Algorithmica (2017) 77:867–901
DOI 10.1007/s00453-016-0114-2

Online Square-into-Square Packing

Sándor P. Fekete[1] · **Hella-Franziska Hoffmann**[2]

Received: 12 February 2014 / Accepted: 2 January 2016 / Published online: 11 January 2016

Abstract In 1967, Moon and Moser proved a tight bound on the critical density of squares in squares: any set of squares with a total area of at most 1/2 can be packed into a unit square, which is tight. The proof requires full knowledge of the set, as the algorithmic solution consists in sorting the objects by decreasing size, and packing them greedily into shelves. Since then, the online version of the problem has remained open; the best upper bound is still 1/2, while the currently best lower bound is 1/3, due to Han et al. (Theory Comput Syst 43(1):38–55, 2008). In this paper, we present a new lower bound of 11/32, based on a dynamic shelf allocation scheme, which may be interesting in itself. We also give results for the closely related problem in which the size of the square container is not fixed, but must be dynamically increased in order to accommodate online sequences of objects. For this variant, we establish an upper bound of 3/7 for the critical density, and a lower bound of 1/8. When aiming for accommodating an online sequence of squares, this corresponds to a 2.82...-competitive method for minimizing the required container size, and a lower bound of 1.33... for the achievable factor.

A preliminary extended abstract appears in the Proceedings of the 16th International Workshop APPROX 2013 [20]. Partially funded by DFG Grant FE407/17-1 within the DFG Research Unit 1800, "Controlling Concurrent Change".

✉ Sándor P. Fekete
s.fekete@tu-bs.de

Hella-Franziska Hoffmann
hrhoffmann@uwaterloo.ca

1 Department of Computer Science, TU Braunschweig, 38106 Braunschweig, Germany

2 David R. Cheriton School of Computer Science, University of Waterloo, 200 University Avenue West, Waterloo, ON N2L 3G1, Canada

Keywords Packing · Online problems · Packing squares · Critical density

1 Introduction

Packing is one of the most natural and common optimization problems. Given a set $\mathcal{O}$ of objects and a container E, find a placement of all objects into E, such that no two overlap. Packing problems are highly relevant in many practical applications, both in geometric and abstract settings. Simple one-dimensional variants (such as the PARTITION case with two containers, or the KNAPSACK problem of a largest packable subset) are NP-hard. Additional difficulties occur in higher dimensions: as Leung et al. [41] showed, it is NP-hard even to check whether a given set of squares fits into a unit-square container.

When dealing with an important, but difficult optimization problem, it is crucial to develop a wide array of efficient methods for distinguishing feasible instances from the infeasible ones. In one dimension, a trivial necessary and sufficient criterion is the total size of the objects in comparison to the container. This makes it natural to consider a similar approach for the two-dimensional version: *What is the largest number δ, such that any family of squares with area at most δ can be packed into a unit square?* An upper bound of $\delta \leq 1/2$ is trivial: two squares of size $1/2 + \varepsilon$ cannot be packed. As Moon and Moser [43] showed in 1967, $\delta = 1/2$ is the correct critical bound: sort the objects by decreasing size, and greedily pack them into a vertical stack of one-dimensional "shelves", i.e., horizontal subpackings whose height is defined by the largest object.

This approach cannot be used when the set of objects is not known a priori, i.e., in an online setting. It is not hard to see that a pure shelf-packing approach can be arbitrarily bad. However, other, more sophisticated approaches were able to prove lower bounds for δ: the current best bound (established by Han et al. [25]) is based on a relatively natural recursive approach and shows that $\delta \geq 1/3$.

Furthermore, it may not always be desirable (or possible) to assume a fixed container: the total area of objects may remain small, so a fixed large, square container may be wasteful. Thus, it is logical to consider the size of the container itself as an optimization parameter. Moreover, considering a possibly *larger* container reflects the natural optimization scenario in which the full set of objects *must* be accommodated, possibly by paying a price in the container size. From this perspective, $1/\sqrt{\delta}$ yields a competitive factor for the minimum size of the container, which is maintained at any stage of the process. This perspective has been studied extensively for the case of an infinite strip, but not for an adjustable square.

1.1 Our Results

We establish a new best lower bound of $\delta \geq 11/32$ for packing an online sequence of squares into a fixed square container, breaking through the threshold of $1/3$ that is natural for simple recursive approaches based on brick-like structures. Our result is based on a two-dimensional system of multi-directional shelves and buffers, which are dynamically allocated and updated. We believe that this approach is interesting in

itself, as it may not only yield worst-case estimates, but also provide a possible avenue for further improvements, and be useful as an algorithmic method.

As a second set of results, we establish the first upper and lower bounds for a square container, which is dynamically enlarged, but must maintain its quadratic shape. In particular, we show that there is an upper bound of $\delta \leq 3/7 < 1/2$ for the critical density, and a lower bound of $1/8 \leq \delta$; when focusing on the minimum size of a square container, these results correspond to a $2.82\ldots$-competitive factor, and a lower bound of $1.33\ldots$ for the achievable factor by any deterministic online algorithm.

1.2 Related Work

Two- and higher-dimensional problems of packing rectangular objects into rectangular containers have received a considerable amount of attention; see Harren's [27] Ph.D. thesis for a relatively recent survey. Many of the involved ideas are closely or loosely related to some of the ideas of our paper. We summarize many of the related papers, with particular attention dedicated to those that are of direct significance for our approach.

Offline Packing of Squares One of the earliest considered packing variants is the problem of finding a dense square packing for a rectangular container. In 1966 Moser [44] first stated the question as follows:

> What is the smallest number A such that any family of objects with total area at most 1 can be packed into a rectangle of area A?

The offline case has been widely studied since 1966; there is a long list of results for packing squares into a rectangle. Already in 1967, Moon and Moser [43] gave the first bounds for A: any set of squares with total area at most 1 can be packed into a square with side lengths $\sqrt{2}$, which shows $A \leq 2$, and thus $\delta \geq 1/2$; they also proved $A \geq 1.2$. Meir and Moser [42] showed that any family of squares each with side lengths $\leq x$ and total area A can be packed into a rectangle of width w and height h, if $w, h \geq x$ and $x^2 + (w - x)(h - x) \geq A$; they also proved that any family of k-dimensional cubes with side lengths $\leq x$ and total volume V can be packed into a rectangular parallelepiped with edge lengths $a_1, \ldots, a_k$ if $a_i \geq x$ for $i = 1, \ldots, k$ and $x^k + \prod_{i=1}^{k} (a_i - x) \geq V$. Kleitman and Krieger improved the upper bound on A to $\sqrt{3} \approx 1.733$ [39] and to $4/\sqrt{6} \approx 1.633$ [40] by showing that any finite family of squares with total area 1 can be packed into a rectangle of size $\sqrt{2} \times 2/\sqrt{3}$.

Novotný further improved the bounds to $1.244 \approx (2 + \sqrt{3})/3 \leq A < 1.53$ in 1995 [45] and 1996 [46]. The current best known upper bound of 1.3999 is due to Hougardy [30]. There is also a considerable number of other related work on offline packing squares, cubes, or hypercubes; see [15,26,33] for prominent examples.

Online Packing of Squares into a Square In 1997, Januszewski and Lassak [37] studied the online version of the dense packing problem. In particular, they proved that for $d \geq 5$, every online sequence of d-dimensional cubes of total volume $2(\frac{1}{2})^d$ can be packed into the unit cube. For lower dimensions, they established online methods for

packing (hyper-) cubes and squares with a total volume of at most $\frac{3}{2}(\frac{1}{2})^d$ and $\frac{5}{16}$ for $d \in \{3, 4\}$ and $d = 2$, respectively. The results are achieved by performing an online algorithm that subsequently divides the unit square into rectangles with aspect ratio $\sqrt{2}$. In the following, we call these rectangles *bricks*. The best known lower bound of $2(\frac{1}{2})^d$ for any $d \geq 1$ was presented by Meir and Moser [42].

Using a variant of the brick algorithm, Han et al. [25] extended the result to packing a 2-dimensional sequence with total area $\leq 1/3$ into the unit square.

A different kind of online square packing was considered by Fekete et al. [21,22]. The container is an unbounded strip, into which objects enter from above in a Tetris-like fashion; any new object must come to rest on a previously placed object, and the path to its final destination must be collision-free. Their best competitive factor is $34/13 \approx 2.6154\ldots$, which corresponds to an (asymptotic) packing density of $13/34 \approx 0.38\ldots$

Other Online Packing of Squares There are various ways to generalize online packing of squares; see Epstein and van Stee [17–19] for online bin packing variants in two and higher dimensions. In this context, also see parts of Zhang et al. [47].

Online Packing of Rectangles A natural generalization of online packing of squares is online packing of rectangles, which have also received a serious amount of attention. Most notably, online strip packing has been considered; for prominent examples, see Azar and Epstein [1], who employ shelf packing, and Epstein and van Stee [17].

Packing into One Container Offline packing of rectangles into a unit square or rectangle has also been considered in different variants; for examples, see [23], as well as [36]. Particularly interesting for methods for online packing into a single container may be the work by Bansal et al. [2], who show that for any complicated packing of rectangular items into a rectangular container, there is a simpler packing with almost the same value of items.

Two-Dimensional Bin Packing Packing squares or rectangles into a minimum number of square boxes amounts to two-dimensional bin packing, which is closely related to packing into a single container. Arguably, bin packing is the two-dimensional packing problem that has received the most attention from an algorithmic perspective. See [3–5,7–11,13,15,28,31,47] for particularly relevant work. Most of these papers consider offline problems, with notable exceptions already cited above.

Resource Augmentation Our study of online packing into a dynamic square container can be interpreted as a variant of resource augmentation, which has been studied in the context of two-dimensional packing by several other authors, including [12,14,24,34].

Strip Packing Dynamically expanding a square container (as presented in Sect. 3) can be seen as a variation of increasing a container along only one dimension, i.e., packing into a strip. Two- and higher-dimensional offline strip packing has been studied intensively, see [6,29,32,35,38] for prominent examples.

2 Packing into a Fixed Container

As noted in the introduction, it is relatively easy to achieve a dense packing of squares in an offline setting: sorting the items by decreasing size makes sure that a shelf-packing approach places squares of similar size together, so the loss of density remains relatively small. This line of attack is not available in an online setting; indeed, it is not hard to see that a brute-force shelf-packing method can be arbitrarily bad if the sequence of items consists of a limited number of medium-sized squares, followed by a large number of small ones. Allocating different size classes to different horizontal shelves is not a remedy, as we may end up saving space for squares that never appear, and run out of space for smaller squares in the process; on the other hand, fragmenting the space for large squares by placing small ones into it may be fatal when a large one does appear after all.

Previous approaches (in particular, the brick-packing algorithm) have side-stepped these difficulties by using a recursive subdivision scheme. While this leads to relatively good performance guarantees (such as the previous record of 1/3 for a competitive ratio), it seems impossible to tighten the lower bound; in particular, 1/3 seems to be a natural upper bound for this relatively direct approach. Thus, making progress on this natural and classical algorithmic problem requires less elegant, but more powerful tools.

In the following we present a different approach for overcoming the crucial impediment of mixed square sizes, and breaking through the barrier of 1/3. Our *Recursive Shelf Algorithm* aims at subdividing the set of squares into different size classes called *large*, *medium* and *small*, which are packed into pre-reserved shelves. The crucial challenge is to dynamically update regions when one of them gets filled up before the other ones do; in particular, we have to protect against the arrival of one large square, several medium-sized squares, or many small ones. To this end, we combine a number of new techniques:

- Initially, we assign carefully chosen horizontal strips for shelf-packing each size class.
- We provide rules for dynamically updating shelf space when required by the sequence of items. In particular, we accommodate a larger set of smaller squares by inserting additional *vertical* shelves into the space for larger squares whenever necessary.
- In order to achieve the desired overall density, we maintain a set of buffers for overflowing strips. These buffers can be used for different size classes, depending on the sequence of squares.

With the help of these techniques, and a careful analysis, we are able to establish $\delta \geq 11/32$. It should be noted that the development of this new technique may be more significant than the numerical improvement of the density bound: we are convinced that tightening the remaining gap towards the elusive 1/2 will be possible by an extended (but more complicated) case analysis.

The remainder of this section is organized as follows. In Sect. 2.1 we give an overview of the algorithm. Sect. 2.2 sketches the placement of large objects, while Sect. 2.3 describes the packing created with medium-sized squares. In Sect. 2.4 we

describe the general concept of shelf-packing that is used for the packing of small squares discussed in Sect. 2.5. The overall performance is analyzed in Sect. 2.6.

2.1 Algorithm Overview

We construct a shelf-based packing in the unit square by packing *small*, *medium* and *large squares* separately. We stop the Recursive Shelf Algorithm when the packings of two different subalgorithms would overlap. As it turns out, this can only happen when the total area of the given squares is $> 11/32$; details are provided in the Sect. 2.6, after describing the approach for individual size classes.

In the following, we will subdivide the set of possible squares into subsets, according to their size: we let H_k denote the height class belonging to the interval $(2^{-(k+1)}, 2^{-k}]$. In particular, we call all squares in H_0 *large*, all squares in H_1 *medium*, and all other squares (in $H_{\geq 2}$) *small*.

2.2 Packing Large Squares

The simplest packing subroutine is applied to large squares, i.e., of size $> 1/2$. We pack a square $Q_0 \in H_0$ into the top right corner of the unit square U. Clearly, only one large square can be part of a sequence with total area $\leq 11/32$. Hence, this single location for the squares in H_0 is sufficient.

2.3 Packing Medium Squares

We pack all medium squares (those with side lengths in $(1/4, 1/2]$) separately; note that there can be at most five of these squares, otherwise their total area is already bigger than $3/8 > 11/32$.

We start with packing the H_1-squares from left to right coinciding with the top of the unit square U. If a square would cross the right boundary of U, we continue by placing the following squares from top to bottom coinciding with the right boundary; see Fig. 1a.

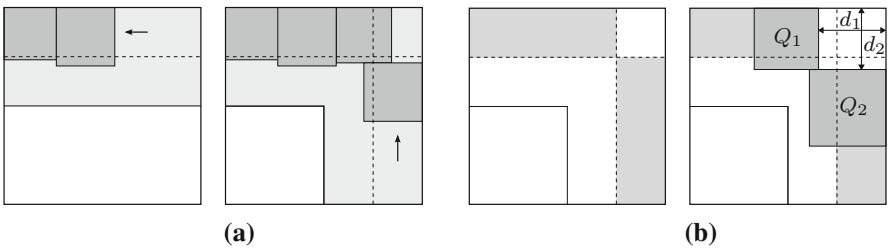


Fig. 1 Packing medium squares (Sect. 2.3). **a**: the L-shaped packing created with medium squares. **b** Density consideration: the Ceiling Packing Algorithm packs at least as much as the area of the *gray region* (R) shown on the *left*. If a portion of R remains uncovered by squares, a larger portion of $U \setminus R$ must be covered

We call the corresponding subroutine the *Ceiling Packing Algorithm*. Without interference of other height classes, the algorithm succeeds in packing any sequence of H_1-squares with total area $\leq 3/8$.

Theorem 1 *The Ceiling Packing Subroutine packs any sequence of medium squares with total area at most* $3/8$ *into the unit square.*

Proof Assume that the Ceiling Packing subroutine fails to pack a square Q. By construction, the algorithm successfully packs squares aligned with the top of U and the squares aligned with the right boundary of U until the space left at the bottom of U is too small to fit square Q. We prove that in this case the total area of the given sequence σ is greater than the area $\|R\| = 3/8$ of the gray region R depicted in Fig. 1b. The idea is that all of R is covered by packed squares except for potentially a small portion of it in the top right that can only be left uncovered as a result of receiving a large square that covers parts of $U \setminus R$. Let Q_2 with side length x_2 be the first square that was not packed aligned with the top boundary of U and let Q_1 with side length x_1 be the square packed aligned with the top of U that touches the top boundary of Q_2. Let d_1 be the distance of Q_1 to the right boundary of U and d_2 the distance of Q_2 to the top boundary of U. Then we have $d_1 < x_2$ and $d_2 = x_1$. Because all medium squares have a side length of at least $1/4$, we have $x_1^2 = 1/4x_1 + (x_1 - 1/4)x_1 \geq 1/4x_1 + (d_2 - 1/4) \times 1/4$ and $x_2^2 = 1/4x_2 + (x_2 - 1/4)x_2 > 1/4x_2 + (d_1 - 1/4) \times 1/4$. Furthermore, we get that the set σ_1 of all squares packed before Q_1 in σ has a total area of at least $1/4 \times (1 - d_1 - x_1)$, and that the set σ_2 of all squares that appeared after Q_2 in σ has a total area of at least $1/4 \times (1 - d_2 - x_2)$. Hence, we conclude

$$\begin{aligned}\|\sigma\| &\geq \|\sigma_1\| + \|Q_1\| + \|Q_2\| + \|\sigma_2\| \\ &> \frac{1}{4}(1 - d_1 - x_1) + \frac{1}{4}x_1 + \left(d_2 - \frac{1}{4}\right)\frac{1}{4} + \frac{1}{4}x_2 + \left(d_1 - \frac{1}{4}\right)\frac{1}{4} \\ &\quad + \frac{1}{4}(1 - d_2 - x_2) \\ &= \frac{1}{4}\left(1 - x_1 - d_1 + x_1 + d_2 - \frac{1}{4} + x_2 + d_1 - \frac{1}{4} + 1 - x_2 - d_2\right) = 3/8.\end{aligned}$$

2.4 Shelf Packing

In this section we revisit the well-known shelf-packing algorithm that is used for packing small squares into the unit square. Given a set of squares with maximum size h, a *shelf* $\mathcal{S}$ is a subrectangle of the container that has height h; the *Next Fit Shelf Algorithm NFS*($\mathcal{S}$) places incoming squares into $\mathcal{S}$ next to each other, until some object no longer fits; see Fig. 2a. When that happens, the shelf is closed, and a new shelf gets opened. Before we analyze the density of the resulting packing, we introduce some notation.

Notation In the following we call a shelf with height 2^{-k} designed to accommodate squares of height class H_k an *H_k-shelf*. We let $w_{\mathcal{S}}$ denote the width of a shelf $\mathcal{S}$,

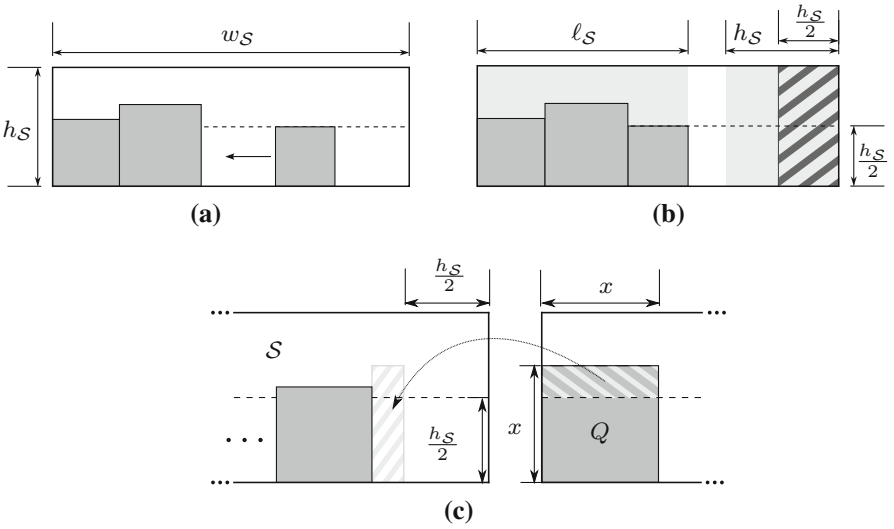


Fig. 2 **a** A shelf S packed by NFS(S) with squares of one height class. **b** Different areas of a shelf S. *occupied*(S): total area of squares in $\mathcal{P}(S)$ (*dark gray*), *usedSection*(S): region with *light gray background* (incl. *occupied*(S)) to the *left*, *head*(S): region with *light gray background* to the *right*, and *end*(S): hatched region to the *right*. **c** Assignment of *extra*(Q) (hatched) to S when square Q causes an overflow of shelf S

h_S denote its height and $\mathcal{P}(S)$ denote the set of squares packed into it. We define *usedSection*(S) as the horizontal section of S that contains $\mathcal{P}(S)$ and ℓ_S as its length; see Fig. 2b. We denote the last h_S-wide section at the end of S by *head*(S) and the last $h_S/2$-wide slice by *end*(S). The total area of the squares packed into a shelf S is *occupied*(S). The part of the square Q packed in the upper half of S is *extra*(Q).

A useful property of the shelf-packing algorithm is that *usedSection*(S) has a packing-density of $1/2$ if we pack S with squares of the same height class only. The gap remaining at the end of a closed shelf may vary depending on the sequence of squares. However, the following density property described in the following lemma (due to Moon and Moser [43]).

Lemma 1 *Let S be an H_k-shelf with width w_S and height h_S that is packed by $NFS(S)$ with a set $\mathcal{P}(S)$ of H_k-squares. Let Q be an additional square of H_k with side length x that does not fit into S. Then the total area $\|\mathcal{P}(S)\|$ of all squares packed into S plus the area $\|Q\|$ of Q is greater than $\|S\|/2 - (h_S/2)^2 + \frac{1}{2}h_S \times x$.*

In other words: if we count the extra area of the overflowing square Q towards the density of a closed shelf S, we can, w.l.o.g., assume that S has a packing density of $1/2$, except for at its end *end*(S). We formalize this charging scheme as follows. When a square Q causes a shelf S to be closed, we assign *extra*(Q) to S; see Fig. 2c. The total area assigned to S this way is referred to as *assigned*(S). Further, define $\widetilde{A}(S)$ as *occupied*(S) plus *assigned*(S) minus *extra*(Q) of all squares Q in S.

Corollary 1 *Let S be a closed shelf packed by the shelf-packing algorithm. Then $\widetilde{A}(S) \geq \|S \setminus end(S)\|/2$.*

Proof If the packing of $\mathcal{P}$ intersects with $end(\mathcal{S})$, then

$$\widetilde{A}(\mathcal{S}) \geq occupied(\mathcal{S}) - \sum_{Q' \in \mathcal{P}(\mathcal{S})} extra(Q') > h_{\mathcal{S}}/2 \times (w_{\mathcal{S}} - h_{\mathcal{S}}/2)\,.$$

Otherwise, square Q with side length x caused shelf $\mathcal{S}$ to be closed and we have:

$$\begin{aligned}
\widetilde{A}(\mathcal{S}) &= occupied(\mathcal{S}) - \sum_{Q' \in \mathcal{P}(\mathcal{S})} extra(Q') + extra(Q) \\
&= \frac{h_{\mathcal{S}}}{2} \times (w_{\mathcal{S}} - x) + x\left(x - \frac{h_{\mathcal{S}}}{2}\right) \\
&\geq \frac{h_{\mathcal{S}}}{2} \times (w_{\mathcal{S}} - x) + \frac{h_{\mathcal{S}}}{2}\left(x - \frac{h_{\mathcal{S}}}{2}\right) \\
&= \frac{h_{\mathcal{S}}\left(w_{\mathcal{S}} - \frac{h_{\mathcal{S}}}{2}\right)}{2}.
\end{aligned}$$

2.5 The *packSmall* Subroutine

As noted above, the presence of one large or few medium squares already assigns a majority of the required area, without causing too much fragmentation. Thus, the critical question is how to deal with small squares in a way that leaves space for larger ones, but allows us to find extra space for a continuing sequence of small squares.

We describe an algorithm for packing any family of H_k-squares with $k \geq 2$ and total area up to 11/32 in Sects. 2.5.1–2.5.4 and discuss the resulting packing density in Sects. 2.5.5–2.5.9. In Sect. 2.5.10 we describe mixed packing of small squares and analyze the corresponding density in Sects. 2.5.11 and 2.5.12.

2.5.1 The packSmall Algorithm: Overview and Notation

In the Recursive Shelf Algorithm we pack all small squares according to the *packSmall* subroutine, independent of the large and medium square packings. The method is based on the Next Fit Shelf (NFS) packing scheme described above. We first give a brief overview of the general distribution of the shelves and the order in which we allocate the shelves for the respective height classes.

Notation and Distribution of the Shelves The general partition of the unit square we use is depicted in Fig. 3a. The regions $M_1, \ldots, M_4$ (in that order) act as shelves for height class H_2. We call the union M of the M_i the *main packing area*; this is the part of U that will definitely be packed with squares by our packSmall subroutine. The other regions may stay empty, depending on the sequence of incoming small squares. The regions $B_1, \ldots, B_4$ provide shelves for H_3. We call the union B of the B_j the *buffer region*. In the region A we reserve H_k-shelf space for every $k \geq 4$. We call A the *initial buffer region*. The ends E_1, E_2 and E_3 of the main packing regions M_1, M_2 and

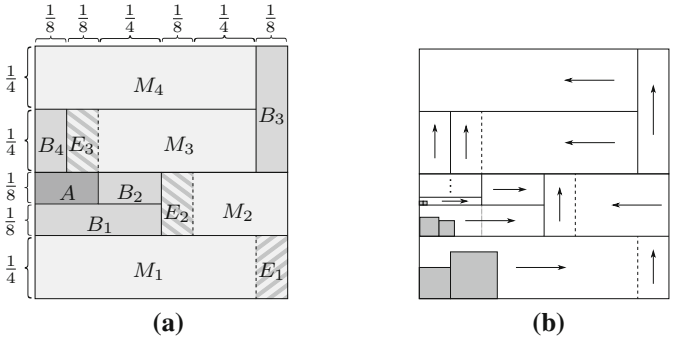


Fig. 3 **a** Distribution of the shelves for the *smallPack* Algorithm. **b** Initial shelf packing and packing directions

M_3 serve as both parts of the main packing region and additional buffer areas. We use $\bar{E}_i$ to refer to the vertical section of M_i that does not intersect with $usedSection(M_i)$.

Shelf Allocation Order During the packing process, we maintain open shelves for all the height classes for which we already received at least one square as input and pack each of them according to *NFS*. The order and location for the shelf allocation are chosen as follows.

- We start packing small squares into shelves that we open on the left side of the lower half $\mathcal{H}_\ell$ of U; see Fig. 3b. The region M_1 serves as the first H_2-shelf, the left half (width $1/4$) of B_1 serves as the first shelf for H_3 and region A is reserved for first shelves for any H_k with $k \geq 4$; see details below.
- Once an overflow occurs in a main packing region M_i, we close the corresponding H_2-shelf and continue packing H_2-squares into M_{i+1}.
- Once the packing in the initial shelf for H_k with $k \geq 3$ reaches a certain length, we cut a vertical slice $\mathcal{V}_k$ out of the currently open H_2-shelf (one of the M_i regions) and use $\mathcal{V}_k$ for the packing of subsequent H_k-squares.
- Once the packing in $\mathcal{V}_k$ reaches a certain height, we allocate space in the buffer region $B \cup E$ to accommodate H_k-squares before returning to pack $\mathcal{V}_k$.
- Once $\mathcal{V}_k$ is full, we cut another vertical slice out of the main packing region and repeat the process.

We claim that we can accommodate any family of small squares with total area up to $11/32$ this way. In the following, we describe the packings for the different small height classes in more detail.

2.5.2 The packSmall Algorithm: Separate Packing of H_2-Squares

In the main packing area, we always maintain an open shelf M_i for height class H_2, which is packed with H_2-squares according to $NFS(M_i)$. In order to avoid early collisions with large and medium squares, we start with packing M_1 from left to right, continuing with packing M_2 from right to left. Then we alternately treat M_3 and M_4 as the current main packing region, placing H_2-squares into the region whose *usedSection*

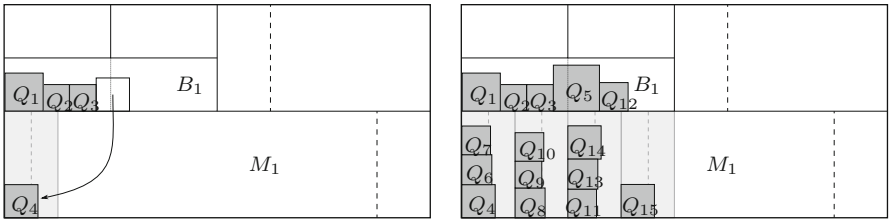


Fig. 4 A sample packing of H_3-squares in the lower half of U. **a** Initial packing and first vertical shelf. **b** Packing after three iterations of Steps 2 and 3

is smaller. When the length of *usedSection*(M_4) becomes larger than 3/8, we prefer M_3 over M_4 until M_3 is full.

2.5.3 The packSmall Algorithm: Separate Packing of H_3-Squares

For the packing of H_3-squares we alternate between using the buffer regions B_1, …, B_4, and vertical slices of width 1/8 cut out of the main packing region as the currently open H_3-shelf; see details below and Fig. 4 for an example.

The algorithm uses variables μ, β, ε_1, ε_2 and ε_3, which are used to quantify the growth of the packings in regions M, B, E_1 ,E_2, and E_3, respectively. In the algorithm we use a comparison of μ and $\beta + \sum_i \varepsilon_i$ to decide whether to place the next incoming square into the main packing region M or the buffer region $B \cup E$. Intuitively, we do this to ensure approximately proportional growth of the two regions (see Lemma 7), which in turn helps avoiding early collisions with large and medium squares. In addition, we use $\mathcal{V}_3$ to denote the (only) currently open vertical H_3-shelf in M. We define the algorithm *packSmall(3)* for H_3 as follows.

0. Set $\mu := 0$, $\beta := 0$, $\varepsilon_i := 0$ $\forall i$, $\mathcal{V}_3 := \emptyset$.
1. Open an H_3-shelf in B_1. Use NFS(B_1) to pack incoming H_3-squares Q and increase β by x_Q each time. Once $\beta + x_Q \geq \mu + 1/4$, for the next incoming square Q, set $\beta := \beta + 1/16 - x_Q$ and continue with Step 2.
2. Open a new vertical shelf $\mathcal{V}$ of width $\frac{1}{8}$ and height $\frac{1}{4}$ at the end of the packing in M. Set $\mathcal{V}_3 := \mathcal{V}$. Use NFS($\mathcal{V}_3$) (from bottom to top) to pack H_3-squares until the packing of the next square Q in $\mathcal{V}_3$ would intersect with *head*($\mathcal{V}_3$).
3. Increase μ by 1/16 and:
   (a) If $\beta + \sum_i \varepsilon_i + x_Q \geq \mu + 1/4$, pack Q into $\mathcal{V}_3$ and increase β by $x_Q - 1/16$.
   (b) Otherwise:
      (i) If there is an open end buffer shelf $\bar{E}_i$ for which M_i is closed, then
         – either pack Q into $\mathcal{V}_3$ and set $\varepsilon_i := \varepsilon_i + 1/16$ if $x_Q > 1/8 - \ell_i$ or
         – use NFS($\bar{E}_i$) to pack Q and set $\varepsilon_i := \varepsilon_i + x_Q$, otherwise.
      (ii) Otherwise, use NFS(B_1) to pack Q and increase β by x_Q.
4. Use NFS($\mathcal{V}_3$) to pack all following H_3-squares until $\mathcal{V}_3$ is full.
5. Repeat Steps 2–4 using regions $M_1, \ldots, M_4$ (in the same order and direction as for the H_2-square packing) for the placement of $\mathcal{V}_3$ in Step 2 and regions $B_1, \ldots, B_4, \mathcal{S}_3$ (in this order) for the placement of Q in Step 3(b)ii. If the algorithm

closes region M_i, set $\varepsilon_i := 2\ell_{E_i}$. If at any point in time $\varepsilon_i \geq 2/16$ or $\ell_{\bar{E}_i} \geq 2/16$, close $\bar{E}_i$ and set $\varepsilon_i := \max\{\varepsilon_i, 2/16\}$.

2.5.4 The packSmall Algorithm: Separate Packing of H_k-Squares with $k \geq 4$

For each H_k with $k \geq 4$, the packing algorithm *packSmall(k)* is defined as follows.

0. Set $\mu := 0$, $\beta := 0$, $\varepsilon_i := 0$ $\forall i$, $\mathcal{V}_k := \emptyset$, $\mathcal{B}_k := \emptyset$.
1. Open an H_k-shelf of length $1/4$ (and height 2^{-k}) on top of the existing shelves in A. Call this shelf $\mathcal{I}_k$. Use NFS($\mathcal{I}_k$) (from left to right) to pack incoming H_k-squares until $\mathcal{I}_k$ is full.
2. Open a vertical shelf $\mathcal{V}$ of width 2^{-k} and height $1/4$ at the end of the packing in M. Set $\mathcal{V}_k := \mathcal{V}$ and use NFS($\mathcal{V}_k$) (from bottom to top) to pack H_k-squares until the packing of the next square Q in $\mathcal{V}_k$ would intersect with $head(\mathcal{V}_k)$.
3. Increase μ by $2^{-k}/2$ and:
   (a) If $\beta + \sum_i \varepsilon_i \geq \mu + 3/16$, pack square Q into $\mathcal{V}_k$.
   (b) Otherwise:
      (i) If there is an open end buffer shelf $\bar{E}_i$ for which M_i is closed, then open a horizontal H_k-shelf $\mathcal{E}$ with width $1/8 - \ell_{\bar{E}_i}$ and height 2^{-k} on top of the current packing in $\bar{E}_i$, set $\mathcal{E}_k := \mathcal{E}$ and $\varepsilon_i := \varepsilon_i + 2^{-k}/2$. Use NFS($\mathcal{E}_k$) to pack incoming H_k-squares until $\mathcal{E}_k$ is full.
      (ii) Otherwise, open a vertical H_k-shelf $\mathcal{B}_k$ (with height $1/8$ and width 2^{-k}) at the end of the current packing in B_1, set $\beta := \beta + 2^{-k}$ and use NFS($\mathcal{B}_k$) to pack incoming H_k-squares until $\mathcal{B}_k$ is full.
4. Use NFS($\mathcal{V}_k$) to pack all following H_k-squares until $\mathcal{V}_k$ is full.
5. Repeat Steps 2–4 using regions $M_1, \ldots, M_4$ (in the same order and direction as for the H_2-square packing) for the placement of $\mathcal{V}_k$ in Step 2 and regions $B_1, \ldots, B_4$, $\mathcal{S}_4$ (in this order) for the placement of $\mathcal{B}_k$ in Step 3(b)ii. If the algorithm closes region M_i, set $\varepsilon_i := 2\ell_{E_i}$. If at any point in time $\varepsilon_i \geq 2/16$ or $\ell_{\bar{E}_i} \geq 2/16$, close $\bar{E}_i$ and set $\varepsilon_i := \max\{\varepsilon_i, 2/16\}$.

2.5.5 packSmall Analysis: Overview

In following sections we prove that the *packSmall* subroutine successfully packs any set of *small* squares with total area at most $11/32$.

In order to quantify the overall density achieved by the *packSmall* Algorithm, we make some simplifying assumptions on the density reached in the respective shelves. We argue that low-density shelves only appear along with high-density regions and define a charging scheme that assigns extra areas from dense regions to sparse regions in order to estimate the overall density. More precisely, we prove the following important invariant for our algorithm, which is essential for the overall density analysis in the case of mixed packings; see Sect. 2.5.11.

Property 1 *In any step of the algorithm, the total area of the small squares packed into U is at least $\|usedSection(M) \setminus E\|/2$.*

We start the density analysis by introducing notation, simplifying assumptions and general packing properties in Sect. 2.5.6. We proceed with analyzing the case of separately packing a set of only H_k-squares. In Sects. 2.5.7, 2.5.8 and 2.5.9, we discuss the cases $k = 2$, $k = 3$ and $k \geq 4$, respectively. We describe and analyze the case of packing a mixed sequence of small squares in Sects. 2.5.10 and 2.5.11 and conclude with a presentation of additional density properties in Sect. 2.5.12.

2.5.6 packSmall Analysis: Preliminaries

For the analysis of the density achieved with the packing of small squares, we use the following notation:

$$
\begin{aligned}
V_X^k &:= \text{set of all vertical } H_k\text{-shelves in region } X \\
V_{\text{open}} &:= \text{set of all open vertical shelves} \\
V_{\text{closed}} &:= \text{set of all closed vertical shelves} \\
V_{\text{head}}^k &:= \text{set of all } \mathcal{V} \in V_M^k \text{ for which we executed Step 3} \\
&\quad \text{in the } \textit{smallPack}(k) \text{ subroutine after opening } \mathcal{V} \\
V_{\text{head}} &:= \bigcup_{k \in K} V_{\text{head}}^k \\
V_{\text{head}}^{\text{B}} &:= \text{set of all } \mathcal{V} \in V_{\text{head}} \text{ for which we placed a square in } B \text{ (Step 3(b)ii)} \\
V_{\text{head}}^{\text{E}} &:= \text{set of all } \mathcal{V} \in V_{\text{head}} \text{ for which we placed a square in } E \text{ (Step 3(b)i)} \\
V_{k \geq 4} &:= \text{set of all } H_{k \geq 4}\text{-shelves} \\
K &:= \text{set containing all } k \text{ for which } V_M^k \neq \emptyset \\
\beta, \varepsilon_i, \mu, \mathcal{V}_k, \mathcal{B}_k &:= \text{variables used in the algorithm (see above)} \\
e &:= \text{the index of the end buffer region } \bar{E}_i \text{ that was closed last} \\
\ell_X &:= \text{total length of} \textit{usedSection}(X) \text{(see Sect. 2.4)} \\
\widetilde{\mathcal{A}}(\mathcal{S}) &:= \textit{occupied}(\mathcal{S}) + \textit{assigned}(\mathcal{S}) - \sum_{Q \in \mathcal{S}} \textit{extra}(Q) \text{ for shelf} \mathcal{S}
\end{aligned}
$$

To make similar simplifying density considerations as in Corollary 1, we define the following charging scheme that assigns area from high-density regions to low-density regions.

Charging Scheme

I: From each square Q that causes an overflow in a shelf $\mathcal{S}$ assign $\textit{extra}(Q)$ to $\mathcal{S}$.
II: From each H_3-square Q that was packed into $\mathcal{V}_3$ in Step 3(a), assign $\textit{extra}(Q)$ to the buffer region B_i.
III: From each H_3-square Q that was packed into $\mathcal{V}_3$ in Step 3(b)i, assign $\textit{extra}(Q)$ to the buffer region $\bar{E}_i$.

In the following we use this charging scheme for the definition of $\textit{assigned}(\mathcal{S})$ and assume, w.l.o.g. that $\ell_{\mathcal{S}} \geq w_{\mathcal{S} \setminus \textit{end}(\mathcal{S})}$ for any closed shelf $\mathcal{S}$ that is packed by NFS($\mathcal{S}$).

Because we only charge squares with their extra area and we do not charge any squares twice, we know that $\widetilde{\mathcal{A}}(\mathcal{S})$ is a lower bound on the actual density of $\mathcal{S}$. In the remainder of this section, we prove some general packing properties, which we use in subsequent density considerations.

Lemma 2 *Let $\mathcal{V}$ be an H_k-shelf in V_{closed}, then $\widetilde{\mathcal{A}}(\mathcal{V}) \geq \|\mathcal{V}\|/2 - (w_{\mathcal{V}}/2)^2$.*

Proof The claim follows directly with Corollary 1 and the fact that each vertical H_k-shelf $\mathcal{V}$ is packed (vertically) by NFS($\mathcal{S}$) with H_k-squares only. □

Lemma 3 *Let $\mathcal{V}$ be an H_k-shelf in V_{open}, then $\widetilde{\mathcal{A}}(\mathcal{V}) \geq (w_{\mathcal{V}}/2)^2$.*

Proof The claim follows directly from the fact that, by construction, each open vertical shelf contains at least one square of size at least $w_{\mathcal{V}_o}/2$. □

Lemma 4 *The total area $\sum_{\mathcal{V}\in V_M^k} \widetilde{\mathcal{A}}(\mathcal{V})$ of vertical H_k-shelves in the main packing area M is greater or equal to $\sum_{\mathcal{V}\in V_M^k} \|\mathcal{V}\|/2 - \sum_{\mathcal{V}\in V_M^k \cap V_{closed}} (w_{\mathcal{V}}/2)^2 - \frac{1}{4} \times 2^{-k}/2 + \widetilde{\mathcal{A}}(\mathcal{V}_k)$.*

Proof By construction, we always close the vertical H_k-shelf $\mathcal{V}_k$ before opening a new one. Thus, $\mathcal{V}_k$ is the only vertical H_k-shelf in M that is open and the claim follows with Lemma 2 and the fact that $\|\mathcal{V}_k\| = 1/4 \times 2^{-k}$. □

Lemma 5 *For any H_k-shelf $\mathcal{S}$ packed by NFS($\mathcal{S}$), it holds $\widetilde{\mathcal{A}}(\mathcal{S}) \geq h_{\mathcal{S}}\ell_{\mathcal{S}}/2$.*

Proof The claim follows directly from the fact that *usedSection*($\mathcal{S}$) is packed with H_k-squares only, which all have size at least $h_{\mathcal{S}}/2$.

Lemma 6 *For all $k \geq 4$, we have*

$$\widetilde{\mathcal{A}}(\mathcal{V}_k) + \widetilde{\mathcal{A}}(\mathcal{B}_k) \geq \begin{cases} \frac{1}{4} \times \frac{2^{-k}}{2} - \left(\frac{2^{-k}}{2}\right)^2 & \text{if } \mathcal{V}_k \in V_{head} \text{ and } \mathcal{B}_k \in V_{closed} \\ \frac{1}{8} \times \frac{2^{-k}}{2} & \text{otherwise} \end{cases}$$

Proof If $\mathcal{B}_k$ is closed, then $\widetilde{\mathcal{A}}(\mathcal{B}_k) \geq (1/8 - 2^{-k}/2) \times 2^{-k}/2)$ by Corollary 1. Otherwise, $\widetilde{\mathcal{A}}(\mathcal{B}_k) \geq 2^{-k}/2$ because $\mathcal{B}_k$ contains at least one H_k-square. We only execute Step 3 of the algorithm if the next square Q would intersect with *head*($\mathcal{V}_k$) when placed in $\mathcal{V}_k$. Thus, $\widetilde{\mathcal{A}}(\mathcal{B}_k) \geq (1/4 - 2^{-k} - 2^{-k}) \times 2^{-k}/2$ if $\mathcal{V} \in V_{\text{head}}$. If $\mathcal{V} \notin V_{\text{head}}$, then $\widetilde{\mathcal{A}}(\mathcal{B}_k) \geq (\frac{2^{-k}}{2})^2$ and $\mathcal{B}_k$ must be closed.

Lemma 7 *After each step in the algorithm it holds $\beta + \sum_i \varepsilon_i \geq \mu + 3/16$.*

Proof To simplify the notation define $\varepsilon := \sum_i \varepsilon_i$. Initially, we have $\mu = \beta = \varepsilon = 0$. Now consider the execution of any step in the algorithm that could change any of these values.

Step 1: The variable values only change if $k = 3$. If $\beta^{\text{old}} + x_Q \geq \mu^{\text{old}} + 1/4$, we have $\beta^{\text{new}} + \varepsilon^{\text{new}} = \beta^{\text{old}} + x_Q - 1/16 + \varepsilon^{\text{old}} \geq \mu^{\text{old}} + 1/4 - x_Q + x_Q - 1/16 = \mu^{\text{new}} + 3/16$. Otherwise, $\Delta\beta = x_Q$ and $\Delta\mu = \Delta\varepsilon = 0$.

Step 2: Nothing changes.
Step 3: Independent of k, we have $\mu^{\text{new}} = \mu^{old} + 2^{-k}/2$.
Subcase (a) & $k = 3$: This step is only executed if $\beta^{\text{old}} + \varepsilon^{\text{old}} + x_Q \geq \mu^{\text{new}} + 1/4$, which implies $\beta^{\text{new}} + \varepsilon^{\text{new}} = \beta^{old} + x_Q - 1/16 + \varepsilon^{old} \geq \mu^{\text{new}} + 1/4 + x_Q - 1/16 - x_Q = \mu^{\text{new}} + 3/16$
Subcase (a) & $k \geq 4$: We have $\Delta\beta = \Delta\varepsilon = 0$ and $\beta^{\text{old}} + \varepsilon^{\text{old}} \geq \mu^{\text{new}} + 1/4$.
Subcase (b) & $k = 3$: We either have $\Delta\beta = x_Q$ and $\Delta\varepsilon = 2^{-k}/2$ (subcase i), or $\Delta\beta = x_Q \geq 2^{-k}/2$ and $\Delta\varepsilon = 0$ (subcase ii).
Subcase (b) & $k \geq 4$: We either have $\Delta\beta = 0$ and $\Delta\varepsilon = 2^{-k}/2$ (subcase i), or $\Delta\beta = 2^{-k}$ and $\Delta\varepsilon = 0$ (subcase ii).
Step 4: Nothing changes.
Step 5: We only increase the left hand side of the equation.

The inequality holds in either of the cases and the claim follows by induction. □

Lemma 8 *In each step of the algorithm, we have* $assigned(B) \geq 1/16 \times (\beta - \ell_B)$.

Proof We give a proof by induction over the changes of $assigned(B)$, β and ℓ_B. Initially, we have $\beta = assigned(B) = \ell_B = 0$. If $k = 3$, then $\Delta\beta = (x_Q - \frac{1}{16})$, $\Delta\ell_B = 0$ and $\Delta assigned(B) = x_Q(x_Q - \frac{1}{16}) \geq \frac{1}{16}(\Delta\beta - \Delta\ell_B)$ in Step 3(a) and $\Delta assigned(B) = 0$ and $\Delta\beta = \Delta\ell_B = x_Q$ in Steps 1 and 3(b). If $k \geq 4$, then $\Delta\beta = \Delta\ell_B = 2^{-k}$ in Step 3(b). In all other cases, $\Delta assigned(B) \geq 0 = \Delta\beta = \Delta\ell_B$.□

Lemma 9 *In each step of the algorithm:* $\mu = \sum_{\mathcal{V} \in V_{head}} \frac{w_{\mathcal{V}}}{2}$ *and* $V_{closed} \subseteq V_{head}$.

Proof By construction, we increase μ by $\frac{2^{-k}}{2} = \frac{w_{\mathcal{V}}}{2}$ each time we execute Step 3 for the packing of H_k-squares. In addition, we only close the currently open vertical main packing shelf $\mathcal{V}_k$ (Step 4) after executing Step 3, which proves the claim. □

2.5.7 packSmall Analysis: Overall Density of Separate H_2-Square Packing

Lemma 10 *The algorithm successfully packs any sequence of* H_2*-squares with total area at most* 11/32.

Proof Using the Next Fit Shelf Algorithm NFS(M), the packing explicitly allocates a position for each incoming H_2-square until on overflow occurs in M_4. In that case, we have $\|\mathcal{P}\| + \|Q\| > 1/4 \times w_{M\setminus E} \times 1/2 = 1/8 \times (7/8 + 3/8 + 5/8 + 7/8) = 22/64$ by Corollary 1, which contradicts $\|\mathcal{P}\| \leq 11/32$. Thus, the algorithm successfully packs all incoming H_2-squares. □

2.5.8 packSmall Analysis: Overall Density of Separate H_3-Square Packing

In this subsection we analyze the overall packing density for the special case of packing a sequence of squares that all belong to height class H_3.

Lemma 11 *If the input sequence contains only* H_3*-squares, then* $\|\mathcal{P}\| \geq \frac{1}{8}\ell_{M\setminus E}$ *after each step of the algorithm.*

Proof By construction, Section M only contains vertical H_3-shelves. Thus, we have $\ell_M = \sum_{\mathcal{V}\in V_M^3} w_{\mathcal{V}}$ and with Lemma 4 we get

$$\widetilde{\mathcal{A}}(M) \geq \sum_{\mathcal{V}\in V_M^3} \widetilde{\mathcal{A}}(\mathcal{V}) \geq \frac{1}{4}\ell_M/2 - \sum_{\mathcal{V}\in V_M^3\cap V_{\text{closed}}} (w_{\mathcal{V}}/2)^2 - \frac{1}{16}\times\frac{1}{4} + \widetilde{\mathcal{A}}(\mathcal{V}_3) \quad (1)$$

With Lemmas 5 and 8, we get

$$\widetilde{\mathcal{A}}(B) \geq \mathit{assigned}(B)+\mathit{occupied}(B)-\mathit{extra}(B) \geq \frac{1}{16}\times(\beta-\ell_B)+\frac{1}{16}\ell_B = \frac{1}{16}\beta \quad (2)$$

By construction, we have $\varepsilon_i = 2\ell_{E_i}\ \forall i \leq e$ and $w_{\mathcal{V}}/2 = 1/16$ Thus, by combining Eqs. 1 and 2 and applying Lemma 7, we get

$$\begin{aligned}\|\mathcal{P}\| \geq \widetilde{\mathcal{A}}(M)+\widetilde{\mathcal{A}}(B) &\geq \frac{1}{8}\ell_M+\frac{1}{16}\left(\beta-\frac{3}{16}\right)-\sum_{\mathcal{V}\in V_M^3\cap V_{\text{closed}}}\left(\frac{w_{\mathcal{V}}}{2}\right)^2-\frac{1}{16^2}+\widetilde{\mathcal{A}}(\mathcal{V}_3)\\ &\geq \frac{1}{8}\ell_{M\setminus E}+\frac{1}{16}\left(\mu-\sum_{\mathcal{V}\in V_M^3\cap V_{\text{closed}}}\frac{w_{\mathcal{V}}}{2}\right)-\frac{1}{16^2}+\widetilde{\mathcal{A}}(\mathcal{V}_3)\end{aligned}$$

The claim follows with Lemmas 9 and 11. □

Lemma 12 *The algorithm successfully packs any sequence of H_3-squares with total area at most* 11/32.

Proof The algorithm explicitly assigns an unoccupied space to the next incoming square or vertical shelf until an overflow occurs in M_4 or B_4. If an overflow occurred in M_4, we would have $\ell_{M_4} = w_{M_4}$ and M_1, M_2 and M_3 are closed. Thus, $\|\mathcal{P}\|+\|Q\| > 1/8 \times w_{M\setminus E} = 22/64$ by Lemma 11, which contradicts $\|\mathcal{P}\| \leq 11/32$. Assume we could not fit a square Q into B_4 in Step 3(b)ii of the algorithm, then $\ell_{B_4}+x_Q > 1/4$ and B_1, B_2 and B_3 are closed. Thus, $\beta+x_Q \geq \sum_i \ell_{B_i} > (7+3+7+4)/16 = 21/16$. However, we only execute Step 3(b)ii if $\beta+x_Q < \mu-\varepsilon+1/4 = \ell_{M\setminus E}/2+\ell_E/2-\sum_{i=1}^{e} 2\ell_{E_i}+1/4 < 20/16$, which is a contradiction. Hence, the algorithm successfully packs any sequence of H_2-squares. □

2.5.9 packSmall Analysis: Overall Density of Separate $H_{k\geq 4}$-Square Packing

In this subsection we analyze the overall packing density for the special case of packing a sequence of squares that all belong to height class H_k for a fixed $k \geq 4$.

Lemma 13 *If the input sequence contains only H_k-squares with $k \geq 4$, then $\|\mathcal{P}\| \geq \frac{1}{8}\ell_{M\setminus E}$ after each step of the algorithm.*

Proof By the same reasoning as for Eq. 3 in Lemma 11 we have

$$\widetilde{\mathcal{A}}(M) \geq \sum_{\mathcal{V}\in V_M^k} \widetilde{\mathcal{A}}(\mathcal{V}) \geq \frac{1}{4}\ell_M/2 - \sum_{\mathcal{V}\in V_M^k \cap V_{\text{closed}}} (w_{\mathcal{V}}/2)^2 - \frac{1}{4}\times\frac{2^{-k}}{2} + \widetilde{\mathcal{A}}(\mathcal{V}_k) \quad (3)$$

By construction, we have

$$|V_{\text{head}}^{\text{B}} \cap V_M^k| = |V_B^k|, \text{ and } w_{\mathcal{V}} = w_{\mathcal{B}}\ \forall \mathcal{V} \in V_{\text{head}}^{\text{B}} \cap V_M^k,\ \mathcal{B} \in V_B^k. \quad (4)$$

Because we maintain at most one open vertical H_k-buffer-shelf ($\mathcal{B}_k$) at all times, the following equation follows with Lemma 2 and Eq. 4.

$$\begin{aligned}\sum_{\mathcal{B}\in V_B^k} \widetilde{\mathcal{A}}(\mathcal{B}) &\geq \widetilde{\mathcal{A}}(\mathcal{B}_k) + \sum_{\mathcal{B}\in V_B^k\setminus\{\mathcal{B}_k\}} \|\mathcal{B}\|/2 - (w_{\mathcal{B}}/2)^2 \\ &\geq \widetilde{\mathcal{A}}(\mathcal{B}_k) - \frac{1}{16}\times\frac{2^{-k}}{2} - \left(\frac{2^{-k}}{2}\right)^2 \\ &\quad + \sum_{\mathcal{V}\in V_{\text{head}}^{\text{B}}} \left(\frac{1}{16}\times(w_{\mathcal{V}}/2) + (w_{\mathcal{V}}/2)^2\right) \end{aligned} \quad (5)$$

Because Section B only contains vertical H_k-shelves and $\beta = \ell_B = \sum_{\mathcal{V}\in V_{\text{head}}^{\text{B}}} w_{\mathcal{V}}$:

$$\widetilde{\mathcal{A}}(B) \geq \widetilde{\mathcal{A}}(\mathcal{B}_k) - \frac{1}{16}\times\frac{2^{-k}}{2} - \left(\frac{2^{-k}}{2}\right)^2 + \frac{1}{16}\times\beta/2 + \sum_{\mathcal{V}\in V_{\text{head}}^{\text{B}}} (w_{\mathcal{V}}/2)^2 \quad (6)$$

Section A contains exactly one horizontal H_k-shelf $\mathcal{I}_k$, which is closed before packing H_k-squares into M. Thus, if M contains at least one vertical H_k-shelf, we have

$$\widetilde{\mathcal{A}}(A) = \widetilde{\mathcal{A}}(\mathcal{I}_k) \geq \left(1/4 - 2^{-k}/2\right)\times 2^{-k}/2. \quad (7)$$

By combining Eqs. 3, 6 and 7 and applying Lemma 7, we get

$$\begin{aligned}\|\mathcal{P}\| &\geq \widetilde{\mathcal{A}}(M) + \widetilde{\mathcal{A}}(B) + \widetilde{\mathcal{A}}(A) \\ &\geq \frac{1}{8}\ell_M - \sum_{\mathcal{V}\in V_M^k\cap V_{\text{closed}}} (w_{\mathcal{V}}/2)^2 + \sum_{\mathcal{V}\in V_{\text{head}}^{\text{B}}} (w_{\mathcal{V}}/2)^2 + \widetilde{\mathcal{A}}(\mathcal{B}_k) + \widetilde{\mathcal{A}}(\mathcal{V}_k) \\ &\quad + \frac{1}{16}\left(\beta - \sum_{\mathcal{V}\in V_{\text{head}}^{\text{B}}\cap V_M^k} (w_{\mathcal{V}}/2)\right) - \frac{1}{16}\times\frac{2^{-k}}{2} - 2\left(\frac{2^{-k}}{2}\right)^2\end{aligned}$$

$$\geq \frac{1}{8}\ell_{M\setminus E} + \frac{1}{16}\left(\mu - \sum_{\mathcal{V}\in V_M^k \cap V_{\text{closed}}} \frac{w_{\mathcal{V}}}{2}\right)$$
$$+ \widetilde{\mathcal{A}}(\mathcal{B}_k) + \widetilde{\mathcal{A}}(\mathcal{V}_k) - \frac{1}{16} \times \frac{2^{-k}}{2} - 2\left(\frac{2^{-k}}{2}\right)^2$$

The claim follows with Lemmas 6 and 9. □

Lemma 14 *The algorithm successfully packs any sequence of H_k-squares with $k \geq 4$ and total area at most* 11/32.

Proof By the same reasoning as in the proof of Lemma 12, no overflow occurs in M as long as $\|\sigma\| \leq 11/32$. Assume we could not fit a vertical H_k-shelf into B_4, then $\ell_{B_4} + 2^{-k} > 1/4$ and B_1, B_2 and B_3 are closed. Thus, $\beta \geq \sum_i \ell_{B_i} > (7 + 3 + 7 + 4)/16 - 2^{-k} = 21.5/16$. However, for $k \geq 4$, we only execute Step 3(b)ii if $\beta < \mu - \varepsilon + 3/16 = \ell_{M\setminus E}/2 + \ell_E/2 - \sum_{i=1}^{e} 2\ell_{E_i} + 1/4 < 20/16$, which is a contradiction. Hence, the algorithm successfully packs any sequence of H_k-squares.□

2.5.10 The packSmall Algorithm: Mixed Packing of Small Squares

In this section we describe the packing created by the *packSmall* Algorithm for the case that the input sequence contains a mixed set of *small* squares.

When receiving squares of different small height classes, not much changes. We allocate shelves and fill them by placing squares (or vertical subshelves) at the end of their *used sections* according to the *packSmall(k)* algorithm given in Sects. 2.5.2–2.5.4 for each class separately. Once we receive a first square of height class H_k, we simply start running *packSmall(k)* in parallel to the other *packSmall* subroutines. For all subsequent H_k-squares Q in the input, we simply perform the next step in *packSmall(k)* to pack Q. The variables μ, β and ϵ_i become shared variables. They are initialized once to 0 in a global Step 0 and are then modified by each of the different subroutines as described above (Steps 1–5). The resulting packing differs from the separate packings in the following two ways.

- The main packing regions M and buffer regions $B \cup \bar{E}$ may now contain vertical shelves from a variety of height classes; see Fig. 5.
- Because the vertical shelves for $H_{k\geq 3}$ do not fit as nicely into M as is the case in the separate packings, gaps may remain at the end of the packing in each M_i.

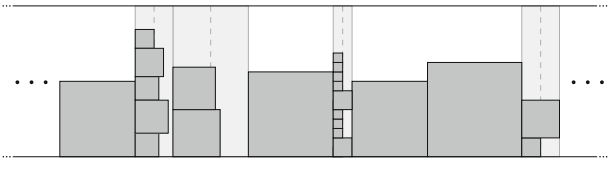


Fig. 5 Sample packing of squares and vertical subshelves for mixed small height classes

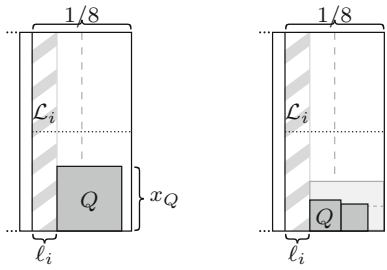


Fig. 6 The buffer packing performed in the end buffer regions: (*left*) packing of a fitting H_3-square and (*right*) subshelves packing of $H_{\geq 4}$ squares

The algorithm uses these gaps for the placement of buffer squares and horizontal buffer shelves; see Fig. 6 and Step 3(b)i of the algorithm.

2.5.11 packSmall Analysis: Density of Mixed Small Square Packing

In this section we analyze the overall density achieved by the *packSmall* algorithm for any input sequence of small squares.

Lemma 15 *After each step of the packSmall Algorithm we have* $\|\mathcal{P}_s\| \geq \frac{1}{8}\ell_{M\setminus E} + \frac{1}{16} \times \sum_{\mathcal{V}\in V_{open}\cap V_{head}} \frac{w_{\mathcal{V}}}{2} + \sum_{k\in K}\left(\widetilde{\mathcal{A}}(\mathcal{V}_k) - \left(\frac{2^{-k}}{2}\right)^2\right) + \sum_{k\in K\setminus 3}\left(\widetilde{\mathcal{A}}(\mathcal{B}_k) - \left(\frac{1}{8} - \frac{2^{-k}}{2}\right)\frac{2^{-k}}{2}\right)$

Proof By construction, M only contains H_2 squares and vertical $H_{k\geq 3}$-shelves. Thus, $\widetilde{\mathcal{A}}(M) \geq \frac{1}{8}\ell_M - \sum_{\mathcal{V}\in V_M}\|\mathcal{V}\| + \sum_{\mathcal{V}\in V_M}\widetilde{\mathcal{A}}(\mathcal{V})$ and with Lemma 4 we get

$$\widetilde{\mathcal{A}}(M) \geq \frac{1}{8}\ell_M - \sum_{\mathcal{V}\in V_M\cap V_{\text{closed}}}(w_{\mathcal{V}}/2)^2 + \sum_{k\in K}\left(\widetilde{\mathcal{A}}(\mathcal{V}_k) - \frac{1}{4}\times\frac{2^{-k}}{2}\right)$$

Analogously, we have $\widetilde{\mathcal{A}}(B) \geq \frac{1}{16}\ell_B - \sum_{\mathcal{B}\in V_B}\|\mathcal{B}\| + \sum_{\mathcal{B}\in V_B}\widetilde{\mathcal{A}}(\mathcal{B})$, which together with Lemma 8, Eqs. 4 and 5 and $w_{\mathcal{V}}/2 = 1/16$ for $\mathcal{V}\in V_M^3$ implies

$$\widetilde{\mathcal{A}}(B) \geq \frac{1}{16}\left(\beta - \sum_{\mathcal{V}\in V_{\text{head}}^{\text{B}}}\frac{w_{\mathcal{V}}}{2}\right) + \sum_{\mathcal{V}\in V_{\text{head}}^{\text{B}}}\left(\frac{w_{\mathcal{V}}}{2}\right)^2 + \sum_{k\in K\setminus 3}\left(\widetilde{\mathcal{A}}(\mathcal{B}_k) - \left(\frac{1}{8} - \frac{2^{-k}}{2}\right)\frac{2^{-k}}{2}\right)$$

Let $\bar{\varepsilon}_i$ be the amount assigned to ε_i in the first subcase of 3(b)i in *smallPack(3)*. By construction, we have $\mathit{assigned}(\bar{E}_i) = (1/16 - \ell_{E_i})\bar{\varepsilon}_i$ and

$$\varepsilon_i = \begin{cases} \bar{\varepsilon}_i + \ell_{\bar{E}_i} - \sum_{\mathcal{V}\in V_{\text{head}}^{\text{E}_i}\cap V_{k\geq 4}} w_{\mathcal{V}}/2 & \text{if } \bar{E}_i \text{ is open} \\ \max\{2/16, 2/16\ell_{E_i}\} & \text{if } \bar{E}_i \text{ is closed} \end{cases} \tag{8}$$

If $\ell_{E_i} \geq 1/16$, we have $V_{\text{head}}^{\text{E}_i} = \emptyset$ and $\varepsilon_i := 2\ell_{E_i}$. Otherwise, $\widetilde{\mathcal{A}}(\bar{E}_i) \geq \frac{1}{16}\ell_{\bar{E}_i} - \sum_{\mathcal{E}\in V_{\bar{E}}} \|\mathcal{E}\| + \sum_{\mathcal{E}\in V_{\bar{E}_i}} \widetilde{\mathcal{A}}(\mathcal{E})$ and with the same reasoning as for Eq. 5 we get

$$\mathit{occupied}(\bar{E}_i) \geq \frac{1}{16}\left(\ell_{\bar{E}_i} - \sum_{\mathcal{V}\in V_{\text{head}}^{\text{E}_i}} \frac{w_\mathcal{V}}{2}\right) + \sum_{\mathcal{V}\in V_{\text{head}}^{\text{E}_i}} \left(\frac{w_\mathcal{V}}{2}\right)^2 - \ell_{E_i} \sum_{\mathcal{V}\in V_{\text{head}}^{\text{E}_i}\cap V_{k\geq 4}} \frac{w_\mathcal{V}}{2}$$

By combining the above equations with $\bar{\varepsilon}_i \leq 2/16$ and $\ell_{E_i} < 1/16$, we get

$$\widetilde{\mathcal{A}}(\bar{E}_i) \geq \frac{1}{16}\left(\varepsilon_i - 2\ell_{E_i} - \sum_{\mathcal{V}\in V_{\text{head}}^{\text{E}_i}\cap V_{k\geq 4}} \frac{w_\mathcal{V}}{2}\right) + \sum_{\mathcal{V}\in V_{\text{head}}^{\text{E}_i}} \left(\frac{w_\mathcal{V}}{2}\right)^2$$

By the same reasoning as for Eq. 7, we get

$$\widetilde{\mathcal{A}}(A) \geq \sum_{k\in K} \widetilde{\mathcal{A}}(\mathcal{I}_k) \geq \sum_{k\in K} (1/4 - 2^{-k}/2) \times 2^{-k}/2$$

The claim follows with $\|\mathcal{P}_s\| \geq \widetilde{\mathcal{A}}(M) + \widetilde{\mathcal{A}}(B) + \widetilde{\mathcal{A}}(\bar{E}) + \widetilde{\mathcal{A}}(A)$ and Lemmas 7 and 9. □

Theorem 2 *The packSmall Algorithm packs any sequence of small squares with total area at most* 11/32 *into the unit square.*

Proof Let Q be the next incoming H_k-square. We consider all possible cases in which the algorithm does not explicitly assign an unoccupied space to Q.

1. Assume Q causes an overflow in M_4. Then either $k = 2$ and $\|Q\| > 1/8 w_Q$ or $k \geq 3$, $\ell_{M\setminus E} + 2^{-k} > 22/16$ and $\widetilde{\mathcal{A}}(\mathcal{V}_k) + \|Q\| > \|\mathcal{V}_k\|/2 = 1/8 \times 2^{-k}$ by construction.
2. Assume the algorithm cannot open a new vertical buffer shelf $\mathcal{B}$ for H_k with $k \geq 5$ in B. Then $\beta + 2^{-k} > 21/16$ and $\ell_{M\setminus E}/2 = \mu - \ell_E/2 > \beta + \varepsilon - \ell_E/2 - 3/16 > 21/16 - 2^{-k} + 4.5/16 - 3/16 = 22.5/16 - w_{\mathcal{V}_k}$.
3. Assume Q causes an overflow in $\mathcal{V}_3$ or $\mathcal{V}_4$. Then, by construction, $\mathcal{V}_k \in V_{\text{open}} \cap V_{\text{head}}$, $\widetilde{\mathcal{A}}(\mathcal{V}_k) + \|Q\| > 1/8 \times w_{\mathcal{V}_k} \geq 1/8 \times 0.5/16$, all B_i regions must have been closed and we have $\beta \geq \sum_i \ell_{B_i} \geq 20/16$. With Lemma 7, we have $\ell_{M\setminus E}/2 = \mu - \ell_E/2 > \beta + \varepsilon - \ell_E/2 - 1/4 > 21.5/16$.

In either of the three cases we get $\|\sigma\| > 11/32$ with Lemmas 3, 6 and 15, a contradiction. Thus, the algorithm successfully packs any sequence of small squares. □

2.5.12 packSmall Analysis: Some Additional Properties

Before we analyze the algorithms performance in the presence of large and medium squares, we state a couple of important properties of the packing created with small squares.

Recall that we use variable β to quantify the growth of the buffer packing. By construction, we can relate the length of the buffer region and the total area of the input as follows.

Lemma 16 *Let Q be a small square with side length x_Q in the buffer region B and let $\mathcal{P}_s$ be the set of small squares received so far. Then the total area of the small input squares $\|\mathcal{P}_s\|$ is greater than $(\beta + \frac{1.5}{16}e + x_Q - 1/16) \times 1/4$.*

Proof By construction, we only pack an H_3-square into B if $\beta + 2\ell_E + \varepsilon + x_Q < \mu + 1/4$. For $Q \in H_k$ with $k \geq 4$ we have $x_Q < 1/16$ and we only extend the buffer packing if $\beta + 2\ell_E + \varepsilon < \mu + 3/16$. In either case we get $\beta + 2\ell_E + \varepsilon + x_Q - 1/16 < \mu$. Recall that μ is defined as the total width of the vertical shelves in M. Thus, with Lemmas 7, 9, and 15 and Eq. 8 we get

$$\begin{aligned}\|\mathcal{P}_s\| \geq 1/8\ell_{M\setminus E} &\geq (\mu - \sum_{\mathcal{V}\in V_E} w_{\mathcal{V}}/2) \times 1/4 \\ &> (\beta + \varepsilon + x_Q - 1/16 - \ell_E/2) \times 1/4\end{aligned}$$

□

As a direct implication of Lemma 16 and the fact that when B_4 is first used for buffer square placement in step 3(b), both end buffer regions $\bar{E}_1$ and $\bar{E}_2$ have successfully been closed by the algorithm before, we get the following lower bounds for the total area $\|\mathcal{P}_s\|$ of small squares packed, as a function of the total length of the packing in B.

Property 2 *Let Q be a small square in the buffer region B_2 with side length x and distance $d > 1/4$ to the left boundary of U, then $\|\mathcal{P}_s\| > (d + x - 1/16) \times 1/4$.*

Property 3 *If there is a small square in B_3, Then $\|\mathcal{P}_s\| > 7/64$.*

Property 4 *Let Q be a small square in B_3 with side length x that was packed in a distance $d > 0$ to the bottom of B_3. Then $\|\mathcal{P}_s\| > (7.5/16 + d + x) \times 1/4$.*

Property 5 *If there is a small square in B_4, then $\|\mathcal{P}_s\| > 17/64$.*

Property 6 *Let Q be a small square in B_4 with side length x and distance $d > 0$ from the bottom of B_4. Then $\|\mathcal{P}_s\| > (1 + d + x) \times 1/4$.*

The following properties follow directly from the algorithm invariant of Property 1.

Property 7 *When the first small square is packed into M_2, then $\|\mathcal{P}_s\| \geq 7/64$.*

Property 8 *When the first small square is packed into M_3, then $\widetilde{A}(\mathcal{H}_\ell) \geq 10/64$.*

2.6 Combined Analysis

In the previous sections we proved that the algorithm successfully packs small, medium and large squares separately, as long as input has a total area of at most 11/32. A case distinction over all possible collisions that may appear between the packings of these height classes can be used to prove the main result.

Theorem 3 *The Recursive Shelf Algorithm packs any sequence of squares with total area at most* 11/32 *into the unit square.*

We prove the claim by showing that if the algorithm fails to pack a square, the total area of the given squares must exceed 11/32. In the following we analyze the packing density at the time a collision of the different packing subroutines would appear. First we consider a collision between a medium and a small square in the upper half $\mathcal{H}_u$ of the unit square container.

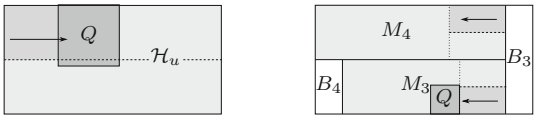
Lemma 17 *If a medium square Q_1 collides with a part of the packing constructed with small squares, then $\|Q_1\| + \widetilde{A}(\mathcal{H}_u) \geq 6/32$.*

Proof Recall that we pack the medium sized squares from left to right aligned with the top boundary of $\mathcal{H}_u$; see Fig. 7a. The packing of small squares (into M_3 and M_4) is performed from right to left; see Fig. 7b. Also recall that we alternatingly use M_3 and M_4 as the current main packing region (choosing which ever half is less full in width) until the packing in M_4 reaches a total length at least 3/8. Then we only pack M_3 until it is completely filled, before finishing the packing in M_4.

Let Q_1 be a medium square that collides with a small square in the upper half $\mathcal{H}_u$ of the unit bin. Then Q_1 either intersects a vertical shelf $\mathcal{S}$ or an H_2-square Q_2. The main idea is to prove that the parts of $\mathcal{H}_u \setminus B_3$ both right and left to $Q_2/\mathcal{S}$ have a density of 1/2. We distinguish six different cases depending on the location of $\mathcal{S}$ or Q_2 in $\mathcal{H}_u$.

1. *Q_1 collides with an H_2-square Q_2 in M_4*:
   We know $\ell_{M_3} > \ell_{M_4} - w_{\mathcal{S}}$, as otherwise we would have packed Q_2 in M_3. Therefore, the entire part of $M_3 \cup M_4$ to the right of Q_2 must be used by small squares, thus having a density of 1/2. Additionally, the section used by the H_1-squares must be filled to a height of at least 1/4. Hence, we know that the sections of $\mathcal{H}_u \setminus B_3$ both right and left to Q_2 are half full; see Fig. 8a. Therefore, with $x_2 \geq 1/8$,

$$\begin{aligned}\widetilde{A}(\mathcal{H}_u) &> \frac{(7/8 - x_2) \times 1/2}{2} + x_2^2 \\ &\geq \frac{7}{32} + x_2\left(x_2 - \frac{1}{4}\right) \\ &\geq \frac{7}{32} + \frac{1}{8} \times \left(\frac{1}{8} - \frac{1}{4}\right) > \frac{6}{32}.\end{aligned}$$



(a) **(b)**

Fig. 7 Packing performed in the *upper half* of U. The feasible packing area has *light gray background*, the medium *gray part* represents the packing created before Q was placed. **a** Packing created with medium squares. **b** Packing created with small squares

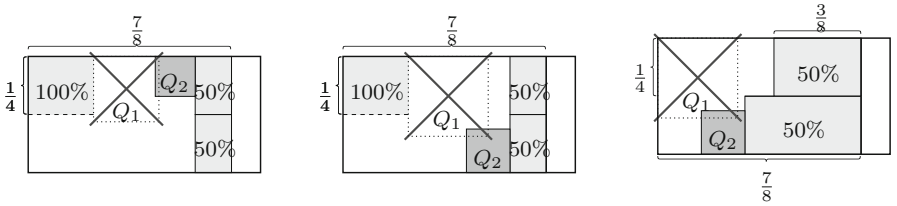


Fig. 8 Collision of a medium square Q_1 with an H_2-square Q_2 in $\mathcal{H}_u$

2. *Q_1 collides with an H_2-square Q_2 in M_3*:
   By construction we have $\ell_{M_3} - x_2 \leq \ell_{M_4}$ or $\ell_{M_4} \geq 3/8$, as we packed Q_2 into M_3 instead of M_4.
   (a) If $\ell_{M_3} - x_2 \leq \ell_{M_4}$, the situation is symmetric to the one in the previous case; see Fig. 8b. Because Q_1 is aligned with the top and Q_2 with the bottom of $\mathcal{H}_u$ and Q_1 and Q_2 collide, we have $x_1 + x_2 > 1/2$. Thus, we get

$$\begin{aligned}\widetilde{\mathcal{A}}(\mathcal{H}_u) &> \frac{(7/8 - x_1 - x_2) \times 1/2}{2} + x_1^2 + x_2^2 \\ &\geq \frac{7}{32} - \frac{x_1 + x_2}{4} + \frac{(x_1 + x_2)^2}{2} \\ &> \frac{7}{32} - \frac{(x_1 + x_2) \times 1/2}{2} + \frac{(x_1 + x_2) \times 1/2}{2} > \frac{6}{32}.\end{aligned}$$

   (b) Otherwise, $\ell_{M_3} - x_2 > \ell_{M_4} \geq 3/8$; see Fig. 8c. Again, we know $x_1 + x_2 > 1/2$. Thus,

$$\begin{aligned}\widetilde{\mathcal{A}}(\mathcal{H}_u) &\geq x_1^2 + x_2^2 + \frac{\|usedSection(M_3)\|}{2} + \frac{\|usedSection(M_4)\|}{2} \\ &> \frac{(x_1 + x_2)^2}{2} + 2 \times \frac{\|usedSection(M_4)\|}{2} \\ &> \frac{(1/2)^2}{2} + \frac{3}{8} \times \frac{1}{4} > \frac{6}{32}.\end{aligned}$$

3. *Q_1 collides with a vertical shelf $\mathcal{S}$ in M_4*:
   Analogously to the first case, we know that the sections of $\mathcal{H}_u \setminus B_4$ both right and left to $\mathcal{S}$ must be half full; see Fig. 9a. Thus, with $w_{\mathcal{S}} \leq 1/8$ we get:

$$\begin{aligned}\widetilde{\mathcal{A}}(\mathcal{H}_u) &> \frac{(7/8 - w_{\mathcal{S}}) \times 1/2}{2} + \frac{\|\mathcal{S}\|}{2} \\ &\geq \frac{7}{32} - \frac{w_{\mathcal{S}}}{4} \geq \frac{6}{32}.\end{aligned}$$

4. *Q_1 collides with a vertical shelf $\mathcal{S}$ in M_3*:
   Analogously to the second case, we must have $\ell_{M_3} - w_{\mathcal{S}} \leq \ell_{M_4}$ or $\ell_{M_4} \geq 3/8$ as we opened $\mathcal{S}$ in M_3.

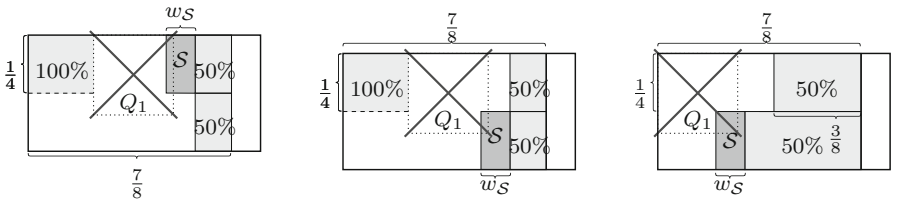


Fig. 9 Collision of a medium square Q_1 with vertical shelf $\mathcal{S}$ in $\mathcal{H}_u$

(a) If $\ell_{M_3} - w_{\mathcal{S}} \leq \ell_{M_4}$, then we have the same conditions as described in the first case; see Fig. 9b. We analogously get

$$\widetilde{\mathcal{A}}(\mathcal{H}_u) > \left(\frac{7}{8} - w_{\mathcal{S}}\right)\frac{1}{4} \geq \frac{6}{32}.$$

(b) Otherwise, $\ell_{M_3} - w_{\mathcal{S}} > \ell_{M_4} \geq 3/8$. Let ℓ^1 be the length of the $\mathcal{H}_u$-section used by H_1. We know $\ell^1 \geq 1/4$ and $\ell_{M_3} > 7/8 - \ell^1$; see Fig. 9b. Thus,

$$\begin{aligned}\widetilde{\mathcal{A}}(\mathcal{H}_u) &\geq \ell^1 \times \frac{1}{4} + \frac{\|usedSection(M_3)\|}{2} + \frac{\|usedSection(M_4)\|}{2} \\ &> \ell^1 \times \frac{1}{4} + \frac{(7/8 - \ell^1) \times 1/4}{2} + \frac{3/8 \times 1/4}{2} \\ &\geq \frac{\ell^1}{8} + \frac{7}{64} + \frac{3}{64} \geq \frac{6}{32}.\end{aligned}$$

□

We are now able to prove Theorem 3.

Proof (of Theorem 3) Let Q be the square at which the algorithm stops. Denote σ the set of all input squares and $\mathcal{P}$ the set of all squares packed at the time Q arrives. We claim $\|Q\| + \|\mathcal{P}\| > 11/32$. To prove this statement we distinguish the different types of collisions that might cause the algorithm to stop with failure. Note that we covered the cases in which σ consists of either all large, all medium or all small squares in the previous sections. In the following we denote $\mathcal{P}_s$ the set of all small squares in $\mathcal{P}$ and $\mathcal{P}_m$ the set of all medium squares in $\mathcal{P}$.

1. *A large square Q_0 collides with a medium square Q_1*:
   In this case, the first (and only) square of H_0 collides with the L-shaped packing produced by the Ceiling Algorithm; see Fig. 10b. We know $\|Q_0\| > (1/2)^2 = 1/4$ and the shelf packing for the H_1-squares must reach from the left boundary to more than a distance of x_0 from the right boundary. Thus, as $x_i \geq 1/4$ for any square $Q_i \in H_1$, the total area of the input sequence $\|\sigma\|$ is at least $\|\mathcal{P}\| + \|Q_0\| > x_0^2 + (1 - x_0) \times \frac{1}{4} \geq 3/8 > 11/32$.
2. *A large square Q_0 collides with a small square Q_s*:

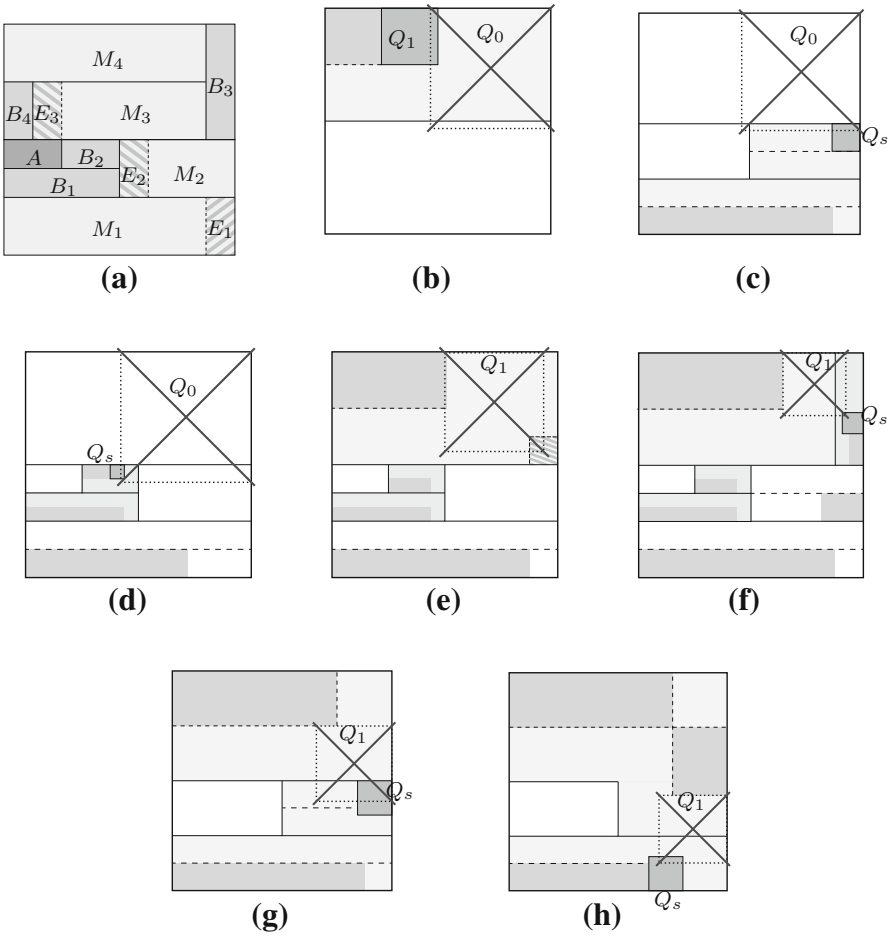


Fig. 10 Different types of collision that may appear if the total area of the input exceeds $11/32$. **a** Packing region denotations. **b** A large square Q_0 only collides with a medium square Q_1 if $\|\sigma\| > 3/8$. **c** If a large square collides with a small square in M_2, then the total area of small squares is at least $3/32$. **d** We can relate the total area of small squares packed to the total length of the buffer packing. **e** If a medium square collides with the packing in B_3, then we packed at least $7/64$ with small squares. **f** The greater the distance of Q_s to the bottom of B_3, the more area of $\mathcal{H}_\ell$ we packed with small squares. **g** The total area of medium squares is $> 1/4$, the total area of small squares is at least $7/64$. **h** The total area of medium squares is $> 5/16$, the total area of small squares is at least $1/16$

If the side length x_0 of Q_0 is greater than $\sqrt{11/32}$, then $\|Q_0\| > 11/32$ and we are done. Therefore, we assume $x_0 \leq \sqrt{11/32} < 5/8$. There are two cases:

(a) *Q_s is in the main packing area*:
Because $x_0 < 5/8 < 3/4$, Q_s must have been packed into M_2, M_3 or M_4. In any case, a small square must be in M_2 and by Property 7 we have a total area of more than $7/64$ from small squares. Additionally, we have $x_0 \geq 1/2$, as Q_0 is large. That is, $\|\sigma\| \geq \|Q_0\| + \|\mathcal{P}_s\| > (1/2)^2 + 7/64 = 23/64 > 11/32$; Fig. 10c.

(b) *Q_s is in the buffer area*:
We have that Q_s is not in B_1 since $x_0 < 5/8$. If Q_s is located in B_3 or B_4, then $\|\mathcal{P}_s\| > 7/64$ according to Property 3 and $\|\sigma\| > (1/2)^2 + 7/64 > 11/32$. Otherwise, Q_s is located in B_2. Let d be the distance of Q_s to the left boundary of the unit square. As Q_0 and Q_s collide, we have $d + x_s + x_0 > 1$; see Fig. 10d. We distinguish two cases for the side length of Q_0:

(i) $x_0 \in (1/2, 9/16)$: Then $d + x_s > 1 - x_0 > 7/16$, which implies $d > 7/16 - 1/8 > 1/4$. Thus, by Property 2 we get

$$\|\mathcal{P}_s\| > \frac{d + x_s - 1/16}{4} > \frac{7/16 - 1/16}{4} = 6/64$$

(ii) $x_0 \in [9/16, 5/8)$: Then $d + x_s > 1 - x_0 > 3/8$, which implies $d > 3/8 - 1/8 = 1/4$. Thus, by Property 2 we get

$$\|\mathcal{P}_s\| > \frac{d + x_s - 1/16}{4} > \frac{3/8 - 1/16}{4} = 5/64$$

In total we get
$\|\sigma\| \geq \|Q_0\| + \|\mathcal{P}_s\| > \min\{(1/2)^2 + 6/64, (9/16)^2 + 5/64\} = 11/32$.

3. *A medium square Q_1 collides with a small square Q_s*:
There are many different types of collisions that might appear between the small square packing and a square of H_1. Note that the ceiling packing never interacts with the buffer area of $\mathcal{H}_\ell$, but might interact with the buffers in $\mathcal{H}_u$.

(a) *Q_s is (a buffer square) in B_3*:
Recall that all medium squares are packed from left to right aligned with the top boundary of U. Let d be the distance of Q_s to the lower boundary of B_3. We distinguish two cases for $x_s + d$:

(i) $x_s + d \leq 1/8$: Then Q_1 intersects the 1/8-high section at the bottom of B_3; see Fig. 10e. That is, either an overflow of H_1-squares occurred in $\mathcal{H}_u$ and we have $\|\mathcal{P}_m\| > 1 \times 1/4$, or Q_1 coincides with the top of U, which implies $x_1 > 3/8$, and we have

$$\|\mathcal{P}_m\| > x_1^2 + \left(\frac{7}{8} - x_1\right) \times \frac{1}{4} \geq \frac{7}{32} + x_1 \times \left(x_1 - \frac{1}{4}\right) \geq \frac{14}{64} + \frac{3}{8} \times \frac{1}{8} = \frac{17}{64} > \frac{1}{4}.$$

As Q_s is in B_3 we get $\|\mathcal{P}_s\| > 7/64$ by Property 3. In both cases, we get

$$\|\sigma\| \geq \|\mathcal{P}_m\| + \|\mathcal{P}_s\| > 1/4 + 7/64 = 11/32.$$

(ii) $x_s + d > 1/8$: This case is depicted in Fig. 10f. By Property 4 we get

$$\|\mathcal{P}_s\| > \frac{7.5/16 + d + x_s}{4} > \frac{7.5/16 + 2/16}{4} = \frac{9.5}{64}.$$

Because Q_1 intersects with B_3, we know that the total length of the medium square packing is $> 7/8$. Thus we get

$$\|\sigma\| \geq \|\mathcal{P}_m\| + \|\mathcal{P}_s\| > 7/8 \times 1/4 + 9.5/64 > 23.5/64 > 11/32.$$

(b) *Q_s is (a buffer square) in B_4*:
Recall that we start packing medium squares coinciding with the top of U. We fill the buffer region B_4 from bottom to top. Let d be the distance of Q_s to the lower boundary of B_4. We distinguish two cases for $x_s + d$:

(i) $x_s + d \leq 1/8$: Then Q_1 perturbs the 1/8-high section at the bottom of B_4, i.e. we have $x_1 > 3/8$. Because Q_s is in B_4, we get $\|\mathcal{P}_s\| > 17/64$ by Property 5. Thus, in total we have

$$\|\sigma\| \geq \|Q_1\| + \|\mathcal{P}_s\| > (3/8)^2 + 17/64 = 26/64 > 11/32.$$

(ii) $x_s + d > 1/8$: By Property 6 we have

$$\|\mathcal{P}_s\| > \frac{1 + d + x_s}{4} > \frac{1 + 1/8}{4} = \frac{9}{32}.$$

Because Q_1 is a medium square, we have $x \geq 1/4$ and get

$$\|\sigma\| \geq \|Q_1\| + \|\mathcal{P}_s\| > (1/4)^2 + 9/32 = 11/32.$$

(c) *Q_s is packed into M_3 or M_4*:
By Property 8 and Lemma 17 we have $\widetilde{\mathcal{A}}(\mathcal{H}_\ell) \geq 5/32$ and $\widetilde{\mathcal{A}}(\mathcal{H}_u) \geq \widetilde{\mathcal{A}}(M_3 \setminus E_i \cup M_4) \geq 6/32$, respectively. Therefore, $\widetilde{\mathcal{A}}(U) \geq 11/32$.

(d) *Q_s is a buffer square in E_i*:
We start treating the end E_i of a main packing area M_i only if $M_i \setminus E_i$ is fully *used*. Therefore, this type of collision can be handled analogously to the collision of Q_0 with a square in M_i.

(e) *Q_1 overlaps with M_2 but not with M_1*:
This only happens if Q_1 provokes an overflow in the upper half of U and is therefore packed into the second shelf of the Ceiling Packing; see Fig. 10g. In this case, the total area of squares from H_1 is $> 1/4$. By assumption, Q_s is placed in M_2 and we get an additional packing area of at least 7/64 from small squares; see Property 7. In total, we have $\|\sigma\| \geq \|\mathcal{P}_m\| + \|\mathcal{P}_s\| > 1/4 + 7/64 > 11/32$.

(f) *Q_1 overlaps with M_1*:
Because Q_1 intersects M_1, the lower boundary of Q_1 must have a distance $> 3/4$ from the top of U; see Fig. 10h. Hence, $\|\mathcal{P}_m\| > 1/4 \times (3/4 + 2/4) = 10/32$. As no H_1-square ever touches the left half of $\mathcal{H}_\ell$, we must have an area of at least $1/2 \times 1/8 = 2/32$ occupied by small squares. In total we get $\|\sigma\| \geq \|\mathcal{P}_m\| + \|\mathcal{P}_s\| > 10/32 + 2/32 > 11/32$. □

This concludes the proof of the main Theorem 3.

3 Packing into a Dynamic Container

Now we discuss the problem of online packing a sequence of squares into a dynamic square container. At each stage, the container must be large enough to accommodate all objects; this requires keeping the container tight early on, but may require increasing its edge length appropriately during the process.

In the following, we give a non-trivial family of instances, which prove that no online algorithm can maintain a packing density $> 3/7$ for an arbitrary input sequence of squares and introduce an online square packing algorithm that maintains a packing density of $1/8$ for an arbitrarily input sequence of squares.

3.1 An Upper Bound on δ

If the total area of the given sequence is unknown in advance, the problem of finding a dense online packing becomes harder. As it turns out, a density of $1/2$ cannot be achieved.

Theorem 4 *There are sequences for which no deterministic online packing algorithm can maintain a density strictly* $> 3/7 \approx 0.4286$.

Proof We construct an appropriate sequence of squares, depending on what choices a deterministic player makes; see Fig. 11. At each stage, the player must place a square Q_3 into a corner position (Fig. 11a) or into a center position (Fig. 11b); the adversary responds by either requesting another square of the same size (a), or two of the size of the current spanning box. This is repeated.

If the player keeps choosing corner positions, the density δ_i for the enclosing square of size x_i satisfies the recursion $\delta_{i+1} = 1/4 \times \delta_i + 1/2$, as shown in Fig. 11d. The sequence is decreasing and bounded from below, so solving the equation $\delta_\infty = 1/4 \times \delta_\infty + 1/2$ yields $\lim_{i\to\infty} \delta_i = \delta_\infty = 2/3$. If the player keeps choosing center positions, the density δ_i for the enclosing square of size x_i satisfies the recursion $\delta_{i+1} = 1/9 \times \delta_i + 2/3$, as shown in Fig. 11e. This sequence is also decreasing and bounded from below, so solving the equation $\delta_\infty = 1/9 \times \delta_\infty + 2/3$ yields $\lim_{i\to\infty} \delta_i = \delta_\infty = 3/4$. For mixed choices, the density lies in between. Therefore the opponent can force the density below $3/4 + \varepsilon$, for any $\varepsilon > 0$. Once that is the case, with the center position occupied, the adversary can request a final square of size $3/4 \times x$, where x is the size of the current spanning box. The resulting density is arbitrarily close to $\frac{3/4\times x^2+(3/4\times x)^2}{(x+3/4\times x)^2} = 3/7$. If the center position does not get occupied, the density is even worse. □

It is an easy consequence of continuity that this upper bound can be lowered by a very small amount by slightly decreasing the value for the center case, while increasing the value for the corner case, until they are balanced. More specifically, we can decrease the density for the center case by increasing the square sizes by more than a factor of 2 at each step of the recursion. When only focusing on the center case, the best such factor is $1 + \sqrt{3}$, for an asymptotic density of $\sqrt{3} - 1 = 0.73204\ldots$, yielding a resulting final density of $0.42265\ldots$ as a lower bound for the achievable value. However, this is

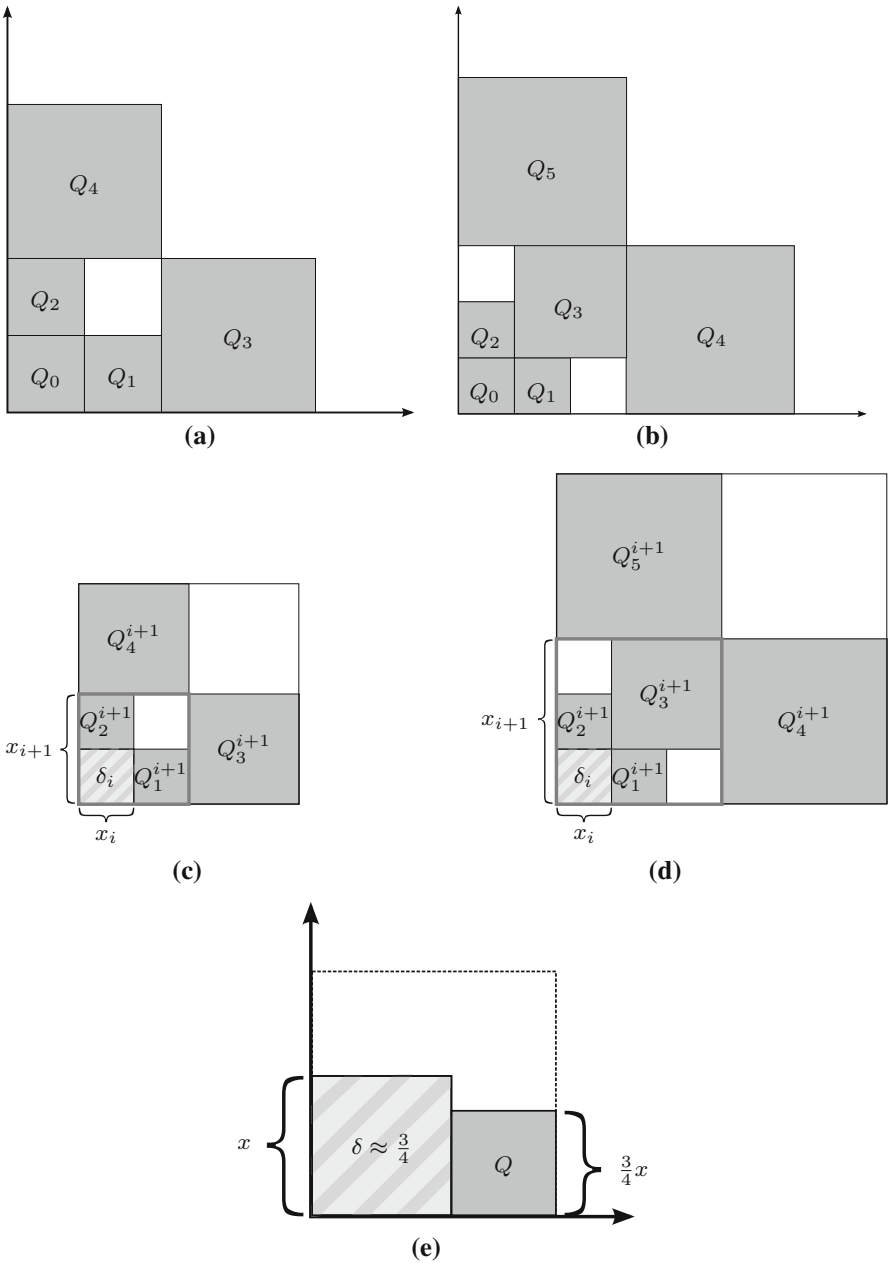


Fig. 11 Different choices in the lower-bound sequence: **a** packing after choosing corner positions. **b** Packing after choosing a center position. **c** Recursion parameters for corner positions. **d** Recursion parameters for center position. **e** Packing a last square

much beyond what can actually be achieved when also accounting for the corner case: the bounding box for each iteration becomes a rectangle, so the worst-case density for the corner case increases quite rapidly. This keeps the total upper bound for the final density much closer to $3/7 = 0.42857\ldots$. As a consequence, we omit the tedious computations for the resulting tiny improvement.

3.2 A Lower Bound on δ

When placing squares into a dynamic container, we cannot use our Recursive Shelf Algorithm, as it requires allocating shelves from all four container boundaries, which are not known in advance. However, we can adapt the Brick Algorithm by [37], which we describe in the following.

The method is based on a partition of the unit square into *bricks*. Bricks are rectangles with aspect ratio $\sqrt{2}$ (or $1/\sqrt{2}$), which are well-known from the international ISO 216 paper formats, in particular the common A series. The most important property of these rectangles is that by bisecting a brick with dimensions $(b, b/\sqrt{2})$, we create two new smaller bricks of size $(b/2, b/\sqrt{2})$. This way, we can construct bricks with side lengths $b2^{-k/2}$ and $b2^{(-k-1)/2}$ for any $k = 0, 1, \ldots$ via a recursive bisection. All of the bricks created this way are called *sub bricks* of B. For any square Q let $S_b(Q)$ denote the smallest brick with side lengths $b/(\sqrt{2})^k$ and $b/(\sqrt{2})^{k+1}$ that may contain Q. Obviously, there is some space left if Q is packed into $S_b(Q)$. Independent of the base b side length we can bound this free space as follows.

Lemma 17.1 *Let Q be a square and b be a real number. Then* $\frac{\|S_b(Q)\|}{2\sqrt{2}} < \|Q\| \leq \frac{\|S_b(Q)\|}{\sqrt{2}}$.

Proof Let s and $\sqrt{2}s$ be the side lengths of $S_B(Q)$. By definition of $S_b(Q)$, we have

$$\frac{s}{\sqrt{2}} = \frac{b}{\left(\sqrt{2}\right)^{k+1}} < x \leq \frac{b}{(\sqrt{2})^k} =: s$$

for some $k \in \mathbb{N}$. With $\|S_b(Q)\| = s \times \sqrt{2}s = \sqrt{2}s^2$ we get

$$\frac{\|S_b(Q)\|}{2\sqrt{2}} = \frac{\sqrt{2}s^2}{2\sqrt{2}} = \left(\frac{s}{\sqrt{2}}\right)^2 < x^2 = \|Q\|$$

$$\text{and} \qquad \|Q\| = x^2 \leq s^2 = \frac{\|S_b(Q)\|}{\sqrt{2}}$$

In other words, with a strategy that packs squares into their respective smallest sub-brick, we cannot hope to generate a packing density higher than $1/(2\sqrt{2})$. We denote the bricks that contain a square *occupied*. All other bricks are called *free*.

Based on this subdivision, Januszewski and Lassak developed a recursive packing algorithm, which they call *the method of the first free fitting subbrick*. They first construct three bricks in the unit square as shown in Fig. 12b. Then they pack each

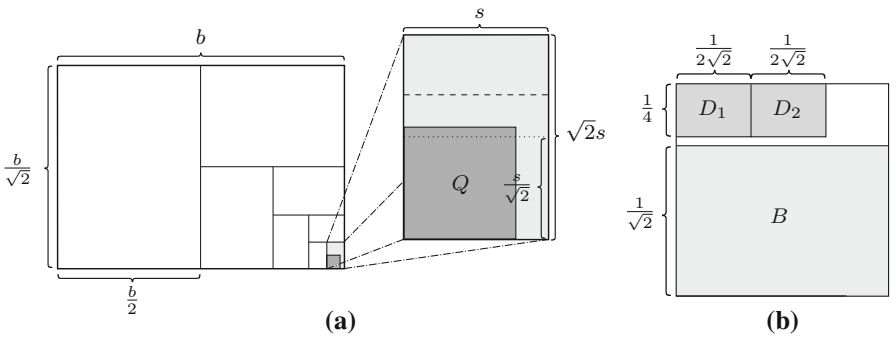


Fig. 12 **a** Subdivision Scheme of the Brick Concept: (*Left*) The recursive bisection of a brick B. (*Right*) A brick (equal to $S_B(Q)$) occupied by a square Q; the *dashed line marks* the upper bound, the *dotted line* the lower bound on the possible side lengths of Q. **b** Partition of the unit square U used by Januszewski and Lassak

square Q into a brick congruent to $S_1(Q)$ after recursively subdividing the smallest free brick that can accommodate Q.

The Brick Algorithm

1. Construct the bricks

$$\begin{aligned} B &= \{(x_1, x_2) : 0 \le x_1 \le 1,\ 0 \le x_2 \le \sqrt{2}\}, \\ D_1 &= \{(x_1, x_2) : 0 \le x_1 \le 2^{-3/2},\ 3/4 \le x_2 \le 1\} \quad \text{and} \\ D_2 &= \{(x_1, x_2) : 2^{-3/2} \le x_1 \le 2 \times 2^{-3/2},\ 3/4 \le x_2 \le 1\} \end{aligned}$$

2. For each incoming square Q:
   (a) Let $\mathcal{B}$ be the first base brick in the order D_1, D_2, B that has a free subbrick $\mathcal{B}'$ of size greater or equal to $S_1(Q)$.
   (b) If $\|\mathcal{B}'\| = \|S_1(Q)\|$, then pack Q into $\mathcal{B}'$.
   (c) Otherwise, recursively bisect (one half of) the smallest subbrick of $\mathcal{B}'$ until a brick $\mathcal{B}''$ of size $S_1(Q)$ is created.
   (d) Pack Q into $\mathcal{B}''$.

We call the bricks B, D_1 and D_2 the *base bricks*. Note that all three base bricks have side length equal to a power of $\sqrt{2}$. That is, all (sub-)bricks created by the algorithm have only side lengths equal to a power of $\sqrt{2}$, too.

In order to adapt this approach to our setting (with increasing instead of decreasing brick size), we keep some properties, but adjust others. We still consider bricks with side lengths equal to a power of $\sqrt{2}$ (and aspect ratio $1/\sqrt{2}$ or $\sqrt{2}$). We let B_k denote the brick of size $\left(\sqrt{2}^k, \sqrt{2}^{k+1}\right)$ and let $S(Q)$ denote the smallest brick B_i that may contain a given square Q.

There are two crucial modifications: (1) The first square Q is packed into a brick of size $S(Q)$ with its lower left corner in the origin and (2) instead of always subdividing the existing bricks (starting with three fixed ones), we may repeatedly double the current maximum existing brick B_{max} to make room for large incoming squares. Apart

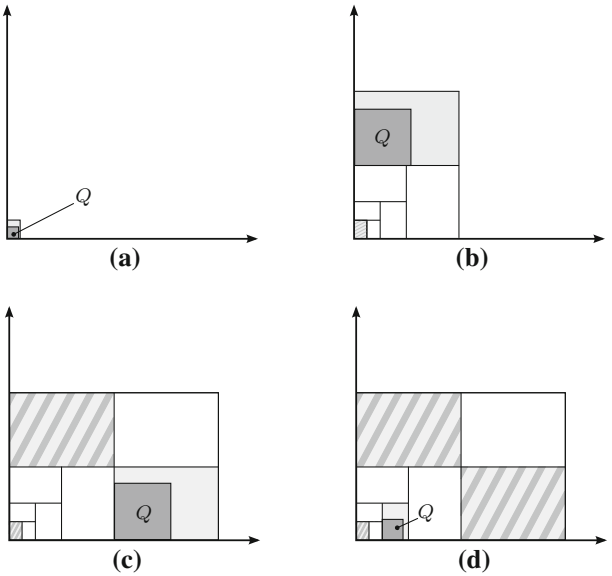


Fig. 13 The modified Brick-Packing algorithm for an input square Q. Occupied bricks are hatched, free bricks are blank. **a** A first square gets placed into the *lower left corner*, $B_{max} = S(Q)$. **b** If $S(Q) > B_{max}$, we double B_{max} until Q fits. **c** If Q does not fit into B_{max}, but $\|S(Q)\| < \|B_{max}\|$, we double B_{max} and subdivide the resulting brick. **d** If Q fits into B_{max}, we pack it into the smallest free fitting subbrick

from that, we keep the same packing scheme: Place each square Q into (a subbrick of) the smallest free brick that can contain Q; see Fig. 13 for an illustration.

Theorem 5 *For any input sequence of squares, the Dynamic Brick Algorithm maintains a packing density of at least* $1/8$.

Proof By construction, every occupied brick has a density of at least $1/(2\sqrt{2})$. It is easy to see that in every step of the algorithm at most half the area of B_{max} consists of free bricks; compare [37]. Because B_{max} always contains all occupied bricks (and thus all packed squares), the ratio of $\|B_{max}\|$ to the area of the smallest enclosing square is at least $1/\sqrt{2}$. Therefore, the algorithm maintains an overall density of at least $(1/(2\sqrt{2})) \times (1/2) \times (1/\sqrt{2}) = 1/8$. □

3.3 Minimizing Container Size

The above results consider the worst-case ratio for the packing density. A closely related question is the online optimization problem of maintaining a square container with minimum edge length. The following is an easy consequence of Theorem 5, as a square of edge length $2\sqrt{2}$ can accommodate a unit area when packed with density 1/8. By considering optimal offline packings for the class of examples constructed in Theorem 4, it is straightforward to get a lower bound of 4/3 for any deterministic online algorithm.

Corollary 2 *Dynamic Brick Packing provides a competitive factor of* $2\sqrt{2} = 2.82\ldots$ *for packing an online sequence of squares into a square container with small edge length. The same problem has a lower bound of 4/3 for the competitive factor.*

4 Conclusion

We have presented progress on two natural variants of packing squares into a square in an online fashion. The most immediate open question remains the critical packing density for a fixed container, where the correct value may actually be $< 1/2$. Even though we invested a considerable amount of work into establishing a lower bound $> 1/3$, we believe that there are alternative schemes that could lead to further improvement.

Online packing into a dynamic container remains wide open. There is still possible slack in both bounds; our feeling is that it should be easier to improve the lower bound rather than the upper bound, as there is still considerable room to employ more sophisticated recursive schemes, just like in the case of a fixed container.

There are many interesting related questions. What is the critical density (offline and online) for packing circles into a unit square? This was raised by Demaine et al. [16]. In an offline setting, there is a lower bound of $\pi/8 = 0.392\ldots$, and an upper bound of $\frac{2\pi}{(2+\sqrt{2})^2} = 0.539\ldots$, which is conjectured to be tight. Another question is to consider the critical density as a function of the size of the largest object. In an offline context, the proof by Moon and Moser provides an answer, but little is known in an online setting.

Acknowledgements We thank the anonymous reviewers for many helpful comments that improved the overall manuscript.

References

1. Azar, Y., Epstein, L.: On two dimensional packing. J. Algorithms **25**(2), 290–310 (1997)
2. Bansal, N., Caprara, A., Jansen, K., Prädel, L., Sviridenko, M.: A structural lemma in 2-dimensional packing, and its implications on approximability. In: Algorithms and Computation, 20th International Symposium, ISAAC, pp. 77–86 (2009)
3. Bansal, N., Caprara, A., Sviridenko, M.: Improved approximation algorithms for multidimensional bin packing problems. In: 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS), pp. 697–708 (2006)
4. Bansal, N., Caprara, A., Sviridenko, M.: A new approximation method for set covering problems, with applications to multidimensional bin packing. SIAM J. Comput. **39**(4), 1256–1278 (2009)
5. Bansal, N., Correa, J.R., Kenyon, C., Sviridenko, M.: Bin packing in multiple dimensions: inapproximability results and approximation schemes. Math. Oper. Res. **31**(1), 31–49 (2006)
6. Bansal, N., Han, X., Iwama, K., Sviridenko, M., Zhang, G.: A harmonic algorithm for the 3D strip packing problem. SIAM J. Comput. **42**(2), 579–592 (2013)
7. Bansal, N., Khan, A.: Improved approximation algorithm for two-dimensional bin packing. In: Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 13–25 (2014)
8. Bansal, N., Lodi, A., Sviridenko, M.: A tale of two dimensional bin packing. In: 46th Annual IEEE Symposium on Foundations of Computer Science (FOCS), pp. 657–666 (2005)
9. Bansal, N., Sviridenko, M.: New approximability and inapproximability results for 2-dimensional bin packing. In: Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 196–203 (2004)

10. Caprara, A.: Packing 2-dimensional bins in harmony. In: 43rd Symposium on Foundations of Computer Science (FOCS), pp. 490–499 (2002)
11. Caprara, A.: Packing d-dimensional bins in d stages. Math. Oper. Res. **33**(1), 203–215 (2008)
12. Caprara, A., Lodi, A., Martello, S., Monaci, M.: Packing into the smallest square: worst-case analysis of lower bounds. Discrete Optim. **3**(4), 317–326 (2006)
13. Caprara, A., Lodi, A., Monaci, M.: Fast approximation schemes for two-stage, two-dimensional bin packing. Math. Oper. Res. **30**(1), 150–172 (2005)
14. Correa, J.R.: Resource augmentation in two-dimensional packing with orthogonal rotations. Oper. Res. Lett. **34**(1), 85–93 (2006)
15. Correa, J.R., Kenyon, C.: Approximation schemes for multidimensional packing. In: Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA) 2004, pp. 186–195 (2004)
16. Demaine, E.D., Fekete, S.P., Lang, R.J.: Circle packing for origami design is hard. In: Origami 5, pp. 609–626. AK Peters/CRC Press (2011)
17. Epstein, L., van Stee, R.: Optimal online bounded space multidimensional packing. In: Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2004, New Orleans, Louisiana, USA, January 11–14, 2004, pp. 214–223 (2004)
18. Epstein, L., van Stee, R.: Online square and cube packing. Acta Inform. **41**(9), 595–606 (2005)
19. Epstein, L., van Stee, R.: Bounds for online bounded space hypercube packing. Discrete Optim. **4**(2), 185–197 (2007)
20. Fekete, S.P., Hoffmann, H.F.: Online square-into-square packing. In: 16th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems Proceedings (APPROX), LNCS, vol. 8096, pp. 126–141 (2013)
21. Fekete, S.P., Kamphans, T., Schweer, N.: Online square packing. In: 11th International Symposium on Algorithms and Data Structures (WADS), LNCS, vol. 5664, pp. 302–314. Springer, Berlin, Heidelberg (2009)
22. Fekete, S.P., Kamphans, T., Schweer, N.: Online square packing with gravity. Algorithmica **68**, 1019–1044 (2014)
23. Fishkin, A.V., Gerber, O., Jansen, K., Solis-Oba, R.: Packing weighted rectangles into a square. In: Mathematical Foundations of Computer Science, International Symposium (MFCS), LNCS, vol. 3618, pp. 352–363 (2005)
24. Fishkin, A.V., Gerber, O., Jansen, K., Solis-Oba, R.: On packing rectangles with resource augmentation: maximizing the profit. Algorithmic Oper. Res. **3**(1), 1–12 (2008)
25. Han, X., Iwama, K., Zhang, G.: Online removable square packing. Theory Comput. Syst. **43**(1), 38–55 (2008)
26. Harren, R.: Approximation algorithms for orthogonal packing problems for hypercubes. Theor. Comput. Sci. **410**(44), 4504–4532 (2009)
27. Harren, R.: Two-dimensional packing problems. Ph.D. thesis, Saarland University (2010)
28. Harren, R., Jansen, K., Prädel, L., Schwarz, U.M., van Stee, R.: Two for one: tight approximation of 2d bin packing. Int. J. Found. Comput. Sci. **24**(8), 1299–1328 (2013)
29. Harren, R., Jansen, K., Prädel, L., van Stee, R.: A $(5/3+\epsilon)$-approximation for strip packing. Comput. Geom. **47**(2), 248–267 (2014)
30. Hougardy, S.: On packing squares into a rectangle. Comput. Geom. Theory Appl. **44**(8), 456–463 (2011)
31. Jansen, K., Prädel, L.: New approximability results for two-dimensional bin packing. In: Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, (SODA), pp. 919–936 (2013)
32. Jansen, K., Solis-Oba, R.: New approximability results for 2-dimensional packing problems. In: Mathematical Foundations of Computer Science, International Symposium (MFCS), LNCS, vol. 4708, pp. 103–114 (2007)
33. Jansen, K., Solis-Oba, R.: A polynomial time approximation scheme for the square packing problem. In: Integer Programming and Combinatorial Optimization, 13th International Conference (IPCO), pp. 184–198 (2008)
34. Jansen, K., Solis-Oba, R.: Rectangle packing with one-dimensional resource augmentation. Discrete Optim. **6**(3), 310–323 (2009)
35. Jansen, K., van Stee, R.: On strip packing with rotations. In: Proceedings of the 37th Annual ACM Symposium on Theory of Computing (STOC), pp. 755–761 (2005)

36. Jansen, K., Zhang, G.: Maximizing the total profit of rectangles packed into a rectangle. Algorithmica **47**(3), 323–342 (2007)
37. Januszewski, J., Lassak, M.: On-line packing sequences of cubes in the unit cube. Geom. Dedic. **67**(3), 285–293 (1997)
38. Kenyon, C., Rémila, E.: Approximate strip packing. In: 37th Annual Symposium on Foundations of Computer Science (FOCS), pp. 31–36 (1996)
39. Kleitman, D., Krieger, M.: Packing squares in rectangles I. Ann. N. Y. Acad. Sci. **175**, 253–262 (1970)
40. Kleitman, D.J., Krieger, M.M.: An optimal bound for two dimensional bin packing. In: 16th Annual Symposium on Foundations of Computer Science (FOCS), pp. 163–168 (1975)
41. Leung, J.Y.T., Tam, T.W., Wong, C.S., Young, G.H., Chin, F.Y.L.: Packing squares into a square. J. Parallel Distrib. Comput. **10**(3), 271–275 (1990)
42. Meir, A., Moser, L.: On packing of squares and cubes. J. Comb. Theory **5**(2), 126–134 (1968)
43. Moon, J., Moser, L.: Some packing and covering theorems. Colloq. Math. **17**, 103–110 (1967)
44. Moser, L.: Poorly formulated unsolved problems of combinatorial geometry. Mimeographed (1966)
45. Novotný, P.: A note on a packing of squares. Stud. Univ. Transp. Commun. Ilina Math. Phys. Ser. **10**, 35–39 (1995)
46. Novotný, P.: On packing of squares into a rectangle. Arch. Math. (Brno) **32**(2), 75–83 (1996)
47. Zhang, Y., Chen, J.C., Chin, F.Y.L., Han, X., Ting, H.F., Tsin, Y.H.: Improved online algorithms for 1-space bounded 2-dimensional bin packing. In: 21st International Symposium on Algorithms and Computation (ISAAC), LNCS, vol. 6507, pp. 242–253 (2010)