On Guillotine Cutting Sequences*

Fidaa Abed[1], Parinya Chalermsook[1], José Correa[2], Andreas Karrenbauer[1], Pablo Pérez-Lantero[3], José A. Soto[4], and Andreas Wiese[1]

1 **Max Planck Institute for Informatics**
Saarbrücken, Germany
{fabed,karrenba,parinya,awiese}@mpi-inf.mpg.de

2 **Department of Industrial Engineering, Universidad de Chile**
Santiago, Chile
correa@uchile.cl

3 **Escuela de Ingeniería Civil Informática, Universidad de Valparaíso**
Valparaiso, Chile
pablo.perez@uv.cl

4 **Department of Mathematical Engineering and CMM, Universidad de Chile**
Santiago, Chile
jsoto@dim.uchile.cl

Abstract

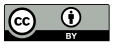
Imagine a wooden plate with a set of non-overlapping geometric objects painted on it. How many of them can a carpenter cut out using a panel saw making guillotine cuts, i.e., only moving forward through the material along a straight line until it is split into two pieces? Already fifteen years ago, Pach and Tardos investigated whether one can always cut out a constant fraction if all objects are axis-parallel rectangles. However, even for the case of axis-parallel squares this question is still open. In this paper, we answer the latter affirmatively. Our result is constructive and holds even in a more general setting where the squares have weights and the goal is to save as much weight as possible. We further show that when solving the more general question for rectangles affirmatively with only axis-parallel cuts, this would yield a combinatorial $O(1)$-approximation algorithm for the Maximum Independent Set of Rectangles problem, and would thus solve a long-standing open problem. In practical applications, like the mentioned carpentry and many other settings, we can usually place the items freely that we want to cut out, which gives rise to the two-dimensional guillotine knapsack problem: Given a collection of axis-parallel rectangles without presumed coordinates, our goal is to place as many of them as possible in a square-shaped knapsack respecting the constraint that the placed objects can be separated by a sequence of guillotine cuts. Our main result for this problem is a quasi-PTAS, assuming the input data to be quasi-polynomially bounded integers. This factor matches the best known (quasi-polynomial time) result for (non-guillotine) two-dimensional knapsack.

1998 ACM Subject Classification G.2.1 Combinatorics, I.3.5 Computational Geometry and Object Modeling

Keywords and phrases Guillotine cuts, Rectangles, Squares, Independent Sets, Packing

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2015.1

* This work was partially supported by Iniciativa Científica Milenio through the Millennium Nucleus Information and Coordination in Networks RC130003.



18th Int'l Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX'15) / 19th Int'l Workshop on Randomization and Computation (RANDOM'15).
Eds.: Naveen Garg, Klaus Jansen, Anup Rao, and José D. P. Rolim; pp. 1–19

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

Two-dimensional cutting stock problems arise naturally in industrial manufacturing. The goal is to cut out a given set of geometric objects from a large piece of parent material such as wood, metal, or glass. Guillotine cutting sequences play an important role in such processes. Starting from the large piece, in each step such a sequence takes one of the available pieces and cuts it along a straight line into two smaller pieces. Eventually, each of the input objects corresponds to one of the resulting pieces. Using guillotine cuts is often required by the available technology since more complex cutting patterns are often not possible.

Guillotine cuts motivate interesting basic problems in combinatorics, computational geometry, and combinatorial optimization. For instance, Urrutia [16] asked the following simple and yet intricate question: Given a set of pairwise non-overlapping compact convex geometric objects in the plane, can we always separate a constant fraction of them using a guillotine cutting sequence? Since the answer would be trivially *no* if cutting through objects is forbidden, such cuts are allowed at the expense of losing partial objects completely. This is equivalent to asking how many of them we can cut out using only guillotine cuts, i.e., each piece must not contain more than one complete object at the end. Pach and Tardos [15] investigated this question and showed that already for straight line segments this cannot always be achieved. They give a family of instances with straight line segments yielding an upper bound of $O(n^{\log 2/\log 3})$ for the number of line segments that can be cut out using only guillotine cuts. On the other hand, they show that we can indeed cut out a constant fraction of the input objects if – loosely speaking – all input objects have roughly the same size.[1] In this paper, we investigate the natural related question when the objects to be cut are rectangles[2] as we describe in the sequel.

1.1 Guillotine cuts for squares and rectangles

Even though Urrutia's general conjecture about convex objects was refuted, Pach and Tardos [15] wrote that "it seems plausible" that the question can be answered in the affirmative if the input objects are axis-parallel rectangles. They provided a cut sequence that saves $\Omega(n/\log n)$ rectangles, and stated that they "were unable to verify [a bound of $\Omega(n)$] even for axis-parallel squares".

In our first result in this paper, we answer the latter open question by giving a guillotine cutting sequence that recovers an 1/81-fraction of any set of axis-parallel squares. We first clean up the instance by placing a hierarchical grid with a random offset and delete some of the squares according to it (a standard step, see e.g., [7]). Then, we show that in each iteration of the cutting sequence we can find a cut that intersects at most $O(1)$ of the remaining squares while there is at least one surviving square on either side of the cut. By viewing this sequence as a binary tree, we can elegantly charge the number of the intersected squares to the number of surviving squares in the leaves.

Furthermore, we extend the previous result to the weighted case of the problem in which each square i has a weight w_i associated to it. As usual in combinatorial optimization, the weight of each object is a measure for its importance and we are looking for a cutting sequence that cuts out squares whose total weight is at least a constant fraction of the total weight of the input squares. As our above techniques do not carry over to this case, we give

[1] For a precise statement see Pach and Tardos [15].
[2] In this paper all rectangles considered are axis-parallel and open: a line going through a boundary side of such a rectangle does not destroy it.

a new algorithm that is based on a suitable conflict graph with one vertex for each input square. The graph has the important property that any independent set corresponds to a set of squares that can be cut out completely without any further loss. Even more, we show that we can color the vertices with at most 9 colors and thus there is an independent set corresponding to a 1/9-fraction of the entire weight of the input squares. Furthermore, our reasoning here directly extends to hypercubes in arbitrary dimensions. Thus, we can recover a 4/729-fraction of the weight of the squares in the two-dimensional plane, and a $1/2^{O(d)}$-fraction in d dimensions.

▶ **Theorem 1** (informal). *For axis-parallel squares there is always a guillotine cutting sequence that recovers a* 1/81*-fraction in the unweighted case, a* 4/729*-fraction in the weighted case, and a* $1/2^{O(d)}$*-fraction in the weighted case in* d *dimensions.*

An interesting aspect of our algorithms is that they only require axis-parallel cuts as opposed to the original question posed by Urrutia [16] (and investigated by Pach and Tardos [15]) where in principle also diagonal ones are allowed. Restricted to axis-parallel cuts we prove that for unit squares there are instances in which any cutting sequence can recover at most a 1/2-fraction of the squares (See Section 3 and Figure 3). Interestingly, although this is the strongest negative result we can obtain, we can easily show an algorithm finding a cutting sequence recovering 1/2-fraction of any set of unit squares, or more generally, for rectangles of equal height or width (see Section 2.3).

1.2 Connection to Independent Set of Rectangles

Inspired by the previous comment, we formulate a conjecture which is slightly stronger than the question investigated by Pach and Tardos [15].

▶ Conjecture 1.1. For any set of n non-overlapping axis-parallel rectangles there is a guillotine cutting sequence with only axis-parallel cuts separating $\Omega(n)$ of them.

If the conjecture was true this would have exciting consequences for the notoriously hard Maximum Independent Set of Rectangles (MISR) problem. Given a set of possibly overlapping axis-parallel rectangles, we want to compute a non-overlapping subset of maximum size. Finding a polynomial time $O(1)$-approximation algorithm for this problem is an important open problem (see e.g, [1, 7, 4, 5] and references therein.) We show that there is a simple dynamic program that computes the largest subset of the given rectangles that can be cut out completely using only guillotine cuts. Now, the conjecture implies that an $\Omega(1)$-fraction of the optimal solution of a MISR-instance can be cut out using guillotine cuts. Assuming this, we show that a simple dynamic programming (DP) algorithm yields an elegant combinatorial $O(1)$-approximation for MISR (see Section 4).

▶ **Theorem 2.** *If Conjecture 1.1 is true, then there is a* $O(1)$*-approximation algorithm for MISR with running time* $O(n^5)$*.*

1.3 Two-dimensional guillotine knapsack

The final contribution of this work concerns the two-dimensional guillotine knapsack problem. In this problem we are given a set of rectangles and a square-shaped knapsack[3], modeling a

[3] While with our techniques we can also handle the case of a rectangular knapsack, in this extended abstract for simplicity we assume the knapsack to be a square.

piece of parent material. We are interested in a placement of as many rectangles as possible in the knapsack so that there is a guillotine cutting sequence extracting them.

The two-dimensional geometric knapsack problem, without taking into account the guillotine cut constraint, is well-studied in the literature. For squares, a $(5/4+\epsilon)$-approximation is presented by Harren [10], which was subsequently improved to a PTAS by Jansen and Solis-Oba [12]. The best known polynomial time result for rectangles is a $(2+\epsilon)$-approximation result due to Jansen and Zhang [14]. The same authors presented a faster and simpler $(2+\epsilon)$-approximation for the unweighted case [13]. Recently, Adamaszek and Wiese presented a quasi-PTAS for rectangles that assumes the input data to be polynomially bounded integers [2]. For rectangles, there are $(1+\epsilon)$-approximation algorithms known if we are allowed to increase the size of the knapsack by a factor of $1+\epsilon$ in both dimensions [9], or even only in one [11]. Also, there is a PTAS if the profit of each item equals its area [3].

It is worth noting that many of the known results for this problem can be easily adjusted to take the guillotine cut constraint into account. However, this is not the case for the result giving the best known approximation factor for the problem: the recent $(1+\epsilon)$-approximation with quasi-polynomial running time (QPTAS) [2]. The algorithm is based on partitioning the placement area into $(\log n)^{O(1)}$ rectangular boxes such that there is a near-optimal solution in which – informally speaking – each box contains either only one big item, or high and narrow items, or wide and thin items. Then, the objects are assigned to the boxes via linear programming. This algorithm does not directly extend to the setting of guillotine cutting sequences. First, the mentioned near-optimal solution might not allow a guillotine cutting sequence (even if it is constructed based on such a solution) and second, the LP-rounding procedure does not necessarily produce such solutions either. We overcome these problems by showing that at additional (marginal) cost, we can construct a near-optimal solution in which essentially the mentioned boxes can be cut out using guillotine cuts. Then, we replace the LP-approach by a dynamic program. For this to work, we carefully round the items such that after rounding there are only $(\log n)^{O(1)}$ many different types of items and our DP guesses the correct guillotine cuts step by step, together with the distribution of the items on either side of the cut. In summary, our result is the following.

▶ **Theorem 3.** *There is a quasi-PTAS for the unweighted two-dimensional guillotine knapsack problem if all input data are quasi-polynomially bounded integers. This holds with and without the possibility to rotate items by 90 degrees.*

2 Guillotine cutting sequences for squares

In this section, we give guillotine cutting sequences recovering a constant fraction of a set of non-overlapping axis-parallel squares [4] of arbitrary sizes. Our results generalize to higher dimensions. First, we give some basic terminologies. In Section 2.1, we present a cutting sequence for unweighted squares, and then in Section 2.2, we prove the result for the weighted case.

Let P be a *piece*, i.e., a rectangle in the plane, and let H_1 and H_2 be the two open disjoint half-planes bounded by a straight line ℓ. Cutting P along ℓ gives us two sub-pieces $P_1 = P \cap H_1$ and $P_2 = P \cap H_2$. A *cutting strategy* is represented by a binary tree $\mathcal{T}$ where each non-leaf node $v \in V(\mathcal{T})$ is equipped with a piece P_v and a straight line ℓ_v such that cutting P_v along the straight line ℓ_v gives us P_{v_1} and P_{v_2}, where v_1, v_2 are the children of v.

[4] In this section, we henceforth implicitly assume all squares to be axis-parallel.

Let $\mathcal{O}$ be a set of objects. We say that the cutting strategy $\mathcal{T}$ *separates* $\mathcal{O}$ if the following statements hold

- The piece P_r associated with the root node r is a rectangle containing all objects in $\mathcal{O}$.
- For each non-leaf node v, the straight line ℓ_v intersects no object inside P_v.
- For each leaf node v, the piece P_v contains only one object in $\mathcal{O}$.

We also say that $\mathcal{O}$ is *guillotine separable* if there is a cutting strategy $\mathcal{T}$ separating $\mathcal{O}$. In the rest of this section, we focus on the case when our input objects are squares of arbitrary sizes. Let $\mathcal{R}$ denote the input set of squares.

Grid Lemma. One of the components in our proof is a collection of (multi-level) grid lines drawn on the plane in a "nice" way. These grid lines will be used to suggest our cut sequence, i.e., most of the straight lines in the strategy coincides with one of these grid lines. [5] We draw grid lines of various granularities and remove squares from $\mathcal{R}$ according to the grid lines, so that the remaining squares admit a guillotine cutting sequence. We say that a square $R \in \mathcal{R}$ is at level-i if its side length is in the interval $(N/2^{i+1}, N/2^i]$, where $N \in \mathbb{N}$ is used for normalization so that level-0 contains the largest squares.

In a first step, we independently pick two random numbers $x, y \in [0, N)$ defining a random shift to draw the grid. For each i (i.e. level), the vertical grid lines at level-i are drawn at $x, x+N/2^i, x+2\cdot N/2^i, \ldots$ (wrapping up appropriately); similarly, the horizontal grid lines at level-i are drawn at $y, y+N/2^i, y+2\cdot N/2^i \ldots$. Grid cells bounded by consecutive grid lines at level $i-1$ are said to be at level-i, so level-i grid cells are squares of size $(2N/2^i)$-by-$(2N/2^i)$. Note that the higher the level the more fine grained the grid is. A square $R \in \mathcal{R}$ is removed from this step if it intersects with grid lines at levels below it, i.e. if R is in level-i, then it is removed if it intersects a line at levels $i-1, \ldots, 0$. Let $\mathcal{R}_1$ be the set of squares that are not removed from this step.

▶ Claim 2.1. A level-i square $R \in \mathcal{R}$ of side length $\ell_R \in (N/2^{i+1}, N/2^i]$ remains in $\mathcal{R}_1$ with probability $(1-\mu_R)^2 \geq 1/4$, where $\mu_R = \ell_R 2^{i-1}/N$.

Proof. The probability that a horizontal grid line at level $i-1$ intersects R is $\mu_R = \ell_R 2^{i-1}/N$ (because of the random shift). Notice that μ_R is between $1/4$ and $1/2$. Now, since the shifts x, y are chosen independently, the probability that the square R survives in $\mathcal{R}_1$ is $(1-\mu_R)^2 \geq 1/4$. ◀

In a second step, we further sample $\mathcal{R}_1$ to obtain $\mathcal{R}_2$, where each square is sampled with relatively large probability. Now we consider each grid cell C at level-i that contains a subset of squares $\mathcal{R}_1^C$ at level-i. Cell C may contain up to 9 squares, so if we are not careful, we might end up paying an extra factor of 9 (giving $1/36$ marginal only). So, we define a distribution on $\mathcal{R}_1^C$ where exactly one square R in $\mathcal{R}_1^C$ is kept, and R is kept with probability

$$\frac{1}{(1-\mu_R)^2 \cdot M_C}, \quad \text{for } M_C = \sum_{S \in \mathcal{R}_1^C} \frac{1}{(1-\mu_S)^2}.$$

Let $\mathcal{R}_2$ be the set of squares remaining after this process. We analyze the probability that a level-i square R remains in $\mathcal{R}_2$. This can be broken down into:

$$\begin{aligned} \mathbf{Pr}\left[R \in \mathcal{R}_1\right] \cdot \mathbf{Pr}\left[R \in \mathcal{R}_2 \mid R \in \mathcal{R}_1\right] &= (1-\mu_R)^2 \cdot \left(\frac{1}{(1-\mu_R)^2 M_C}\right) \\ &= 1/M_C. \end{aligned}$$

[5] We may deviate from this strategy when considering subsets with a constant number of squares.

▶ Claim 2.2. $M_C \leq 81/4$.

Proof. Each square $R \in \mathcal{R}_1^C$ satisfies $\mu_R \in (1/4, 1/2]$, and has side length ℓ_R strictly larger than $1/4$ of the side length of C. This implies $|\mathcal{R}_1^C| \leq 9$. Furthermore, R consumes a μ_R^2-fraction of the area of C, and $\sum_{R \in \mathcal{R}_1^C} \mu_R^2 \leq 1$ must be satisfied. We argue below that M_C is maximized when $|\mathcal{R}_1^C| = 9$ and $\mu_R = 1/3$ for each square $R \in \mathcal{R}_1^C$, i.e., when $\mathcal{R}_1^C$ is a set of 9 equal squares of area $1/9$ of that of C. This gives $M_C \leq 9 \cdot (1 - 1/3)^{-2} = 9 \cdot (9/4) = 81/4$.

We ignore the geometry of the squares and focus only on the values of μ_R that satisfy the following:

$$\begin{aligned} \max \quad & \sum_{R \in \mathcal{R}_1^C} \frac{1}{(1-\mu_R)^2} \\ \text{s.t.} \quad & \sum_{R \in \mathcal{R}_1^C} \mu_R^2 \leq 1, \quad |\mathcal{R}_1^C| \leq 9, \quad \mu_R \in (1/4, 1/2]. \end{aligned}$$

Let q be an integer in $\{1, \ldots, 9\}$. Notice that for a fixed choice of $|\mathcal{R}_1^C| = q$, the objective function is maximized when the values of μ_R are "balanced": One can check that if there are two values $\mu_R \neq \mu_{R'}$, then we could have averaged these values to $\mu'_R = \mu'_{R'} = \sqrt{(\mu_R^2 + \mu_{R'}^2)/2}$; the new choices of μ'_R and $\mu'_{R'}$ still satisfy all constraints while increasing the objective. Because $1/4 < \mu_R \leq 1/2$, the objective values cannot exceed $\frac{q}{(1-1/2)^2} \leq 20 < 81/4$ for $q \leq 5$. Moreover, for a fixed $q \in \{6, \ldots, 9\}$, the objective is optimized by setting μ_R such that $q \cdot \mu_R^2 = 1$, i.e., $\mu_R = 1/\sqrt{q}$. An enumeration of the corresponding objective values reveals the maximum of $81/4$ for $q = 9$, which proves the claim. ◀

Notice that in the random subset $\mathcal{R}_2 \subseteq \mathcal{R}$, each square at level-$i$ is not intersected by grid lines at levels $i - 1, \ldots, 0$, and each level-i grid cell has at most one square in level-i. Then, using claims 2.1 and 2.2, we can state the main result of this section:

▶ **Lemma 4.** *There exists a distribution $\mathcal{D}\colon 2^{\mathcal{R}} \to [0, 1]$ such that each subset $\mathcal{R}'$ in its support admits a grid drawing, with some shift, satisfying the following properties:*

- *Each square at level-i is not intersected by grid lines at levels $i - 1, \ldots, 0$.*
- *Each level-i grid cell has at most one square in level-i.*

Moreover, each square $R \in \mathcal{R}$ appears in a randomly drawn subset with probability at least $\epsilon_1 = 4/81$, i.e., $\mathbf{Pr}_{\mathcal{R}' \sim \mathcal{D}}[R \in \mathcal{R}'] \geq \epsilon_1$.

2.1 Unweighted case

Before we treat the more general weighted case, we first show how to save a linear number of squares using guillotine cuts. The approach of this section is not subsumed by the one from Section 2.2, as it yields a better constant than applying the latter with uniform weights. The high-level idea comes from the observation that the number of leaves in a proper binary tree is at least half the number of nodes. Thus, bounding the number of squares that are cut in each node yields a lower bound on the number of square that are saved.

▶ **Definition 5.** We call a line a *k-good separator*, if it intersects at most k squares and each side contains at least one square completely. We call a binary tree of a cutting strategy *k-good*, if each internal node has a k-good separator.

▶ **Lemma 6.** *If an instance admits a k-good tree, then there is a guillotine sequence such that $n/(k+1)$ squares survive.*

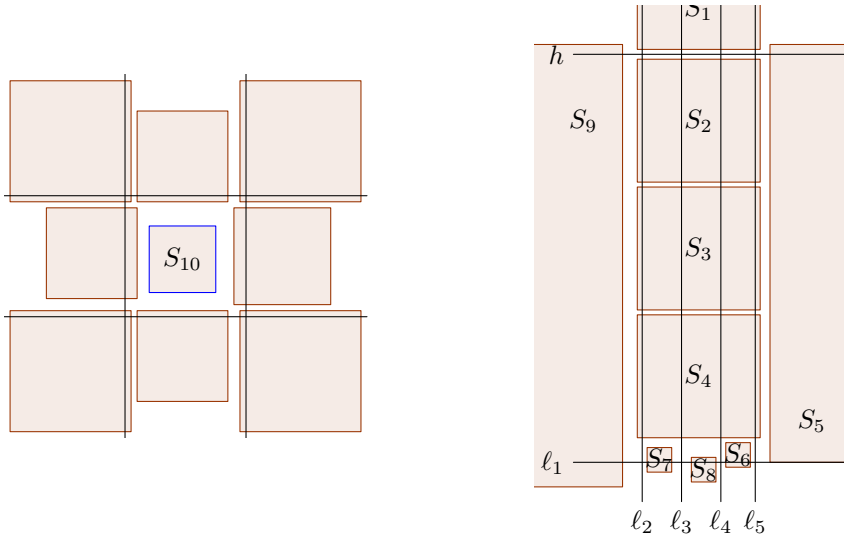


Figure 1 Arising cases in the proof of Lemma 7.

Proof. Observe that the number of surviving squares is equal to the number of leaves, say s, in the k-good tree. Let t be the number of nodes in this tree. The instance contains one square for each leaf and at most k squares for each internal node. Thus, we obtain $n \leq s + k \cdot (t - s) = s + k \cdot (s - 1) \leq (k + 1) \cdot s$, since $t = 2s - 1$ for a proper binary tree. ◀

▶ **Lemma 7.** *There is a guillotine sequence with a 3-good tree for any subset in the support of a distribution according to Lemma 4.*

Proof. We iteratively define the cut sequence that gives us a 3-good tree. Starting from the piece P_0 that contains every square, we show that, given a piece P', a 3-good separator for P' exists. Now we describe the existence proof of the separator. There are two cases to consider. First, if P' contains at least 10 squares, consider the squares in P' as a sequence $S_1, S_2, \ldots$ ordered by non-increasing side lengths. Let i be the level of S_{10}. We consider the grid cell of level i containing S_{10}. Observe (see left of Figure 1 for an illustration) that the edges of that cell can be covered by at most eight squares because these squares must be at lower levels than i and the distance between two adjacent corners of the cell, which is $2N/2^i$, is not wide enough to contain two squares of lower levels, which have side lengths of at least $N/2^i$, in its interior. Thus, one of the four grid lines defining the cell separates S_{10} from some other square while intersecting at most three. Since we choose a separator by the grid line at the level of S_{10}, this line cannot intersect any square in $\{S_{11}, S_{12}, \ldots\}$.

Let us now consider the other case when P' contains at most 9 squares. In this case, we would choose a separator that does not necessarily correspond to a grid line. We order the squares $S_1, \ldots, S_9$ by non-increasing y-coordinates of the bottom boundaries. Consider a horizontal line ℓ_1 that coincides with the bottom boundary of S_5, so ℓ_1 cannot "stab" any square in $\{S_1, \ldots, S_5\}$. If ℓ_1 stabs at most 3 squares in $\{S_6, S_7, S_8, S_9\}$, we would be done, ℓ_1 is our separator. Otherwise, ℓ_1 must stab all four squares and shares the border with S_5.

There are (at least) 4 combinatorially different vertical lines that separate squares in $\{S_5, \ldots, S_9\}$ without intersecting them. By "combinatorially different" we mean that they do not separate the squares in the exact same way. Denote by $\mathcal{L}$ a set of four such vertical lines. If there is a vertical line $\ell' \in \mathcal{L}$ that intersects at most 3 squares in $\{S_1, \ldots, S_4\}$, we

would be done, as we can use ℓ' as our separator. Otherwise, each of these four lines stabs four squares in $\{S_1, \ldots, S_4\}$, and in this case, we can use the horizontal line h that coincides with the bottom boundary of S_1 as our separator; this line cannot overlap with any square in $\{S_1, \ldots, S_4\}$, and it can only overlaps with at most two of the squares in $\{S_5, \ldots, S_9\}$. ◀

▶ **Theorem 8.** *There is always a cutting strategy for a $\frac{1}{81}$-fraction of squares.*

Proof. We first apply the construction of Lemma 4. This guarantees the existence of a 3-good tree due to Lemma 7. This implies with Lemma 6 that a fraction of $\frac{4}{81} \cdot \frac{1}{4} = \frac{1}{81}$ of the given squares can be separated by guillotine cuts. ◀

2.2 Weighted case

In this setting, we are additionally given a weight function $w\colon \mathcal{O} \to \mathbb{R}_{\geq 0}$, and we want a cutting strategy separating a subset with the largest possible weight. We restate this question in an equivalent form with the following notion. An *ϵ-guillotine sampling* for objects $\mathcal{O}$ is a distribution $\mathcal{D}\colon 2^{\mathcal{O}} \to [0,1]$ such that any object $r \in \mathcal{O}$ is sampled by $\mathcal{D}$ with probability at least ϵ, i.e., $\mathbf{Pr}_{\mathcal{O}' \sim \mathcal{D}}[r \in \mathcal{O}'] \geq \epsilon$, and each subset $\mathcal{O}'$ in the support of $\mathcal{D}$ is guillotine separable.

▶ **Lemma 9.** *For any set of objects $\mathcal{O}$, the following are equivalent: (i) there is an ϵ-guillotine sampling for $\mathcal{O}$ and (ii) for any weight function $w : \mathcal{O} \to \mathbb{R}_{\geq 0}$, there is a subset $\mathcal{O}' \subseteq \mathcal{O}$ that is guillotine separable and $w(\mathcal{O}') \geq \epsilon \cdot w(\mathcal{O})$.*

Proof. The forward implication is easy to see. Suppose we have an ϵ-guillotine sampling $\mathcal{D}$ for $\mathcal{O}$. Let $w : \mathcal{O} \to \mathbb{R}_{\geq 0}$ be a weight function. We pick a random set $\mathcal{O}'$ according to the distribution $\mathcal{D}$. Then, we have

$$\mathbf{E}[w(\mathcal{O}')] = \sum_{r \in \mathcal{O}} w(r) \cdot \mathbf{Pr}[r \in \mathcal{O}'] \geq \epsilon \cdot w(\mathcal{O}).$$

This shows the existence of such a subset. For the backward implication, the proof is by LP duality. Let $\mathcal{F}_{\mathcal{O}}$ be the set of all guillotine separable subsets of $\mathcal{O}$. We write the following linear program that reflects the best ϵ guillotine sampling:

$$\begin{aligned} \text{(LP)} \quad \max \quad & \gamma \\ \text{s.t.} \quad & \gamma \leq \sum_{\mathcal{O}' \in \mathcal{F}_{\mathcal{O}} : r \in \mathcal{O}'} p_{\mathcal{O}'} \text{ for all object } r \in \mathcal{O}, \\ & \sum_{\mathcal{O}' \in \mathcal{F}_{\mathcal{O}}} p_{\mathcal{O}'} = 1, \\ & p_{\mathcal{O}'} \geq 0 \text{ for all } \mathcal{O}'. \end{aligned}$$

The dual of the above LP can be written as:

$$\begin{aligned} \text{(LP')} \quad \min \quad & \beta \\ \text{s.t.} \quad & \beta \geq \sum_{r \in \mathcal{O}'} w_r \text{ for all } \mathcal{O}' \in \mathcal{F}_{\mathcal{O}}, \\ & \sum_{r \in \mathcal{O}} w_r = 1, \\ & w_r \geq 0 \text{ for all object } r \in \mathcal{O}. \end{aligned}$$

Now, suppose that we can find a guillotine separable subset for any weight function w. This means that (LP') cannot be feasible for any (w, β) if $\beta < \epsilon$, so the optimal value of (LP') is at

least ϵ. By duality, the optimal solution for (LP) gives us the distribution that is ϵ-guillotine sampling for $\mathcal{O}$. ◀

Lemma 9 allows us to focus on finding ϵ-guillotine samplings instead (this is an unweighted question). In what follows, we present such a sampling for any input set $\mathcal{R}$ of squares. We start by invoking Lemma 4 to find a distribution $\mathcal{D}'$ for $\mathcal{R}$. Consider each subset $\mathcal{R}'$ in the support of $\mathcal{D}'$ together with its grid drawing $\mathcal{G}'$. We will show that each such subset $\mathcal{R}'$ can be further partitioned into 9 guillotine separable subsets, which will imply the following theorem:

▶ **Theorem 10.** *Any set of squares $\mathcal{R}$ is ϵ-guillotine samplable for $\epsilon = 4/729$.*

For each square R at level-i, let $\mathsf{cell}(R)$ be the level-i cell containing R. We say that two squares R and S are *conflicting* if either R overlaps the boundary of $\mathsf{cell}(S)$ or S overlaps the boundary of $\mathsf{cell}(R)$. Observe that any pair of conflicting squares belong to different levels and if R overlaps the boundary of $\mathsf{cell}(\mathsf{S})$, then the level of R is smaller than that of S.

We define a conflict graph that encodes the conflict structures between squares. Let H be the graph such that the vertex set $V(H)$ corresponds to the squares in $\mathcal{R}'$, and there is an edge between squares R and S if and only if R and S are conflicting. The following two lemmas complete the proof.

▶ **Lemma 11.** *The graph H is 9-colorable.*

Proof. We prove this by induction on the number of vertices. The base case when $|V(H)| = 1$ is obvious. Now, consider any graph H with at least two vertices, and any square $R \in V(H)$ whose size is minimum among the squares in $V(H)$. Let ℓ be the side length of $\mathsf{cell}(R)$. Consider the set $N_H(R) \subset V(H)$ of the squares S defining an edge with R. It can only be that each square S in $N_H(R)$ is at the level below of that of R; so the lengths of these squares are strictly greater than $\ell/2$. We claim that $|N_H(R)| \leq 8$: There can be at most 4 squares in $N_H(R)$ that contain some corner of $\mathsf{cell}(R)$, and for each side of $\mathsf{cell}(R)$, there can be at most one square in $N_H(R)$ overlapping it and without containing a corner of $\mathsf{cell}(R)$. By the induction hypothesis, the graph obtained from H by removing the vertex R can be colored with 9 colors. Since the degree of R is at most 8, we can always assign a color to R, distinct from the colors of its neighbors $N_H(R)$. ◀

▶ **Lemma 12.** *Let $I \subseteq V(H)$ be an independent set. The squares $\{R\}_{R \in I}$ are guillotine separable.*

Proof. We prove this by iteratively defining the cutting strategy. Our cut sequence always cuts along grid lines, and any piece P produced in this process satisfies the following property: P contains at most one square, or, otherwise, let $\mathcal{R}_P$ be the set of squares inside P and ℓ the level of the *second* largest square in $\mathcal{R}_P$. Then, the sides of P are aligned with grid lines of levels at most ℓ.

Initially, let P_0 be the single piece that contains all squares, so the above properties are satisfied: One can assume that P_0 is bounded by four grid lines at level-0. Now, if every piece contains at most one square, then we are done. Otherwise, consider a piece P' with more than one square in $\mathcal{R}_{P'}$. Let R and R' be the largest and second largest squares in $\mathcal{R}_{P'}$, respectively.

There are two cases. See Figure 2 for reference. First, if R and R' belong to the same level ℓ, then we simply cut along any level-$(\ell - 1)$ line L that separates the grid cells $\mathsf{cell}(R)$ and $\mathsf{cell}(R')$. Line L cannot intersect any square in $\mathcal{R}_{P'}$: By Lemma 4, these squares are

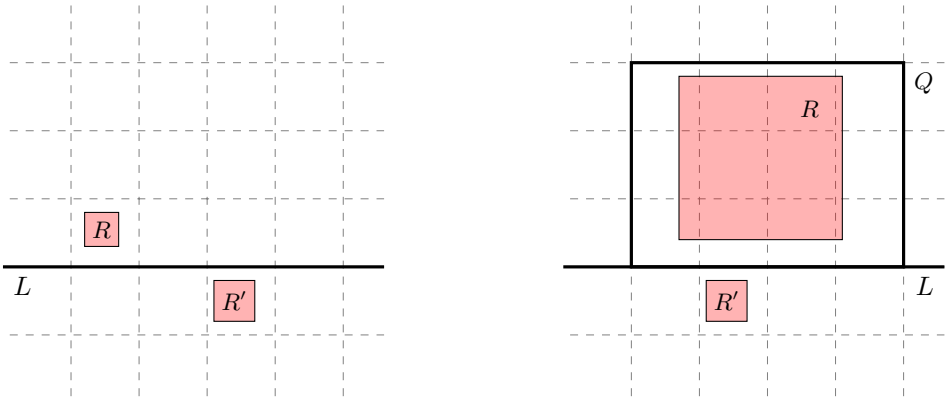


Figure 2 An illustration of the proof of Lemma 12. The figure on the left shows the case when the levels of R and R' are the same, and the right figure shows the other case when their levels are different.

at level at least ℓ, and cannot be intersected by level-$(\ell-1)$ grid lines. Otherwise, suppose R and R' belong to different levels $\ell < \ell'$, respectively. Let Q be the union of level-ℓ' grid cells that overlap with square R. Notice that Q is a rectangle, and R' cannot be inside Q; otherwise, R would have intersected the boundaries of $\mathsf{cell}(R')$, contradicting the fact that they are not conflicting. This implies that Q and $\mathsf{cell}(R')$ are disjoint. Let L be any level-$(\ell'-1)$ grid line that goes through a side of Q and separates Q from $\mathsf{cell}(R')$. Again, L cannot intersect any square in $\mathcal{R}_{P'}$ (since squares in $\mathcal{R}_{P'}$ are only at levels $\ell', \ell'+1, \ldots$), and it separates R from R' in a way that maintains the properties. ◀

2.3 A cutting strategy for rectangles of the same width/height

Let $\mathcal{R}$ be a set of weighted rectangles of the same width or height . In this section, we prove the following lemma:

▶ **Lemma 13.** *There is a polynomial time algorithm to find a guillotine strategy that separates a set of rectangles of $\mathcal{R}$ with at least 1/2-fraction of its weight.*

Proof. Without loss of generality, we can assume that the rectangles in $\mathcal{R}$ have unit width. For every $x \in [0,1]$, consider the set L_x of vertical lines at coordinates $\{x+2k\colon k \in \{0,\ldots,N\}$. Let $\mathcal{R}_x$ be the set of rectangles in $\mathcal{R}$ obtained by removing all rectangles intersecting a line in L_x, and all rectangles whose right side is contained in a line of L_x.

It is easy to see that $\mathcal{R}_x$ is guillotine separable. Indeed, we can first cut through all the lines in L_x to obtain a collection of vertical slabs of width 2 (In fact, we do not need to cut through all the $O(N)$ lines, since this could be non-polynomial in the number of rectangles. Precisely, we do not cut through the lines separating slabs without rectangles in $\mathcal{R}_x$). The rectangles in $\mathcal{R}_x$ of each vertical slab can then be separated using horizontal cuts. This is always possible, since the width of the slab does not allow two unit width rectangles side by side (recall that we remove the ones whose right side is aligned with a line in L_x).

Observe that if we choose x uniformly at random from $[0,1]$, then every given rectangle $R \in \mathcal{R}$ belongs to $\mathcal{R}_x$ with probability $1/2$. This means that there is a value x for which $\mathcal{R}_x$ contains at least $1/2$ of the total weight of $\mathcal{R}$. In fact, this value can be found deterministically by standard techniques (the number of values of x with different sets $\mathcal{R}_x$ is linear in the number of rectangles). This concludes the proof of the lemma. ◀

2.4 Guillotine Cuts for d-Dimensional Boxes

By extending the ideas of this section, we can to get a $1/2^{O(d)}$-guillotine sampling for any set of d-dimensional cubes of arbitrary sizes. We first need to generalize the notion of guillotine cuts to higher dimensions. We say that a hyperplane is a *canonical hyperplane* if it is orthogonal to a vector in the standard basis, i.e. those hyperplanes illustrated by $x_i = a$ for $i \in \{1, \ldots, d\}$ and $a \in \mathbb{R}$ would be canonical. When we say a piece $P \subseteq \mathbb{R}^d$, we mean the region that are bounded by $2d$ canonical hyperplanes, i.e. P can be represented by the intersection of $2d$ halfspaces and corresponds to $\bigcap_{i=1}^{d} \{x : (a_i \leq x_i \leq b_i)\}$.

Cutting a piece P along a canonical hyperplane h gives us two sub-pieces. A cutting strategy is represented by a binary tree $\mathcal{T}$ where each non-leaf node $v \in V(\mathcal{T})$ is equipped with a piece P_v and a hyperplane h_v such that cutting P_v along h_v gives us P_{v_1} and P_{v_2} where v_1 and v_2 are the children of v. We say that a set of objects $\mathcal{O}$ is guillotine separable if there is a cutting strategy $\mathcal{T}$ for $\mathcal{O}$ such that:

- The piece P_r associated with the root node contains all objects in $\mathcal{O}$.
- For each non-leaf node v, the canonical hyperplane h_v intersects no object inside P_v.
- For each leaf node v, the piece P_v contains only one object in $\mathcal{O}$.

To handle the weighted case, we can define a similar concept of guillotine sampling for objects. We prove the following theorem:

▶ **Theorem 14.** *There is an ϵ-guillotine sampling for any set of cubes of arbitrary sizes, for $\epsilon = 1/2^{O(d)}$.*

The rest of this section is spent on proving this theorem. The proof follows the same line of ideas used in the case of two dimensions. There are two steps. In the first step, we define the "high-dimensional grid" that will be used to suggest how the hyperplane will be selected. In the second step, we define the conflict graph that is shown to be $2^{O(d)}$ colorable.

Let $\mathcal{R}$ be a set of input cubes of arbitrary sizes. We say that a cube $r \in \mathcal{R}$ is at level-i if its edge length is in $(N/2^{i+1}, N/2^i]$. We define *special hyperplanes* of various granularities. For each integer i, for each axis x_j, the special hyperplanes of type-j are drawn so that consecutive hyperplanes are $N/2^i$ apart in distance (i.e. one can think of special hyperplanes of type-j as those corresponding to $x_j = kN/2^i$ for all integers k). The level-i grid cells are those regions that are bounded by consecutive special hyperplanes from level-$(i-1)$. The following lemma encapsulates the first step:

▶ **Lemma 15.** *There exists a distribution $\mathcal{D} : 2^{\mathcal{R}} \to [0, 1]$ such that each subset $\mathcal{R}'$ in its support admits a drawing of special hyperplanes with the following properties:*

- *Each cube at level-i is not intersected by hyperplanes at levels $i-1, \ldots, 0$.*
- *Each level-i grid cell has at most one cube in level-i.*

Moreover, each cube $r \in \mathcal{R}$ appears in a randomly drawn subset with probability at least $\epsilon_1 = 1/2^{3d}$.

Proof. For each dimension j, we pick a random shift s_j and draw level-i special hyperplanes at $s_j + N/2^i, s_j + 2 \times N/2^i$, and so on. The probability that each level-i cube $r \in \mathcal{R}$ of edge length ℓ_r is intersected by special hyperplanes of type-$(i-1)$ is exactly $(1 - \ell_r 2^{i-1}/N)^d \geq 1/2^d$ since $\ell_r \leq N/2^i$. This is the probability that each cube remains after placing the special hyperplanes.

Now, inside each cell, there can be more than one cube, and in such a case, we randomly select one of them to keep. This ensures that each cube that survives the first phase remains with probability at least $1/2^{2d}$ because there can be at most 2^{2d} cubes inside each cell (just because of the volume). This implies the lemma. ◀

Next, we consider a subset $\mathcal{R}'$ in the support of the distribution given by the above lemma. For each level-i cube $r \in \mathcal{R}$, we define $\mathsf{cell}(r)$ as the level-i grid cell that contains r. We say that two cubes r and r' are *conflicting* if either r overlaps the boundary of $\mathsf{cell}(r')$ or r' overlaps the boundary of $\mathsf{cell}(r)$.

We define the conflict graph H such that the vertex set $V(H)$ corresponds to the cubes in $\mathcal{R}'$, and there is an edge between r and r' if and only if r and r' are conflicting. The following two lemmas finish the proof.

▶ **Lemma 16.** *The graph H is $2^{O(d)}$ colorable.*

Proof. We prove this by induction on the number of vertices. The base case, when $|V(H)| = 1$, is obvious. Now consider any graph H with at least two vertices, and a cube $r \in V(H)$ whose size is minimum among the cubes in $V(H)$. Notice that $\mathsf{cell}(r)$ has $2d$ bounding hyperplanes.

Now, consider the neighborhood $N_H(r) \subset V(H)$ of r. There can be at most 2^d elements of $N_H(r)$ that contain some corner of $\mathsf{cell}(r)$. We then count those that do not contain any corner. Each cube $r' \in N_H(r)$ must be at the level below of that of r, so the overlapping volume of the intersection between r' and the bounding hyperplane of $\mathsf{cell}(r)$ must be at least $1/4^d$ fraction of the total surface volume of such bounding hyperplane. Then, each such hyperplane supports at most 4^d cubes in $N_H(r)$, and since there are $2d$ bounding hyperplanes in total, we have $2d4^d$ cubes that do not intersect any corner of $\mathsf{cell}(r)$.

This implies that $|N_H(r)| \leq 2d4^d + 2^d \leq 2d2^{3d} = 2^{O(d)}$. By induction hypothesis, we are done: We can inductively color the graph resulting from removing the vertex r from H, and since there are only $2^{O(d)}$ neighbors of r, we can assign a distinct color to r. ◀

Finally, the following lemma can be proved similarly to the 2-dimensional case.

▶ **Lemma 17.** *Let I be an independent set of H. Then I is guillotine separable.*

Proof. We argue that we can iteratively define a sequence of cuts that separate all cubes. Let P' be a piece that currently contains at least 2 cubes. Let R and R' be the largest and second largest cubes contained in P', respectively. If they are from the same level, we would be done: Pick any special hyperplane that separates the two cells $\mathsf{cell}(R)$ and $\mathsf{cell}(R')$. Otherwise, if R and R' are from levels $\ell < \ell'$, respectively, we define Q as the union of level-ℓ' cells that overlap with R. Notice that R' cannot be inside Q; otherwise, this would have contradicted the fact that they are independent in H. Now, we can pick any special hyperplane that separates Q from R'. ◀

3 Negative result for unit squares

In this section, we give a sequence of instances W_n for which no guillotine strategy separates more than a fraction of $\frac{1}{2} + o(1)$. All these instances are formed by unit squares.

Given positive integers a, and b, the *brick wall* $W(a, b) = \{S(i, j) \colon 1 \leq i \leq a, 1 \leq j \leq b\}$ is the set of unit squares such that the lower left corner of $S(i, j)$ is at coordinates $i \cdot (1, \delta) + j \cdot (-\delta, 1) = (i - j\delta, j + i\delta)$, where δ is an arbitrarily small positive constant (say, smaller than $1/(ab)$). For example, Figure 3 shows $W(5, 4)$. Let $W_n = W(n, n)$.

▶ **Lemma 18.** *The number of squares of W_n that can be separated by a guillotine strategy is at most $\left\lceil \frac{n^2+2n-2}{2} \right\rceil \leq \frac{n^2}{2} + n$.*

In fact, we prove a more general version of Lemma 18. Define a *subwall* of a brick wall $W(a, b)$ as the subset of squares of $W(a, b)$ contained inside a given rectangle. For example,

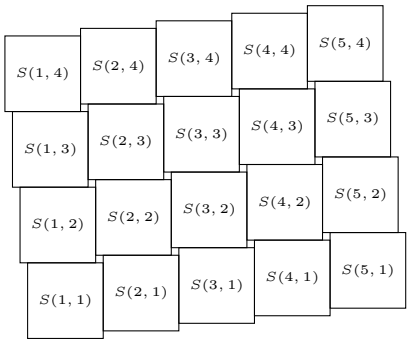


Figure 3 The grid wall $W(5,4)$.

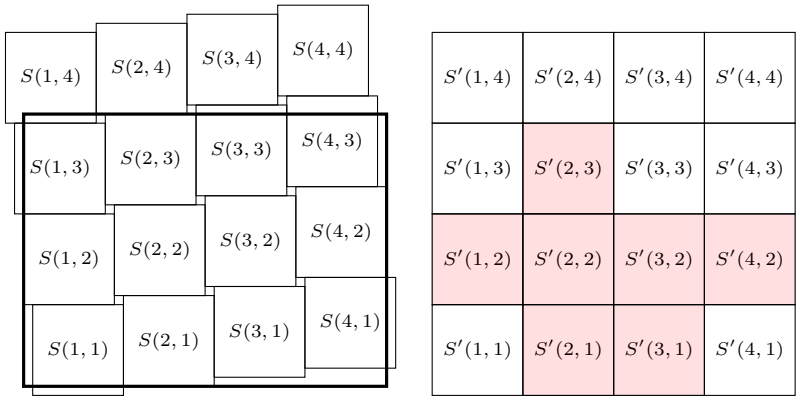


Figure 4 A subwall q of $W(4,4)$. On the right, the corresponding squares of q in the standard unit grid. The region has an area of 7 and a perimeter of 14.

in Figure 4 (left) the set $\{S(2,1), S(3,1), S(1,2), S(2,2), S(3,2), S(4,2), S(2,3)\}$ is a subwall of $W(4,4)$. It is easier to note the subwall in the "standard" square grid, as shown in Figure 4 (right). Define the *area* $A(q)$ of a subwall q as the number of squares it contains, and the *perimeter* $P(q)$ as the number of edges (i.e. sides of squares) such that only one of its incident squares is in q. The previous concepts coincide with the actual geometric area and perimeter of the union of the corresponding squares in a standard square grid. Let also $S(q)$ denote the maximum number of squares that can be separated using a guillotine cutting sequence.

▶ **Lemma 19.** *For any subwall q,*

$$S(q) \;\leq\; \left\lceil \frac{2A(q) + P(q) - 4}{4} \right\rceil.$$

Proof. The proof is by induction on $A(q)$. When $A(q) = 1$, $S(q) = 1$ and $\lceil \frac{2A(q)+P(q)-4}{4} \rceil = \lceil \frac{2+4-4}{4} \rceil = 1$, so assume that $A(q) \geq 2$. Consider the first guillotine cut of an optimal cutting sequence for q (the one achieving $S(q)$). Without loss of generality, assume it is an horizontal cut. This cut divides q into two subwalls q_1 and q_2, both with at least one complete square inside. Let r be the number of squares cut in this first iteration. See Figure 5 for reference, where the cut squares are shown in black. It is easy to see that:

$$A(q) \;=\; A(q_1) + A(q_2) + r,$$

and

$$P(q_1) + P(q_2) - 2(r+1) \;\leq\; P(q) \;\leq\; P(q_1) + P(q_2) - 2r.$$

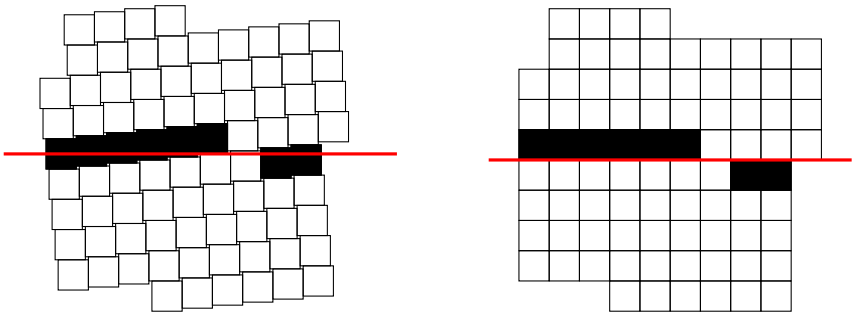


Figure 5 An horizontal guillotine cut.

Note that $P(q)$ attains the lower bound if the horizontal cut goes exactly through a square edge (like in Figure 5). Otherwise, an extra square is destroyed and $P(q)$ attains the upper bound. Using the previous observation we have that

$$\begin{aligned}
S(q) &= S(q_1) + S(q_2) && \text{(by optimality)}\\
&\leq \left\lceil \frac{2A(q_1) + P(q_1) - 4}{4} \right\rceil + \left\lceil \frac{2A(q_1) + P(q_1) - 4}{4} \right\rceil && \text{(inductive step)}\\
&\leq \left\lceil \frac{(2A(q_1) + 2A(q_2)) + (P(q_1) + P(q_2)) - 6}{4} \right\rceil && (\star)\\
&\leq \left\lceil \frac{(2A(q) - 2r) + (P(q) + 2r + 2) - 6}{4} \right\rceil\\
&= \left\lceil \frac{2A(q) + P(q) - 4}{4} \right\rceil .
\end{aligned}$$

To conclude the proof we need to check the validity of inequality $(\star)$. Note first that the perimeter of any subwall is always even (because it can be computed as the length of a closed path on the integer grid). Using this we only need to prove, for every pair of even integers x and y, that

$$\lceil x/4 \rceil + \lceil y/4 \rceil \ \leq \ \lceil (x + y + 2)/4 \rceil,$$

or equivalently, for every pair of integers a and b, that

$$\lceil a/2 \rceil + \lceil b/2 \rceil \ \leq \ \lceil (a + b + 1)/2 \rceil.$$

This is direct: if both a and b are even, then the left hand side (LHS) is $(a + b)/2$ and the right hand side (RHS) is $(a + b)/2 + 1$. If only one of them is even, then both the LHS and the RHS are equal to $(a + b + 1)/2$. If both numbers are odd, then both the LHS and RHS are equal to $(a + b + 2)/2$. ◀

Lemma 18 follows immediately from Lemma 19.

4 Proof of Theorem 2

We assume that Conjecture 1.1 is true, and present an $O(1)$-approximation algorithm for MISR. In our reasoning here, we assume that the given rectangles are open sets. Suppose we are given a set of axis-parallel rectangles $\mathcal{R}$. Since we consider the cardinality case in MISR, we can assume w.l.o.g. that the input does not contain any two rectangles R, R' such that

$R \subseteq R'$. The algorithm is essentially the algorithm GEO-DP from [1], when parametrized by $k = 4$. First, we can assume w.l.o.g. that all rectangles in $\mathcal{R}$ have integer coordinates in the range $\{0, ..., 2n\}$. This can be achieved easily by suitable stretching of the instance. Our algorithm is a dynamic program (DP) with a cell for every open rectangle $P \subseteq [0, 2n] \times [0, 2n]$ whose corners have integer coordinates. Note that there are $O(n^4)$ many of them. Each of these cells represents the problem of selecting the optimal Independent Set that consists only of rectangles that lie completely within P, i.e., rectangles R with $R \subseteq P$. Intuitively, the DP stores a near-optimal solution to this subproblem in the respective cell.

When computing the entry for such a cell, representing a rectangle P, the DP does the following procedure. In the case where the rectangle P does not (completely) contain any input rectangle, we define the empty set to be the solution corresponding to the cell. Also, in the case where P coincides with an input rectangle R, i.e. $P = R$, then from our assumption above we know that there is no rectangle R' with $R \neq R'$ such that $R' \subseteq P$ and therefore, we define $\{R\}$ to be the solution corresponding to the cell. Otherwise, it tries all possibilities of dividing P into two smaller pieces using a horizontal or vertical guillotine cut such that the horizontal/vertical coordinate of this cut is an integer (since the rectangles have integral coordinates we can safely restrict ourselves to those). Consider one such cut and let $P_1 \neq \emptyset \neq P_2$ denote the resulting pieces. The DP looks up the solutions for the cells representing P_1 and P_2 and combines them to a solution for P. It selects the cut yielding the optimal total profit from the resulting two subproblems. Since there are $O(n)$ possible cuts for each rectangle P, it takes $O(n)$ time to compute the solution for a given cell. Finally, we output the solution that the DP computes for the cell corresponding to the rectangle $[0, 2n] \times [0, 2n]$. Since there are $O(n^4)$ cells in total, we obtain a total running time of $O(n^5)$.

To analyze this algorithm, we first show that it finds the largest subset of the input rectangles that is guillotine separable. This can be easily shown by induction over the cells, i.e., for each cell the algorithm computes a solution that is at least as profitable as the best guillotine separable solution consisting only of rectangles completely contained in the rectangle that the cell represents.

▶ **Lemma 20.** *Let $\mathcal{R}' \subseteq \mathcal{R}$ be a largest subset of $\mathcal{R}$ that is guillotine separable. Then, the DP finds a set of size $|\mathcal{R}'|$.*

If now Conjecture 1.1 holds, then $|\mathcal{R}'| \geq \Omega(|OPT|)$, where OPT denotes the optimal solution to the given MISR instance. Hence, the DP finds a solution with at least $\Omega(|OPT|)$ rectangles and it is hence a $O(1)$-approximation algorithm for MISR. This completes the proof of Theorem 2.

We would like to note that the resulting algorithm is purely combinatorial and that it is faster than solving the natural LP-relaxation of the problem (which is a natural candidate for obtaining an $O(1)$-approximation algorithm for MISR).

5 QPTAS for two-dimensional geometric knapsack

In this section, we present a QPTAS for the unweighted two-dimensional guillotine knapsack problem. It builds on the recent QPTAS for the problem without the guillotine constraint [2] but requires substantial new ideas to handle this constraint. Like the latter QPTAS we require all input data to be quasi-polynomially bounded integers.

Let $\epsilon > 0$ and suppose that the knapsack has a capacity of $N \times N$ with $N \in \mathbb{N}$. For each item i denote by h_i and w_i its height and width, respectively. By standard shifting arguments we can assume that there are values $\mu, \delta > 0$ such that for each item i we have

that $h_i \in [0, \mu \cdot N] \cup [\delta \cdot N, N]$ and $w_i \in [0, \mu \cdot N) \cup [\delta \cdot N, N]$, while losing only factor $1 + \epsilon$ in the approximation ratio. The values μ, δ can be chosen such that $1 \geq \delta > \mu \geq (\frac{\epsilon}{\log n})^{O_\epsilon(1)}$ and $\mu = (\delta \cdot \frac{\epsilon}{\log n})^{O_\epsilon(1)}$ (see Lemma 1 in [2]). Moreover, since we study the unweighted case of the problem and allow quasi-polynomial running time, it is sufficient to compute a solution with a set of items I' such that $|I'| \geq (1 - \epsilon)|OPT| - (\frac{\log n}{\epsilon})^{O_\epsilon(1)}$ (see Proposition 2.1 in [2]). We will call such solutions *near-optimal* solutions in the sequel. Using this property, we assume from now on that there are no items i with $h_i \geq \delta \cdot N$ and $w_i \geq \delta \cdot N$ since there can be only $1/\delta^2 = O_\epsilon(\log n)^{O_\epsilon(1)}$ of them in the optimal solution OPT. We partition the remaining items into horizontal, vertical, and tiny items, given by sets H, V, and T, respectively. An item i is *horizontal* if $w_i \geq \delta \cdot N$ and $h_i \leq \mu \cdot N$, *vertical* if $w_i \leq \mu \cdot N$ and $h_i \geq \delta \cdot N$, and *tiny* if $w_i \leq \mu \cdot N$ and $h_i \leq \mu \cdot N$.

Box partition. The key ingredient in the QPTAS in [2] is a partition of the knapsack into boxes which intuitively describe the topology of the optimal solution. More precisely, it is shown that if $OPT \cap T = \emptyset$ then there exists a partition of the knapsack into a set $\mathcal{B}$ of at most $(\frac{\log n}{\epsilon})^{O_\epsilon(1)}$ axis-parallel rectangular boxes and a near-optimal solution OPT' such that each box $B \in \mathcal{B}$ either contains only items from $OPT' \cap H$ or only items from $OPT' \cap V$ (the tiny items are added later into the remaining empty space). By being more careful in the construction, we can prove the following stronger statement which also takes our guillotine cutting constraint into account. Note that the following lemma is non-constructive, so our algorithm has to guess the partition given by it.

▶ **Lemma 21.** *There is a partition of the $N \times N$ knapsack into at most $(\frac{\log n}{\epsilon})^{O_\epsilon(1)}$ rectangular boxes $\mathcal{B}$ with integral coordinates and a near-optimal solution $OPT' \subseteq OPT$ with the following properties:*

- *each item in OPT' is fully contained in some box $B \in \mathcal{B}$,*
- *each box $B \in \mathcal{B}$ is either a horizontal box which contains only items in $OPT' \cap (H \cup T)$ or a vertical box which contains only items in $OPT' \cap (V \cup T)$,*
- *OPT' is constructed by removing some items from OPT and moving the remaining items within the knapsack so that no horizontal item is moved to the left or right and no vertical items is moved up or down,*
- *for each box $B \in \mathcal{B}$ there exists a guillotine cutting sequence which cuts all items of OPT' that are contained in B.*

Proof. We start with the solution OPT that is the optimal solution satisfying the guillotine cut constraint. We use the same construction as in the proof of Lemma 2 in [2]. The proof of the lemma follows. Note that the last property is satisfied since OPT satisfies the guillotine cuts constraint by assumption, and after the modifications in the proof of Lemma 2 in [2] the property is still true for each box $B \in \mathcal{B}$. ◀

Guillotine cutting sequence. Using Lemma 21 we construct a guillotine cutting sequence which intersects the items of OPT' in a very controlled way, as given by the following lemma.

▶ **Lemma 22.** *There exists a guillotine cutting sequence with at most $(\frac{\log n}{\epsilon})^{O_\epsilon(1)}$ cuts, cutting the knapsack into a set $\mathcal{B}'$ of rectangular pieces, such that the following properties hold:*

- *for each piece $B' \in \mathcal{B}'$ there is a box $B \in \mathcal{B}$ such that $B' \subseteq B$,*
- *at most $(\frac{\log n}{\epsilon})^{O_\epsilon(1)}$ items from $OPT' \cap (H \cup V)$ are intersected,*
- *the intersected items from $OPT' \cap T$ have a total area of $\leq (\frac{\epsilon}{\log n})^{O_\epsilon(1)} \cdot N^2$.*

The intersected items from $OPT' \cap (H \cup V)$ will be lost which is justified since we are only interested in a near-optimal solution. For the intersected tiny items, we will argue that we can free up some space in the boxes in $\mathcal{B}'$ at only marginal cost to accommodate them.

We construct our guillotine cutting sequence now. Consider the first cut of the cutting sequence of OPT (which does not intersect any item of OPT!) and assume w.l.o.g. that it is a horizontal cut given by the line segment $[0, N] \times \{h\}$. By Lemma 21 we can show that if we used exactly the same cut in OPT' then we would not intersect any item in $OPT' \cap V$. Denote by h_1 the smallest integer such that $h_1 \geq h$ and either $[0, N] \times \{h_1 + 1\}$ split an item in $OPT' \cap V$, or there is a box $B \in \mathcal{B}$ such that $[0, N] \times \{h_1\}$ cuts along the top edge of B. Similarly, denote by h_2 the largest integer such that $h_2 \leq h$ and either $[0, N] \times \{h_2 - 1\}$ split an item in $OPT' \cap V$, or there is a box $B' \in \mathcal{B}$ such that $[0, N] \times \{h_2\}$ cuts along the bottom edge of B'. Our first two cuts are $[0, N] \times \{h_1\}$ and $[0, N] \times \{h_2\}$. We then cut the resulting piece between the two cuts by vertical cuts, such that we obtain smaller pieces having non-empty intersection with at most one box in $\mathcal{B}$. We continue iteratively on the other resulting pieces. Consider the piece $B_1 := [0, N] \times [0, h_1]$. Observe that it is contained in the piece $B_0 := [0, N] \times [0, h]$ which is the first piece that the optimal cutting sequence obtains. For cutting B_1 further, we consider the next cut on B_0 of the optimal cutting sequence which also cuts B_1 (i.e., we ignore cuts of the form $[0, N] \times \{\bar{h}\}$ with $h_1 < \bar{h} < h$). Suppose that it is a vertical cut $\{h'\} \times [0, h]$. Similarly as before, we denote by h'_1 the smallest integer such that $h'_1 \geq h'$ and either $\{h'_1 + 1\} \times [0, h_1]$ split an item in $OPT' \cap H$, or there is a box $B \in \mathcal{B}$ such that $\{h'_1\} \times [0, h_1]$ cuts along the right edge of B. We define h'_2 accordingly. Our next two cuts for B_1 are $\{h'_1\} \times [0, h_1]$ and $\{h'_2\} \times [0, h_1]$. We continue in the same manner till we end up with the set $\mathcal{B}'$.

For each box in $B \in \mathcal{B}$ we bound the number of cuts that go through B. There are two types of these cuts. Cuts of the first type contains the corner of some box in $\mathcal{B}$. Thus, the number of these cuts is at most $(\frac{\log n}{\epsilon})^{O_\epsilon(1)}$. For the other type of cuts we can give a charging scheme such that each box pays for at most $O(1/\delta)$ of them. Thus, the number of boxes in $\mathcal{B}'$ is bounded by $O(1/\delta) \cdot (\frac{\log n}{\epsilon})^{O_\epsilon(1)} \cdot |\mathcal{B}| \leq (\frac{\log n}{\epsilon})^{O_\epsilon(1)}$. Therefore, the total number of cuts is bounded by the same value. Each cut intersects at most $1/\delta = (\frac{\log n}{\epsilon})^{O_\epsilon(1)}$ horizontal or vertical items and tiny items with a total area of at most $N \cdot \mu N \leq (\frac{\epsilon}{\log n})^{O_\epsilon(1)} \cdot N^2$.

Adding back tiny items. In the above process, we intersected some tiny items (with small total area). We do not want to lose them so we add them back now. To this end, we drop some of the remaining items and create an empty space in each box $B' \in \mathcal{B}'$. Suppose that B' contains only items from $OPT' \cap (H \cup T)$ and has height $h_{B'}$. Denote by $OPT'_{B'}$ the items of OPT' inside B'. We identify a horizontal slice of height $\epsilon \cdot h_{B'}$ inside B' such that there are at most $O(\epsilon) \cdot |OPT'_{B'}| + O(\frac{1}{\delta})$ items in $OPT'_{B'}$ that intersect this slice. We drop all these items. By doing this with each box $B' \in \mathcal{B}'$ we can show that this creates enough empty space to put back almost all tiny items that were intersected by our cutting sequence above. Even more, by assigning them into the empty space by the Next-Fit-Decreasing-Height (NFDH) algorithm [6] we can ensure that also a guillotine cutting sequence for them exists. Overall, we obtain the property that for the items in each box $B' \in \mathcal{B}'$ there exists a guillotine cutting sequence.

Details of this operation are the following: Observe that the number of guillotine cuts that separate the boxes in $\mathcal{B}'$ is exactly $|\mathcal{B}'| - 1$. Every cut intersects tiny items with total area at most μN^2. The total area of the tiny items intersected in all cuts is $\mu N^2 |\mathcal{B}'|$ which we can upper-bound by $\epsilon^3 N^2$ by choosing μ and δ such that $\mu |\mathcal{B}'| \leq \epsilon^3$. We call a box $B' \in \mathcal{B}'$ a *good box* if $h(B') \geq \frac{\mu N}{\epsilon^2}$ and $w(B') \geq \frac{\mu N}{\epsilon^2}$. Otherwise the box is called a *bad box*.

The total area of bad boxes is at most $(\frac{\mu N}{\epsilon^2})N|\mathcal{B}'|$. Recall that $\mu|\mathcal{B}'| \leq \epsilon^3$. This implies that the total area of bad boxes is at most ϵN^2. This means that the total area of good boxes is at least $(1-\epsilon)N^2$. In good boxes, we create empty space to accommodate the tiny items whose total area is at least $(\epsilon - \epsilon^2)N^2$. Since we use the NFDH algorithm to accommodate the tiny items, we only lose a constant fraction of the area [6]. Note that by definition any tiny item fits into the created empty space of a good box. This implies that we are able to accommodate all tiny items. We note that using the NFDH algorithm we can ensure that a guillotine cutting sequence for the tiny items exist.

Rounding of item sizes. As the last step of the non-constructive part, we round the sizes of the items such that at the end we have only $(\frac{\log n}{\epsilon})^{O_\epsilon(1)}$ many items types. We say that two items i, i' are of the same type if $h_i = h_{i'}$ and $w_i = w_{i'}$. We round the item sizes in each box in $\mathcal{B}'$ separately. Like above, we create empty space of height $\epsilon \cdot h_{B'}$ inside each horizontal box $B' \in \mathcal{B}'$ while dropping only few items, at most $\epsilon|OPT'| + (\frac{\log n}{\epsilon})^{O_\epsilon(1)}$ many (and we do a similar adjustment for the vertical boxes). We use this empty space to round up the height of each item to the next larger power of $1+\epsilon$, yielding $O_\epsilon(\log n)^{O(1)}$ many height classes (we use here that the input data are quasi-polynomially bounded integers). Using harmonic grouping like in De La Vega and Lueker [8] we round the widths of the items such that among each height class there are only $O_\epsilon(1)$ many widths arising, yielding $O_\epsilon(\log n)^{O(1)}$ different items types in total.

Algorithm. The algorithm first guesses the cutting sequence according to Lemma 22 for which there are only $n^{(\frac{\log n}{\epsilon})^{O_\epsilon(1)}}$ many options. Then, it guesses the item types that arise after the previous rounding. Note here that the height and width of each item type is either a power of $1+\epsilon$ or coincides with the height or width of an input item, and thus we have to choose $(\frac{\log n}{\epsilon})^{O_\epsilon(1)}$ types from $n^{O(1)}$ possible ones. So there are only $n^{(\frac{\log n}{\epsilon})^{O_\epsilon(1)}}$ many options for this in total. For each item type we guess how many items there are in the solution that we constructed above. Then we verify that our input items are consistent with this guess which can be done by finding a perfect bipartite matching. In this matching we have a node for every input item, and a set of nodes for each item type. For each item type the number of nodes equals the guessed number of items of this type in the searched-for solution. An edge between a node for an input item and a node for a guessed item of a type is added if the input item can be drawn inside the guessed item. In this case finding a perfect matching means that the guess is consistent with the input items, otherwise the guess is rejected. For each box $B' \in \mathcal{B}'$ we guess how many items of each type we have to assign in the box such that for them there is a guillotine cutting sequence. It remains to verify for each box $B' \in \mathcal{B}'$ that the items we guessed to be assigned to it actually fit into B'. To this end, we use a dynamic program. It guesses the first cut of the (existent) cutting sequence of the items and then guesses how we have to partition the items to the two sides of the cut. Then we recurse on both sides. Each arising subproblem is specified by a remaining rectangular piece and a set of items that is to be assigned to it. Since for both quantities together there are only $n^{(\frac{\log n}{\epsilon})^{O_\epsilon(1)}}$ many options, also this dynamic program runs in quasi-polynomial time. For the version of the problem where we are allowed to rotate items by 90 degrees the above methods can be adjusted easily. This completes the proof of Theorem 3.

Acknowledgements. J. Correa, P. Pérez-Lantero, and J. Soto are supported by Millennium Nucleus Information and Coordination in Networks ICM/FIC RC130003 (Chile). J. Soto is additionally supported by FONDECYT Grant 11130266. A. Karrenbauer is supported by the Max Planck Center for Visual Computing and Communication (`http://www.mpc-vcc.org`).

References

1 Anna Adamaszek and Andreas Wiese. Approximation schemes for maximum weight independent set of rectangles. In *Proceedings of the 54th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2013)*, pages 400–409. IEEE, 2013.

2 Anna Adamaszek and Andreas Wiese. A quasi-PTAS for the two-dimensional geometric knapsack problem. In *Proceedings of the 26th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2015)*, pages 1491–1505, 2015.

3 Nikhil Bansal, Alberto Caprara, Klaus Jansen, Lars Prädel, and Maxim Sviridenko. A structural lemma in 2-dimensional packing, and its implications on approximability. In *Algorithms and Computation (ISAAC 2009)*, volume 5878 of *LNCS*, pages 77–86. Springer, 2009.

4 Parinya Chalermsook and Julia Chuzhoy. Maximum independent set of rectangles. In *Proceedings of the 20th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2009)*, pages 892–901. SIAM, 2009.

5 Timothy M. Chan and Sariel Har-Peled. Approximation algorithms for maximum independent set of pseudo-disks. *Discrete & Computational Geometry*, 48:373–392, 2012.

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

7 Thomas Erlebach, Klaus Jansen, and Eike Seidel. Polynomial-time approximation schemes for geometric intersection graphs. *SIAM Journal on Computing*, 34(6):1302–1323, 2005.

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

9 Aleksei V. Fishkin, Olga Gerber, Klaus Jansen, and Roberto Solis-Oba. Packing weighted rectangles into a square. In *Mathematical Foundations of Computer Science (MFCS 2005)*, volume 2337 of *LNCS*, pages 352–363. Springer, 2005.

10 Rolf Harren. Approximating the orthogonal knapsack problem for hypercubes. In *Automata, Languages and Programming (ICALP 2006)*, volume 4051 of *LNCS*, pages 238–249. Springer, 2006.

11 Klaus Jansen and Roberto Solis-Oba. New approximability results for 2-dimensional packing problems. In *Mathematical Foundations of Computer Science (MFCS 2007)*, volume 4708 of *LNCS*, pages 103–114. Springer, 2007.

12 Klaus Jansen and Roberto Solis-Oba. A polynomial time approximation scheme for the square packing problem. In *Proceedings of the 13th Integer Programming and Combinatorial Optimization Conference (IPCO 2008)*, pages 184–198. Springer, 2008.

13 Klaus Jansen and Guochuan Zhang. Maximizing the number of packed rectangles. In *Proceedings of the 9th Scandinavian Workshop on Algorithm Theory (SWAT 2004)*, pages 362–371. Springer, 2004.

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

15 János Pach and Gábor Tardos. Cutting glass. In *Proceedings of the 16th Annual Symposium on Computational Geometry (SOCG 2000)*, pages 360–369. ACM, 2000.

16 Jorge Urrutia. Problem presented at ACCOTA'96: Combinatorial and Computational Aspects of Optimization, Topology, and Algebra, Taxco, Mexico, 1996.