Online Packing to Minimize Area or Perimeter

Mikkel Abrahamsen
BARC, University of Copenhagen, Denmark

Lorenzo Beretta
BARC, University of Copenhagen, Denmark

Abstract

We consider online packing problems where we get a stream of axis-parallel rectangles. The rectangles have to be placed in the plane without overlapping, and each rectangle must be placed without knowing the subsequent rectangles. The goal is to minimize the perimeter or the area of the axis-parallel bounding box of the rectangles. We either allow rotations by 90° or translations only.

For the perimeter version we give algorithms with an absolute competitive ratio slightly less than 4 when only translations are allowed and when rotations are also allowed.

We then turn our attention to minimizing the area and show that the competitive ratio of any algorithm is at least $\Omega(\sqrt{n})$, where n is the number of rectangles in the stream, and this holds with and without rotations. We then present algorithms that match this bound in both cases and the competitive ratio is thus optimal to within a constant factor. We also show that the competitive ratio cannot be bounded as a function of $\textsc{Opt}$. We then consider two special cases.

The first is when all the given rectangles have aspect ratios bounded by some constant. The particular variant where all the rectangles are squares and we want to minimize the area of the bounding square has been studied before and an algorithm with a competitive ratio of 8 has been given [Fekete and Hoffmann, Algorithmica, 2017]. We improve the analysis of the algorithm and show that the ratio is at most 6, which is tight.

The second special case is when all edges have length at least 1. Here, the $\Omega(\sqrt{n})$ lower bound still holds, and we turn our attention to lower bounds depending on $\textsc{Opt}$. We show that any algorithm for the translational case has a competitive ratio of at least $\Omega(\sqrt{\textsc{Opt}})$. If rotations are allowed, we show a lower bound of $\Omega(\sqrt[4]{\textsc{Opt}})$. For both versions, we give algorithms that match the respective lower bounds: With translations only, this is just the algorithm from the general case with competitive ratio $O(\sqrt{n}) = O(\sqrt{\textsc{Opt}})$. If rotations are allowed, we give an algorithm with competitive ratio $O(\min\{\sqrt{n}, \sqrt[4]{\textsc{Opt}}\})$, thus matching both lower bounds simultaneously.

2012 ACM Subject Classification Theory of computation → Computational geometry; Theory of computation → Packing and covering problems; Theory of computation → Online algorithms

Keywords and phrases Packing, online algorithms

Digital Object Identifier 10.4230/LIPIcs.SoCG.2021.6

Related Version *Full Version*: https://arxiv.org/abs/2101.09024

Funding Research of both authors partly supported by Investigator Grant 16582, Basic Algorithms Research Copenhagen (BARC), from the VILLUM Foundation.
Lorenzo Beretta: receives funding from the European Union's Horizon 2020 research and innovation program under the Marie Skłodowska-Curie grant agreement No 801199.

1 Introduction

Problems related to packing appear in a plethora of big industries. For instance, two-dimensional versions of packing arise when a given set of pieces have to be cut out from a large piece of material so as to minimize waste. This is relevant to clothing production where cutting patterns are cut out from a roll of fabric, and similarly in leather, glass, wood, and sheet metal cutting.

37th International Symposium on Computational Geometry (SoCG 2021).
Editors: Kevin Buchin and Éric Colin de Verdière; Article No. 6; pp. 6:1–6:15
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

In some applications, it is important that the pieces are placed in an *online* fashion. This means that the pieces arrive one by one and we need to decide the placement of one piece before we know the ones that will come in the future. This is in contrast to *offline* problems, where all the pieces are known in advance. Problems related to packing were some of the first for which online algorithms were described and analyzed. Indeed, the first use of the terms "online" and "offline" in the context of approximation algorithms was in the early 1970s and used for algorithms for bin-packing problems [14].

In this paper, we study online packing problems where the pieces can be placed anywhere in the plane as long as they do not overlap. The goal is to minimize the region occupied by the pieces. The pieces are axis-parallel rectangles, and they may or may not be rotated by 90°. We want to minimize the size of the axis-parallel bounding box of the pieces, and the size of the box is either the perimeter or the area. This results in four problems: PerimeterRotation, PerimeterTranslation, AreaRotation, and AreaTranslation.

Competitive analysis

The *competitive ratio* of an online algorithm is the equivalent of the *approximation ratio* of an (offline) approximation algorithm. The usual definitions [7, 9, 11] of competitive ratio (or *worst case ratio*, as it may also be called [11]) can only be used to describe that the cost of the solution produced by an online algorithm is at most some constant factor higher than the cost Opt of the optimal (offline) solution. In the study of approximation algorithms, it is often the case that the approximation ratio is described not just as a constant, but as a more general function of the input. In the same way, we generalize the definition of competitive ratios to support such statements about online algorithms.

Consider an algorithm A for one of the packing problems studied in this paper. Let $\mathcal{L}$ be the set of non-empty streams of rectangular pieces. For a stream $L \in \mathcal{L}$, we define $A(L)$ to be the cost of the packing produced by A and let $\textsc{Opt}(L)$ be the cost of the optimal (offline) packing. We say that A has a *competitive ratio* of $f(L)$, for some function $f : \mathcal{L} \longrightarrow \mathbb{R}^+$ which may just be a constant, if

$$\sup_{L \in \mathcal{L}} \frac{A(L)}{\textsc{Opt}(L) f(L)} \leq 1.$$

In this paper, the functions $f(L)$ that we consider will be (i) constants, (ii) functions of the number of pieces $n = |L|$, (iii) functions of $\textsc{Opt}(L)$.

Results and structure of the paper

We develop online algorithms for the perimeter versions PerimeterRotation and PerimeterTranslation, both with a competitive ratio slightly less than 4. These algorithms are described in Section 2. The idea is to partition the positive quadrant into *bricks*, which are axis-parallel rectangles with aspect ratio $\sqrt{2}$. In each brick, we build a stack of pieces which would be too large to place in a brick of smaller size. Online packing algorithms using higher-dimensional bricks were described by Januszewski and Lassak [15] and our algorithms are inspired by an algorithm of Fekete and Hoffmann [13].

In Section 3, we study the area versions AreaRotation and AreaTranslation. We show in Section 3.1 that any algorithm A processing a stream of n pieces cannot achieve a better competitive ratio than $\Omega(\sqrt{n})$, and this holds for all online algorithms and with and without rotations allowed. It also holds in the special case where all the edges of pieces have length at least 1. We furthermore show that when the pieces can be arbitrary, no bound on the competitive ratio as a function of $\textsc{Opt}$ for $\textsc{AreaRotation}$ nor $\textsc{AreaTranslation}$. In Section 3.2 we describe the algorithms $\textsc{DynBoxTrans}$ and $\textsc{DynBoxRot}$, which achieve a $O(\sqrt{n})$ competitive ratio for $\textsc{AreaTranslation}$ and $\textsc{AreaRotation}$, respectively, for an arbitrary stream of n pieces. This is thus optimal up to a constant factor when measuring the competitive ratio as a function of n. Both algorithms use a row of boxes of exponentially increasing width and dynamically adjusted height. In these boxes, we pack pieces using a next-fit shelf algorithm, which is a classic online strip packing algorithm first described by Baker and Schwartz [6].

We then turn our attention to two special cases. The first special case is when the aspect ratio is bounded by a constant $\alpha \geq 1$. A case of particular interest is when all pieces are squares, i.e., $\alpha = 1$. It is natural to have the same requirement to the container as to the pieces, so let us assume that the goal is to minimize the area of the axis-parallel bounding square of the pieces, and call the problem $\textsc{SquareInSquareArea}$. This problem was studied by Fekete and Hoffmann [13], and they gave an algorithm for the problem and proved that it was 8-competitive. We prove that the same algorithm is in fact 6-competitive and that this is tight. It easily follows that if the aspect ratio is bounded by an arbitrary constant $\alpha \geq 1$ or if the goal is to minimize the area of the axis-parallel bounding rectangle, we also get a $O(1)$-competitive algorithm.

The second special case is when all edges are *long*, that is, when they have length at least 1 (any other constant will work too). In Section 3.4, we show that under this assumption, there is a lower bound of $\Omega(\sqrt{\textsc{Opt}})$ for the competitive ratio of $\textsc{AreaTranslation}$, whereas for $\textsc{AreaRotation}$, we get the lower bound $\Omega(\sqrt[4]{\textsc{Opt}})$. In Section 3.5, we provide algorithms for the area versions when the edges are long. For both problems $\textsc{AreaRotation}$ and $\textsc{AreaTranslation}$, we give algorithms that match the lower bounds of Section 3.4 to within a constant factor. With translations only, this is just the algorithm from the general case with competitive ratio $O(\sqrt{n}) = O(\sqrt{\textsc{Opt}})$. The algorithm with ratio $O(\sqrt[4]{\textsc{Opt}})$ for the rotational case follows the same scheme as the algorithms for arbitrary rectangles of Section 3.2, but differ in the way we dynamically increase boxes' heights. We finally describe an algorithm for the rotational case with competitive ratio $O(\min\{\sqrt{n}, \sqrt[4]{\textsc{Opt}}\})$, thus matching the lower bounds $\Omega(\sqrt{n})$ and $\Omega(\sqrt[4]{\textsc{Opt}})$ simultaneously. Actually, the two lower bounds for $\textsc{AreaRotation}$ can be summarized by $\Omega(\max\{\sqrt{n}, \sqrt[4]{\textsc{Opt}}\})$, while we manage to achieve a competitive ratio of $O(\min\{\sqrt{n}, \sqrt[4]{\textsc{Opt}}\})$. However, this gives no contradiction, it simply proves that the *edge cases* that have a competitive ratio of at least $\Omega(\sqrt[4]{\textsc{Opt}})$ must satisfy $\textsc{Opt} = O(n^2)$, and those for which the competitive ratio is at least $\Omega(\sqrt{n})$ satisfy $n = O(\sqrt{\textsc{Opt}})$. We summarize the results in Table 1.

Related work

The literature on online packing problems is rich. See the surveys of Christensen, Khan, Pokutta, and Tetali [9], van Stee [25, 26], and Csirik and Woeginger [11] for an overview. It seems that the vast majority of previous work on online versions of two-dimensional packing problems is concerned with either bin packing (packing the pieces into a minimum number of unit squares) or strip packing (packing the pieces into a strip of unit width so as to minimize the total height of the pieces). From a mathematical point of view, we find the problems studied in this paper perhaps even more fundamental than these important problems in the sense that we give no restrictions on where to place the pieces, whereas the pieces are restricted by the boundaries of the bins and the strip in bin and strip packing.

Table 1 Results of this paper.

Measure	Version	Trans./Rot.	Lower bound	Upper bound
Perimeter	General	Translation	4/3, Sec. 2.2	$4-\varepsilon$, Sec. 2.1
		Rotation	5/4, Sec. 2.2	$4-\varepsilon$, Sec. 2.1
Area	General	Translation	$\Omega(\sqrt{n})$ & $\forall f : \Omega(f(\textsc{Opt}))$, Sec. 3.1	$O(\sqrt{n})$, Sec. 3.2
		Rotation	$\Omega(\sqrt{n})$ & $\forall f : \Omega(f(\textsc{Opt}))$, Sec. 3.1	$O(\sqrt{n})$, Sec. 3.2
	Sq.-in-sq.	N/A	16/9, Sec. 3.3	6, Sec. 3.3
	Long edges	Translation	$\Omega(\sqrt{\textsc{Opt}})$, Sec. 3.4	$O(\sqrt{n}) = O(\sqrt{\textsc{Opt}})$, Sec. 3.5
		Rotation	$\Omega(\max\{\sqrt{n}, \sqrt[4]{\textsc{Opt}}\})$, Sec. 3.1 and 3.4	$O(\min\{\sqrt{n}, \sqrt[4]{\textsc{Opt}}\})$, Sec. 3.5

Another related problem is to find the critical density of online packing squares into a square. In other words, what is the maximum $\Sigma \leq 1$ such that there is an online algorithm that packs any stream of squares of total area at most Σ into the unit square? This was studied, among others, by Fekete and Hoffmann [13] and Brubach [8]. Lassak [16] and Januszewski and Lassak [15] studied higher-dimensional versions of this problem.

Milenkovich [20, 21] and Milenkovich and Daniels [22] studied generalized offline versions of the minimum area problem where the pieces are simple or convex polygons. Some algorithms have been described for computing the packing of two or three convex polygons that minimizes the perimeter or area of the convex hull or the bounding box [1, 5, 17, 23].

Alt [2] and Alt, de Berg, and Knauer [4] gave constant factor approximation algorithms for the offline versions of AreaTranslation and AreaRotation when the pieces are axis parallel rectangles or convex polygons, with translations only or arbitrary rotations allowed.

Lubachevsky and Graham [18] used computational experiments to find the rectangles of minimum area into which a given number $n \leq 5000$ of congruent circles can be packed; see also the follow-up work by Specht [24]. In another paper, Lubachevsky and Graham [19] studied the problem of minimizing the perimeter instead of the area.

Another fundamental packing problem is to find the smallest square containing a given number of *unit* squares, with arbitrary rotations allowed. A long line of mathematical research has been devoted to this problem, initiated by Erdős and Graham [12] in 1975, and it is still an active research area [10].

2 The perimeter versions

In Section 2.1, we present two online algorithms to minimize the perimeter of the bounding box: the algorithm BrickTranslation solves the problem PerimeterTranslation, where we can only translate pieces; the algorithm BrickRotation solves the problem PerimeterRotation, where also rotations are allowed. Both algorithms achieve a competitive ratio of 4. In Section 2.2, we show a lower bound of 4/3 for the version with translations and 5/4 for the version with rotations.

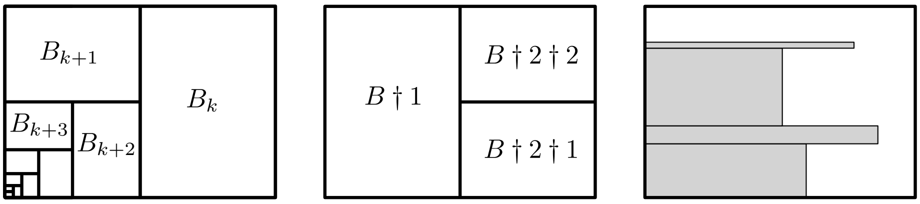

Figure 1 Left: Fundamental bricks. Middle: Splitting a brick. Right: Pieces packed in a brick.

2.1 Algorithms to minimize perimeter

Algorithm for translations

We pack the pieces into non-overlapping *bricks*; a technique first described by Januszewski and Lassak [15] which was also used by Fekete and Hoffmann [13] for the problem SQUARE-INSQUAREAREA. Let a *k-brick* be a rectangle of size $\sqrt{2}^{-k} \times \sqrt{2}^{-k-1}$ if k is even and $\sqrt{2}^{-k-1} \times \sqrt{2}^{-k}$ if k is odd. A *brick* is a k-brick for some integer k.

We tile the positive quadrant using one k-brick B_k for each integer k as in Figure 1 (left): if k is even, B_k is the k-brick with lower left corner $(0, \sqrt{2}^{-k-1})$ and otherwise, B_k is the k-brick with lower left corner $(\sqrt{2}^{-k-1}, 0)$. The bricks B_k are called the *fundamental* bricks. We define $B_{>k} := \bigcup_{i>k} B_i$ and $B_{\geq k} := B_{>k-1}$, so that $B_{>k}$ is the k-brick immediately below (if k is even) or to the left (if k is odd) of B_k.

An important property of a k-brick B is that it can be split into two $(k+1)$-bricks: $B \dagger 1$ and $B \dagger 2$; see Figure 1 (middle). We introduce a uniform naming and define $B \dagger 1$ to be the left half of B if k is even and the lower half of B if k is odd.

We define a *derived* brick recursively as follows: a derived brick is either (i) a fundamental brick B_k or (ii) $B \dagger 1$ or $B \dagger 2$, where B is a derived brick. We introduce an ordering $\prec$ of the derived k-bricks as follows. Consider two derived k-bricks D_1 and D_2 such that $D_1 \subset B_i$ and $D_2 \subset B_j$. If $i > j$, then $D_1 \prec D_2$. Else, if $i = j$ then the bricks D_1 and D_2 are both obtained by splitting the fundamental brick B_i, and the number of splits is $\ell := i - k$. Hence the bricks have the forms $D_1 = B_i \dagger b_{11} \dagger b_{12} \dagger \ldots \dagger b_{1\ell}$ and $D_2 = B_i \dagger b_{21}\, b_{22} \ldots b_{2\ell}$, where $b_{ij} \in \{1,2\}$ for $i \in \{1,2\}$ and $j \in \{1,\ldots,\ell\}$. We then define $D_1 \prec D_2$ if $(b_{11}, b_{12}, \ldots, b_{1\ell})$ precedes $(b_{21}, b_{22}, \ldots, b_{2\ell})$ in the lexicographic ordering.

We say that a k-brick is *suitable* for a piece p of size $w \times h$ if the width and height of the brick are at least w and h, respectively, and if that is not the case for a $(k+1)$-brick. We will always pack a given piece p in a derived k-brick that is suitable for p.

We now explain how we pack pieces into one specific brick; see Figure 1 (right). The first piece p that is packed in a brick B is placed with the lower left corner of p at the lower left corner of B. Suppose now that some other pieces $p_1, \ldots, p_i$ have been packed in B. If k is even, then $p_1, \ldots, p_i$ form a stack with the left edges contained in the left edge of B, and we place p on top of p_i (again, with the left edge of p contained in the left edge of B). Otherwise, $p_1, \ldots, p_i$ form a stack with the bottom edges contained in the bottom edge of B, and we place p to the right of p_i (again, with the bottom edge of p contained in the bottom edge of B). We say that a brick *has room* for a piece p if the packing scheme above places p within B, and it is apparent that an empty suitable brick for p has room for p.

The algorithm BRICKTRANSLATION maintains the collection $\mathcal{D}$ of non-overlapping derived bricks, such that one or more pieces have been placed in each brick in $\mathcal{D}$; see Figure 2. Before the first piece arrives, we set $\mathcal{D} := \emptyset$. Suppose that some stream of pieces have been packed, and that a new piece p appears. Choose k such that a k-brick is suitable for p. If there exists

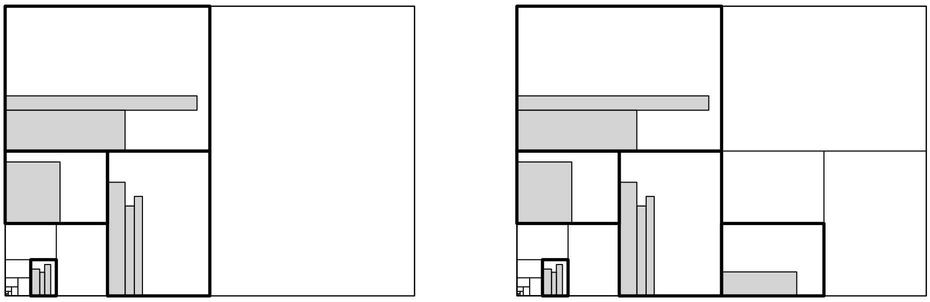

Figure 2 Left: Some pieces have been packed by the algorithm. The bricks in $\mathcal{D}$ are drawn with fat edges. Right: A new piece arrives. There is already a brick of the suitable size in $\mathcal{D}$, but there is not enough room, so a new brick of the same size is added to $\mathcal{D}$ where the piece is placed.

a derived k-brick $D \in \mathcal{D}$ such that D has room for p, then we pack p in D. Else, let D be the minimum derived k-brick (with respect to the ordering $\prec$ described before) such that D is interior-disjoint from each brick in $\mathcal{D}$; we then add D to $\mathcal{D}$ and pack p in D.

▶ **Theorem 1.** *The algorithm* BrickTranslation *has a competitive ratio strictly less than 4 for* PerimeterTranslation*.*

Proof. This proof relies on a careful case analysis of which bricks are contained in $\mathcal{D}$ and is deferred to the full version. ◀

Algorithm using rotations

The algorithm BrickRotation is almost identical to BrickTranslation, but with the difference that we rotate each piece so that its height is at least its width.

▶ **Theorem 2.** *The algorithm* BrickRotation *has a competitive ratio of strictly less than 4 for* PerimeterRotation*.*

Proof. This proof is deferred to the full version. ◀

2.2 Lower bounds

In this section we state two lower bounds worth reporting. However, their proofs are of modest interest and are deferred to the full version.

▶ **Lemma 3.** *Any algorithm for* PerimeterTranslation *has competitive ratio* $\geq 4/3$*.*

▶ **Lemma 4.** *Any algorithm for* PerimeterRotation *has competitive ratio* $\geq 5/4$*.*

3 Area versions

3.1 General lower bounds

In this section we show that, if we allow pieces to be arbitrary rectangles, we cannot bound the competitive ratio for neither AreaTranslation nor AreaRotation as a function of the area Opt of the optimal packing. However we will be able to bound the competitive ratio as a function of the total number n of pieces in the stream.

▶ **Lemma 5.** *Consider any algorithm A solving* AreaTranslation *or* AreaRotation *and let any $m \in \mathbb{N}$ and $p \in \mathbb{R}$ be given. There exists a stream of $n = m^2 + 1$ rectangles such that (i) the rectangles can be packed into a bounding box of area $2p^2$, and (ii) algorithm A produces a packing with a bounding box of area at least mp^2.*

Proof. We first feed A with m^2 rectangles of size $p \times \frac{p}{m^2}$. These rectangles have total area p^2. Let $a \times b$ be the size of the bounding box of the produced packing.

Suppose first that $a \geq \frac{p}{m}$ and $b \geq \frac{p}{m}$ hold. We then feed A with a long rectangle of size $pm^2 \times \frac{p}{m^2}$. The produced packing has a bounding box of area at least $\frac{p}{m} \cdot pm^2 = mp^2$. The optimal packing is to pack the m^2 small rectangles along the long rectangle, which would produce a packing with bounding box of size $pm^2 \times \frac{2p}{m^2} = 2p^2$.

Otherwise, we must have $b > pm$ or $a > pm$, since $ab \geq p^2$. We then feed A with a square of size $p \times p$. The produced packing has a bounding box of area at least $p \cdot pm = mp^2$. The optimal packing is obtained stacking the m^2 thin rectangles on top of the big square, which produces a packing with bounding box of size $p \times 2p = 2p^2$. ◀

▶ **Corollary 6.** *Let A be an algorithm for* AreaTranslation *or* AreaRotation*. Then A does not have a competitive ratio which is a function of* Opt*.*

Proof. Let f be any function of Opt. For any value Opt $= c$, we choose $p := \sqrt{c/2}$. We now choose $m > 2f(c)$ and obtain that the competitive ratio is at least $\frac{mp^2}{2p^2} = m/2 > f(c) = f(\text{Opt})$. ◀

▶ **Corollary 7.** *Let A be an algorithm for* AreaTranslation *or* AreaRotation*. If A has a competitive ratio of $f(n)$, where $n = |L|$ is the number of pieces in the stream, then $f(n) = \Omega(\sqrt{n})$. This holds even when all edges of the pieces are required to have length at least 1.*

Proof. We choose $p := m^2$. Then all edges have length at least 1, and the competitive ratio is at least $\frac{mp^2}{2p^2} = m/2 = \Omega(\sqrt{n})$. Here, Opt can be arbitrarily big by choosing m big enough, so it is a lower bound on the competitive ratio. ◀

3.2 Algorithms for arbitrary pieces

In this section we provide algorithms that solve AreaTranslation and AreaRotation with a competitive ratio of $O(\sqrt{n})$, where n is the total number of pieces. Thus we match the bounds provided in the previous section.

We first describe the algorithm DynBoxTrans that solves AreaTranslation. We assume to receive a stream of pieces $p_1, \ldots, p_n$ of unknown length n, such that piece p_i has size $w_i \times h_i$. For each $k \in \mathbb{Z}$, we define a rectangular box B_k with a size varying dynamically. After pieces $p_1, \ldots, p_j$ have been processed B_k has size $2^k \times T_j$, where $T_j := H_j\sqrt{j} + 7H_j$ and $H_j := \max_{i=1,\ldots,j} h_i$. We place the boxes with their bottom edges on the x-axis and in order such that the right edge of B_{k-1} is contained in the left edge of B_k; see Figure 3. Furthermore, we place the lower left corner of box B_0 at the point $(1, 0)$. It then holds that all the boxes are to the right of the point $(0, 0)$.

We say that the box B_k is *wide enough* for a piece $p_i = w_i \times h_i$ if $w_i \leq 2^k$. If a box B_k is wide enough for p_i, we can pack p_i in B_k using the online strip packing algorithm NFS_k that packs rectangles into a strip of width 2^k. The algorithm NFS_k is the *next-fit shelf algorithm* first described by Baker and Schwartz [6]. The algorithm packs pieces in *shelves* (rows), and each shelf is given a fixed height of 2^j for some $j \in \mathbb{Z}$ when it is created; see Figure 4. The width of each shelf is 2^k, since this is the width of the box B_k.

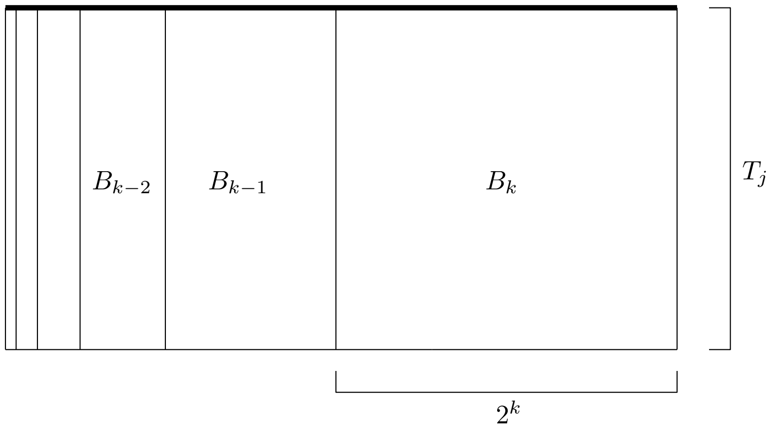

Figure 3 The algorithm DynBoxTrans packs pieces into the boxes B_k that form a row. Every box has height T_j that depends on $p_1 \dots p_j$ and is dynamically updated.

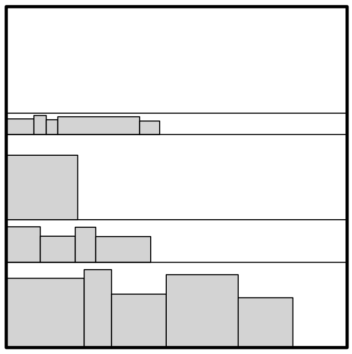

Figure 4 A packing produced by the next-fit shelf algorithm using four shelves.

A piece of height h, where $2^{j-1} < h \le 2^j$, is packed in a shelf of height 2^j. We divide the shelves into two types. If the total width of pieces in a shelf is more than 2^{k-1} we call that shelf *dense*, otherwise we say it is *sparse*. The algorithm NFS$_k$ places each piece as far left as possible into the currently sparse shelf of the proper height. If there is no sparse shelf of this height or the sparse shelf has not room for the piece, a new shelf of the appropriate height is created on top of the top shelf, and the piece is placed there at the left end of this new shelf. This ensures that at any point in time there exists at most one sparse shelf for each height 2^j.

If we allow the height of the box B_k to grow large enough with respect to shelves' heights, the space wasted by sparse shelves becomes negligible and we obtain a constant density strip packing, as stated in the following lemma.

▶ **Lemma 8.** *Let $\widetilde{H}$ be the total height of shelves in B_k, and H_{max} be the maximum height among pieces in B_k. If $\widetilde{H} \ge 6H_{max}$, then the pieces in B_k are packed with density at least* 1/12.

Proof. Let $2^{m-1} < H_{max} \le 2^m$, so that $\widetilde{H} \ge 3 \cdot 2^m$. For each $i \le m$ we have at most one sparse shelf of height 2^i and each shelf of B_k has height at most 2^m, hence the total height of sparse shelves is at most $\sum_{i \le m} 2^i = 2^{m+1}$, so the total height of dense shelves is at least $\widetilde{H} - 2^{m+1} \ge \widetilde{H}/3$. Thus, the total area of the dense shelves is at least $2^k \cdot \widetilde{H}/3$.

Consider a dense shelf of height 2^i. Into that shelf, we have packed pieces of height at least 2^{i-1}, and the total width of these pieces is at least 2^{k-1}. Hence, the density of pieces in the shelf is at least $1/4$. Therefore, the total area of pieces in B_k is at least $2^k \cdot \widetilde{H}/12$. On the other hand, the area of the bounding box is $2^k \cdot \widetilde{H}$, that yields the desired density. ◀

Now we are ready to describe how the algorithm works. When the first piece p_1 arrives, let $2^{k-1} < w_1 \leq 2^k$, then we pack it in the box B_k according to NFS_k and define B_k to be the *active box*. Suppose now that B_i is the active box when the piece p_j arrives, first we update the value of the threshold T_{j-1} to T_j, then we have two cases. If $w_j > 2^i$ we choose ℓ such that $2^{\ell-1} < w_j \leq 2^\ell$, pack p_j in B_ℓ and define B_ℓ to be the active box. Else, B_i is wide enough for p_j and we try to pack p_j into B_i. Since B_i has size $2^i \times T_j$ it may happen that NFS_i exceeds the threshold T_j while packing p_j, generating an overflow. In this case, instead of packing p_j in B_i, we pack p_j into B_{i+1} and define that to be the active box.

▶ **Theorem 9.** *The algorithm* DynBoxTrans *has a competitive ratio of $O(\sqrt{n})$ for the problem* AreaTranslation *on a stream of n pieces.*

Proof. First, define Σ_j as the total area of the first j pieces, $W := \max_{i=1,\dots,n} w_i$ and recall that $H_j = \max_{i=1,\dots,j} h_i$ and $T_j = H_j\sqrt{n} + 7H_j$. Let B_k be the last active box, so that we can enclose all the pieces in a bounding box of size $2^{k+1} \times T_n$, and bound the area returned by the algorithm as $\text{Alg} = O(2^k H_n \sqrt{n})$. On the other hand we are able to bound the optimal offline packing as $\text{Opt} = \Omega(\Sigma_n + WH_n)$.

If the active box never changed, then we have $2^k < 2W$ that implies $\text{Alg} = O(WH_n\sqrt{n}) = \text{Opt} \cdot O(\sqrt{n})$. Otherwise, let B_ℓ be the last active box before B_k, and p_j be the first piece put in B_k. Here we have two cases.

Case (1) $w_j > 2^\ell$ In this case we have $2^k < 2W$ that implies $\text{Alg} = O(WH_n\sqrt{n}) = \text{Opt} \cdot O(\sqrt{n})$.

Case (2) $w_j \leq 2^\ell$ In this case we have $k = \ell + 1$. Denote with $\widetilde{H_i}$ the total height of shelves in B_i. Then we have $\widetilde{H_\ell} \geq T_j - H_j = H_j\sqrt{n} + 6H_j$, otherwise we could pack p_j in B_ℓ. Thus, we can apply Lemma 8 and conclude that the box B_ℓ of size $2^\ell \times T_j$ is filled with constant density. Here we have two cases.

Case (2.1) $\widetilde{H_k} \leq T_j$ In this case we have $\text{Alg} = O(2^k T_j)$ and, thanks to the constant density packing of B_ℓ we have $\Sigma_j = \Theta(2^\ell \widetilde{H_\ell}) = \Theta(2^k T_j)$. Since $\text{Opt} \geq \Sigma_j$, we get $\text{Alg} = O(\text{Opt})$.

Case (2.2) $\widetilde{H_k} > T_j$ In this case we have $\text{Alg} = O(2^k \widetilde{H_k})$. Moreover, $\widetilde{H_k} = O(H_n + \Sigma_n/2^k)$, in fact if $2^{s-1} < H_n \leq 2^s$, then the total height of sparse shelves is $\sum_{i \leq s} 2^i = 2^{s+1} = O(H_n)$. Furthermore, dense shelves are filled with constant density, therefore their total height is at most $O(\Sigma_n/2^k)$. Finally, we need to show that $2^k = O(W\sqrt{n})$. Thanks to the constant density packing of B_ℓ, we have $2^k H_j \sqrt{j} = O(2^\ell T_j) = O(\Sigma_j)$. We can upper bound the size of every piece p_i for $i \leq j$ with $W \times H_j$ and obtain $\Sigma_j \leq n \cdot WH_j$. Plugging it in the previous estimate and dividing both sides by $H_j\sqrt{n}$ we get $2^k = O(W\sqrt{n})$. Now we have $\text{Alg} = O(2^k\widetilde{H_k}) = O(2^k H_n + \Sigma_n) = O(WH_n\sqrt{n} + \Sigma_n) = \text{Opt} \cdot O(\sqrt{n})$. ◀

The algorithm DynBoxRot is obtained from DynBoxTrans with a slight modification: before processing any piece p_i we rotate it so that $w_i \leq h_i$. In this way, it still holds that $\text{Opt} = \Omega(\Sigma_n + WH_n)$ and the proof of Theorem 9 works also for the following.

▶ **Theorem 10.** *The algorithm* DynBoxRot *has a competitive ratio of $O(\sqrt{n})$ for the problem* AreaRotation *on a stream of n pieces.*

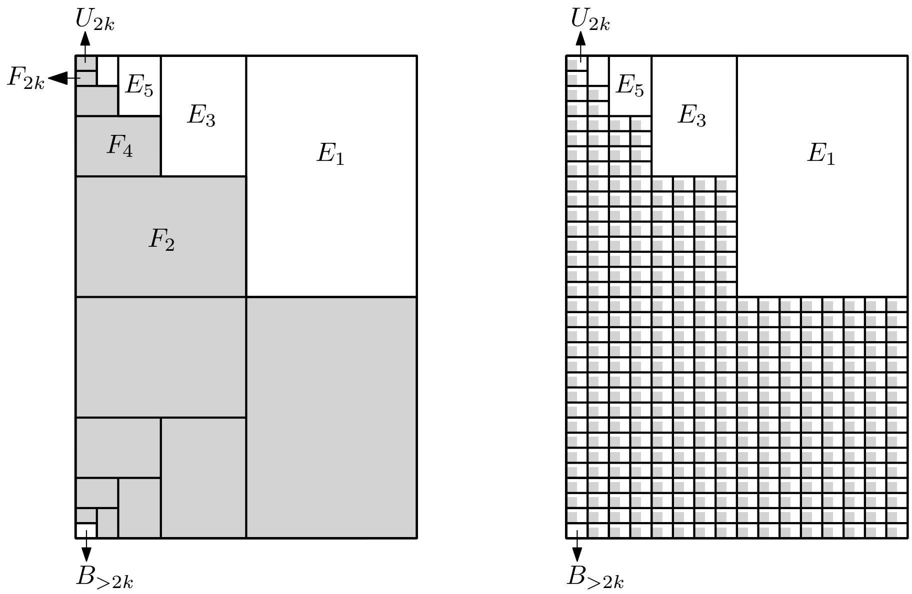

Figure 5 Left: A $2k$-packing. The grey bricks are non-empty and may have been split into smaller bricks. Right: The $2k$-packing produced by BrickTranslation when providing the algorithm with enough copies of the square S_k (the small grey squares), showing that the competitive ratio can be arbitrarily close to 6.

3.3 Bounded aspect ratio

In this section, we will consider the special case where the aspect ratio of all pieces is $\alpha = 1$, i.e., all the pieces are squares. Furthermore, we will measure the size of the packing as the area of the minimum axis-parallel bounding *square*, and we call the resulting problem SquareInSquareArea. Since we get a constant competitive ratio in this case, it follows that for other values of α and when allowing the bounding box to be a general rectangle, one can likewise achieve a constant competitive ratio. Here we give a lower bound, that is proven in the full version.

▶ **Lemma 11.** *Consider any algorithm A for the problem SquareInSquareArea. Then the competitive ratio of A is at least* $16/9$.

We are now going to analyze the competitive ratio of the algorithm BrickTranslation when it is fed with squares only. Note that a brick can never contain more than one piece. The algorithm is almost the same as the one described by Fekete and Hoffmann [13]. Their analysis proved that the algorithm is 8-competitive, with a more careful analysis we prove the following.

▶ **Theorem 12.** *The algorithm BrickTranslation has a competitive ratio of* 6 *for SquareInSquareArea. The analysis is tight.*

Proof. Suppose a stream of squares have been packed by BrickTranslation, and let Alg be the area of the bounding square of the resulting packing. Let B_k be the largest elementary brick in which a square has been placed. Suppose without loss of generality that $k = 0$, so that B_k has size $1 \times 1/\sqrt{2}$ and $B_{\geq k}$, which contains all the packed squares, has size $1 \times \sqrt{2}$.

We now recursively define a type of packing that we call a $2k$-packing, for a non-negative integer k; see Figure 5 (left). As k increases, so do the requirements to a $2k$-packing, in the sense that a $(2k+2)$-packing is also a $2k$-packing, but the other way is in general not the case. Define $F_0 := B_{\geq 1}$ and $U_0 := B_0$. A packing is a 0-packing if pieces have been placed in U_0 (the brick U_0 may or may not have been split in smaller bricks). Hence, the considered packing is a 0-packing by the assumption that a piece has been placed in B_0. Suppose that we have defined a $2k$-packing for some integer k. A $(2k+2)$-packing is a $2k$-packing with the additional requirements that

- the brick U_{2k} has been split into $L := U_{2k} \dagger 1$ and $E_{2k+1} := U_{2k} \dagger 2$,
- the right brick E_{2k+1} is empty,
- the left brick L has been split into $F_{2k+2} := L \dagger 1$ and $U_{2k+2} := L \dagger 2$, and
- U_{2k+2} is non-empty, and thus also F_{2k+2} is non-empty.

The symbols U_j, E_j, F_j have been chosen such that the brick is a j-brick, i.e., the index tells the size of the brick.

Consider a $2k$-packing. It follows from the definition that along the top edge of $B_{\geq 0}$ from the right corner $(1, \sqrt{2})$ to the left corner $(0, \sqrt{2})$, we meet a sequence $E_1, E_3, \ldots, E_{2k-1}$ of empty bricks of decreasing size, and finally meet a non-empty brick U_{2k} which may have been split into smaller bricks.

In the full version the following claim is proven.

▷ Claim 13. If the packing is a $2k$-packing and not a $(2k+2)$-packing, then $\text{ALG}/\text{OPT} < 6$.

Since we pack a finite number of squares, the produced packing is a $2k$-packing but not a $(2k+2)$-packing for some sufficiently large k, so Claim 13 implies Theorem 12.

Moreover we prove in the full version that this analysis is tight. The idea is to show that for any given k and a small $\varepsilon > 0$, we can force the algorithm to produce a $2k$-packing, such that as $k \longrightarrow \infty$ and $\varepsilon \longrightarrow 0$, the ratio $\frac{\text{ALG}}{\text{OPT}}$ tends to 6. We use a stream where all pieces are a square S_k of size slightly more than $\sqrt{2}^{-k}/2 \times \sqrt{2}^{-k}/2$; see Figure 5 (right). ◀

3.4 More lower bounds when edges are long

We already saw in Corollary 7 that as a function of n, the competitive ratio of an algorithm for AreaTranslation or AreaRotation must be at least $\Omega(\sqrt{n})$, even when all edges have length 1. In this section, we give lower bounds in terms of Opt for the same case. Note that the assumption that the edges are long is needed for these bounds to be matched by actual algorithms, since Corollary 6 states that without the assumption, the competitive ratio cannot be bounded as a function of Opt.

▶ **Theorem 14.** *Consider any algorithm A for the problem AreaTranslation with the restriction that all edges of the given rectangles have length at least 1. If A has a competitive ratio $f(\text{OPT})$ as a function of Opt, then $f(\text{OPT}) = \Omega(\sqrt{\text{OPT}})$.*

▶ Remark 15. Note that when the edges are long, $\sqrt{\text{OPT}} = \Omega(\sqrt{n})$, so this bound is stronger than the $\Omega(\sqrt{n})$ bound of Corollary 7.

Proof of Theorem 14. For any $n \in \mathbb{N}$, we do as follows. We first provide A with n^2 unit squares. Let the bounding box of the produced packing of these squares have size $a \times b$. Assume without loss of generality that $a \leq b$, so that $b \geq n$. We now give A the rectangle $n^2 \times 1$. The optimal offline solution to this set of rectangles has a bounding box of size $n^2 \times 2$. The packing produced by A has a bounding box of size at least $n^2 \times n = \Omega(\sqrt{\text{OPT}}) \cdot \text{OPT}$. ◀

▶ **Theorem 16.** *Consider any algorithm A for the problem AreaRotation with the restriction that all edges of the given rectangles have length at least 1. If A has a competitive ratio $f(\text{OPT})$ as a function of Opt, then $f(\text{OPT}) = \Omega(\sqrt[4]{\text{OPT}})$.*

Proof. For any $n \in \mathbb{N}$, we do as follows. We first provide A with n^2 unit squares. Let the bounding box of the produced packing of these squares have size $a \times b$. Assume without loss of generality that $a \leq b$. If $a \geq n^{1/2}$, we give A the rectangle $1 \times n^2$. Otherwise, we have $b > n^{3/2}$, and then we give A the square $n \times n$. In either case, there is an optimal offline solution of area $2n^2$, but the bounding box of the packing produced by A has area at least $n^{5/2} = \Omega(\sqrt[4]{\text{OPT}}) \cdot \text{OPT}$. ◀

3.5 Algorithms when edges are long

In this section, we describe algorithms that match lower bounds of Section 3.4. We analyze these algorithms under the assumption that we feed them with rectangles with edges of length at least 1 (of course, any other positive constant will also work), but we require no bound on the aspect ratio. Under this assumption, we observe that DynBoxTrans has competitive ratio $O(\sqrt{\text{OPT}})$ for AreaTranslation. We then describe the algorithm DynBoxRot$_{\sqrt[4]{\text{OPT}}}$, which we prove to have competitive ratio $O(\sqrt[4]{\text{OPT}})$ for AreaRotation. By Theorems 14 and 16, both algorithms are optimal to within a constant factor.

In previous sections we proved lower bounds of $\Omega(\sqrt{n})$ and $\Omega(\sqrt[4]{\text{OPT}})$ for AreaRotation. They can be summarized stating that AreaRotation has a competitive ratio of $\Omega(\max\{\sqrt{n}, \sqrt[4]{\text{OPT}}\})$. The last theorem of this section, describes the algorithm DynBoxRot$_{\sqrt{n}\wedge\sqrt[4]{\text{OPT}}}$ that simultaneously matches both lower bounds achieving a competitive ratio of $O(\min\{\sqrt{n}, \sqrt[4]{\text{OPT}}\})$. At a first sight it may seem that this algorithm contradicts the lower bound of $\Omega(\max\{\sqrt{n}, \sqrt[4]{\text{OPT}}\})$; however this simply proves that the *edge cases* that achieve a competitive ratio of at least $\Omega(\sqrt[4]{\text{OPT}})$ must satisfy $\text{OPT} = O(n^2)$. Likewise, those for which the competitive ratio is at least $\Omega(\sqrt{n})$ satisfy $n = O(\sqrt{\text{OPT}})$.

Translations only

Under the long edge assumption, we have $n \leq \text{OPT}$. Therefore, DynBoxTrans achieves a competitive ratio of $O(\sqrt{n}) = O(\sqrt{\text{OPT}})$ for AreaTranslation and matches the bound stated in Theorem 14.

Rotations allowed

Now we tackle the AreaRotation problem and describe the algorithm DynBoxRot$_{\sqrt[4]{\text{OPT}}}$. We define the threshold function $T_j = \Sigma_j^{3/4} + 7H_j$, where $H_j = \max_{i=1,\dots,j} h_i$ and Σ_j is the total area of pieces $p_1, \dots, p_j$. DynBoxRot$_{\sqrt[4]{\text{OPT}}}$ is obtained by running DynBoxRot, as described in Section 3.2, employing this new threshold T_j.

▶ **Theorem 17.** *The algorithm DynBoxRot$_{\sqrt[4]{\text{OPT}}}$ has a competitive ratio of $O(\sqrt[4]{\text{OPT}})$ for the problem AreaRotation, where Opt is the area of the optimal offline packing.*

Proof. This proof is similar to the one of Theorem 9. Define $W := \max_{i=1,\dots,n} w_i$. Recall that in DynBoxRot we preprocess every piece p rotating it so that $w_p \leq h_p$, hence $W \leq \sqrt{\Sigma_n}$. Let B_k be the last active box, so that we can enclose all the pieces in a bounding box of size $2^{k+1} \times T_n$, and bound the area returned by the algorithm as $\text{ALG} = O(2^k H_n + 2^k \Sigma_n^{3/4})$. On the other hand we are able to bound the optimal offline packing as $\text{OPT} = \Omega(\Sigma_n + W H_n)$.

If the active box never changed, then we have $2^k < 2W$ that implies $\text{ALG} = O(W H_n + \Sigma_n^{5/4}) = \text{OPT} \cdot O(\sqrt[4]{\text{OPT}})$. Otherwise, let B_ℓ be the last active box before B_k, and p_j be the first piece put in B_k. Here we have two cases.

Case (1) $w_j > 2^\ell$ In this case we have $2^k < 2W$ that implies $\text{ALG} = O(WH_n + \Sigma_n^{5/4}) = \text{OPT} \cdot O(\sqrt[4]{\text{OPT}})$.

Case (2) $w_j \leq 2^\ell$ In this case we have $k = \ell + 1$. Denote with $\widetilde{H_i}$ the total height of shelves in B_i. Then we have $\widetilde{H_\ell} \geq T_j - H_j = \Sigma_j^{3/4} + 6H_j$, otherwise we could pack p_j in B_ℓ. Thus, we can apply Lemma 8 and conclude that the box B_ℓ of size $2^\ell \times T_j$ is filled with constant density. Here we have two cases.

Case (2.1) $\widetilde{H_k} \leq T_j$ In this case we have $\text{ALG} = O(2^k T_j)$ and, thanks to the constant density packing of B_ℓ we have $\Sigma_j = \Theta(2^\ell \widetilde{H_\ell}) = \Theta(2^k T_j)$. Since $\text{OPT} \geq \Sigma_j$, we get $\text{ALG} = O(\text{OPT})$.

Case (2.2) $\widetilde{H_k} > T_j$ In this case we have $\text{ALG} = O(2^k \widetilde{H_k})$. Moreover, $\widetilde{H_k} = O(H_n + \Sigma_n/2^k)$, in fact if $2^{s-1} < H_n \leq 2^s$, then the total height of sparse shelves is $\sum_{i \leq s} 2^i = 2^{s+1} = O(H_n)$. Furthermore, dense shelves are filled with constant density, therefore their total height is at most $O(\Sigma_n/2^k)$. Finally, we need to show that $2^k = O(\sqrt[4]{\Sigma_n})$. Thanks to the constant density packing of B_ℓ, we have $2^k \Sigma_j^{3/4} = O(2^\ell T_j) = O(\Sigma_j)$. Dividing both sides by $\Sigma_j^{3/4}$ we get $2^k = O(\Sigma_j^{1/4})$. In the end notice that, thanks to the long edge hypotheses $H_n \leq \Sigma_n$ and we have $\text{ALG} = O(2^k \widetilde{H_k}) = O(2^k H_n + \Sigma_n) = O(\Sigma_n^{5/4}) = \text{OPT} \cdot O(\sqrt[4]{\text{OPT}})$. ◀

So far we managed to match the competitive ratio lower bounds of $\Omega(\sqrt{n})$ and $\Omega(\sqrt[4]{\text{OPT}})$ employing two different algorithms: DynBoxRot and DynBoxRot$_{\sqrt[4]{\text{OPT}}}$. A natural question is whether is it possible to match the performance of these algorithms simultaneously, having an algorithm that achieves a competitive ratio of $O(\min\{\sqrt{n}, \sqrt[4]{\text{OPT}}\})$. We give an affirmative answer by describing the algorithm DynBoxRot$_{\sqrt{n} \wedge \sqrt[4]{\text{OPT}}}$.

Again, we employ the same scheme of DynBoxRot with a different threshold function. This time the definition of T_j is slightly more involved:

$$T_j = \begin{cases} 0 & \text{if } j = 0 \\ \max\left\{T_{j-1}, \widetilde{T_j}\right\} & \text{if } j \geq 1 \end{cases} \quad \text{where} \quad \widetilde{T_j} = \begin{cases} \Sigma_j^{3/4} + 7H_j & \text{if } \Sigma_j < j^2 \\ H_j\sqrt{n} + 7H_j & \text{otherwise.} \end{cases}$$

▶ **Theorem 18.** *The algorithm DynBoxRot$_{\sqrt{n} \wedge \sqrt[4]{\text{OPT}}}$ achieves a competitive ratio of $O(\min\{\sqrt{n}, \sqrt[4]{\text{OPT}}\})$ on the problem AreaRotation, where OPT is the area of the optimal offline packing and n is the total number of pieces in the stream.*

Proof. This proof is similar to the one of Theorem 17 and is deferred to the full version. ◀

4 Further questions

It is natural to consider problems where the given pieces are more general, such as convex polygons. Here, we may allow the pieces to be rotated by arbitrary angles. In that case, it follows from the technique described by Alt [2] that one can obtain a constant competitive ratio for computing a packing with a minimum perimeter bounding box: For each new piece, we rotate the piece so that a diameter of the piece is horizontal. We then use the algorithm BrickRotation to pack the bounding boxes of the pieces. Since the area of each piece is at least half of the area of its bounding box, the density of the produced packing is at least half of the density of the packing of the bounding boxes. This results in an increase of the competitive ratio by a factor of at most $\sqrt{2}$.

For the problem of minimizing the perimeter of the bounding box (or convex hull) with convex polygons as pieces and only translations allowed, we do not know if it is possible to get a competitive ratio of $O(1)$, and this seems to be a very interesting question for future research. In order to design such an algorithm, it would be sufficient to show that for some constants $\delta > 0$ and $\Sigma > 0$, there is an online algorithm that packs any stream of convex polygons of diameter at most δ and total area at most Σ into the unit square, which is in itself an interesting problem. The three-dimensional version of this question has a negative answer, even for offline algorithms: Alt, Cheong, Park, and Scharf [3] showed that for any $n \in \mathbb{N}$, there exists a finite number of 2D unit disks embedded in 3D that cannot all be packed by translation in a cube with edges of length n.

References

1 Hee-Kap Ahn and Otfried Cheong. Aligning two convex figures to minimize area or perimeter. *Algorithmica*, 62(1-2):464–479, 2012.

2 Helmut Alt. Computational aspects of packing problems. *Bulletin of the EATCS*, 118, 2016.

3 Helmut Alt, Otfried Cheong, Ji-won Park, and Nadja Scharf. Packing 2D disks into a 3D container. In *International Workshop on Algorithms and Computation (WALCOM 2019)*, pages 369–380, 2019.

4 Helmut Alt, Mark de Berg, and Christian Knauer. Approximating minimum-area rectangular and convex containers for packing convex polygons. In *23rd Annual European Symposium on Algorithms (ESA 2015)*, pages 25–34, 2015.

5 Helmut Alt and Ferran Hurtado. Packing convex polygons into rectangular boxes. In *18th Japanese Conference on Discrete and Computational Geometry (JCDCGG 2000)*, pages 67–80, 2000.

6 Brenda S. Baker and Jerald S. Schwarz. Shelf algorithms for two-dimensional packing problems. *SIAM Journal on Computing*, 12(3):508–525, 1983.

7 Allan Borodin and Ran El-Yaniv. *Online computation and competitive analysis*. Cambridge University Press, 2005.

8 Brian Brubach. Improved bound for online square-into-square packing. In *12th International Workshop on Approximation and Online Algorithms (WAOA 2014)*, pages 47–58, 2014. doi:10.1007/978-3-319-18263-6_5.

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

10 Fan Chung and Ron Graham. Efficient packings of unit squares in a large square. *Discrete & Computational Geometry*, 2019.

11 János Csirik and Gerhard J. Woeginger. On-line packing and covering problems. In Amos Fiat and Gerhard J. Woeginger, editors, *Online Algorithms: The State of the Art*, pages 147–177. Springer, 1998. doi:10.1007/BFb0029568.

12 Paul Erdős and Ron Graham. On packing squares with equal squares. *Journal of Combinatorial Theory, Series A*, 19(1):119–123, 1975.

13 Sándor P. Fekete and Hella-Franziska Hoffmann. Online square-into-square packing. *Algorithmica*, 77(3):867–901, 2017. doi:10.1007/s00453-016-0114-2.

14 Amos Fiat and Gerhard J. Woeginger. Competitive analysis of algorithms. In Amos Fiat and Gerhard J. Woeginger, editors, *Online Algorithms: The State of the Art*, pages 1–12. Springer, 1998. doi:10.1007/BFb0029562.

15 Janusz Januszewski and Marek Lassak. On-line packing sequences of cubes in the unit cube. *Geometriae Dedicata*, 67(3):285–293, 1997.

16 Marek Lassak. On-line potato-sack algorithm efficient for packing into small boxes. *Periodica Mathematica Hungarica*, 34(1-2):105–110, 1997.

17 Hyun-Chan Lee and Tony C. Woo. Determining in linear time the minimum area convex hull of two polygons. *IIE Transactions*, 20(4):338–345, 1988. doi:10.1080/07408178808966189.

18 Boris D. Lubachevsky and Ronald L. Graham. Dense packings of congruent circles in rectangles with a variable aspect ratio. In Boris Aronov, Saugata Basu, János Pach, and Micha Sharir, editors, *Discrete and Computational Geometry: The Goodman-Pollack Festschrift*, pages 633–650. Springer, 2003.

19 Boris D. Lubachevsky and Ronald L. Graham. Minimum perimeter rectangles that enclose congruent non-overlapping circles. *Discrete Mathematics*, 309(8):1947–1962, 2009. `doi:10.1016/j.disc.2008.03.017`.

20 Victor J. Milenkovic. Translational polygon containment and minimal enclosure using linear programming based restriction. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of Computing (STOC 1996)*, pages 109–118, 1996.

21 Victor J. Milenkovic. Rotational polygon containment and minimum enclosure using only robust 2D constructions. *Computational Geometry*, 13(1):3–19, 1999. `doi:10.1016/S0925-7721(99)00006-1`.

22 Victor J. Milenkovic and Karen Daniels. Translational polygon containment and minimal enclosure using mathematical programming. *International Transactions in Operational Research*, 6(5):525–554, 1999. `doi:10.1111/j.1475-3995.1999.tb00171.x`.

23 Dongwoo Park, Sang Won Bae, Helmut Alt, and Hee-Kap Ahn. Bundling three convex polygons to minimize area or perimeter. *Computational Geometry*, 51:1–14, 2016. `doi:10.1016/j.comgeo.2015.10.003`.

24 E. Specht. High density packings of equal circles in rectangles with variable aspect ratio. *Computers & Operations Research*, 40(1):58–69, 2013. `doi:10.1016/j.cor.2012.05.011`.

25 Rob van Stee. SIGACT news online algorithms column 20: the power of harmony. *SIGACT News*, 43(2):127–136, 2012. `doi:10.1145/2261417.2261440`.

26 Rob van Stee. SIGACT news online algorithms column 26: Bin packing in multiple dimensions. *SIGACT News*, 46(2):105–112, 2015. `doi:10.1145/2789149.2789167`.