A Parameterized Approximation Scheme for the Geometric Knapsack Problem with Wide Items

Mathieu Mari
Institute of Informatics, University of Warsaw, Poland
IDEAS-NCBR, Warsaw, Poland

Timothé Picavet 🄳
ENS de Lyon, France
Aalto University, Finland

Michał Pilipczuk ✉ 🄳
Institute of Informatics, University of Warsaw, Poland

---- **Abstract** ----

We study a natural geometric variant of the classic KNAPSACK problem called 2D-KNAPSACK: we are given a set of axis-parallel rectangles and a rectangular bounding box, and the goal is to pack as many of these rectangles inside the box without overlap. Naturally, this problem is NP-complete. Recently, Grandoni et al. [ESA'19] showed that it is also W[1]-hard when parameterized by the size k of the sought packing, and they presented a parameterized approximation scheme (PAS) for the variant where we are allowed to rotate the rectangles by 90° before packing them into the box. Obtaining a PAS for the original 2D-KNAPSACK problem, without rotation, appears to be a challenging open question.

In this work, we make progress towards this goal by showing a PAS under the following assumptions:

- both the box and all the input rectangles have integral, polynomially bounded sidelengths;
- every input rectangle is wide – its width is greater than its height; and
- the aspect ratio of the box is bounded by a constant.

Our approximation scheme relies on a mix of various parameterized and approximation techniques, including color coding, rounding, and searching for a structured near-optimum packing using dynamic programming.

1 Introduction

We study a natural geometric variant of the classic KNAPSACK problem, called 2D KNAPSACK and defined as follows. On input, we are given a rectangular box B and a set $\mathcal{R}$ of items, each being a rectangle. The task is to place as many items from $\mathcal{R}$ as possible in B so that the placed items do not overlap. Note that this problem generalizes classic KNAPSACK: given

18th International Symposium on Parameterized and Exact Computation (IPEC 2023).
Editors: Neeldhara Misra and Magnus Wahlström; Article No. 33; pp. 33:1–33:20
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

an instance of KNAPSACK with items of sizes $a_1, \ldots, a_n$ and a knapsack of size K, we can create an instance of 2D KNAPSACK with B being the $K \times 1$ rectangle and $\mathcal{R}$ consisting of $a_i \times 1$ rectangles, for all $i \in \{1, \ldots, n\}$.

As in the case of KNAPSACK, there are two natural variants of the problem depending on how the input is encoded. In the *binary variant*, both B and the rectangles of $\mathcal{R}$ have integral sidelengths encoded in binary, hence these sidelengths can be exponential in the total input size. In the *unary variant* the difference is that the sidelengths are encoded in unary, or equivalently, one assumes that all the sidelengths are bounded polynomially in the total input size. In this work we focus on the unary variant.

Again as in the case KNAPSACK, adopting the unary variant helps tremendously for the design of algorithms for 2D KNAPSACK, for instance due to allowing to perform dynamic programming over the dimensions of the box. While the problem remains NP-hard even in the unary variant [10], Adamaszek and Wiese [1] gave a QPTAS for the problem in this setting. The best approximation factor known to be achievable in polynomial time in the unary variant is $4/3 + \varepsilon$ due to Galvez et al. [6]; earlier, a $(2 + \varepsilon)$-approximation was given in [8] and a $(558/325 + \varepsilon)$-approximation was given in [5]. It is believed that the problem should admit a PTAS, but this question remains widely open to this day.

We remark that the abovementioned works also study the weighted variant of the problem. In this work we only consider the unweighted version, hence an interested reader is invited to the relevant discussion in the references.

Recently, Grandoni et al. [7] proposed to approach the question about the existence of a PTAS for 2D KNAPSACK by adding parameterization by the solution size to the picture. That is, they presented a *parameterized approximation scheme* (*PAS*) with running time of the form $k^{\mathcal{O}(k/\varepsilon)} \cdot n^{\mathcal{O}(1/\varepsilon^3)}$ that either finds a packing of size at least $(1 - \varepsilon)k$ or correctly concludes that there is no packing of size k. However, this result applies *only* to the variant of the problem where each input rectangle can be rotated by 90° before packing it into the box, and the question about the existence of a PAS for 2D KNAPSACK without rotation was explicitly left open by Grandoni et al. This is in contrast with the other mentioned works on 2D KNAPSACK which all apply both to the variant with rotation and without rotation (for the variant with rotation, Galvez et al. [6] reported a better approximation ratio of $5/4 + \varepsilon$).

We note that the PAS of Grandoni et al. actually works in the binary variant of the problem. Also, reliance on approximation is probably necessary: as proved in [7], the exact version of the problem is W[1]-hard when parameterized by k.

Our contribution.   In this work we approach – though not completely solve – the open problem left by Grandoni et al. [7] by giving a parameterized approximation scheme with running time of the form $f(k, \varepsilon, \delta) \cdot n^{g(\varepsilon)}$ for 2D KNAPSACK under the following assumptions:

- we consider the unary variant of the problem, thus the dimensions of the box are bounded polynomially in n;
- we assume that every item is *wide*: its width is not smaller than its height; and
- we assume that the *aspect ratio* (ratio between the dimensions, always greater or equal to 1) of the box is at most δ.

See Theorem 1 for a formal statement of our result and an explicit formula for the running time. Note that in the context of the variant with rotation, the second assumption can be always achieved by rotating every input rectangle so that it is wide, while the third assumption for $\delta = 1$ can be obtained by scaling both the box and all rectangles on input.

Let us elaborate on our approach and how it is different from the approach of Grandoni et al. [7]. The approach of [7] can be summarized as follows.

- Consider a hypothetical packing $\mathcal{S}$ of size k.
- *Freeing a strip:* Remove a small fraction of $\mathcal{S}$ and shift the items slightly in order to free up a horizontal strip of height N/k^b at the bottom of the box, where N is the sidelength of the box and $b = \mathcal{O}(1/\varepsilon)$ is an integer. This strip can now accommodate all *thin* items: those of height at most N/k^{b+1}.
- *Resource augmentation:* After the previous step, we may assume that all items are *large* – both dimensions are at least N/k^{b+1} – and there is still a considerable strip free at the bottom of the box. Now, one can round the heights of items up to the nearest multiplicity of, say, N/k^{b+2} and argue that even the rounded items can be packed, due to the free strip at the bottom of the box. After rounding the rectangles have at most $k^{\mathcal{O}(1)}$ different heights, so keeping k narrowest rectangles of each possible height gives us a polynomial in k number of candidate rectangles that can be reasonably used in the packing. This easily leads to a PAS.

The possibility of rotating rectangles is crucially used in the second step, freeing a strip. Without this assumption, thin rectangles come in two different flavors: there are *wide* rectangles, of very small height and possibly large width, and symmetric *tall* rectangles. The strategy of freeing a strip presented in [7] can be applied also in the setting without rotation, but then it results in either freeing a horizontal strip at the bottom of the box, or a vertical strip at the left side of the box; there is no control over which strip will be freed. Consequently, only one type of thin rectangles can be disposed of as a result of freeing the strip, and there is no control over which one it is.

In the setting of Grandoni et al., our assumptions on the problem essentially mean that we allow the existence of wide rectangles, but not of the tall ones. The application of the approach of Grandoni et al. could result in freeing a vertical strip, in which the wide rectangles cannot fit. Consequently, we do not see how to fix the approach presented in [7] to solve our case where only wide rectangles are present and no rotation is allowed. We therefore abandon this approach and propose a completely new one.

Instead, we prove a different result about the existence of a well-structured near-optimum solution. Our structural lemma (Lemma 9) says that at the cost of sacrificing a small fraction of rectangles, the considered packing can be divided into regions $B_1, B_2, \ldots, B_m$ so that:

- Every region B_i is delimited by the left side of the box, the right side of the box, and two x-monotone axis-parallel polylines connecting the left and the right side. Moreover, each of the polylines defining the division $B_1, B_2, \ldots, B_m$ consists of $\mathcal{O}(1/\varepsilon)$ segments.
- Every region B_i is either *light* – contains only $\mathcal{O}(1/\varepsilon^2)$ rectangles from the packing – or *roundable* – rectangles within B_i could be packed inside B_i even after rounding them to the nearest multiple of (roughly) N_1/k^2, where N_1 is the width of the box.

Having such a structural lemma, a near-optimum solution can be constructed using a bottom-up dynamic programming that guesses the regions B_i one by one. For each region B_i we consider two cases: either B_i is light and a solution within it can be guessed (essentially) by brute-force, or B_i is roundable and using the same trick as in [7], one can restrict attention to $k^{\mathcal{O}(1)}$ many different candidates for rectangles that will be packed into B_i.

There is a technical caveat in the plan presented above. Namely, in dynamic programming we need to make sure that we do not reuse the same rectangle from $\mathcal{R}$ in two or more different regions B_i. We resolve this issue using color-coding. Namely, by applying color-coding upfront we may assume that all the rectangles in $\mathcal{R}$ are colored with k colors, and we look for a packing consisting of rectangles of pairwise different colors. Then our dynamic programming keeps track of the subset of colors that have already been used, which adds only another dimension of size 2^k to the dynamic programming table.

2 Preliminaries

Basic terminology. For a positive integer N, we write $[N] = \{1, 2, \ldots, N\}$. For a pair of reals (x, y), we call x the vertical coordinate and y the horizontal coordinate.

A *rectangle* is a pair of positive integers $R = (w, h) \in \mathbb{Z}_+^2$, and a *placed rectangle* is a set of the form $Q = [x, x+w] \times [y, y+h] \subseteq \mathbb{R}^2$, where $R = (w, h)$ is a rectangle and $(x, y) \in \mathbb{Z}^2$ is the bottom-left corner of Q; we will also say that such Q is a *placement* of R. In the notation, we will sometimes treat placed rectangles as their non-placed counterparts; the meaning of this will be always clear from the context.

Both for placed and non-placed rectangles, w and h are called the *width* (length on the horizontal dimension) and the *height* (length on the vertical dimension), respectively, and may be denoted by $w(P)$ and $h(P)$, where P is the (placed) rectangle in question. The *interior* of a placed rectangle $Q = [x, x+w] \times [y, y+h]$ is the set $I(Q) = (x, x+w) \times (y, y+h)$. Two placed rectangles *overlap* if their interiors intersect.

A *zone* is simply a subset of $\mathbb{R}^2$. For a zone Z and a set of placed rectangles $\mathcal{R}$, by $\mathcal{R}[Z] = \{R \in \mathcal{R} \mid R \subseteq Z\}$ we denote the set of all rectangles from $\mathcal{R}$ that are entirely contained in Z. For a zone Z and a set of non-placed rectangles $\mathcal{R}$, a *packing* of $\mathcal{R}$ in Z is a set $\mathcal{R}' = \{R' : R \in \mathcal{R}\}$ consisting of pairwise non-overlapping placed rectangles contained in Z, where R' is a placement of R for each $R \in \mathcal{R}$.

The problem and the main result. In the (parameterized variant of) 2D KNAPSACK problem, we are given a rectangular zone $B = [0, N_1] \times [0, N_2] \subseteq \mathbb{R}^2$ called the *box*, where N_1, N_2 are positive integers, a set $\mathcal{R}$ of rectangles called *items*, and an integer k. The question is whether there exists a packing of some k items from $\mathcal{R}$ in the box B.

In the context of an instance $(B, \mathcal{R}, k)$ of 2D KNAPSACK, the *size* of the box B is $\|B\| = N_1 + N_2$, and the *aspect ratio* of B is $\delta(B) = \max\left(\frac{N_1}{N_2}, \frac{N_2}{N_1}\right) \geqslant 1$. Further, an item $R \in \mathcal{R}$ is *wide* if $w(R) \geqslant h(R)$. Note that in the variant of the problem where rotations by $90°$ are allowed, one may always rotate the items so that they are wide. When the instance $(B, \mathcal{R}, k)$ is clear from the context, by a *packing* we mean a packing of a subset of $\mathcal{R}$ in B.

With these definitions in place, we can state our main result.

▶ **Theorem 1.** *There exists an algorithm that given an accuracy parameter $\varepsilon > 0$ and an instance $(B, \mathcal{R}, k)$ of* 2D KNAPSACK*, where $\mathcal{R}$ consists only of wide items, either returns a packing of size at least $(1 - \varepsilon)k$ or correctly concludes that there is no packing of size k. The running time of the algorithm is $\delta(B)^{\mathcal{O}(k)} \cdot (k + 1/\varepsilon)^{\mathcal{O}(k + 1/\varepsilon^2)} \cdot (|\mathcal{R}|\|B\|)^{\mathcal{O}(1/\varepsilon^2)}$.*

Polylines and containers. In our algorithm for 2D KNAPSACK we will decompose the box into zones delimited by borders of low complexity, allowing those borders to be efficiently guessed. Formally, each border will be a polyline defined as follows.

▶ **Definition 2** (Axis-parallel polyline). *An* axis-parallel polyline $\mathcal{P}$ *is a union of horizontal or vertical segments $S_1, S_2, \ldots, S_m$ such that for $1 \leqslant i \leqslant m - 1$, the end of segment S_i is the beginning of segment S_{i+1}. Then m is called the* complexity *of $\mathcal{P}$.*

For brevity, axis-parallel polylines will be just called polylines. We will only work with *monotone* polylines, meaning that all horizontal coordinates of points on S_j will not be smaller than the horizontal coordinates of the points on S_i, whenever $i < j$. A polyline P *crosses* a placed rectangle R if P intersects the interior of R.

Next we introduce *containers*. We will use them to capture the idea of decomposing the box into zones.

▶ **Definition 3** (Container). *A container $\mathcal{C}$ is a union of horizontal or vertical segments $S_1, S_2, \ldots, S_m$ such that:*

- *for $1 \leqslant i \leqslant m-1$, the end of segment S_i is the beginning of segment S_{i+1}, and*
- *the end of segment S_m is the beginning of segment S_1.*

Furthermore, we require that $\mathcal{C}$ is weakly-simple (as introduced in [9, 3, 2]) in the following sense: if S^1 is the unit circle and $\gamma \colon S^1 \to \mathbb{R}^2$ is a parameterization of $\mathcal{C}$, then for every $\varepsilon > 0$ there exists an injective continuous $\gamma_\varepsilon \colon S^1 \to \mathbb{R}^2$ such that $\max_{v \in S^1} \|\gamma(v) - \gamma_\varepsilon(v)\| \leqslant \varepsilon$ for some norm $\|\cdot\|$.

The inside *of the container, denoted $I(\mathcal{C})$, is the bounded open region delimited by the segments. Moreover, the* complexity *of the container is defined as m.*

Note that the inside of a container is not necessarily connected. For clarification, see Figure 1.

3 Exact algorithm

In this section, we give an exact algorithm for the problem, which will be later used as a subroutine in the proof of Theorem 1. The point here is that we allow the box to be delimited by an arbitrary container, and we measure the running time in the complexity of the container. Formally, we will prove the following statement.

▶ **Lemma 4.** *Given a set of rectangles $\mathcal{R}$ and a container $\mathcal{C}$ of complexity m, one can determine whether there is a packing of the rectangles of $\mathcal{R}$ inside $\mathcal{C}$ in time $(m + |\mathcal{R}|)^{\mathcal{O}(|\mathcal{R}|)}$.*

The main idea of our algorithm is to push the packing bottom-left, as explained in the next definition.

▶ **Definition 5.** *A packing $\mathcal{R}$ inside a container $\mathcal{C}$ is said to be* pushed bottom-left *if for every rectangle $R \in \mathcal{R}$, its left (resp. bottom) side intersects either a vertical (resp. horizontal) segment of the container, or a right (resp. top) side of another rectangle $R' \in \mathcal{R}$.*

It is not hard to see that if a packing is pushed bottom-left, then there must be a rectangle in the packing whose left and bottom sides rest on the perimeter of the container. This is formally proved in the following statement. (Proofs of statements marked with (♠) can be found in Appendix B.)

▶ **Proposition 6** (♠). *Suppose $\mathcal{R}$ is a non-empty packing of rectangles inside a container $\mathcal{C}$ that is pushed bottom-left. Then there exists a rectangle $R \in \mathcal{R}$ such that its left side intersects a vertical segment of the container and its bottom side intersects an horizontal segment of the container.*

With Proposition 6 established, we can conclude our goal using a simple branching strategy.

Proof of Lemma 4. We prove a stronger statement where we allow $\mathcal{C}$ to be the union of several disjoint containers, and we let m be the sum of their complexities.

Suppose there is a packing $\mathcal{S}$ of the rectangles of $\mathcal{R}$ into $\mathcal{C}$. We can assume without loss of generality that $\mathcal{S}$ is pushed bottom-left within every container of $\mathcal{C}$. Now by Proposition 6, there exists a rectangle R such that its left (resp. bottom) side intersects a vertical (resp. horizontal) segment of a container in $\mathcal{C}$.

So here is a recursive procedure to solve the problem. First, guess (by trying all possibilities) the rectangle R satisfying the condition above; there are n different possibilities for R, where $n = |\mathcal{R}|$. Second, guess which pair of segments of the containers intersect the left and the bottom side of R; there are at most m^2 possibilities. Place rectangle R according

to the latter guess and verify that it is indeed fully contained in $\mathcal{C}$. Then, "carve out" the rectangle, i.e., define a new union of containers $\mathcal{C}'$ so that the $I(\mathcal{C}') = I(\mathcal{C}) \setminus R$. It is easy to see that the total complexity of the new union of containers $\mathcal{C}'$ is at most $m + 6$ and it can be computed in time polynomial in m. Then, recurse on $\mathcal{R}'$ and $\mathcal{C}'$ where $\mathcal{R}' = \mathcal{R} \setminus \{R\}$.

It is clear that the algorithm is correct. To analyze its running time, note that the recursion tree has depth bounded by n and branching bounded by $n(m + 6n)^2$, hence it consists of $(m + n)^{\mathcal{O}(n)}$ nodes. The internal computation at each node take time polynomial in n and m, so the total running time of $(m + n)^{\mathcal{O}(n)}$ follows. ◄

4 Giving structure to the packing

In this section we prove structural results that can be summarized as follows: at the cost of sacrificing a small fraction of the packing, one can apply resource augmentation – round the packing – so that it gains a certain structure. Once this structure is achieved, we will argue later that structured packings can be efficiently computed using dynamic programming.

Throughout this section we fix an instance $(B, \mathcal{R}, k)$ of 2D KNAPSACK, where $B = [0, N_1] \times [0, N_2]$ and $\mathcal{R}$ consists only of wide rectangles: $w(R) \geqslant h(R)$ for all $R \in \mathcal{R}$.

To perform resource augmentation, we need the following notion of rounding a rectangle. Informally, a rounded rectangle is the original rectangle with its width rounded up to the nearest multiple of $\ell' = \ell^2/N_1$; here is a formal definition.

▶ **Definition 7** (Rounded rectangles). *Let $R = (w, h)$ be a rectangle and $\ell > 0$ be a positive real. Then the ℓ-rounded rectangle $round_\ell(R)$ is the rectangle $(\ell' \lceil w/\ell' \rceil, h)$ where $\ell' = \ell^2/N_1$. For a set $\mathcal{R}$ of rectangles, we define similarly $round_\ell(\mathcal{R}) = \{round_\ell(R) \colon R \in \mathcal{R}\}$.*

As mentioned, the key idea behind our algorithm is to look for a specifically structured packing. This structure is quantified formally in the following definition. Broadly speaking, we look for a packing that is partitioned into regions of low complexity and such that the rectangles in each region behave well.

▶ **Definition 8** (Structured packing). *Fix any $\varepsilon, \ell > 0$. Consider a set of pairwise non-intersecting monotone polylines $P_1, P_2, \ldots, P_m$ contained in the box B, where each P_i starts at the left side of B and finishes at the right side of B, and the polylines $P_1, \ldots, P_m$ are naturally numbered from bottom to top. We define the partition of the box B into regions $B_0, B_1, \ldots, B_m$ so that each region B_i is delimited by the polylines P_i and P_{i+1} and the left and the right side of B (here we define for convenience P_0 to be the bottom side of B and P_{m+1} to be the top of B).*

We say that a packing of rectangles $\mathcal{Q}$ in B is an (ε, ℓ)-structured packing if every rectangle in $\mathcal{Q}$ has width at least 2ℓ and there exist polylines $P_1, P_2, \ldots, P_m$ as above, each of complexity at most $4/\varepsilon + 1$, such that no rectangle of $\mathcal{Q}$ is crossed by any polyline P_i, $i \in [m]$, and for each $i, 0 \leqslant i \leqslant m$ at least one of the following conditions holds:
- *$|\mathcal{Q}[B_i]| \leqslant 2/\varepsilon^2$, or*
- *$round_\ell(\mathcal{Q}[B_i])$ can be packed into B_i.*

The rest of the section is dedicated to proving the following structural lemma (recall that the instance $(B, \mathcal{R}, k)$ is fixed in the context):

▶ **Lemma 9** (structural lemma). *Suppose $\ell > 0$ is a positive real such that there is a packing of size k consisting of rectangles of width at least 2ℓ each. Then for every $\varepsilon > 0$, there exists also an (ε, ℓ)-structured packing of size at least $(1 - 3\varepsilon)k$.*

This section is divided into 3 parts. In the first subsection we study the assumed packing of size k and define an associated *conflict graph*, which turns out to be planar. In the second subsection, we show that if there exists a packing of rectangles in a specific zone on the box, then at the cost of removing a few rectangles, there exists a packing of the rounded rectangles into a slightly bigger rounded version of the zone. In the last section, we define the specific polylines that will divide the zones and finish the proof of the Lemma 9.

By assumption, there exists a packing $\mathcal{S}$ in B consisting of k rectangles from $\mathcal{R}$, each of width at least 2ℓ. Fix $\mathcal{S}$ for the remainder of this section.

4.1 Conflict graph

For the definition of the conflict graph, we need the following notion of horizontal visibility.

▶ **Definition 10.** *Two different placed rectangles $R, R' \in \mathcal{S}$ see each other if there is an horizontal segment s intersecting the interior of the right side of R and the interior of the left side of R' (or vice versa) such that s does not intersect any other rectangle of $\mathcal{S}$. Notice that s may consist of a single point, if R and R' are intersecting. For convenience, we extend this definition to the case where R is the left side of B or R' is the right side of B. For instance with the left side of B we associate the placed rectangle $R_{left} = [-1, 0] \times [0, N_2]$ and say that R and the left side of B see each other if R_{left} and R see each other; similarly for the right side of B. The left side of B and the right side of B do not see each other.*

Note two rectangles intersecting only at their common corner do *not* see each other.

▶ **Definition 11** (Conflict graph). *For a packing $\mathcal{S}$, we define the conflict graph of $\mathcal{S}$ to be the graph G defined as follows: the vertex set contains all the rectangles of $\mathcal{S}$, and in addition there are two special vertices s and t identified with the left side and the right side of B, respectively. Two vertices of G are adjacent if and only if they see each other.*

It is easy to see that the conflict graph is planar; see Figure 2. We formalize this intuition in the following lemma.

▶ **Lemma 12** (♠). *For any packing $\mathcal{Q}$, the conflict graph of $\mathcal{Q}$ is planar.*

4.2 Packing rounded rectangles

Next, we analyze a packing within some zone $Z \subseteq \mathbb{R}^2$, with the goal of understanding when and how the rectangles of this packing can be rounded to obtain a rounded packing of substantial size. We fix some positive real $\ell > 0$ for the rest of this subsection.

First, we need some definitions about expanding zones.

▶ **Definition 13.** *Let $Z \subseteq \mathbb{R}^2$. We define:*
- *the negatively shifted zone $\overleftarrow{Z}\langle \ell \rangle = \left(\bigcup_{(x,y) \in Z} [x - \ell, x] \times \{y\} \right) \cap [0, N_1 - \ell] \times [0, N_2]$,*
- *the positively shifted zone $\overrightarrow{Z}\langle \ell \rangle = \left(\bigcup_{(x,y) \in Z} [x, x + \ell] \times \{y\} \right) \cap [0, N_1] \times [0, N_2]$,*
- *and the rounded zone $\overleftrightarrow{Z}\langle \ell \rangle = \left(\bigcup_{(x,y) \in Z} [x - \ell, x + \ell] \times \{y\} \right) \cap [0, N_1] \times [0, N_2]$.*

Note that if $Z' = \overleftarrow{Z}\langle \ell \rangle$ then $\overleftrightarrow{Z}\langle \ell \rangle = \overrightarrow{Z'}\langle \ell \rangle$, and that the first two definitions are not symmetric.

Our main goal in this subsection is to prove the following lemma. It intuitively says that at the cost of removing an *st*-separator in the conflict graph, one can find a packing of the rounded rectangles into a slightly extended zone. Here, an *st-separator* is a set of vertices (rectangles) that hits every *s-t* path.

▶ **Lemma 14.** *Let $\mathcal{Q}$ be a packing in a zone $Z \subseteq B$ such that every rectangle of $\mathcal{Q}$ has width at least 2ℓ. Further, let C be an st-separator in the conflict graph of $\mathcal{Q}$. Then $round_\ell(\mathcal{Q} \setminus C)$ can be packed inside the zone $\overleftrightarrow{Z}\langle\ell\rangle$.*

The first step towards the proof of Lemma 14 is to repack $\mathcal{Q}$ into the negatively shifted zone Z at the cost of deleting a few rectangles.

▶ **Proposition 15 (♠).** *Let $\mathcal{Q}$ be a packing in a zone $Z \subseteq B$ such that every rectangle of $\mathcal{Q}$ has width at least 2ℓ. Further, let C be an st-separator in the conflict graph of $\mathcal{Q}$. Then $\mathcal{Q} \setminus C$ can be packed inside the zone $\overleftarrow{Z}\langle\ell\rangle$.*

For an illustration of the proof, see Figure 3.

Now that we have emptied a strip to the right of the zone, we can do some resource augmentation in order to replace the original rectangles by their rounded versions, while still being able to pack them inside the rounded zone.

▶ **Proposition 16 (♠).** *Let $\mathcal{Q}$ be a packing in a zone $Z \subseteq [0, N_1 - \ell] \times [0, N_2]$ such that every rectangle of $\mathcal{Q}$ has width at least 2ℓ. Then $round_\ell(\mathcal{Q})$ can be packed inside the zone $\overrightarrow{Z}\langle\ell\rangle$.*

For clarification, see Figure 4. We may now combine Proposition 15 and Proposition 16 to achieve our goal.

Proof of Lemma 14. Apply Proposition 15 and Proposition 16 to get that $round_\ell(\mathcal{Q} \setminus C)$ can be packed in $\overrightarrow{Z'}\langle\ell\rangle$, where $Z' = \overleftarrow{Z}\langle\ell\rangle$. As $\overrightarrow{Z'}\langle\ell\rangle = \overleftrightarrow{Z}\langle\ell\rangle$, the proof is finished. ◀

4.3 Proof of the Structural Lemma

Finally, in this subsection we define the polylines that we are interested in and prove some results about zones and polylines to finish the proof of Lemma 9. The main idea is to construct some well-chosen polylines by looking at the rectangles on short s-t paths. These polylines are then used to delimit zones in which we can find a separator of bounded size, and apply the ideas of the previous subsections.

Recall that we are working with a packing $\mathcal{S}$ of size k consisting of rectangles of width at least 2ℓ each. Let G be the conflict graph of $\mathcal{S}$. For every $R \in \mathcal{S}$, by v_R we denote the vertex of G corresponding to R. First, we need to understand how s-t paths in G can be mapped to polylines.

▶ **Definition 17** (Bottom polyline of a path). *Consider an s-t path $P = (s, v_{R_1}, v_{R_2}, \ldots, v_{R_m}, t)$ in G, and suppose that for each $i \in \{0, 1, \ldots, m\}$, that R_i and R_{i+1} see each other is witnessed by the segment $s_i = [x(R_i) + w(R_i), x(R_{i+1})] \times \{y_i\}$ (where $R_0 = s$ and $R_{m+1} = t$). Then define the* bottom polyline *of P as the polyline $\mathcal{P}$ formed by the union of the following segments:*

- $[x(R_i), x(R_i) + w(R_i)] \times \{y(R_i)\}$ *for each $i \in [m]$,*
- $\{x(R_i)\} \times [\min\{y(R_i), y_{i-1}\}, \max\{y(R_i), y_{i-1}\}]$ *for each $i \in [m]$,*
- $\{x(R_i) + w(R_i)\} \times [\min\{y(R_i), y_i\}, \max\{y(R_i), y_i\}]$ *for each $i \in [m]$, and*
- s_i *for each $i \in \{0, 1, \ldots, m\}$.*

Less formally, $\mathcal{P}$ is the union of the segments s_i joining the rectangles of the path, the bottom sides of the rectangles, and parts of the left/right sides of the rectangles to join the segments to the bottom sides.

Similarly, we define the notion of the top polyline of an s-t path in G. When defining at the same time the top and the bottom polyline of the same path, we always use the same segments s_i to define how rectangles R_i and R_{i+1} should be linked. Finally, we will also need the middle polyline.

▶ **Definition 18** (Middle polyline of a path). *Consider an s-t path $P = (s, v_{R_1}, v_{R_2}, \ldots, v_{R_m}, t)$ in G, and suppose that for each $i \in \{0, 1, \ldots, m\}$, that R_i and R_{i+1} see each other is witnessed by the segment $s_i = [x(R_i) + w(R_i), x(R_{i+1})] \times \{y_i\}$ (where $R_0 = s$ and $R_{m+1} = t$). Then define the* middle polyline *of P as the polyline $\mathcal{P}$ formed by the union of the following segments:*

- *$\{x(R_i) + w(R_i)/2\} \times [\min\{y(R_i) + h(R_i)/2, y_i\}, \max\{y(R_i) + h(R_i)/2, y_i\}]$ for each $i \in [m]$,*
- *$\{x(R_i) + w(R_i)/2\} \times [\min\{y(R_i) + h(R_i)/2, y_{i+1}\}, \max\{y(R_i) + h(R_i)/2, y_{i+1}\}]$ for each $i \in [m]$, and*
- *$[\max(x(R_i) + w(R_i)/2, 0), \min(x(R_{i+1}) + w(R_{i+1})/2, N_2)] \times \{y_i\}$ for each $i \in \{0, 1, \ldots, m\}$.*

Less formally, $\mathcal{P}$ is the union of:

- a vertical segment from the center of each R_i to the vertical position of s_i,
- a vertical segment from the center of each R_i to the vertical position of s_{i+1},
- all segments s_i extended so that they reach the horizontal coordinates of the centers of the corresponding rectangles.

For a visual representation, see Figure 5. The following is clear.

▶ **Proposition 19.** *The top, bottom and middle polylines of a path P have complexity at most $4|P| + 1$, where $|P|$ denotes the number of vertices on P.*

Moreover, the middle polyline is defined so that we have space to the left and the right when performing resource augmentation. This will be made clear in the following definitions and lemmas; see Figure 6.

▶ **Proposition 20** (♠). *Suppose P is an s-t path in the conflict graph G of the packing $\mathcal{S}$. Let $\mathcal{P}$ be the middle polyline of P and let $\mathcal{Q}$ be the packing obtained from $\mathcal{S}$ by removing all the rectangles participating in P. Then $\mathcal{P}$ does not cross $\overleftarrow{\mathcal{Q}}\langle\ell\rangle$. The same goes for $\overrightarrow{\mathcal{Q}}\langle\ell\rangle$, and therefore also for $\overleftrightarrow{\mathcal{Q}}\langle\ell\rangle$.*

Next, we need the following graph-theoretic observation.

▶ **Proposition 21.** *Let G be a graph containing vertices s and t. Suppose every s-t path in G contains at least $1/\varepsilon$ internal vertices. Then G contains an st-separator of size at most $\varepsilon(|V(G)| - 2)$.*

Proof. As every s-t path in G contains at least $1/\varepsilon$ internal vertices, one cannot find more than $\varepsilon(|V(G)| - 2)$ internally disjoint s-t paths in G. By Menger's theorem, there is an st-separator of size at most $\varepsilon(|V(G)| - 2)$. ◀

We can now wrap up the section by proving the Structural Lemma.

Proof of Lemma 9. Based on the assumed packing $\mathcal{S}$, we construct another packing $\mathcal{S}'$ and then we prove that it is structured and has size at least $(1 - 3\varepsilon)k$. Let G be the conflict graph of $\mathcal{S}$. Let $\mathcal{F}$ be an inclusion-wise maximal family $\mathcal{F}$ of internally disjoint s-t paths in G, each with at most $1/\varepsilon$ internal vertices. As the paths from $\mathcal{F}$ are internally disjoint, we can naturally enumerate them from bottom to top: $\mathcal{F} = \{P_1, P_2, \ldots, P_m\}$. For convenience, let $P_0 = P_{m+1} = \emptyset$.

Because the conflict graph is planar by Lemma 12, by the Jordan Curve theorem, for each $i \in \{0, 1, \ldots, m\}$ there is a set V_i of vertices of G that lies inside the cycle $P_i \cup P_{i+1}$. By construction of the conflict graph, V_i is exactly the set of rectangles lying in the area Z_i delimited by the box, the top polyline of P_i and the bottom polyline of P_{i+1}. For each V_i we construct a separating polyline $\mathcal{P}_i$ as follows:

- If $|V_{i-1}| \leqslant 1/\varepsilon^2$ and $|V_i| \leqslant 1/\varepsilon^2$, select the bottom polyline of P_i as the separating polyline.
- Otherwise, select the middle polyline of P_i as the separating polyline.

All the polylines created are of complexity at most $4/\varepsilon + 1$ by Proposition 19. They partition the box into regions $B_0, B_1, \ldots, B_{m+1} \subseteq B$, from the bottom to the top. Note that $B_i \supseteq Z_i$ for each relevant i.

Notice that in $G[V_i \cup \{s, t\}]$ there is no s-t path of length at most $1/\varepsilon$, because $\mathcal{F}$ is maximal. Let C_i be the separator given by Proposition 21 for the graph $G[V_i \cup \{s, t\}]$. Then we have $|C_i| \leqslant \varepsilon |V_i|$.

We can now specify which rectangles we want to include in $\mathcal{S}'$. We define $\mathcal{S}'$ to be the union of sets V_i' for $i \in \{0, 1, \ldots, m\}$, where

$$
V_i' = \begin{cases} V_i \cup V(P_i) \setminus \{s, t\} & \text{if } |V_{i-1}| \leqslant 1/\varepsilon^2 \text{ and } |V_i| \leqslant 1/\varepsilon^2, \\ V_i & \text{if } |V_{i-1}| > 1/\varepsilon^2 \text{ and } |V_i| \leqslant 1/\varepsilon^2, \\ V_i \setminus C_i & \text{otherwise.} \end{cases}
$$

We now argue that for each $i \in \{0, 1, \ldots, m\}$, either $|V_i'| \leqslant 2/\varepsilon^2$ and $V_i' \subseteq \mathcal{S}[B_i]$, or $\mathrm{round}_\ell(V_i')$ can be packed in B_i.

First, observe that if $|V_i| \leqslant 1/\varepsilon^2$, then $|V_i'| \leqslant |V_i \cup V(P_i) \setminus \{s, t\}| \leqslant 1/\varepsilon^2 + 1/\varepsilon \leqslant 2/\varepsilon^2$. Further, if $|V_{i-1}| > 1/\varepsilon^2$ then $V_i' = V_i$ and trivially $\mathcal{S}[B_i] \supseteq \mathcal{S}[Z_i] = V_i$, and if $|V_{i-1}| \leqslant 1/\varepsilon^2$ then $\mathcal{P}_i$ is the bottom polyline of P_i and we have $\mathcal{S}[B_i] \supseteq V_i \cup V(P_i) \setminus \{s, t\} = V_i'$ as well.

Second, consider the case when $|V_i| > 1/\varepsilon^2$. Notice that then $\mathcal{P}_i$ is the middle polyline of P_i and $\mathcal{P}_{i+1}$ is the middle polyline of P_{i+1}, and $V_i' = V_i \setminus C_i$. Because V_i can be packed inside Z_i, we can use Lemma 14 on V_i and Z_i to pack $\mathrm{round}_\ell(V_i')$ into $\overleftrightarrow{Z_i}\langle\ell\rangle$. By Proposition 20, we know that $\overleftrightarrow{Z_i}\langle\ell\rangle \subseteq B_i$, hence we can pack $\mathrm{round}_\ell(V_i')$ into B_i.

We conclude that indeed, $\mathcal{S}'$ is an (ε, ℓ)-structured packing, as witnessed by the polylines $\mathcal{P}_i$ for $i \in [m]$. What is left to show is that $|\mathcal{S}'| \geqslant (1 - 3\varepsilon)k$. Call an index $i \in [m]$ *heavy* if $|V_i| > 1/\varepsilon^2$. Observe that $\mathcal{S}' \supseteq \mathcal{S} \setminus \bigcup_{i:\,\mathrm{heavy}}(C_i \cup V(P_i) \cup V(P_{i+1}))$, hence it suffices to prove that $\left|\bigcup_{i:\,\mathrm{heavy}}(C_i \cup V(P_i) \cup V(P_{i+1})) \setminus \{s, t\}\right| \leqslant 3\varepsilon k$. Fix a heavy index i. First, observe that $|(V(P_i) \cup V(P_{i+1})) \setminus \{s, t\}| \leqslant 2/\varepsilon \leqslant 2\varepsilon |V_i|$, as each path P_i has at most $1/\varepsilon$ internal vertices. Second, by construction we have $|C_i| \leqslant \varepsilon |V_i|$. Summing those inequalities throughout all heavy i yields that

$$
\left|\bigcup_{i:\,\mathrm{heavy}}(C_i \cup V(P_i) \cup V(P_{i+1})) \setminus \{s, t\}\right| \leqslant \sum_{i:\,\mathrm{heavy}} 3\varepsilon |V_i| \leqslant 3\varepsilon k,
$$

as required. ◀

5 The algorithm

In this section we finalize the proof of Theorem 1. The section is divided into two parts. The first subsection describes an algorithm working under the assumption that the input set $\mathcal{R}$ only contains rectangles of width at least 2ℓ, for some $\ell > 0$. In the second subsection, we show how to obtain the assumption that $\mathcal{R}$ only contains rectangles of width at least 2ℓ for $\ell = N_1/(\delta(B)k^2)$, at the expense of deleting an ε fraction of the rectangles in the packing. Therefore, we get a full algorithm as a corollary.

Throughout this section, fix an instance $(B, \mathcal{R}, k)$ of 2D KNAPSACK, where $B = [0, N_1] \times [0, N_2]$ and $\mathcal{R}$ consists of wide items.

5.1   The algorithm for rectangles of substantial width

The dynamic programming algorithm will gradually guess a good partition of the box into regions (that we know exists by Lemma 9), and then solve the problem in each region independently. In order to avoid repeating the use of the same rectangles in different regions, we use color-coding.

▶ **Definition 22** (Good coloring). *Given a set of rectangles $\mathcal{R}$ and a subset $\mathcal{S} \subseteq \mathcal{R}$ of size k, a function col: $\mathcal{R} \to [k]$ is a* good coloring *for $\mathcal{S}$ if rectangles of $\mathcal{S}$ have pairwise different colors under col.*

We cannot directly guess a good coloring of the rectangles, as a priori there are too many candidates. We instead use the following classic result of Naor et al. [11], which says that there is only an fpt-sized family of candidates for a good coloring.

▶ **Proposition 23** (Naor et al. [11]). *For every set $\mathcal{R}$ and positive integer k, there exists a family $\mathcal{F}$ of colorings of $\mathcal{R}$ with color set $[k]$ such that $|\mathcal{F}| \leqslant e^k k^{\mathcal{O}(\log k)} \log |\mathcal{R}|$ and for every $\mathcal{S} \subseteq \mathcal{R}$ of size k, in $\mathcal{F}$ there is a good coloring for $\mathcal{S}$. Moreover, $\mathcal{F}$ can be computed in time $e^k k^{\mathcal{O}(\log k)} |\mathcal{R}| \log |\mathcal{R}|$.*

Next, we observe that once the number of different widths in the instance has been bounded, one can restrict attention to a small set of candidate rectangles. For this, notice the following: if we have a colored packing (a packing of colored rectangles) of size k that contains a rectangle R, and in the packing we did not use another rectangle R' of the same color and width as R, but satisfying $h(R') \leqslant h(R)$, then we can replace R with R' and we will still have a colored packing. This observation leads to defining the following operation.

▶ **Definition 24** (reduce$_k(\mathcal{R}, \mathsf{col})$). *Suppose col: $\mathcal{R} \to [k]$ is a coloring of a set of rectangles $\mathcal{R}$ with color set $[k]$. Then for a positive integer w and color $i \in [k]$, let $\mathcal{R}_{w,i}$ be the set of k smallest-height rectangles among the rectangles of $\{R \in \mathcal{R} \mid w(R) = w, col(R) = i\}$. In case $|\{R \in \mathcal{R} \mid w(R) = w, col(R) = i\}| < k$, we set $\mathcal{R}_{w,i} = \{R \in \mathcal{R} \mid w(R) = w, col(R) = i\}$. We define reduce$_k(\mathcal{R}) = \bigcup_{w \in w(\mathcal{R}), i \in [k]} \mathcal{R}_{w,i}$.*

Notice that reduce$_k(\mathcal{R})$ contains at most $k^2 |w(\mathcal{R})|$ elements: for every possible width and every possible color, the at most k rectangles of this specific width and of smallest height. Also, we have the following very simple observation.

▶ **Lemma 25** (♠). *Suppose $\mathcal{R}$ is a set of rectangles and col: $\mathcal{R} \to [k]$ is a coloring function such that $k' \leqslant k$ rectangles from $\mathcal{R}$ with pairwise different colors can be packed in a zone $Z \subseteq \mathbb{R}^2$. Then one can also pack in Z a set of k' rectangles from reduce$_k(\mathcal{R}, col)$ with pairwise different colors.*

Next, we use the following definitions to guess the polylines in a bottom to top order. For two monotone polylines $\mathcal{P}, \mathcal{P}'$ that start at the left side of B and finish at the right side of B, we say that $\mathcal{P}'$ is *below* $\mathcal{P}$ (denoted by $\mathcal{P}' \leqslant \mathcal{P}$), if for every x, y, y', $(x, y) \in \mathcal{P}$ and $(x, y') \in \mathcal{P}'$ implies $y' \leqslant y$. We write $\mathcal{P}' < \mathcal{P}$ if $\mathcal{P}' \leqslant \mathcal{P}$ and $\mathcal{P}' \neq \mathcal{P}$. Given two polylines $\mathcal{P}' < \mathcal{P}$, we want to be able to solve the problem in the following sub-region:

▶ **Definition 26** (container$(\mathcal{P}', \mathcal{P})$). *For polylines $\mathcal{P}' < \mathcal{P}$, container$(\mathcal{P}', \mathcal{P})$ is the container (c.f. Definition 3) delimited by the box B, $\mathcal{P}'$ at the bottom and $\mathcal{P}$ at the top.*

Notice that if $\mathcal{P}$ has complexity m and $\mathcal{P}'$ has complexity m' then container$(\mathcal{P}', \mathcal{P})$ has complexity $m + m' + 2$.

Now we give the algorithm in the case when all rectangles in $\mathcal{R}$ have substantial width. This algorithm is encapsulated in the following lemma.

▶ **Lemma 27.** *There is an algorithm that given $\varepsilon > 0$ and an instance $(B, \mathcal{R}, k)$ of* 2D
KNAPSACK *in which all items are wide and have width at least N_1/α, either returns a packing
of size at least $(1 - \varepsilon)k$ or correctly concludes that there is no packing of size k. The running
time is $(k + 1/\varepsilon)^{\mathcal{O}(k+1/\varepsilon)} \cdot \alpha^{\mathcal{O}(k)} \cdot (|\mathcal{R}|\|B\|)^{\mathcal{O}(1/\varepsilon^2)}$.*

Proof. Let $\ell = N_1/(2\alpha)$; thus every rectangle on input has width at least 2ℓ. For clarity of
presentation we allow the algorithm to output a packing of size at least $(1 - 3\varepsilon)k$; then the
result as stated in the lemma can be obtained by rescaling ε by factor 3.

We first explain the algorithm. Compute $\mathcal{F}$ as given by Proposition 23. Then, guess
(by trying all choices) a coloring col $\in \mathcal{F}$. The idea is now to use dynamic programming to
compute a maximum-size structured packing for the colored instance. More precisely, for
every monotone polyline $\mathcal{P}$ of complexity at most $4/\varepsilon + 1$ connecting the left and the right
side of B, and for every $C \subseteq [k]$, we shall compute the value $\mathrm{dp}[\mathcal{P}, C]$ defined as follows:
$\mathrm{dp}[\mathcal{P}, C]$ is a maximum-size packing that contains only rectangles with colors in C, is colored
injectively by col, and is placed entirely below $\mathcal{P}$ with the added constraint that it is a subset
of some (ε, ℓ)-structured packing.

To compute the value $\mathrm{dp}[\mathcal{P}, C]$ for given $\mathcal{P}$ and C, we iterate over all polylines $\mathcal{P}'$ of
complexity at most $4/\varepsilon + 1$ that are below $\mathcal{P}$. Let $B' = \mathrm{container}(\mathcal{P}', \mathcal{P})$ be the container
between $\mathcal{P}$ and $\mathcal{P}'$. Iterate over all $C' \subseteq C$; this is the set of colors guessed to be used in B'.
Let $\mathcal{R}' = \mathrm{reduce}_k(\mathrm{round}_\ell(\mathcal{R}), \mathrm{col})$ be the reduced set of rounded rectangles, where colors are
naturally inherited from $\mathcal{R}$ during rounding. Compute the following packings:

- $\mathcal{S}_1$ is the largest packing in B' consisting of at most $2/\varepsilon^2$ rectangles with pairwise different
  colors from C'. This packing can be computed in time $|\mathcal{R}|^{\mathcal{O}(1/\varepsilon^2)} \cdot (1/\varepsilon)^{\mathcal{O}(1/\varepsilon^2)}$ by first
  guessing the set of rectangles participating in it, and then checking whether the packing
  can be realized using the algorithm of Lemma 4.
- $\mathcal{S}_2$ is the largest packing in B' consisting of at most k rectangles from $\mathcal{R}'$ with pairwise
  different colors from C'. Again, this packing can be computed in time $|\mathcal{R}'|^{\mathcal{O}(k)} \cdot (k+1/\varepsilon)^{\mathcal{O}(k)}$
  by first guessing the set of rectangles participating in it, and then checking whether the
  packing can be realized using the algorithm of Lemma 4.

Iterate over $\mathcal{S} \in \{\mathcal{S}_1, \mathcal{S}_2\}$, and keep as $\mathrm{dp}[\mathcal{P}, C]$ the set $\mathrm{dp}[\mathcal{P}', C \setminus C'] \cup \mathcal{S}$ of maximum size
over all the sets iterated on. Finally, as the solution to the overall problem, return $\mathrm{dp}[\mathcal{P}, [k]]$
where $\mathcal{P}$ is the top side of B, provided this packing has size at least $(1 - 3\varepsilon)k$. Otherwise,
return that there is no packing of size k.

This concludes the description of the algorithm. We are left with (i) analyzing its running
time and (ii) arguing that in case there is a packing of size at least k, the algorithm will
output a packing of size at least $(1 - 3\varepsilon)k$.

Let us start with assertion (ii). For this, suppose there exists a packing $\mathcal{S}$ of size k.
Since all rectangles of $\mathcal{S}$ have width at least 2ℓ, by Lemma 9 there exists an (ε, ℓ)-structured
packing $\mathcal{S}'$ of size at least $(1 - 3\varepsilon)k$. Further, by the properties of $\mathcal{F}$, there exists col $\in \mathcal{F}$
such that col is injective on $\mathcal{S}'$. Now, let $\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_m$ be the polylines witnessing the
structuredness of $\mathcal{S}'$, and let $\emptyset = C_0 \subseteq C_1 \subseteq C_2 \subseteq \ldots \subseteq C_m \subseteq C_{m+1} = [k]$ be such that C_i
is the sets of colors used by the rectangles of $\mathcal{S}'$ lying below $\mathcal{P}_i$, where $\mathcal{P}_0$ and $\mathcal{P}_{m+1}$ are the
bottom and the top side of B, respectively. A straightforward inductive argument using the
structuredness of $\mathcal{S}'$ and Lemma 25 shows now that for $i = 0, 1, \ldots, m+1$, the cell $\mathrm{dp}[\mathcal{P}_i, C_i]$
will contain a packing of size at least as large as the number of rectangles of $\mathcal{S}'$ lying below $\mathcal{P}_i$.
Hence, the algorithm will return a packing of size at least $|\mathcal{S}'| \geqslant (1 - 3\varepsilon)k$, as promised.

We are left with analyzing the running time. The number of different colorings col $\in \mathcal{F}$ is
$|\mathcal{F}| \leqslant 2^{\mathcal{O}(k)} \cdot \log |\mathcal{R}|$. Further, observe that the number of different polylines considered by the
algorithm is bounded by $\|B\|^{\mathcal{O}(1/\varepsilon)}$ and there are 2^k different subsets of colors. Hence, the

total number of cells $\mathrm{dp}[\mathcal{P}, C]$ considered by the algorithm is bounded by $2^k \cdot \|B\|^{\mathcal{O}(1/\varepsilon)}$. As argued, the time spent on computing a single value of $\mathrm{dp}[\mathcal{P}, C]$ is bounded by $2^k \cdot \|B\|^{\mathcal{O}(1/\varepsilon)}$ (the number of choices for $\mathcal{P}'$ and C') times

$$|\mathcal{R}|^{\mathcal{O}(1/\varepsilon^2)} \cdot (1/\varepsilon)^{\mathcal{O}(1/\varepsilon^2)} + |\mathcal{R}'|^{\mathcal{O}(k)} \cdot (k + 1/\varepsilon)^{\mathcal{O}(k)}.$$

Observe now that the rectangles of $\mathrm{round}_\ell(\mathcal{R})$ have at most $\mathcal{O}(N_1/\ell')$ different widths, where $\ell' = \ell^2/N_1$. Since $\ell = N_1/2\alpha$, we conclude that the total number of different widths of the rectangles of $\mathrm{round}_\ell(\mathcal{R})$ is bounded by

$$\mathcal{O}(N_1/\ell') = \mathcal{O}(N_1^2/\ell^2) \leqslant \mathcal{O}(\alpha^2).$$

Therefore,

$$|\mathcal{R}'| = |\mathrm{reduce}_k(\mathrm{round}_\ell(\mathcal{R}, \mathrm{col}))| \leqslant \mathcal{O}(\alpha^2 k^2).$$

Putting everything together, we infer that the running time of the algorithm is bounded by

$$2^{\mathcal{O}(k)} \cdot \log |\mathcal{R}| \cdot 2^{\mathcal{O}(k)} \cdot \|B\|^{\mathcal{O}(1/\varepsilon)} \cdot \left(|\mathcal{R}|^{\mathcal{O}(1/\varepsilon^2)} \cdot (1/\varepsilon)^{\mathcal{O}(1/\varepsilon^2)} + (\alpha^2 k^2)^{\mathcal{O}(k)} \cdot (k + 1/\varepsilon)^{\mathcal{O}(k)} \right)$$

$$\leqslant (k + 1/\varepsilon)^{\mathcal{O}(k + 1/\varepsilon^2)} \cdot \alpha^{\mathcal{O}(k)} \cdot (|\mathcal{R}| \|B\|)^{\mathcal{O}(1/\varepsilon^2)},$$

as promised. ◀

5.2 Full algorithm

We now present the complete algorithm, which essentially boils down to making a reduction to the case when all rectangles on input have width at least 2ℓ, where $\ell = N_1/(\delta(B)k^2)$. In the next lemma, we explain how to perform this reduction at the cost of removing εk rectangles from the packing.

▶ **Lemma 28.** *Let $\varepsilon > 0$. Suppose there is an algorithm $\mathcal{A}$ that, given a* 2D KNAPSACK *instance $(B = [0, N_1] \times [0, N_2], \mathcal{R}, p)$ in which all items are wide and have width at least $N_1/(\delta q^2)$ and the aspect ratio of B is δ, returns a packing of size at least $(1 - \varepsilon)p$ or attests that there is no packing of size p in time $f(p, q, \varepsilon, \delta, \|B\|, |\mathcal{R}|)$. Then there is an algorithm $\mathcal{B}$ that, given a* 2D KNAPSACK *instance $(B = [0, N_1] \times [0, N_2], \mathcal{R}, k)$ in which all items are wide and the aspect ratio of B is δ, returns a packing of size $(1 - 2\varepsilon)k$ or attests that there is no packing of size k in time $f(k, k, \varepsilon, \delta, \|B\|, |\mathcal{R}|) + (1/\varepsilon + |\mathcal{R}|)^{\mathcal{O}(1/\varepsilon)}$.*

Proof. We present the algorithm $\mathcal{B}$. Without loss of generality, we can assume $k > 1/\varepsilon$, as otherwise the number of rectangles in the sought packing is at most $1/\varepsilon$ and we can solve the problem in time $(1/\varepsilon + |\mathcal{R}|)^{\mathcal{O}(1/\varepsilon)}$ by applying Lemma 4 to every k-tuple of rectangles in $\mathcal{R}$.

Let $\mathcal{W}$ be the set of rectangles of $\mathcal{R}$ that have width at most $N_1/(\delta k^2)$, and let $w = |\mathcal{W}|$. Note that since all rectangles are wide, the rectangles of $\mathcal{W}$ also have height bounded by $N_1/(\delta k^2)$. If $w \geqslant k$, then we can immediately construct a packing of size k by stacking any k rectangles of $\mathcal{W}$ vertically: they fit in the vertical dimension, because $k \cdot N_1/(\delta k^2) \leqslant N_2$. Otherwise, let $k' = k - w$. Run $\mathcal{A}$ on a modified instance where all rectangles of $\mathcal{W}$ are removed, with parameter k'. If there is no packing of size k' for this instance, then clearly there is no packing of size k for the original instance, and this conclusion may be reported by the algorithm. Otherwise, $\mathcal{B}$ returns a packing $\mathcal{S}'$ of size at least $(1 - \varepsilon)k'$ consisting of rectangles from $\mathcal{R} \setminus \mathcal{W}$. If $\mathcal{S}'$ consists only of rectangles of height at most $N_1/(\delta k)$, then we can again immediately obtain a packing of size k by stacking the rectangles of $\mathcal{S}' \cup \mathcal{W}$ vertically; again they fit in the vertical dimension, because $k \cdot N_1/(\delta k) \leqslant N_2$. Otherwise,

we modify $\mathcal{S}'$ by removing any single rectangle R present in $\mathcal{S}'$ whose height (and therefore also width) is at least $N_1/(\delta k)$, and putting all the rectangles of $\mathcal{W}$ into the space freed by the removal of R, by simply stacking them horizontally. They fit horizontally because $w \cdot N_1/(\delta k^2) \leqslant k \cdot N_1/(\delta k^2) = N_1/(\delta k) \leqslant w(R)$, and their heights are not greater than the height of R. The obtained modified packing $\mathcal{S}'$ is returned by the algorithm.

It is clear that the algorithm outputs a packing and that when it concludes that there is no packing of size k, this conclusion is correct. What remains to show is that the packing eventually output by the algorithm has always size at least $(1 - 2\varepsilon)k$. And indeed, the algorithm always is able to pack all rectangles packed in $\mathcal{S}'$, except for possibly one rectangle removed to accommodate $\mathcal{W}$, and all rectangles of $\mathcal{W}$. Hence, the packing output by the algorithm has always size at least

$$(1 - \varepsilon)k' - 1 + w = (1 - \varepsilon)k - (1 - \varepsilon)w - 1 + w \geqslant (1 - \varepsilon)k - 1 > (1 - 2\varepsilon)k,$$

because $\varepsilon k > 1$ due to $k > 1/\varepsilon$. ◀

Now, Theorem 1 follows immediately by combining the algorithm of Lemma 27 with the reduction of Lemma 28. Observe that the running time is $\delta(B)^{\mathcal{O}(k)} \cdot (k + 1/\varepsilon)^{\mathcal{O}(k+1/\varepsilon^2)} \cdot (|\mathcal{R}| \|B\|)^{\mathcal{O}(1/\varepsilon^2)}$, as promised.

6 Conclusion

The correctness of our entire algorithm heavily relies on the assumption that every input rectangle is wide. Indeed, this assumption is used in the greedy arguments in the proof of Lemma 28, which allows us to reduce to the case when every rectangle has a substantial width: at least $N_1/\mathrm{poly}(\delta(B), k)$. This assumption is again heavily used later on: in the proof of Lemma 9 it ensures that upon removing the rectangles corresponding to an st-separator in the conflict graph, there is enough space available for vertical shifting. This eventually leads to rounding the rectangles so that there are only $\mathrm{poly}(\delta(B), k)$ different possible widths, and thus effectively bounding the number of candidate rectangles to $\mathrm{poly}(\delta(B), k)$. So while the original problem – the existence of a parameterized approximation scheme for 2D KNAPSACK – remains open, we hope that the new structural techniques proposed in this work might give insight leading to its resolution.

—— **References** ——

1   Anna Adamaszek and Andreas Wiese. A quasi-PTAS for the two-dimensional geometric knapsack problem. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015*, pages 1491–1505. SIAM, 2015. `doi:10.1137/1.9781611973730.98`.

2   Hsien-Chih Chang, Jeff Erickson, and Chao Xu. Detecting weakly simple polygons. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015*, pages 1655–1670. SIAM, 2015. `doi:10.1137/1.9781611973730.110`.

3   Erik D Demaine and Joseph O'Rourke. *Geometric folding algorithms: linkages, origami, polyhedra.* Cambridge university press, 2007.

4   Manfredo P Do Carmo. *Differential geometry of curves and surfaces: revised and updated second edition.* Courier Dover Publications, 2016.

5   Waldo Gálvez, Fabrizio Grandoni, Salvatore Ingala, Sandy Heydrich, Arindam Khan, and Andreas Wiese. Approximating geometric knapsack via L-packings. *ACM Trans. Algorithms*, 17(4), 2021. `doi:10.1145/3473713`.

6 Waldo Gálvez, Fabrizio Grandoni, Arindam Khan, Diego Ramírez-Romero, and Andreas Wiese. Improved approximation algorithms for 2-Dimensional Knapsack: Packing into multiple L-shapes, spirals, and more. In *37th International Symposium on Computational Geometry, SoCG 2021*, volume 189 of *LIPIcs*, pages 39:1–39:17. Schloss Dagstuhl — Leibniz-Zentrum für Informatik, 2021. `doi:10.4230/LIPIcs.SoCG.2021.39`.

7 Fabrizio Grandoni, Stefan Kratsch, and Andreas Wiese. Parameterized approximation schemes for Independent Set of Rectangles and Geometric Knapsack. In *27th Annual European Symposium on Algorithms, ESA 2019*, volume 144 of *LIPIcs*, pages 53:1–53:16. Schloss Dagstuhl — Leibniz-Zentrum für Informatik, 2019. `doi:10.4230/LIPIcs.ESA.2019.53`.

8 Klaus Jansen and Guochuan Zhang. On rectangle packing: maximizing benefits. In *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2004*, pages 204–213. SIAM, 2004. URL: `http://dl.acm.org/citation.cfm?id=982792.982822`.

9 Yoshiyuki Kusakari, Hitoshi Suzuki, and Takao Nishizeki. A shortest pair of paths on the plane with obstacles and crossing areas. *International Journal of Computational Geometry & Applications*, 9(02):151–170, 1999.

10 Joseph Y.-T. Leung, Tommy W. Tam, C. S. Wong, Gilbert H. Young, and Francis Y. L. Chin. Packing squares into a square. *J. Parallel Distributed Comput.*, 10(3):271–275, 1990. `doi:10.1016/0743-7315(90)90019-L`.

11 Moni Naor, Leonard J Schulman, and Aravind Srinivasan. Splitters and near-optimal derandomization. In *Proceedings of IEEE 36th Annual Foundations of Computer Science*, pages 182–191. IEEE, 1995.
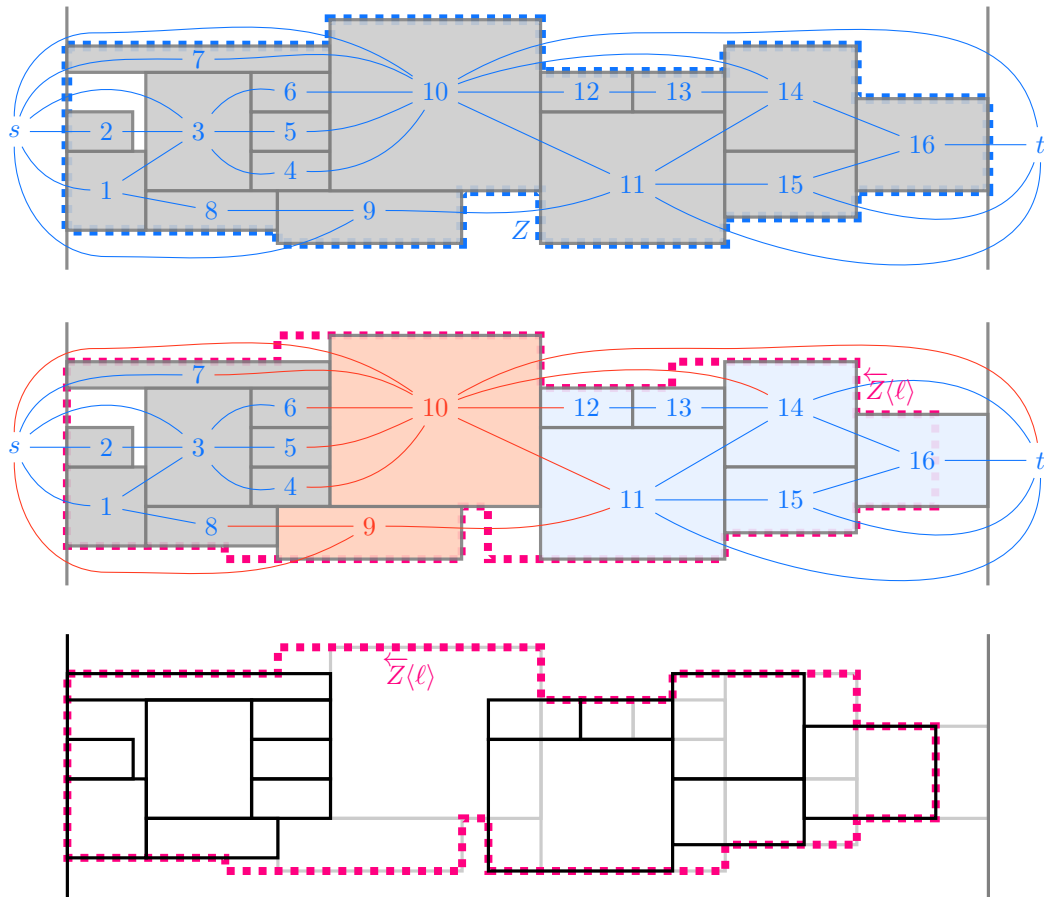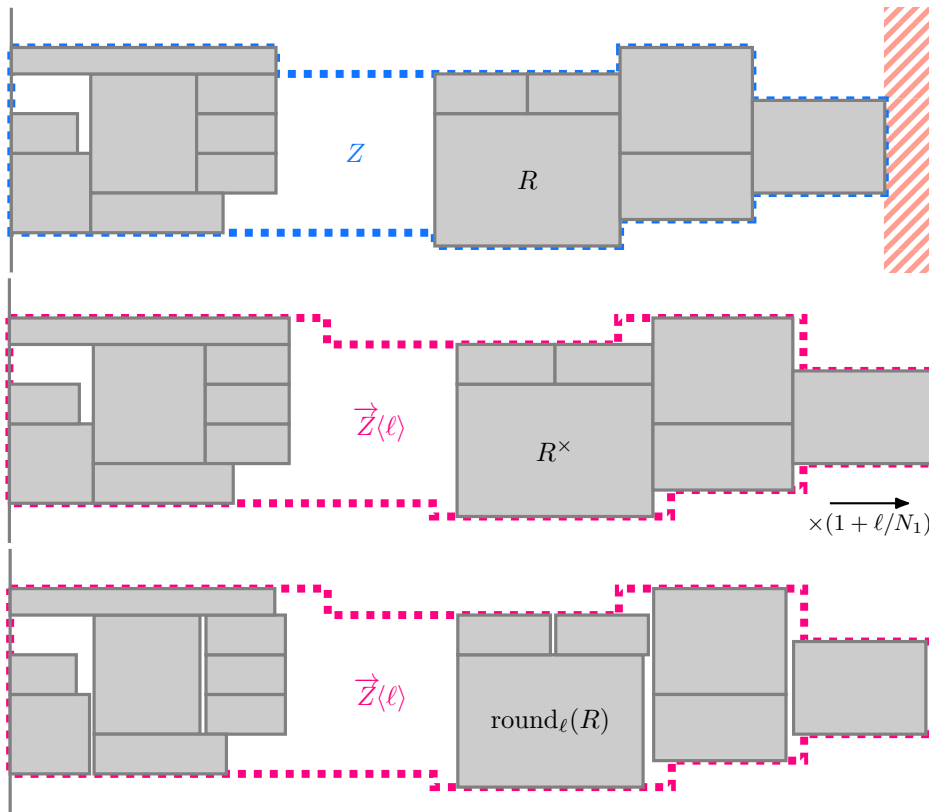
A Figures
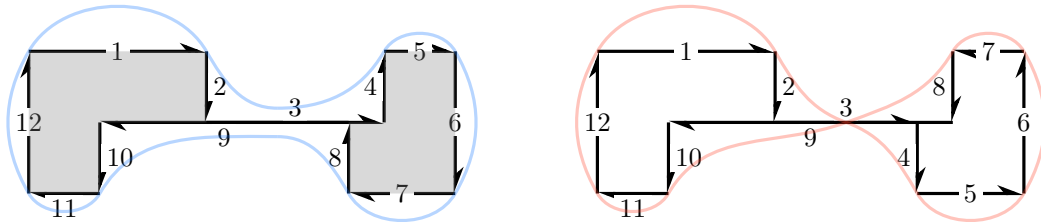
Figure 1 Left panel: an example of a container where the order of the segments is given by the numbers, and some injective candidate γ_ε in light blue. Right panel: an example of a non-container (the paths cross in the middle), and some candidate γ_ε in light red, that is not injective.
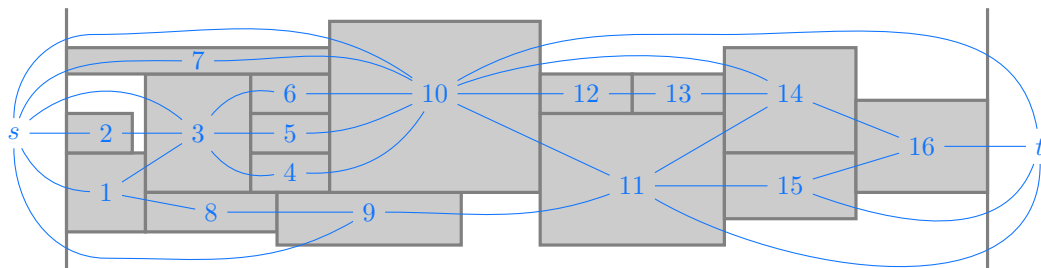

**Figure 2** Example of a conflict graph.

■ **Figure 3** Illustration of the proof of Proposition 15. From top to bottom: First, a blue conflict graph of the gray packing. The packing is entirely in Z, delimited by the blue border. Secondly, we remove the orange separator C and want to pack the leftover rectangles inside the red region $\overleftarrow{Z}\langle \ell \rangle$. The set of rectangles referred to as Y in the proof is in light blue, and X is left gray. Finally, we pack in $\overleftarrow{Z}\langle \ell \rangle$ by shifting the rectangles at the right of the separator by ℓ to the left.

Figure 4 Illustration of the proof of Proposition 16. From top to bottom: First, $\mathcal{Q}$ is packed into Z (blue zone). Then, the rectangles in $\mathcal{Q}$ are scaled horizontally by a factor $\lambda = 1 + \ell/N_1$. We argue in the proof that these scaled-up rectangles are packed in $\overrightarrow{Z}\langle\ell\rangle$ (red zone). Finally, we replace each scaled-up rectangle by its rounded version, which has smaller width.
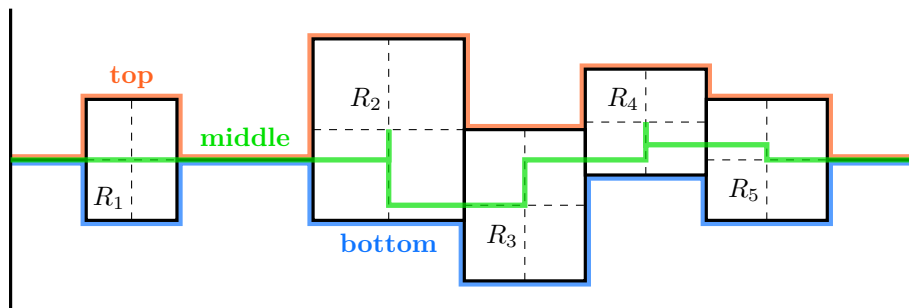
Figure 5 In orange, the top polyline of the st-path formed by R_1, R_2, R_3, R_4 and R_5. In blue, its bottom polyline, and in green, its middle polyline. The dashed lines split their respective rectangles into 4 equal parts.
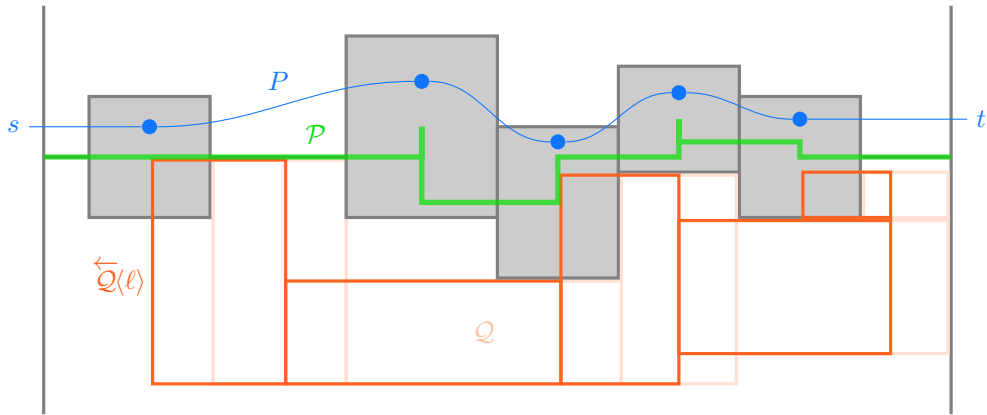
Figure 6 In green, the middle polyline $\mathcal{P}$ of the blue st-path P constituted of the gray rectangles. In light orange, $\mathcal{Q}$, and in orange, $\overleftarrow{\mathcal{Q}}\langle\ell\rangle$. Notice that the green polyline does not cross any orange rectangles.

B Omitted proofs

Proof of Proposition 6. Create a directed graph D with vertex set $\mathcal{R}$, where there is an edge (R, R') if the bottom side of R intersects the top side of R' on more than a single point, or if the left side of R intersects the right side of R' on more than a single point. Let us show that D has no directed cycle, so for contradiction suppose $R_1, \ldots, R_\ell$ is a directed cycle in D. In what follows, all indices behave cyclically modulo ℓ.

For each $i \in [\ell]$, select an arbitrary point p_i in the intersection of R_{i-1} and R_i that is neither a corner of R_{i-1} nor a corner of R_i. Further, observe that one can construct a curve $\gamma_i \colon [0, d_i] \to R_i$, where d_i is the length of γ_i, such that:

- γ_i is smooth (formally, C^1) and monotone in both directions,
- $\|\gamma_i'(t)\|_2 = 1$ for all $t \in [0, d_i]$,
- $\gamma_i(0) = p_i$ and $\gamma_i(d_i) = p_{i+1}$, and
- the tangent of γ_i at p_i and p_{i-1} is perpendicular to the respective side and faces the inside (resp. outside) of R_i. For instance if p_i is on the top side of R_i, we require $\gamma_i'(0) = (0, -1)$, and if p_{i+1} is on the left side of R_i, we require $\gamma_i'(d_i) = (-1, 0)$.

An example of such a construction is shown below.
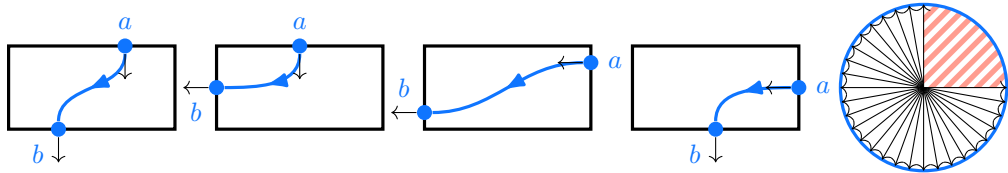
Concatenating all the curves γ_i in order yields a smooth closed curve $\gamma \colon S \to \bigcup_{i=1}^{\ell} R_i$ without self-crossings such that $\|\gamma'(t)\|_2 = 1$ for all $t \in S$, where S is the circle of length $\sum_{i=1}^{\ell} d_i$. Here is the crucial observation: by the way we oriented the arcs in D, the vector $\gamma'(t)$ is never in the positive orthant (i.e. $\gamma'(t)$ has not both coordinates positive), for any $t \in S$.
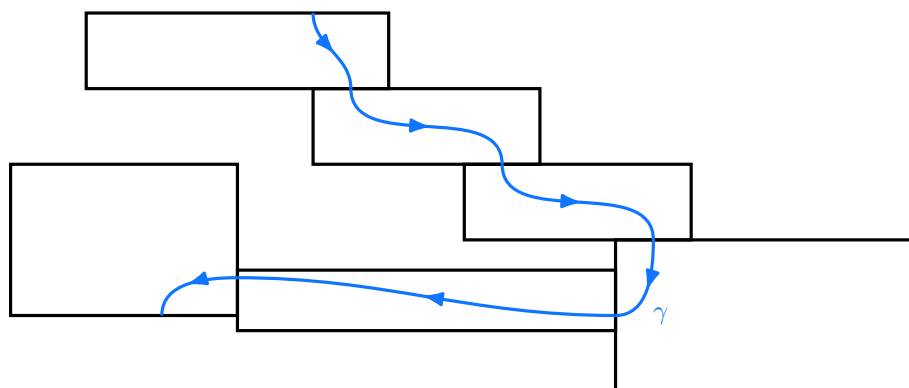
However by Theorem 2 of [4, section 5-7, page 402], a smooth closed curve in the plane without self-crossings has rotation index ± 1, where the rotation index of a curve is the number of times its tangent vector turns around the origin. This means that by the intermediate value theorem, for every $\alpha \in]0, 2\pi[$, there exists a point of the curve where the tangent vector is at angle α with the x-axis, and hence belongs to the positive orthant. This is a contradiction.

We conclude that D is acyclic, hence it has a sink R – a rectangle with out-degree 0. Therefore, R is the rectangle we want: its left side intersects a vertical segment of container and its bottom side intersects a horizontal segment of the container.   ◀

Proof of Lemma 12. Let G be the conflict graph of $\mathcal{Q}$. For a rectangle $R \in \mathcal{Q}$, we denote its associated vertex in G by v_R. We define a planar embedding of G as follows. We define the position of a vertex v_R to be the center $c(R) = (x(R) + w(R)/2, y(R) + h(R)/2)$ of the corresponding rectangle. If there is an edge $e = v_R v_{R'}$, choose $y \in \mathbb{R}$ such that the horizontal segment $s = [x(R) + w(R), x(R')] \times \{y\}$ witnesses that R and R' that see each other, where we assume w.l.o.g. that $x(R) + w(R) \leqslant x(R')$. We define the embedding γ_e of e as the union of 3 internally disjoint segments:

- $s_e^1 = [c(R), (x(R) + w(R), y))]$,
- $s_e^2 = s$,
- $s_e^3 = [(x(R'), y), c(R')]$.

It is straightforward to check that all the curves $\gamma_e \ e \in E(G)$ are pairwise internally disjoint, hence they constitute a planar embedding of G.   ◀

Proof of Proposition 15. Let G be the conflict graph of $\mathcal{Q}$. Since C is an st-separator in G, we may partition $\mathcal{Q}$ into three disjoint sets X, C, and Y so that vertices of X are not connected to t, vertices of Y are not connected to s, and no vertex of X is adjacent to any vertex of Y. Now, construct a new set of placed rectangles $\mathcal{Q}'$ by removing all rectangles of C and shifting every rectangle of Y by ℓ to the left. It remains to prove that $\mathcal{Q}'$ is a packing and that all rectangles of $\mathcal{Q}'$ are entirely contained in $\overleftarrow{Z}\langle \ell \rangle$.

For the second assertion, we need to prove that (i) no $R' \in \mathcal{Q}'$ crosses the left side of the box, i.e., no $R' \in \mathcal{Q}'$ is such that $x(R') < 0$, and (ii) no rectangle $R' \in \mathcal{Q}'$ contains a point with horizontal coordinate larger than $N_2 - \ell$, i.e. $x(R') + w(R') > N_2 - \ell$. To prove (i), suppose for the sake of a contradiction that there exists $R' \in Y$ such that $x(R') < 0$. We must have $R' \in Y$ because R' was shifted, and hence R' cannot see the left side of the box. Let $R \in \mathcal{Q}$ be R' before shifting. We know that $x(R) < \ell$, therefore as every rectangle has width at least ℓ, there is no rectangle in $\mathcal{Q}$ that would be placed between R and the left side of the box. Therefore, the R must see the left side of the box, which is a contradiction because $R \in Y$. A symmetric argument involving the right side of the box proves (ii).

For the first assertion, we need to prove that no two rectangles in $\mathcal{Q}'$ overlap. The only case when this could a priori happen is if $R_1 \in X$ and $R_2 \in Y$ are overlapping after the shift. This would mean that $x(R_1) + w(R_1) < x(R_2) - \ell$. However, again in $\mathcal{Q}$ there cannot be any rectangle lying in between R_1 and R_2, because every rectangle has width at least ℓ. Therefore, the R_1 and R_2 must see each other, which is a contradiction because $R_1 \in X$ and $R_2 \in Y$. ◀

Proof of Proposition 16. First, scale horizontally every rectangle in $\mathcal{Q}$ by a factor $\lambda = 1 + \ell/N_1$, i.e., for a rectangle $R = [x, x + w] \times [y, y + h]$ we define the rectangle $R^\times = [\lambda x, \lambda(x + w)] \times [y, y + h]$. These rectangles fit inside $\text{round}_\ell(Z)$. Indeed, the maximum possible displacement of a point is $N_1 \cdot \ell/N_1 = \ell$, i.e. the image of a point under scaling is at horizontal distance at most ℓ to the right of the original point. Next, observe that every rectangle $\text{round}_\ell(R)$ can be entirely placed inside the corresponding rectangle $R^\times$, because $\lambda w = w + w\ell/N_1 \geqslant w + \ell^2/N_1 = \ell' + w = \ell'(1 + w/\ell') \geqslant \ell' \lceil w/\ell' \rceil$. (Recall here that we assumed all rectangles to have width at least ℓ.) Now, $\mathcal{Q}'$ can be obtained from $\mathcal{Q}$ by replacing each $R \in \mathcal{Q}'$ with $R^\times$, fitting $\text{round}_\ell(R)$ inside $R^\times$, and finally shifting all rectangles to the left so that they have integer coordinates. The last step is always possible as every rectangle has integer length. ◀

Proof of Proposition 20. Suppose $\mathcal{P}$ crosses $\overleftarrow{R}\langle\ell\rangle$ for some $R \in \mathcal{Q}$. Then there exists $R_1, R_2 \in V(P)$ (which are possibly the left or the right side of the box) such that R_1 and R_2 see each other through a segment $s = [x(R_1) + w(R_1), x(R_2)] \times \{y\}$ and R crosses one of the following segments:
1. $\{x(R_1) + w(R_1)/2\} \times [\min\{y(R_1) + h(R_1)/2, y\}, \max\{y(R_1) + h(R_1)/2, y\}]$,
2. $[x(R_1) + w(R_1)/2, x(R_2) + w(R_2)/2] \times \{y\}$,
3. $\{x(R_2) + w(R_2)/2\} \times [\min\{y, y(R_2) + h(R_2)/2\}, \max\{y, y(R_2) + h(R_2)/2\}]$.
We show that every case leads to a contradiction.
1. Assume case 1. $\mathcal{P}$ crosses $\overleftarrow{R}\langle\ell\rangle$ but not R so $x(R) \geqslant x(R_1) + w(R_1)/2$ and $x(R) - \ell \leqslant x(R_1) + w(R_1)/2$. Therefore $x(R_1) \leqslant x(R) \leqslant x(R_1) + w(R_1)/2 + \ell \leqslant x(R_1) + w(R_1)$ because $w(R_1) \geqslant 2\ell$. Moreover, $[\min\{y(R_1) + h(R_1)/2, y\}, \max\{y(R_1) + h(R_1)/2, y\}] \subseteq [y(R_1), y(R_1) + h(R_1)]$ by the definition of y. This means that R and R_1 intersect at $(x(R), y')$ where $y' \in [y(R), y(R) + h(R)] \cap [\min\{y(R_1) + h(R_1)/2, y\}, \max\{y(R_1) + h(R_1)/2, y\}]$, which is not possible.
2. Assume case 2. This would mean that $y \in [y(R), y(R) + h(R)]$, $x(R) \geqslant x(R_2) + w(R_2)/2$ and $x(R) - \ell \leqslant x(R_2) + w(R_2)/2$ because $|[x(R_1) + w(R_1)/2, x(R_2) + w(R_2)/2]| \geqslant 2\ell$ and $\mathcal{P}$ crosses $\overleftarrow{R}\langle\ell\rangle$ but not R. Therefore $x(R_2) \leqslant x(R) \leqslant x(R_2) + w(R_2)/2 + \ell \leqslant x(R_2) + w(R_2)$ because $w(R_2) \geqslant 2\ell$. This means that R and R_2 intersect at $(x(R), y)$, which is not possible.
3. Assume case 3. This is a similar argument as case 1, replacing R_1 by R_2. ◀

Proof of Lemma 25. Let $\mathcal{Q}$ be the assumed packing of $k' \leqslant k$ rectangles from $\mathcal{R}$ of pairwise different colors in the zone Z. Note that if $\mathcal{Q}$ contains some rectangle of $R \in \mathcal{R} \setminus \text{reduce}_k(\mathcal{R}, \text{col})$, then there exists another rectangle $R' \in \text{reduce}_k(\mathcal{R}, \text{col})$ with $w(R') = w(R)$, $\text{col}(R') = \text{col}(R)$ and $h(R') \leqslant h(R)$ such that R' was not used in the packing $\mathcal{Q}$. Hence, we can substitute R with R' in the packing $\mathcal{Q}$, fitting R' within the area freed by removing R from the packing. By applying such substitutions exhaustively, we obtain a packing in Z consisting of k' rectangles from $\text{reduce}_k(\mathcal{R}, \text{col})$. ◀