apart from some items $I'_{small} \subset I_{small}$ which have an area at most $\varepsilon\mathrm{OPT} \cdot W$, the other items can be packed nicely in the containers in $\mathcal{B}_{small} \setminus B_{small}$, where B_{small} has height $9\varepsilon\mathrm{OPT}$ and width W. Thus, we use NFDH for packing the remaining small items. It can be shown that the small items that remain unpacked can be packed nicely in B_{small}, which is placed on the top of our packing. We pack all medium items in I_{medium} into one single container B_{med} of height $2\varepsilon \cdot \mathrm{OPT}$ by the following lemma.

Lemma 2.12. *In time $n^{O(1)}$ we can find a nice placement of all items in I_{medium} inside one container B_{med} of height $2\varepsilon \cdot \mathrm{OPT}$ and width W.*

Proof. By choosing the function f in Lemma 2.1 appropriately (see Section 4 for details on how to choose f), we ensure that the total area of medium items is at most $\varepsilon \cdot \mathrm{OPT} \cdot W$. Now, since the height of the medium items is at most $\delta \cdot \mathrm{OPT} \le \varepsilon \cdot \mathrm{OPT}$, we use Steinberg's algorithm (see Lemma A.6) to pack them in a container B_{med} of height $2\varepsilon \cdot \mathrm{OPT}$ and width W, i.e., according to

Lemma A.6, we choose $w = W$ and $h = 2\varepsilon \cdot \mathrm{OPT}$ and observe that $2h'_{\max} \leq 2\varepsilon \cdot \mathrm{OPT} = h$, where $h'_{\max}$ refers to the maximum height of a medium item. □

Lemma 2.13. *There is an algorithm with a running time of $(nh_{\max})^{|\mathcal{B}_{hor}|}$ that computes a packing of the items in I_{hor} into the containers $\mathcal{B}_{hor}$. Similarly, there is an algorithm with a running time of $(nW)^{|\mathcal{B}_{tall+ver}|}$ that computes a packing of the items in $I_{tall} \cup I_{ver}$ into the containers $\mathcal{B}_{tall+ver}$.*

Proof. We need to pack the items nicely in the containers for which we convert the instance to an instance of the Maximum **Generalized Assignment Problem (GAP)** with one bin per container and the size (area) of jth container B_j is $a(B_j) = w(B_j) \times h(B_j)$. We build an instance of GAP as follows. There is one item R per rectangle $R \in I$, with profit $a(R)$. For each horizontal container B_j, we create a knapsack j of capacity $C_j := h(B_j)$. Furthermore, we define the size $s(R, j)$ of a horizontal rectangle R with respect to knapsack j as $h(R)$ if $h(R) \leq h(B_j)$ and $w(R) \leq w(B_j)$. Otherwise $s(R, j) = \infty$ (meaning that R does not fit in B_j). The profit for rectangle R, $p_R = a(R)$ throughout all the bins. The construction for vertical containers is symmetric. We now use the exact algorithm for GAP mentioned in Lemma A.2 and finish the packing of all the vertical items in the respective containers in $(nW)^{|\mathcal{B}_{tall+ver}|}$ time and pack the horizontal items in the respective containers in $(nh_{\max})^{|\mathcal{B}_{hor}|}$ time. Note that the dependence on $h_{\max}$ in the running time of packing the horizontal items in their respective containers can be significantly reduced via Lemma 2.15. □

Finally, we need to pack the small items. Let $\mathcal{B}_{small} \subseteq \mathcal{B}$ denote the boxes in $\mathcal{B}$ that contain small items in the packing due to Lemma 2.5. Note that for each small item $i \in I_{small}$ only some of the boxes in $\mathcal{B}_{small}$ are allowed since we require that $w_i \leq \varepsilon \cdot w(B)$ and $h_i \leq \varepsilon \cdot h(B)$ if i is nicely packed inside a box B.

On a high level, we pack the small items in two steps, similarly as in [22, 31, 40]. First, we assign at least $I'_{small} \subset I_{small}$ items to the containers $\mathcal{B}_{small}$ such that for each box $B \in \mathcal{B}_{small}$, each item $i \in I_{small}$ assigned to B satisfies that $w_i \leq \varepsilon \cdot w(B)$ and $h_i \leq \varepsilon \cdot h(B)$ and the total area of the items assigned to B is at most $h(B) \cdot w(B)$, and the items not assigned have area at most $\varepsilon\mathrm{OPT} \cdot W$. Then, for each box $B \in \mathcal{B}_{small}$ we try to pack its assigned items greedily using NFDH [14, 19] (see Lemmas A.4 and A.5). For the items in I_{small} that remain unassigned (by our first step or by NFDH), we will show that their total area is at most $3\varepsilon \cdot \mathrm{OPT} \cdot W$. Therefore, we can pack them using NFDH into an additional container B_{small} of height $9\varepsilon \cdot \mathrm{OPT}$ and width W.

Lemma 2.14. *There is an algorithm with a running time of $(n|\mathcal{B}_{small}|)^{O(1)}$ that packs all items in I_{small} into $\mathcal{B}_{small}$ and an additional container B_{small} of height $9\varepsilon \cdot \mathrm{OPT}$ and width W.*

Proof. By Lemma 2.11 we know that all items in I_{small} can be nicely packed in $\mathcal{B}_{small}$. Hence, we convert this problem of nicely packing items in I_{small} into containers in $\mathcal{B}_{small}$ to an instance of GAP with one bin per container and the size of jth container B_j is $a(B_j) = w(B_j) \times h(B_j)$. We build an instance of GAP as follows. There is one item R per rectangle $R \in I_{small}$, with profit $a(R)$. For each container B_j, we create a knapsack j of size $S_j := a(B_j)$. Furthermore, we define the size $s(R, j)$ of a horizontal rectangle R with respect to knapsack j as $a(R)$ *iff* $h(R) \leq \varepsilon \cdot h(B_j)$ and $w(R) \leq \varepsilon \cdot w(B_j)$. Otherwise $s(R, j) = \infty$ (meaning that R is not small enough to nicely fit in B_j). The profit for rectangle R, $p_R = a(R)$ throughout all the bins. Similarly as in Lemma 2.13, we now use the exact algorithm for GAP which runs in pseudo-polynomial time. Hence, the total area of nicely assigned small items to the containers in $\mathcal{B}_{small}$ is at least the area of such items packed according to the packing in Lemma 2.11. Then we use NFDH to pack all items in their respective assigned containers according to GAP, in which case we might miss out on at most 3ε fraction of the total area. All such items are packed in another box B_{small} of height at most $9\varepsilon \cdot \mathrm{OPT}$ and width W by NFDH. □

We pack B_{small} on top of B_{med} and obtain a packing of height $(1 + O(\varepsilon))$OPT. This yields an algorithm with a running time of $(nh_{\max}W)^{O_\varepsilon(1)}$. With a minor modification, we can remove the dependence on $h_{\max}$ and obtain a running time of $(nW)^{O_\varepsilon(1)}$, see Lemma 2.15 for details.

Lemma 2.15. *By increasing the height of the optimal solution by at most a factor* $1 + \varepsilon$*, we can assume that* $h_i \leq \lceil n/\varepsilon \rceil$ *for each item* $i \in I$ *and that the corners of each item are placed on integral coordinates in the optimal solution.*

Proof. Assume that $h_{\max} > n/\varepsilon$ as otherwise the lemma statement holds trivially. Let the height of the optimum packing be OPT. For each item $i \in I$, we round its height h_i to $h'_i = \left\lceil \frac{h_i n}{\varepsilon \cdot h_{\max}} \right\rceil$. Let the height of the optimum packing for this new instance be OPT′ and call this optimum packing P'.

Consider another instance where for each item $i \in I$, we normalize its height h_i to $h''_i = \frac{h_i n}{\varepsilon \cdot h_{\max}}$. Note that this new instance can have heights for the rectangular items which are not necessarily integers. Let its optimum packing have height OPT″ and call this optimum packing P. Now, consider the packing P, and for each item $i \in I$ round its height h_i to h'_i. Call this packing P_1. Then we claim that

$$\mathrm{OPT}' \leq h(P_1) \leq (1+\varepsilon)\mathrm{OPT}'',$$

where $h(P_1)$ denotes the height of P_1.

The first inequality holds since the heights of items in P_1 are at least as much as the heights of items in the packing P'. For the second inequality,

$$h(P_1) - \mathrm{OPT}'' \leq \sum_i (h'_i - h''_i) \leq n(1) \leq \varepsilon(n/\varepsilon) \leq \varepsilon \cdot h''_{\max} \leq \varepsilon \cdot \mathrm{OPT}''.$$

The last inequality holds since $h''_{\max}$ is a trivial lower bound on the height of the optimal packing OPT″ for the instance with heights h''_i for each item $i \in I$. Now, consider the packing P' but with the heights of each item $i \in I$ to be their original height h_i. Call this packing P''':

$$h(P''') \leq \mathrm{OPT}' \left(\frac{\varepsilon \cdot h_{\max}}{n}\right) \leq (1+\varepsilon)\mathrm{OPT}'' \left(\frac{\varepsilon \cdot h_{\max}}{n}\right) \leq (1+\varepsilon)\mathrm{OPT}.$$

Thus, the required conditions of the lemma are satisfied for the respective heights of each item $i \in I$ normalized to h'_i. □

We run our pseudo-polynomial time approximation algorithm on the resulting instance for which $h_{\max} \leq n/\varepsilon + 1$ holds. Thus, we obtain a running time of $(nW)^{O_\varepsilon(1)}$.

Theorem 2.16. *There is a* $(1+\varepsilon)$*-approximation algorithm for the GSP problem with a running time of* $(nW)^{O_\varepsilon(1)}$.

Proof. Follows from Lemma 2.5, the aforementioned algorithm, and by choosing the parameter to be $\varepsilon/33$ to finally achieve a $(1+\varepsilon)$-approximation. □

3 Polynomial Time $(\frac{3}{2} + \varepsilon)$-approximation

In this section, we first present the structural lemma for our polynomial time $(3/2 + \varepsilon)$-approximation algorithm for GSP. Then we describe our algorithm.

To derive our structural lemma, we start with the packing due to Lemma 2.5. The problem is that with a polynomial time algorithm (rather than a pseudo-polynomial time algorithm) we might not be able to pack all the tall items in their respective boxes. If there is even one single tall item i that we cannot pack, then we need to place i on top of our packing, which can increase the height of the packing by up to OPT.

Therefore, we make our packing more robust to small errors when we pack the items into their boxes. In our modified packing, the tall items are *bottom-left-flushed* (see Figure 9(f)), the remaining items are packed into $O_\varepsilon(1)$ boxes, and there is one extra box B^* of height OPT/2 and width $\Omega_\varepsilon(W)$ which is empty. We will use the extra box B^* in order to compensate for small errors when we pack the vertical items.

Formally, we say that in a packing, a set of items I' is *bottom-left-flushed* if they are ordered non-increasingly by height and stacked next to each other in this order within the strip $[0, W] \times [0, \infty)$ starting at the left edge of the strip, such that the bottom edge of each item $i \in I'$ touches the line segment $[0, W] \times \{0\}$. We now state the modified structural lemma for our polynomial time $(3/2 + \varepsilon)$-approximation algorithm formally.

Lemma 3.1 (Structural Lemma 2). *There exists a packing of the items I within $[0, W] \times [0, (3/2 + O(\varepsilon))\mathrm{OPT})$ such that*

- *The items I_{tall} are bottom-left-flushed,*
- *There is a set $\mathcal{B}$ of $O_\varepsilon(1)$ containers that are pairwise non-overlapping and do not intersect the items in I_{tall},*
- *There is a partition of $I \setminus I_{tall} = \bigcup_{B \in \mathcal{B}} I_B$ such that for each $B \in \mathcal{B}$ the items in I_B can be placed nicely into B,*
- *There is a container $B^* \in \mathcal{B}$ of height OPT/2 and width $\varepsilon_1 W$ such that $I_{B^*} = \emptyset$,*
- *The items I_{tall} and the containers $\mathcal{B}$ together are guillotine separable.*

We now prove Lemma 3.1 in the following subsection.

3.1 Proof of Structural Lemma 2

We start with the packing due to Lemma 2.5 and transform it step by step. To obtain our packing, we first argue that we can ensure that all tall items are placed on the bottom of the strip, i.e., their bottom edges touch the bottom edge of the strip. Here, we use that the initial packing is guillotine separable. Then we place the box B^* as follows. Suppose that there are initially C containers that cross the horizontal line with $y = \mathrm{OPT}/2$. Note that $C = O_\varepsilon(1)$. Then, by an averaging argument, we can show that there is a line segment l^* of length at least $\Omega(\frac{W}{C})$ which is the top edge of one of the containers B in the packing at some height $h^* \geq \mathrm{OPT}/2$. We push all the containers which completely lie above the line $y = h^*$ vertically upward by OPT/2 and this creates enough space to pack B^* on top of B. After that, we take advantage of the gained extra space in order to ensure that the tall items are bottom-left-flushed.

Now we describe the proof formally. First, we define some constants. Let $g(\delta, \varepsilon) = O_\varepsilon(1)$ denote an upper bound on the number of containers in the packing obtained using Lemma 2.5, depending on ε and δ. Let $\varepsilon_1 = \frac{1}{3g(\delta,\varepsilon)}$, $\varepsilon_2 = \frac{\varepsilon}{4|\mathcal{B}_{hor}|}$, $\varepsilon_3 = \frac{\varepsilon_1}{4|\mathcal{B}_{ver}|}$, $\varepsilon_4 = \mu$, $\varepsilon_5 = \frac{\varepsilon_1\delta}{6}$, $\varepsilon_6 = \frac{\varepsilon\delta}{6}$.

Our first goal is to make sure that the tall items are all placed on the bottom of the strip $[0, W] \times [0, \infty)$. For this, we observe the following: Suppose that in the guillotine cutting sequence, a horizontal cut is placed. This cut separates the current rectangular piece R into two smaller pieces R_1 and R_2. Suppose that R_1 lies on top of R_2. Then only one of the two pieces R_1, R_2 can contain a tall item. Also, we obtain an alternative guillotine separable packing if we swap R_1 and R_2—together with the items contained in them—within R. We perform this swap if R_1 contains a tall item. We apply this operation to each horizontal cut in the guillotine cutting sequence. As a result, we obtain a new packing in which all tall items are placed on the bottom of the strip (but possibly not yet bottom-left-flushed) as shown in Figure 8.

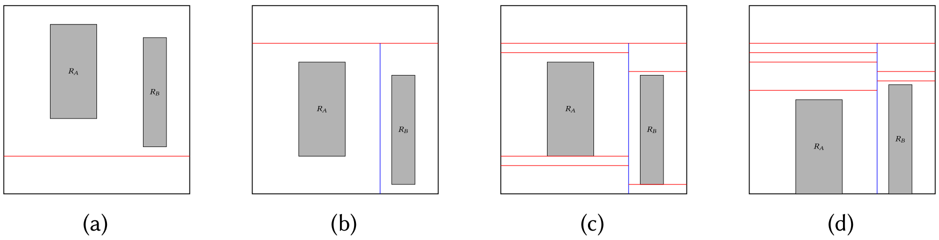


Fig. 8. R_A and R_B are tall containers and by swapping the respective boxes (forming as a result of guillotine cuts) that contain them, they can be packed such that the bottoms of both containers intersect the bottom of the strip.

Lemma 3.2. *There exists a set $\mathcal{B}$ of $O_\varepsilon(1)$ pairwise non-overlapping and guillotine separable boxes that are all placed inside $[0, W] \times [0, (1 + 16\varepsilon)\mathrm{OPT})$ and a partition $I = \bigcup_{B \in \mathcal{B}} I_B$ such that for each $B \in \mathcal{B}$ the items in I_B can be placed nicely into B. Also, for each box $B \in \mathcal{B}$ with $I_B \cap I_{tall} \neq \emptyset$ we have that the bottom edge of B intersects the line segment $[0, W] \times \{0\}$.*

Proof. The proof follows from the proof of Lemma 2.5 and the algorithm described at the start of Section 3.1 for shifting the tall items using guillotine cuts until their bottom edges intersect the bottom of the half-strip. First of all, for a packing, in a stage k (be it vertical or horizontal) in a guillotine cutting sequence if a box B is subdivided into j boxes $B_1, B_2, ..., B_j$ by guillotine cuts, then swapping any of these boxes does not affect the guillotine separability of the packing. Now, for the sake of contradiction assume that at the end of a valid guillotine cutting sequence, after applying the algorithm mentioned before for swapping tall items, a tall item i is such that $bottom(i) > 0$. Then, since i has been separated from all other items by guillotine cuts at the end of the cutting sequence, assume that the box B which contains it (on account of the guillotine cuts) at the end is exactly the size of i itself. Now, to have such a box B containing i, at some stage in the guillotine cutting sequence, we would have had a horizontal cut that intersects the bottom edge of i. But then i would have been swapped across that cut with the resulting guillotine box below B in that stage (by virtue of the algorithm), which is a contradiction. Hence, the lemma statements follow. □

Let $\mathcal{B}$ be the set of containers due to Lemma 3.2. We want to move some of them up in order to make space for the additional box B^*. To this end, we identify a horizontal line segment ℓ^* in the following lemma.

Lemma 3.3. *There is a horizontal line segment ℓ^* of width at least $\varepsilon_1 W$ that does not intersect any container in $\mathcal{B}$, and such that the y-coordinate of ℓ^* is at least $\mathrm{OPT}/2$.*

Proof. Consider the containers in $\mathcal{B}$ that intersect with the horizontal line segment $\ell := [0, W] \times \{\mathrm{OPT}/2\}$ and let $p_1, ...p_k$ be the maximally long line segments on ℓ that do not intersect any container. Since the line segments $\{p_1, ...p_k\}$ are between containers in $\mathcal{B}$, we have that $k \leq |\mathcal{B}| + 1$. Therefore by an averaging argument, we can find a horizontal line segment ℓ^* of width at least $\frac{W}{2(g(\delta,\varepsilon))+1} \geq \frac{W}{3g(\delta,\varepsilon)} \geq \varepsilon_1 W$ that either contains the top edge of one of these containers such that ℓ^* does not intersect any other container in $\mathcal{B}$ or ℓ^* is one of the line segments in the set $\{p_1, ..., p_k\}$. Hence, the y-coordinate of ℓ^* is at least $\mathrm{OPT}/2$. □

Let h^* be the y-coordinate of ℓ^*. We take all containers in $\mathcal{B}$ that lie "above h^*," i.e., that lie inside $[0, W] \times [h^*, \infty)$. We translate them up by $\mathrm{OPT}/2$. We define a container B^*, which has height $\mathrm{OPT}/2$ and width $\varepsilon_1 W$ to be packed such that ℓ^* is the bottom edge of B^* (see Figure 9(b)). We claim

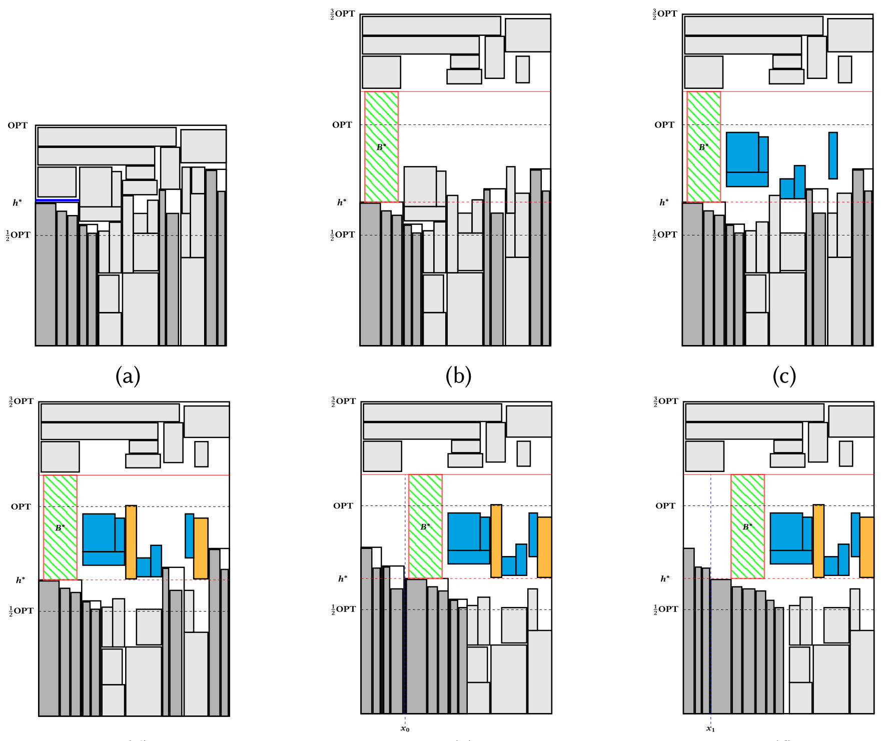


Fig. 9. (a) A guillotine separable packing with items nicely packed in containers. The dark-gray rectangles are the tall items and the light-gray rectangles are containers with items nicely packed inside. The blue line segment indicates l^*. (b) Items completely packed in $[h^*, \text{OPT}]$ are shifted by $\frac{1}{2}$OPT vertically upward. The thick red line indicates $y = h^* + \frac{1}{2}$OPT which separates the items shifted up from the items below. The dashed red line indicates the height h^* and B^* is packed in the strip of sufficient width and lowest height h^*. (c) The containers of type 1 (colored blue) are moved accordingly so they do not intersect $y = h^*$. (d) The containers of type 2 (colored yellow) are moved accordingly so they do not intersect $y = h^*$. (e) The containers in $\mathcal{B}_{tall}$ are *bottom-left-flushed* while other non-tall containers are moved accordingly to the right. The blue vertical dashed line $x = x_0$ separates containers in $\mathcal{B}^+_{tall}$ to its left-hand side from other containers to the right. (f) Final packing where tall items are *bottom-left-flushed* and the blue vertical dashed line $x = x_1$ separates items $i \in I_{tall}$ with $h_i > h^*$ to the left from other items and containers to the right. Notice that the final height of our packing is missing $O(\varepsilon)$ OPT as we do not include the associated steps in our algorithm in this figure. For simplicity, we only focus on showing the crucial aspects of our algorithm here.

that the resulting packing of $\mathcal{B} \cup \{B^*\}$ (we call this packing P_1) is feasible, guillotine separable, and has height $(3/2 + O(\varepsilon))$OPT.

Lemma 3.4. *The packing P_1 is feasible, guillotine separable, and has height* $(3/2 + O(\varepsilon))$OPT.

Proof. Since $h^* > \text{OPT}/2$, observe that no containers are intersecting the line $[0, W] \times \{h^* + \text{OPT}/2\}$. This is because any containers which were lying above the line $[0, W] \times \{h^*\}$ before were pushed up by OPT/2, and the height of such containers is at most OPT/2. Thus, the first guillotine cut is applied at $y = h^* + \text{OPT}/2$ so that we get two pieces R and R_{top}. For the guillotine separability of the top piece R_{top}, we use the fact that the packing to begin with was guillotine separable and we have moved a subset of the items in the initial packing vertically upwards by the same height. For the bottom piece R, which has a subset of the initial packing, we have packed B^* on the top edge (which is part of the line $[0, W] \times \{h^*\}$) of another container (say B) whose width is more than the width of B^*. In the guillotine cutting sequence of this piece without the addition of B^*, consider the horizontal cuts at the height of at least h^*. Note that there is no container lying completely above the line $[0, W] \times \{h^*\}$ in R. Hence, we can remove such horizontal cuts and extend the vertical cuts that were intercepted by these horizontal cuts until they hit the topmost horizontal edge of R. Now, if we follow this new guillotine-cutting sequence, we would finally have a rectangular region with only the container B. As there is no container in the region $[left(B), right(B)] \times [h^*, h^* + \mathrm{OPT}/2]$, we can pack B^* in this region without violating the guillotine separability condition. Now, observe that the height of the piece R_{top} is $(1 + O(\varepsilon))\mathrm{OPT} - h^*$ and height of the piece R is $h^* + \mathrm{OPT}/2$. Hence the height of the packing P_1 is $(3/2 + O(\varepsilon))\mathrm{OPT}$. □

Our next goal is to rearrange the tall items and their containers such that the tall items are bottom-left flushed. Let $\mathcal{B}_{tall} \subseteq \mathcal{B}$ denote the containers in $\mathcal{B}$ that contain at least one tall item. Consider the line segment $\ell := [0, W] \times \{h^*\}$ and observe that it might be intersected by containers in $\mathcal{B}_{tall}$. Let $\ell_1, \ell_2, ., \ell_t$ be the connected components of $\ell \setminus \bigcup_{B \in \mathcal{B}_{tall}} B$. For each $j \in \{1, ., t\}$ we do the following. Consider the containers in $\mathcal{B} \setminus \mathcal{B}_{tall}$ whose bottoms are contained in $\ell_j \times [\mathrm{OPT}/2, h^*]$ (we call them *type* 1 *containers*). We move them up by $h^* - \mathrm{OPT}/2$ units. There is enough space for them since the top edge of any of these containers lies below the line segment $[0, W] \times \{h^* + \mathrm{OPT}/2\}$ after shifting.

Then we take all containers in $\mathcal{B} \setminus \mathcal{B}_{tall}$ that intersect ℓ_j and also the line segment $[0, W] \times \{\mathrm{OPT}/2\}$ (*type* 2 *containers*). We move them up such that their respective bottom edges are contained in ℓ_j. Again there is enough space for this since the containers have height at most $\mathrm{OPT}/2$ and hence, their top edges cannot cross the line segment $[0, W] \times \{h^* + \mathrm{OPT}/2\}$. Note that in this step we do not necessarily move the affected containers uniformly. See Figure 9(c) and (d) for a sketch. Note that due to the way ℓ^* is defined, no type 1 or type 2 container after being shifted overlaps with the region occupied by B^*.

One can show that the resulting packing is still guillotine separable. In particular, there is such a sequence that starts as follows: The first cut of this sequence is a horizontal cut with y-coordinate $h^* + \mathrm{OPT}/2$. For the resulting bottom piece R, there are vertical cuts that cut through the vertical edges of the containers in $\mathcal{B}_{tall}$ whose height is strictly greater than h^*; denote these containers by $\mathcal{B}^+_{tall}$. Let $R_1, \ldots, R_{t'}$ denote the resulting partition of R. We can rearrange our packing by reordering the pieces $R_1, \ldots, R_{t'}$. We reorder them such that on the left we place the pieces containing one container from $\mathcal{B}^+_{tall}$ each, sorted non-increasingly by their heights. Then we place the remaining pieces from $\{R_1, \ldots, R_{t'}\}$ (which hence, do not contain any containers in $\mathcal{B}^+_{tall}$), denote their union by R'. Let the left end of R' be $x = x_0$. We can assume that the guillotine cutting sequence places a vertical cut that separates R' from the other pieces in $\{R_1, \ldots, R_{t'}\}$ at $x = x_0$. From Lemma 3.3, we know that there is a container B (or possibly the case when $h^* = \mathrm{OPT}/2$ and we have a line segment ℓ' of width at least $\varepsilon_1 W$ on top of which we can pack B^*) whose top is at height h^*, has width at least ℓ^* which now lies to the right of x_0 in R'. Thus, the region $[left(B), left(B) + \varepsilon_1 W] \times [h^*, h^* + \mathrm{OPT}/2]$ is empty and can be used to place B^*.

We change now the placement of the containers within R'. Due to our rearrangements, no container inside R' intersects the line segment $[0, W] \times \{h^*\}$, so we can assume that R' is cut by the horizontal cut $[0, W] \times \{h^*\}$; let R'' be the resulting bottom piece and R''' be the piece above. We first show why R''' is guillotine separable. First, we separate B^* using vertical guillotine cuts at its left and right edges. Then we prove that the shifting operation for type 2 and type 1 containers does not violate the guillotine separability of the packing for any region defined by some horizontal segment ℓ_j for $j \in [t]$. Consider any type 2 container B'. Its top edge was initially lying above

$y = h^*$ and its bottom below OPT/2. Hence, before shifting this container no item could have been packed such that it was in the region $[0, W] \times [h^*, h^* + \text{OPT}/2]$ and was intersecting the vertically extended line segments from the left and right edges of B' because any item packed in $[0, W] \times [h^*, \infty)$ initially was shifted upward by OPT/2. Hence, after shifting B' such that its bottom touches $y = h^*$, after considering the cut $y = h^*$ in ℓ_j, extend its left and right edges vertically upward to separate B' using guillotine cuts. For the type 1 containers, after the aforementioned cuts observe that all such containers have been shifted by an equal amount vertically upward, and using the fact that they were guillotine separable initially, we claim that they are guillotine separable afterward. This is proved by considering the initial guillotine cuts that were separating such items and shifting the horizontal cuts upward by $h^* - \text{OPT}/2$ (equal to the distance the type 1 containers were shifted upward by).

To show that R'' is guillotine separable, observe that due to our rearrangements there are no containers that are completely contained in $R'' \cap ([0, W] \times [\text{OPT}/2, h^*])$. Therefore, we can assume that the next cuts for R'' are vertical cuts that contain all vertical edges of the boxes in $\mathcal{B}_{tall}$ that are contained in R''. Let $R''_1, \ldots, R''_{t''}$ denote the resulting pieces. Like above, we change our packing such that we reorder the pieces in $R''_1, \ldots, R''_{t''}$ non-increasingly by the height of the respective box in $\mathcal{B}_{tall}$ contained in them, and at the very right we place the pieces from $R''_1, \ldots, R''_{t''}$ that do not contain any container from $\mathcal{B}_{tall}$ (see Figure 9(e)).

Finally, we sort the tall items inside the area $\bigcup_{B \in \mathcal{B}_{tall}} B$ non-increasingly by height so that they are bottom-left-flushed, and we remove the containers $\mathcal{B}_{tall}$ from $\mathcal{B}$ (see Figure 9(f)). We now prove that the tall items can be sorted inside the area $\bigcup_{B \in \mathcal{B}_{tall}} B$ non-increasingly by height without violating guillotine separability and feasibility. Note that the area $\bigcup_{B \in \mathcal{B}_{tall}} B$ can possibly contain some vertical items. Now, we reorder the tall items within R' such that they are sorted in non-increasing order of their heights. We do the same for all the tall items on the left of R'. There may be tall items (or vertical items) on the left-hand side of R' such that for any such item, its height is less than the tallest tall item in R'. Note that such tall items have to have a height of at most h^*. Such items can be repeatedly swapped with their neighboring tall item till they are in the correct position according to the *bottom-left-flushed* packing of the tall items while maintaining guillotine separability. Such a swap operation between consecutive tall items ensures that all of the tall items and possibly some vertical items that were initially packed in tall containers remain inside the area $\bigcup_{B \in \mathcal{B}_{tall}} B$. We ensure that the vertical items that were packed to the left of R' get swapped so that they are packed on the right of all the tall items in a container. Now, to prove that guillotine separability of the packing is maintained after all such swapping operations, that is, after all tall items are sorted according to their heights in a non-increasing order consider the x-coordinate (say x_1) of the right edge of the shortest tall item which has height strictly greater than h^*. Observe that there were no tall containers of height strictly greater than h^* beyond $x = x_0$, which implies $x_1 \le x_0$ and hence, now, for the guillotine cutting sequence, we can have a vertical guillotine cut at $x = x_1$ instead of at $x = x_0$, the rest being the same as mentioned before. This yields the packing claimed by Lemma 3.1.

3.2 Algorithm for Polynomial Time $(\frac{3}{2} + \varepsilon)$-approximation

First we guess a value OPT′ such that $\text{OPT} \le \text{OPT}' \le (1 + \varepsilon)\text{OPT}$ in $n^{O_\varepsilon(1)}$ time. We do this by computing a 2-approximation APX by Steinberg's algorithm [46] (which is a guillotine separable packing [35]) and then run our algorithm for all values of $\text{OPT}' = (1 + \varepsilon)^j \cdot \frac{\text{APX}}{2}$ which fit in the range $[\frac{\text{APX}}{2}, \text{APX}(1 + \varepsilon)]$, i.e., for $j \in \mathbb{Z}$ such that $1 \le j \le 1 + \lfloor \log_{(1+\varepsilon)} 2 \rfloor$. One of these values will satisfy the claim. In order to keep the notation light we denote OPT′ by OPT. We want to compute a packing of height at most $(\frac{3}{2} + O(\varepsilon))\text{OPT}$ using Lemma 3.1.

Intuitively, we first place the tall items in a bottom-left-flushed way. Then we guess approximately the sizes of the boxes, place them in the free area, and place the items inside them via guessing the relatively large items, solving an instance of the GAP, using NFDH for the small items, and invoking again Lemma 2.12 for the medium items (see Appendix Section A.1 for the definition of GAP). This is similar as in, e.g., [21, 32].

Formally, first we place all items in I_{tall} inside $[0, W] \times [0, (3/2 + \varepsilon)\mathrm{OPT})$ such that they are bottom-left-flushed. Then, we guess approximately the sizes of the containers in $\mathcal{B}$. Note that in polynomial time we cannot guess the sizes of the containers exactly. Let $B \in \mathcal{B}$. Depending on the items packed inside B, we guess different quantities for B.

— If there is only one single large item $i \in I$ packed inside B then we guess i.
— If B contains only items from I_{hor} then we guess the widest item packed inside B. This defines our guessed width of B. Also, we guess all items packed inside B whose height is at least $\varepsilon_2\mathrm{OPT}$ (at most $O(1/\varepsilon_2)$ many), denote them by I'_B. We guess the total height of the remaining items $I_B \setminus I'_B$ approximately by guessing the quantity $\hat{h}(B) := \left\lfloor \frac{h(I_B \setminus I'_B)}{\varepsilon_2\mathrm{OPT}} \right\rfloor \varepsilon_2\mathrm{OPT}$. Our guessed height for B is then $\sum_{i \in I'_B} h(i) + \hat{h}(B)$.
— Similarly, if B contains only items from I_{ver} then we guess the highest item packed inside B, which defines our guessed height of B. Also, we guess all items packed inside B whose width is at least $\varepsilon_3 W$ (at most $O(1/\varepsilon_3)$ many), denote them by I'_B. We guess the total width of the remaining items $I_B \setminus I'_B$ approximately by guessing the quantity $\hat{w}(B) := \left\lfloor \frac{w(I_B \setminus I'_B)}{\varepsilon_3\mathrm{OPT}} \right\rfloor \varepsilon_3\mathrm{OPT}$ and our guessed width of B is then $\sum_{i \in I'_B} w(i) + \hat{w}(B)$.
— If B contains only small items, then our guessed heights and widths of B are $\left\lfloor \frac{h(B)}{\varepsilon_4\mathrm{OPT}} \right\rfloor \varepsilon_4\mathrm{OPT}$ and $\left\lfloor \frac{w(B)}{\varepsilon_4 W} \right\rfloor \varepsilon_4 W$, respectively.

Note that here $\varepsilon_2 = \frac{\varepsilon}{4|\mathcal{B}_{hor}|}$, $\varepsilon_3 = \frac{\varepsilon_1}{4|\mathcal{B}_{ver}|}$, and $\varepsilon_4 = \mu$ are chosen so that the unpacked horizontal items, unpacked vertical items, and unpacked small items due to container rounding can be packed in containers B_{hor} (defined below), B^* and B_{small}, respectively (see Lemmas 3.6 and 3.7).

We have at most $O_\varepsilon(1)$ containers and for each container $B \in \mathcal{B}$ we guess the type of container B and its respective width and height (depending on the type) in $n^{O_\varepsilon(1)}$ time.

Additionally, we guess three containers B_{med} of height $2\varepsilon \cdot \mathrm{OPT}$, B_{hor} of height $\varepsilon \cdot \mathrm{OPT}$, and B_{small} of height $27\varepsilon \cdot \mathrm{OPT}$ and width W each that we will use to place all medium items, and to compensate for errors due to inaccuracies of our guesses for the sizes of the containers for horizontal and small items, respectively. Let $\mathcal{B}'$ denote the guessed containers (including B_{med}, B_{hor}, and B_{small}). Since $|\mathcal{B}'| = O_\varepsilon(1)$ and the containers in $\mathcal{B}'$ are not larger than the containers in $\mathcal{B}$, we can guess a placement for the containers $\mathcal{B}'$ such that together with I_{tall} they are guillotine separable. We place the containers B_{med}, B_{hor}, and B_{small} on top of the packing of the rest of the containers in $\mathcal{B}'$, and I_{tall}.

Lemma 3.5. *In time $n^{O_\varepsilon(1)}$ we can compute a placement for the containers in $\mathcal{B}'$ such that together with the items I_{tall}, they are guillotine separable.*

Proof. We guess the structure as guaranteed by Lemma 3.1. We first sort all the tall items according to the non-increasing order of their heights and place them in that order starting from the left end of the strip $[0, W] \times [0, \infty)$ such that their bottom edge touches the line segment $[0, W] \times \{0\}$. Let the tall items in this order be $I_{t_1}, I_{t_2}, \ldots, I_{t_k}$. Then as mentioned in Section 3.2, we have that sizes of the containers belong to a set (let's say S) that can be computed in $n^{O_\varepsilon(1)}$ time.

The height h^* is one among the heights of the tall items and we have at most $O_\varepsilon(1)$ containers in $\mathcal{B}'$. Hence, the position of the bottom of a container can be either of the following:

(1) A linear combination of heights from the set S.
(2) Sum of h^* with a linear combination of heights from the set S.
(3) Sum of $h^* + \frac{1}{2} \cdot \text{OPT}$ with a linear combination of heights from the set S.

Hence, the possible positions for the bottoms of the containers can be at most $n^{O_\varepsilon(1)}$ many and which can be computed in $n^{O_\varepsilon(1)}$ time. Similarly, the positions for the left end of the containers can be either of the following:

(1) A linear combination of widths from the set S.
(2) Sum of $\sum_{i=1}^{i=j} w(I_{t_i})$ with a linear combination of widths from the set S for some $j \leq t$.
(3) Sum of $\sum_{i=1}^{i=j} w(I_{t_i})$ with a linear combination of widths from the set S for some $j \leq t$ and $\varepsilon_1 W$.

Hence, the possible positions for the left end of the containers can be at most $n^{O_\varepsilon(1)}$ many and which can be computed in $n^{O_\varepsilon(1)}$ time. The three containers B_{med}, B_{hor}, and B_{small} of height $O(\varepsilon\text{OPT})$ and width W are placed on top of the packing. The guillotine separability of this placement of containers in $\mathcal{B}'$ along with the tall items is checked with the help of the algorithm in Lemma A.7 in polynomial time. □

Next, we place the vertical items. Recall that for each container $B \in \mathcal{B}$ containing items from I_{ver} we guessed the items packed inside B whose width is at least $\varepsilon_3 W$. For each such container B we pack these items into the container $B' \in \mathcal{B}'$ that corresponds to B. With a similar technique as used for the GAP [21], we place all but items with width at most $\varepsilon_3 W$ for each container in I_{ver}. Further using the PTAS for this variant of GAP (see Lemma A.3), we can ensure that items of the total area at most $3\varepsilon_5 \cdot \text{OPT} \cdot W$ are not packed. Hence, items of total width at most $(3\varepsilon_5/\delta)W$ remain unpacked as each such item has height at least δOPT. We pack these remaining items into B^*, using that each of them has a height of at most OPT/2 and that their total width is at most $|\mathcal{B}'| \cdot 2\varepsilon_3 W + (3\varepsilon_5/\delta)W \leq \varepsilon_1 W = w(B^*)$ (see Lemma 3.6 and Section 4). In other words, we fail to pack some of the vertical items since we guessed the widths of the containers only approximately and since our polynomial time approximation algorithm for GAP might not find the optimal packing. We use a similar procedure for the items in I_{hor} where instead of B^* we use B_{hor} in order to place the unassigned items.

Lemma 3.6. *In time $n^{O_\varepsilon(1)}$ we can compute a placement for all items in $I_{ver} \cup I_{hor}$ in B^*, B_{hor}, and their corresponding boxes in $\mathcal{B}'$.*

Proof. We reduce the given instance to an instance of GAP exactly as is done in Lemma 2.13. The only difference here is that for the vertical items, due to approximation in the widths of the containers as done in Section 3.2, per container we are not able to pack at most $\varepsilon_3 W$ width of items which are of height at most OPT/2. The combined width of such items which are unable to the packed in the containers is at most $2\varepsilon_3 \cdot |\mathcal{B}_{ver}| W$. For each vertical container, we guess all items packed inside B whose width is at least $\varepsilon_3 W$ (at most $O(1/\varepsilon_3)$ many). We do a symmetric procedure for the horizontal containers. Since we have at most $O_\varepsilon(1)$ containers in total, we then make use of the PTAS for this variant of GAP (see Lemma A.3) which ensures that for the vertical items, we are unable to pack at most $3\varepsilon_5$ fraction of the total area. This means that the total width of such items is at most $(3\varepsilon_5/\delta)W$. Hence, for the appropriate choice of $\varepsilon_3, \varepsilon_5$, after computing the packing of vertical items we might have unpacked vertical items with width at most

$$2\varepsilon_3 \cdot |\mathcal{B}_{ver}|\, W + (3\varepsilon_5/\delta)W \leq \left(\frac{2\varepsilon_1|\mathcal{B}_{ver}|}{4|\mathcal{B}_{ver}|}\right) W + \left(\frac{3\varepsilon_1\delta}{6\delta}\right) W \leq \varepsilon_1 W$$

which is due to the inefficiency of the algorithm for GAP and due to the approximation of container widths (see Section 4). All of such items can now be packed in B^*. Similarly, for the horizontal items, the total area of the items that are unable to be packed because of the inefficiency of GAP is at most $3\varepsilon_6 \cdot \text{OPT} \cdot W$. Also due to container rounding for the horizontal items, we are unable to pack items with height at most $2\varepsilon_2 \cdot |\mathcal{B}_{hor}| \cdot \text{OPT}$. These items have height at most $\varepsilon \cdot \text{OPT}$ by choosing $\varepsilon_2, \varepsilon_6 = O_\varepsilon(1)$ appropriately (see Section 4). We pack all such items simply by stacking them on top of each other in B_{hor} which adds an additional height of $\varepsilon \cdot \text{OPT}$ apart from the $3\varepsilon \cdot \text{OPT}$ height from resource augmentation and width W. □

For the medium items we invoke again Lemma 2.12 and we place B_{med} on top of the containers in $\mathcal{B}$ which increases the height of the packing only by $2\varepsilon \cdot \text{OPT}$.

Finally, we use NFDH again to pack the small items into their corresponding containers in $\mathcal{B}'$, which we denote by $\mathcal{B}'_{small}$, and B_{small}. We need B_{small} due to inaccuracies of NFDH and our guesses of the container sizes.

Lemma 3.7. *In time $n^{O(1)}$ we can compute a placement for all items in I_{small} in $\mathcal{B}'_{small}$ and B_{small}.*

Proof. The proof follows in the same vein as the one for Lemma 2.11, only that now we have to additionally account for inaccuracies in the container sizes. There are at most ε^2/μ^2 containers with width at least $(\mu/\varepsilon)W$ and height at least $(\mu/\varepsilon)\text{OPT}$ and one container of height at most $O(\varepsilon\text{OPT})$ and width W on top of the packing according to Lemma 2.11. By the container rounding mentioned in the algorithm in Section 3.2, for each such container B_i we are unable to pack at most $(\varepsilon_4 + \mu)(h(B) \cdot W + w(B) \cdot \text{OPT})$ area of small items. Thus, the ratio of this area of B is given by $(\varepsilon_4 + \mu)W/w(B) + 2(\varepsilon_4 + \mu)\text{OPT}/h(B)$ and this quantity is maximized when we consider the minimum values of $h(B)$ and $w(B)$ which are $h(B) = (\mu/\varepsilon)\text{OPT}$ and $w(B) = (\mu/\varepsilon)W$ and hence, we get that this ratio of area of small items that we are unable to pack due to container rounding is at most 4ε for $\varepsilon_4 = \mu$. All such items can be packed on top of the packing in another container B_{extra} of height $12\varepsilon \cdot \text{OPT}$ and width W using NFDH (two other containers each of height $6\varepsilon \cdot \text{OPT}$ and width W are used to account for inaccuracies due to NFDH packing and some unassigned area as in the PPTAS for small items).

Now that we have shown the existence of such a packing, for the algorithmic part we follow the same procedure of converting to an instance of GAP and assigning items according to the polynomial-time approximation algorithm for GAP (Lemma A.3). We might lose out on at most $2\varepsilon \cdot \text{OPT} \cdot W$ area due to inefficiency in the algorithm. Such small items can be packed in a container of height $6\varepsilon \cdot \text{OPT}$ and width W using NFDH along with the items we are unable to pack due to container rounding. Note that there are inaccuracies due to packing using NFDH nicely as well which accounts for at most $2\varepsilon\text{OPT} \cdot W$ area of small items not being packed. Such items can be packed using NFDH again using another container of height $6\varepsilon \cdot \text{OPT}$ and width W on top of the packing. Thus, we just need one container of height at most $27\varepsilon \cdot \text{OPT}$ and width W. □

Theorem 3.8. *There is a $(3/2 + \varepsilon)$-approximation algorithm for the GSP problem with a running time of $n^{O_\varepsilon(1)}$.*

Proof. The proof follows from Lemma 3.1 and the algorithm from Section 3.2 and guessing OPT to within a $1 + \varepsilon$ factor as mentioned at the beginning of Section 3.2. Here we take the parameter to be $\varepsilon/79$ to finally achieve a $(1 + \varepsilon)$-approximation. □

4 Relationship between Different Constants

Now let us define a function $g(\delta, \varepsilon)$ to denote an upper bound on the number of containers in the packing obtained using Lemma 2.5. Then, $g(\delta, \varepsilon) \geq |\mathcal{B}_{hor}| + |\mathcal{B}_{tall+ver}| + \left|\mathcal{B}_{large}\right| + |\mathcal{B}_{small}|$, i.e., the upper bound on the total number of containers for items in $I_{hor} \cup I_{tall} \cup I_{vert} \cup I_{large} \cup I_{small}$. This function is used to get an upper bound on μ (to be chosen sufficiently small compared to δ as defined in the paragraph below) and to define the function f from Lemma 2.1.

For the pseudo-polynomial time algorithm, $\varepsilon_{ra} = \varepsilon$, $|\mathcal{B}_{hor}| \leq \frac{96C_{ra}}{\varepsilon\delta^2} + C_{ra}$ and $|\mathcal{B}_{tall+ver}| \leq \frac{96}{\varepsilon\delta}(\frac{1}{\delta}2^{(1/\delta^2)})^{\frac{1}{\delta}}$, $\left|\mathcal{B}_{large}\right| \leq \frac{1}{\delta^2}$, and $|\mathcal{B}_{small}| \leq 4(|\mathcal{B}_{hor}| + |\mathcal{B}_{tall+ver}| + \left|\mathcal{B}_{large}\right| + 1)^2$. Note that C_{ra} is the number of containers we get from resource augmentation as in Lemma A.8. Let $g(\delta, \varepsilon) = |\mathcal{B}_{hor}| + |\mathcal{B}_{tall+ver}| + \left|\mathcal{B}_{large}\right| + |\mathcal{B}_{small}|$. From the condition for small containers in Lemma 2.11, we get that $\mu \leq \frac{\delta\varepsilon}{(g(\delta,\varepsilon))^2}$. Since, $g(\delta, \varepsilon)$ is a decreasing function in ε, we choose the function f from Lemma 2.1 as $f(x) = \frac{x\varepsilon}{g(x,\varepsilon)^2}$.

For the polynomial time algorithm, $\varepsilon_1 = \frac{1}{3g(\delta,\varepsilon)}$, which is the constant associated with the length of line segment l^*. Further, $\varepsilon_2 = \frac{\varepsilon}{4|\mathcal{B}_{hor}|}$, $\varepsilon_3 = \frac{\varepsilon_1}{4|\mathcal{B}_{ver}|}$, $\varepsilon_4 = \mu$, $\varepsilon_5 = \frac{\varepsilon_1\delta}{6}$, $\varepsilon_6 = \frac{\varepsilon\delta}{6}$, $\mu \leq \frac{\delta\varepsilon}{(g(\delta,\varepsilon))^2}$. The function g is as defined before.

5 Hardness

In this section, we show hardness results for GSP in both the pseudo-polynomial time and the polynomial time regimes.

THEOREM 5.1. *There exists no exact pseudo-polynomial time algorithm for the 2D GSP problem unless* P = NP.

PROOF. BIN PACKING is a strongly NP-Hard problem [26] and 2D GSP is a generalization of the same. To see why, consider the reduction: Reduce an instance of BIN PACKING where given, items of sizes $i_1, ..., i_n$ and the problem is to find whether it is possible to pack said items in k bins ($k \in \mathbb{Z}^+$, $i_1, ... i_n \in [0, 1]$ and $i_1, ... i_n \in \mathbb{Q}$) to an instance of GSP where width W of the half-strip is 1 and for each $j \in [n]$ we have a rectangle r_j such that $h(r_j) = 1$ and $w(r_j) = i_j$. Note that here the widths of the rectangles may not be integers but they can be appropriately scaled (along with the width of the half-strip) to ensure that. The objective in the GSP instance is to determine if there exists a guillotine separable packing of height at most k. The proof of the equivalence of this reduction follows in the same vein as the proof of Theorem 5.2. □

THEOREM 5.2. *There exists no polynomial time algorithm for the 2D GSP problem with an approximation ratio* $(\frac{3}{2} - \varepsilon)$ *for any* $\varepsilon > 0$ *unless* P = NP.

PROOF. Consider the following reduction from the PARTITION problem. For an instance of the PARTITION problem P where we are given positive integers $i_1, ..., i_n$ such that $T = \sum_{j=1}^{j=n} i_j$ and where we have to check if we can partition the given numbers into two sets S_1 and S_2 such that $\sum_{i_j \in S_1} i_j = \sum_{i_k \in S_2} i_k = T/2$, we construct the following instance I of the 2D GSP problem: Rectangles $\mathcal{R} = \{R_1, ..., R_n\}$ such that $h(R_k) = 1$ for any $k \in [n]$ and $w(R_k) = i_k$ and we want to check if there exists a guillotine separable packing of the rectangles in $\mathcal{R}$ in a half-strip of width $T/2$ such that height of this packing is at most 2.

We now show that for the above reduction, if the answer to the PARTITION instance P is "Yes," there exists a guillotine separable packing of height exactly 2 for instance I. And if the answer to the instance P is "No," the optimal guillotine separable packing has a height of at least 3. Note that any optimal packing can have only an integral height as all rectangles have a height of exactly 1. Now, if the answer to P is Yes, we have 2 sets S_1 and S_2 such that $S_1 \cup S_2 = \{i_1, \ldots, i_n\}$ and

$\sum_{i_j \in S_1} i_j = \sum_{i_k \in S_2} i_k = T/2$ where $T = \sum_{j=1}^{j=n} i_j$. Thus, we first pack all rectangles corresponding to numbers in S_1 from left to right at the bottom of the half-strip starting from $x = 0$ and without leaving any gap. Since the height of each rectangle is 1, we pack all rectangles similarly as before corresponding to numbers in S_2 from left to right on top of this packing. This results in a packing of height 2. It is a 2-stage guillotine separable packing because we first consider the horizontal cut $y = 1$ and then we separate all the rectangles in the resulting two boxes by way of vertical cuts.

We show that if there exists a guillotine separable packing of rectangles in $\mathcal{R}$ of height at most 2, then the answer to the instance P would be Yes. Observe that any packing of the rectangles has to have a height of at least 2 since the area of the rectangles in $\mathcal{R}$ is T and the width of the half-strip is $T/2$. If we have a guillotine separable packing of height 2, then by the area lower bound and the fact that all the rectangles have height 1, we have a guillotine cut at $y = 1$ and both the resulting pieces are completely filled by rectangles packed side by side without any space in between. Hence, we consider all items corresponding to rectangles packed in 1 box as S_1 and the others as S_2. Hence, we have a positive Partition instance. Taking the contrapositive of this statement proves our first claim of equivalence of the reduction.

If we have a polynomial time algorithm A for the 2D GSP problem with an approximation ratio $(3/2 - \varepsilon)$ for $\varepsilon > 0$, then

(1) If the instance I is a Yes instance, we have a guillotine separable packing of height 2 and by applying the algorithm, we get a guillotine separable packing of height at most $2(3/2 - \varepsilon) < 3$. And since only integral height packings are possible, we get a packing of height 2.
(2) If the instance I is a No instance, from our reduction we have a guillotine packing of height at least 3.

Consider the following polynomial time algorithm for the Partition problem. For an instance P of Partition, we reduce the problem to an instance I of 2D GSP as described. Then we apply the approximation algorithm A on this instance. If we get a packing of height 2, then by our previous claim for the reduction, P is a Yes instance. Else if we get a packing of height at least 3 we have a No instance.

This proves the theorem. □

6 Conclusion and Open Problems

We have studied the GSP problem and designed algorithms with the best possible approximation guarantees in both the pseudo-polynomial and the polynomial time regimes. Some of the prominent related open problems are as follows:

(1) Is it possible to adapt our techniques to show a tight $(3/2 + \varepsilon)$-approximate polynomial time algorithm for SP for any fixed $\varepsilon > 0$? Alternatively, show a hardness of $(3/2 + \varepsilon)$ for a fixed $\varepsilon > 0$. The current best polynomial time approximation guarantee of $(5/3 + \varepsilon)$ is by Harren et al. [28].
(2) For the 2GBP problem, Bansal et al. [6] showed an APTAS. For the general 2BP problem, the best asymptotic polynomial time approximation of 1.405 is by Bansal and Khan [5]. It has been conjectured that the worst-case ratio between the best 2GBP and the best general 2BP is 4/3 [5]. If true, this would imply a $(\frac{4}{3} + \varepsilon)$-approximation algorithm for 2BP.
(3) For the 2GGK problem, Khan et al. [35] have shown the existence of a PPTAS. However, in the polynomial time regime, the approximability of this problem remains unresolved. It remains an open problem to find a PTAS for 2GGK.

Acknowledgments

An extended abstract of this work was published at the 49th International Colloquium onAutomata, Languages, and Programming, 2022. A part of this work was done when A.Lonkar, A. Maiti, and A. Sharma were students/interns at the Indian Institute of Science.

References

[1] Fidaa Abed, Parinya Chalermsook, José R. Correa, Andreas Karrenbauer, Pablo Pérez-Lantero, José A. Soto, and Andreas Wiese. 2015. On guillotine cutting sequences. In *Proceedings of the Approximation, Randomization, and Combinatorial Optimization, Algorithms and Techniques (APPROX/RANDOM '15). LIPIcs, Vol.* 40, Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 1–19. DOI: https://doi.org/10.4230/LIPIcs.APPROX-RANDOM.2015.1

[2] Anna Adamaszek, Sariel Har-Peled, and Andreas Wiese. 2019. Approximation schemes for independent set and sparse subsets of polygons. *Journal of the ACM* 66, 4 (2019), 29:1–29:40. DOI: https://doi.org/10.1145/3326122

[3] Brenda S. Baker, Edward G. Coffman Jr, and Ronald L. Rivest. 1980. Orthogonal packings in two dimensions. *SIAM Journal on Computing* 9, 4 (1980), 846–855. DOI: https://doi.org/10.1145/3092026

[4] Nikhil Bansal, Jose R. Correa, Claire Kenyon, and Maxim Sviridenko. 2006. Bin packing in multiple dimensions: Inapproximability results and approximation schemes. *Mathematics of Operations Research* 31, 1 (2006), 31–49. DOI: https://doi.org/10.1287/moor.1050.0168

[5] Nikhil Bansal and Arindam Khan. 2014. Improved approximation algorithm for two-dimensional bin packing. In *Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '14).* SIAM, 13–25. DOI: https://doi.org/10.1137/1.9781611973402.2

[6] Nikhil Bansal, Andrea Lodi, and Maxim Sviridenko. 2005. A tale of two dimensional bin packing. In *Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science (FOCS '05).* IEEE Computer Society, 657–666. DOI: https://doi.org/10.1109/SFCS.2005.10

[7] István Borgulya. 2019. An EDA for the 2D knapsack problem with guillotine constraint. *Central European Journal of Operations Research* 27, 2 (2019), 329–356. DOI: https://doi.org/10.1007/s10100-018-0551-x

[8] Adam L. Buchsbaum, Howard J. Karloff, Claire Kenyon, Nick Reingold, and Mikkel Thorup. 2004. OPT versus LOAD in dynamic storage allocation. *SIAM Journal on Computing* 33, 3 (2004), 632–646. DOI: https://doi.org/10.1137/S0097539703423941

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

[10] Henrik I. Christensen, Arindam Khan, Sebastian Pokutta, and Prasad Tetali. 2017. Approximation and online algorithms for multidimensional bin packing: A survey. *Computer Science Review* 24 (2017), 63–79. DOI: https://doi.org/10.1016/j.cosrev.2016.12.001

[11] Nicos Christofides and Charles Whitlock. 1977. An algorithm for two-dimensional cutting problems. *Operations Research* 25, 1 (1977), 30–44. DOI: https://doi.org/10.1287/opre.25.1.30

[12] François Clautiaux, Ruslan Sadykov, François Vanderbeck, and Quentin Viaud. 2018. Combining dynamic programming with filtering to solve a four-stage two-dimensional guillotine-cut bounded knapsack problem. *Discrete Optimization* 29 (2018), 18–44. DOI: https://doi.org/10.1016/j.disopt.2018.02.003

[13] François Clautiaux, Ruslan Sadykov, François Vanderbeck, and Quentin Viaud. 2019. Pattern-based diving heuristics for a two-dimensional guillotine cutting-stock problem with leftovers. *EURO Journal on Computational Optimization* 7, 3 (2019), 265–297. DOI: https://doi.org/10.1007/s13675-019-00113-9

[14] Edward G. Coffman Jr, Michael R. Garey, David S. Johnson, and Robert E. Tarjan. 1980. Performance bounds for level-oriented two-dimensional packing algorithms. *SIAM Journal on Computing* 9, 4 (1980), 808–826. DOI: https://doi.org/10.1137/0209062

[15] Max A. Deppert, Klaus Jansen, Arindam Khan, Malin Rau, and Malte Tutas. 2023. Peak demand minimization via sliced strip packing. *Algorithmica* 85, 12 (2023), 3649–3679. DOI: https://doi.org/10.1007/s00453-023-01152-w

[16] Alessandro Di Pieri. 2013. *Algorithms for Two-Dimensional Guillotine Packing Problems.* Master's thesis. University of Padova, Italy.

[17] Mohammad Dolatabadi, Andrea Lodi, and Michele Monaci. 2012. Exact algorithms for the two-dimensional guillotine knapsack. *Computers & Operations Research* 39, 1 (2012), 48–53. DOI: https://doi.org/10.1016/j.cor.2010.12.018

[18] Fabio Furini, Enrico Malaguti, and Dimitri Thomopulos. 2016. Modeling two-dimensional guillotine cutting problems via integer programming. *INFORMS Journal on Computing* 28, 4 (2016), 736–751. DOI: https://doi.org/10.1287/ijoc.2016.0710

[19] Waldo Gálvez, Fabrizio Grandoni, Afrouz Jabal Ameli, Klaus Jansen, Arindam Khan, and Malin Rau. 2023. A tight $(3/2+\epsilon)$-approximation for skewed strip packing. *Algorithmica* 85, 10 (2023), 3088–3109. DOI: https://doi.org/10.1007/s00453-023-01130-2

[20] Waldo Gálvez, Fabrizio Grandoni, Afrouz Jabal Ameli, and Kamyar Khodamoradi. 2021. Proceedings of approximation algorithms for demand strip packing. In *Proceedings of the Approximation, Randomization, and Combinatorial Optimization, Algorithms and Techniques (APPROX/RANDOM '21)*. LIPIcs, *Vol.* 207, Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 20:1–20:24. DOI: https://doi.org/10.4230/LIPICS.APPROX/RANDOM.2021.20

[21] Waldo Gálvez, Fabrizio Grandoni, Salvatore Ingala, Sandy Heydrich, Arindam Khan, and Andreas Wiese. 2021. Approximating geometric knapsack via L-packings. *ACM Transactions on Algorithms* 17, 4 (2021), 1–67. DOI: https://doi.org/10.1145/3473713

[22] Waldo Gálvez, Fabrizio Grandoni, Salvatore Ingala, and Arindam Khan. 2016. Improved pseudo-polynomial-time approximation for strip packing. In *Proceedings of the 36th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS '16)*. LIPIcs, *Vol.* 65, Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 9:1–9:14. DOI: https://doi.org/10.4230/LIPIcs.FSTTCS.2016.9

[23] Waldo Gálvez, Fabrizio Grandoni, Arindam Khan, Diego Ramírez-Romero, and Andreas Wiese. 2021. Improved approximation algorithms for 2-dimensional knapsack: Packing into multiple L-shapes, spirals, and more. In *Proceedings of the 37th International Symposium on Computational Geometry (SoCG '21)*. LIPIcs, *Vol.* 189, Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 39:1–39:17. DOI: https://doi.org/10.4230/LIPIcs.SoCG.2021.39

[24] Waldo Gálvez, Arindam Khan, Mathieu Mari, Tobias Mömke, Madhusudhan Reddy Pittu, and Andreas Wiese. 2021. A $(2+\epsilon)$-approximation algorithm for maximum independent set of rectangles. arXiv:2106.00623. Retrieved from https://arxiv.org/abs/2106.00623

[25] Waldo Gálvez, Arindam Khan, Mathieu Mari, Tobias Mömke, Madhusudhan Reddy Pittu, and Andreas Wiese. 2022. A 3-approximation algorithm for maximum independent set of rectangles. In *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms (SODA '22)*. SIAM, 894–905. DOI: https://doi.org/10.1137/1.9781611977073.38

[26] Michael R. Garey and David S. Johnson. 1978. "Strong" np-completeness results: Motivation, examples, and implications. *Journal of the ACM* 25, 3 (1978), 499–508. DOI: https://doi.org/10.1145/322077.322090

[27] P. C. Gilmore and Ralph E. Gomory. 1965. Multistage cutting stock problems of two and more dimensions. *Operations Research* 13, 1 (1965), 94–120. DOI: https://doi.org/10.1287/opre.13.1.94

[28] Rolf Harren, Klaus Jansen, Lars Prädel, and Rob van Stee. 2014. A $(5/3 + \epsilon)$-approximation for strip packing. *Computational Geometry* 47, 2 (2014), 248–267. DOI: https://doi.org/10.1007/978-3-642-22300-6_40

[29] Sören Henning, Klaus Jansen, Malin Rau, and Lars Schmarje. 2020. Complexity and inapproximability results for parallel task scheduling and strip packing. *Theory of Computing Systems* 64, 1 (2020), 120–140. DOI: https://doi.org/10.1007/s00224-019-09910-6

[30] Dorit S. Hochbaum and Wolfgang Maass. 1985. Approximation schemes for covering and packing problems in image processing and VLSI. *Journal of the ACM* 32, 1 (1985), 130–136. DOI: https://doi.org/10.1145/2455.214106

[31] Klaus Jansen and Malin Rau. 2019. Closing the gap for pseudo-polynomial strip packing. In *Proceedings of the 27th Annual European Symposium on Algorithms (ESA '19)*. LIPIcs, *Vol.* 144, Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 62:1–62:14. DOI: https://doi.org/10.4230/LIPIcs.ESA.2019.62

[32] Klaus Jansen and Guochuan Zhang. 2004. On rectangle packing: Maximizing benefits. In *Proceedings of the 15th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '04)*. J. Ian Munro (Ed.), SIAM, 204–213. Retrieved from http://dl.acm.org/citation.cfm?id=982792.982822

[33] Claire Kenyon and Eric Rémila. 2000. A near-optimal solution to a two-dimensional cutting stock problem. *Mathematics of Operations Research* 25, 4 (2000), 645–656. DOI: https://doi.org/10.1287/moor.25.4.645.12118

[34] Arindam Khan. 2015. *Approximation Algorithms for Multidimensional Bin Packing*. Ph.D. Dissertation. Georgia Institute of Technology.

[35] Arindam Khan, Arnab Maiti, Amatya Sharma, and Andreas Wiese. 2021. On guillotine separable packings for the two-dimensional geometric knapsack problem. In *Proceedings of the 37th International Symposium on Computational Geometry (SoCG '21)*. LIPIcs, Vol. 189, Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 48:1–48:17. DOI: https://doi.org/10.4230/LIPIcs.SoCG.2021.48

[36] Arindam Khan and Madhusudhan Reddy Pittu. 2020. On guillotine separability of squares and rectangles. In *Proceedings of the Approximation, Randomization, and Combinatorial Optimization, Algorithms and Techniques (APPROX/RANDOM '20)*. LIPIcs, *Vol.* 176, Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 47:1–47:22. DOI: https://doi.org/10.4230/LIPIcs.APPROX/RANDOM.2020.47

[37] Arindam Khan and Eklavya Sharma. 2023. Tight approximation algorithms for geometric bin packing with skewed items. *Algorithmica* 85, 9 (2023), 2735–2778. DOI: https://doi.org/10.1007/s00453-023-01116-0

[38] Andrea Lodi, Michele Monaci, and Enrico Pietrobuoni. 2017. Partial enumeration algorithms for two-dimensional bin packing problem with guillotine constraints. *Discrete Applied Mathematics* 217 (2017), 40–47. DOI: https://doi.org/10.1016/j.dam.2015.09.012
[39] Joseph S. B. Mitchell. 2021. Approximating maximum independent set for rectangles in the plane. In *Proceedings of the 62nd IEEE Annual Symposium on Foundations of Computer Science (FOCS '21)*. IEEE, 339–350. DOI: https://doi.org/10.1109/FOCS52979.2021.00042
[40] Giorgi Nadiradze and Andreas Wiese. 2016. On approximating strip packing with a better ratio than 3/2. In *Proceedings of the 27th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '16)*. SIAM, 1491–1510. DOI: https://doi.org/10.1137/1.9781611974331.ch102
[41] János Pach and Gábor Tardos. 2000. Cutting glass. *Discrete & Computational Geometry* 24, 2 (2000), 481–496. DOI: https://doi.org/10.1007/s004540010050
[42] Enrico Pietrobuoni. 2015. *Two-Dimensional Bin Packing Problem with Guillotine Restrictions*. Ph.D. Dissertation. University of Bologna, Italy.
[43] Schiermeyer Ingo. 1994. Reverse-fit: A 2-optimal algorithm for packing rectangles. In *Proceedings of the 2nd European Symposium on Algorithms (ESA '94)*. Lecture Notes in Computer Science, Vol. 855, Springer, 290–299. DOI: https://doi.org/10.1007/BFb0049416
[44] Steven S. Seiden and Gerhard J. Woeginger. 2005. The two-dimensional cutting stock problem revisited. *Mathematical Programming* 102, 3 (2005), 519–530. DOI: https://doi.org/10.1007/s10107-004-0548-1
[45] Daniel Dominic Sleator. 1980. A 2.5 times optimal algorithm for packing in two dimensions. *Information Processing Letters* 10, 1 (1980), 37–40. DOI: https://doi.org/10.1016/0020-0190(80)90121-0
[46] A. Steinberg. 1997. A strip-packing algorithm with absolute performance bound 2. *SIAM Journal on Computing* 26, 2 (1997), 401–409. DOI: https://doi.org/10.1137/S0097539793255801
[47] Paul E. Sweeney and Elizabeth Ridenour Paternoster. 1992. Cutting and packing problems: A categorized, application-orientated research bibliography. *Journal of the Operational Research Society* 43, 7 (1992), 691–706. DOI: https://doi.org/10.1057/jors.1992.101
[48] Lijun Wei and Andrew Lim. 2015. A bidirectional building approach for the 2D constrained guillotine knapsack packing problem. *European Journal of Operational Research* 242, 1 (2015), 63–71. DOI: https://doi.org/10.1007/978-3-642-38577-3_24

Appendix

A.1 Maximum GAP

In this section, we state two results (without proof) in regard to the Maximum GAP, if the number of bins is constant.

Definition A.1 (Maximum GAP [21]*).* In GAP, we are given a set of k bins with capacity constraints and a set $I = \{I_1, ..., I_n\}$ of n items that may have different sizes and profits for each bin, and the objective is to pack a maximum-profit subset of items into the bins. In particular, to pack item i in bin j, the associated size is $s_{ij} \in \mathbb{Z}$ and the associated profit is $p_{ij} \in \mathbb{Z}$. The capacity of bin j for $j \in [k]$ is denoted by C_j. Then, we have to find a packing of the items $P : I \longrightarrow [k]$ such that for each bin $j \in [k]$, $\sum_{i:P(I_i)=j} s_{ij} \leq C_j$ and the total profit denoted by $\sum_{j=1}^{j=k} \sum_{i:P(I_i)=j} p_{ij}$ is maximized.

Let the cost of an optimal assignment be denoted by $p(\text{OPT})$. We have an exact pseudo-polynomial time algorithm for GAP when the number of bins is constant, as stated in the following lemma.

LEMMA A.2 [21]. *There is an $O(n \prod_{j=1}^{k} C_j)$ time algorithm for the maximum GAP with k bins with capacities C_i for bin $i \in [k]$, and returns a solution with a maximum profit $p(OPT)$.*

We also have the following algorithm for GAP running in time $n^{O_\varepsilon(1)}$ which ensures a profit of at least $(1 - O(\varepsilon))p(\text{OPT})$ if the number of bins is constant.

Lemma A.3 [21]. *There is an $O((\frac{1+\varepsilon}{\varepsilon})^k n^{k/\varepsilon^2+k+1})$ time algorithm for the maximum GAP with k bins with capacities C_i for bin $i \in [k]$, which returns a solution with profit at least $(1-3\varepsilon)p(OPT)$ for any fixed $\varepsilon > 0$.*

A.2 NFDH

We will use NFDH as a subroutine in our algorithms. For our purposes, we state two standard results (without proof) concerning NFDH. The first result stated below provides an upper bound on the height of the NFDH packing.

Lemma A.4 [14]. *For a list L of rectangles ordered by non-increasing height*:

$$\text{NFDH}(L) \leq 2A(L)/W + H_1 \leq 3\text{OPT},$$

where OPT *denotes the height of the optimal packing, W denotes the width of the half-strip, H_1 denotes the height of the tallest rectangle in L, and $A(L)$ denotes the total area of the rectangles in L.*

Since $\text{OPT} \geq H_1$ and $A(L)/W \leq \text{OPT}$, NFDH is a polynomial time 3-approximation. The second result is in the context of using NFDH to pack items inside a box of fixed size. Suppose we have a set of items I' and we are given a box C of size $w \times h$ such that each item fits in the box (without rotations). Then, NFDH computes a packing (without rotations) in polynomial time of $I'' \subset I'$ and if the items are small enough, we obtain a near-optimal packing of items inside the box. The following lemma describes the result regarding this packing.

Lemma A.5 [19]. *Assume that for some parameter $\varepsilon \in (0,1)$, for each $i \in I'$ one has $w_i \leq \varepsilon w$ and $h_i \leq \varepsilon h$. Then NFDH is able to pack in C a subset $I'' \subset I'$ of area at least $a(I'') \geq \min\{a(I'), (1-2\varepsilon)w \cdot h\}$. In particular, if $a(I') \leq (1-2\varepsilon)w \cdot h$, all items are packed.*

A.3 Steinberg's Algorithm

We will make use of Steinberg's algorithm [46] as a subroutine in our algorithms.

Theorem A.6 (Steinberg [46]). *We are given a set of rectangles I' and box Q of size $w \times h$. Let $w_{\max} \leq w$ and $h_{\max} \leq h$ be the maximum width and the maximum height among the items in I', respectively. Also we denote $x_+ := \max\{x, 0\}$. If*

$$2a(I') \leq wh - (2w_{\max} - w)_+(2h_{\max} - h)_+,$$

then I' can be packed into Q.

A.4 Algorithm for Checking Guillotine Separability

We present an algorithm that checks whether a set of axis-aligned packed rectangles are guillotine separable.

Lemma A.7. *Given a set of packed rectangles I' specified by their positions $(left(i), right(i)) \times (bottom(i), top(i))$ for each $i \in I'$ with $|I'| = n$, we can check in $O(n^2)$ time whether they are guillotine separable.*

Proof. Using standard shifting arguments, we can pack all the rectangles in a box of $[0, 2n-1] \times [0, 2n-1]$ where all the rectangles have integer coordinates for all of their four corners. To demonstrate this, we show how to do this in the x-direction, i.e., we consider the projections of the rectangles on the x-axis and consider both of the endpoints for each rectangle and then we assign them integer coordinates from $[0, 2n-1]$ in the same order. We do this similarly for the y-coordinates. Now, we specify a recursive procedure where for a box C in which we have a subset of rectangles $I'' \subseteq I'$ packed, we check all of the horizontal cuts and the vertical cuts at integral points that are feasible, incorporate such feasible cuts and recurse on the resulting smaller boxes. That is, we check all cuts that are line segments joining $(i, 0)$ and $(i, 2n-1)$ for $i \in [2n-1]$ and line segments joining $(0, j)$ and $(2n-1, j)$ for $j \in [2n-1]$ and incorporate either all feasible horizontal cuts or all feasible vertical cuts and recurse further (with alternating cuts). We only check cuts at integral coordinates since all rectangles are packed at locations that have integer coordinates by our preprocessing. We claim that this step can be done in linear time. We sort x-coordinates (and separately the y-coordinates) of the rectangles in $O(n \log n)$ time (note that there are at most $O(n)$ of these coordinates). We scan the x-coordinates from left to right, i.e., consider all positions $c = 1, \ldots, 2n-2$ in this order, and calculate a counter $count(c)$ for each $c \in [2n-2]$ of the number of rectangles for which we have already passed the left coordinate but not the right coordinate, i.e., we count the number of rectangles such that $left(R)$ is less than c and $right(R)$ is greater than c. If we have already calculated the value $count(c-1)$ for some c, then we can easily calculate the value $count(c)$ by considering all rectangles that start or end at position c. In particular, in this way we can calculate the value $count(c)$ for each $c \in [2n-2]$ in time $O(n)$. If $count(c) = 0$, then at position c there is a valid vertical guillotine cut. We find all such valid vertical cuts and recurse into the sub-boxes defined by these vertical cuts to find valid horizontal guillotine cuts in the sub-boxes. If there are no valid vertical guillotine cuts, we try the same with a scan from top to bottom to find valid horizontal cuts, and recurse into the corresponding subboxes.

It is easy to see that in each level of the guillotine cutting sequence, at least one rectangle is separated from a box or else we can declare that they are not guillotine separable. At each level of the guillotine cutting sequence, we spend at most $O(n)$ time overall checking all the possible feasible cuts for each respective box. Since the previous argument implies we can have at most $O(n)$ levels for the guillotine cutting sequence, the algorithm runs in $O(n^2)$ time. In fact, for a $(n-1)$ stage packing (see Figure 1) this algorithm may indeed require $\Omega(n^2)$ runtime. □

A.5 Resource Augmentation

In this section, we state without their respective proofs the necessary resource augmentation lemmas.

Lemma A.8. (Resource Augmentation Packing Lemma [21]). *Let I' be a collection of rectangles that can be packed into a box of size $a \times b$, and $\varepsilon_{ra} > 0$ be a given constant. Here a denotes the height of the box and b denotes the width. Then there exists a container packing of $I'' \subseteq I'$ inside a box of size $a \times (1+\varepsilon_{ra})b$ (resp. $(1+\varepsilon_{ra})a \times b$) such that:*

(1) $p(I'') \geq (1 - O(\varepsilon_{ra}))p(I')$;
(2) *the number of containers is $C_{ra} = O_{\varepsilon_{ra}}(1)$ and their sizes belong to a set of cardinality $n^{O_{\varepsilon_{ra}}(1)}$ that can be computed in polynomial time*;
(3) *the total area of the containers is at most $a(I') + \varepsilon_{ra}ab$.*

Lemma A.9 [35]. *If we have a guillotine separable packing of items I in a rectangular box B, the container packing of Lemma A.8 is also a guillotine separable nice packing.*

Received 4 February 2024; revised 30 January 2025; accepted 10 May 2025