Packing Squares into a Disk with Optimal Worst-Case Density

Sándor P. Fekete
Department of Computer Science, TU Braunschweig, Germany

Vijaykrishna Gurunathan
Department of Computer Science & Engineering, IIT Bombay, India

Kushagra Juneja
Department of Computer Science & Engineering, IIT Bombay, India

Phillip Keldenich
Department of Computer Science, TU Braunschweig, Germany

Linda Kleist
Department of Computer Science, TU Braunschweig, Germany

Christian Scheffer
Department of Computer Science, TU Braunschweig, Germany

Abstract

We provide a tight result for a fundamental problem arising from packing squares into a circular container: The critical density of packing squares into a disk is $\delta = {}^{8}/_{5\pi} \approx 0.509$. This implies that any set of (not necessarily equal) squares of total area $A \leq {}^{8}/_{5}$ can always be packed into a disk with radius 1; in contrast, for any $\varepsilon > 0$ there are sets of squares of total area ${}^{8}/_{5} + \varepsilon$ that cannot be packed, even if squares may be rotated. This settles the last (and arguably, most elusive) case of packing circular or square objects into a circular or square container: The critical densities for squares in a square (${}^{1}/_{2}$), circles in a square (${}^{\pi}/_{(3+2\sqrt{2})} \approx 0.539$) and circles in a circle (${}^{1}/_{2}$) have already been established, making use of recursive subdivisions of a square container into pieces bounded by straight lines, or the ability to use recursive arguments based on similarity of objects and container; neither of these approaches can be applied when packing squares into a circular container. Our proof uses a careful manual analysis, complemented by a computer-assisted part that is based on interval arithmetic. Beyond the basic mathematical importance, our result is also useful as a blackbox lemma for the analysis of recursive packing algorithms. At the same time, our approach showcases the power of a general framework for computer-assisted proofs, based on interval arithmetic.

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

Keywords and phrases Square packing, packing density, tight worst-case bound, interval arithmetic, approximation

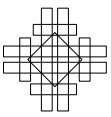
Digital Object Identifier 10.4230/LIPIcs.SoCG.2021.36

Related Version *Full Version*: `https://arxiv.org/abs/2103.07258`

Supplementary Material *Software (Source Code)*:
`https://github.com/phillip-keldenich/squares-in-disk`
archived at `swh:1:dir:7b372039e1b09587f86959e9e1fb013bf2789658`

1 Introduction

Geometric packing and covering problems arise in a wide range of natural applications. They also have a long history of spawning many demanding (and often still unsolved) mathematical challenges. These difficulties are also notable from an algorithmic perspective, as relatively straightforward one-dimensional variants of packing and covering are already NP-hard [16];



however, deciding whether a given set of one-dimensional segments can be packed into a given interval can be checked by computing their total length. This simple criterion is no longer available for two-dimensional, geometric packing or covering problems, for which the total area often does not suffice to decide feasibility of a set, making it necessary to provide an explicit packing or covering. A recent result by Abrahamsen et al. [1] indicates that these difficulties have far-reaching consequences: Two-dimensional packing problems are $\exists\mathbb{R}$-hard, so they are unlikely to even belong to NP.

We provide a provably optimal answer for a natural and previously unsolved case of *tight worst-case area bounds*, based on the notion of *critical packing density*: What is the largest number $\delta \leq 1$, such that any set of squares with a total area of at most δ can always be packed (in a not necessarily axis-parallel fashion) into a disk C of area 1, regardless of the individual sizes of the squares? We show the following theorem that implies $\delta = {}^{8}/_{5\pi} \approx 0.509$ for squares in a disk.

▶ **Theorem 1.** *Every set of squares with a total area of at most* ${}^{8}/_{5}$ *can be packed into the unit disk. This is worst-case optimal, i.e., for every* $A > {}^{8}/_{5}$ *there exists a set of squares with total area* A *that cannot be packed into the unit disk.*

This critical density δ is of mathematical importance, as it settles the last open case of packing circular or square objects into a circular or square container. Figure 1 provides an overview of the critical densities in similar settings, i.e., the critical density for packing squares in a square (${}^{1}/_{2}$), disks in a square (${}^{\pi}/_{(3+2\sqrt{2})} \approx 0.539$), and disks in a disk (${}^{1}/_{2}$).

This result is also of algorithmic interest, because it provides a simple sufficient criterion for feasibility. Note that the previous results illustrated in Figure 1 benefitted from recursive subdivisions of a square container into subpieces bounded by straight lines, or from recursion based on the similarity of objects and container when both are disks; neither applies when objects are squares and the container is a disk. This gives our approach added methodical significance, as it showcases a general framework for establishing computer-assisted proofs for difficult packing problems for which concise manual arguments may be elusive.

A proof of Theorem 1 consists of (i) a class of instances that provide the upper bound of ${}^{8}/_{5\pi}$ for the critical packing density δ and (ii) an algorithm that achieves the matching lower bound for δ by packing any set of squares with a total area of at most ${}^{8}/_{5}$ into the unit

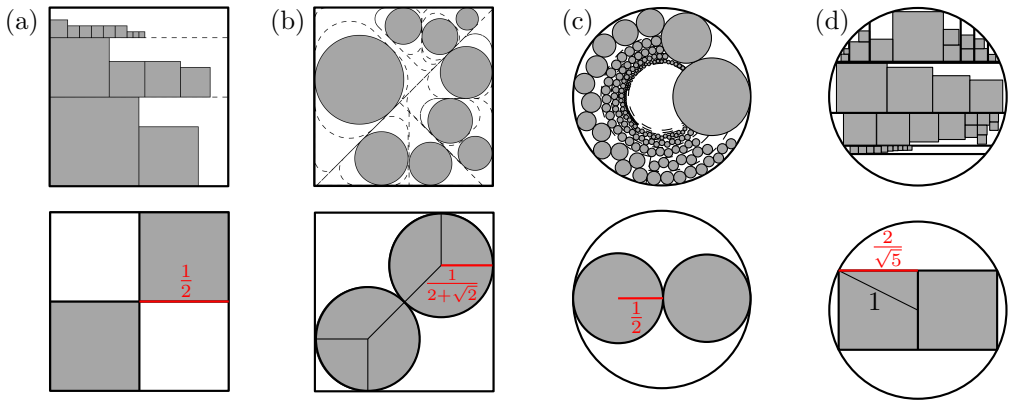


Figure 1 Worst-case optimal approaches and matching worst-case instances for packing: (a) Squares into a square with Shelf Packing by Moon and Moser [23]. (b) Disks into a square by Morr et al. [24, 15]. (c) Disks into a disk by Fekete et al. [14]. (d) Squares into a disk [this paper].

disk. The first part is relatively simple: As shown in Figure 1(d), a critical configuration consists of two squares of side length $s = 2/\sqrt{5}$ and a disk $\mathcal{D}$ of radius 1. It is easy to see that any infinitesimally larger square (of side length $s + \varepsilon$ for any $\varepsilon > 0$) must contain the center of $\mathcal{D}$ in its interior, so two such squares cannot be packed.

The remainder of our paper focuses on the difficult part: providing a strategy for packing sets of squares into a disk (described in Section 2), and then proving that any set of squares with a total area of at most $8/5$ can indeed be packed into the unit disk. This proof is set up with two sets of tools: In Section 3, we describe a general technique that we employed for automated parts of our proof, while Section 4 provides a number of helpful lemmas. Section 5 gives an outline of the actual analysis of our algorithm. Due to space constraints, some detailed proofs are omitted and can be found in the full version of the paper.

1.1 Related work: geometric packing

Problems of geometric packing have been studied for a long time. Providing a survey that does justice to the wide range of relevant work goes beyond the scope of this paper; therefore, we strictly focus on very closely related results, in particular, concerning critical packing density. We refer to Fejes Tóth [10, 28], Lodi, Martello and Monaci [22], Brass, Moser and Pach [8] and Böröczky [7] for more comprehensive surveys.

Even the decision problem whether it is possible to pack a given set of squares into the unit square was shown to be strongly NP-complete by Leung et al. [21], using a reduction from 3-Partition. Already in 1967, Moon and Moser [23] proved that it is possible to pack a set of squares into the unit square if their total area does not exceed $1/2$. This bound is best possible, because two squares even infinitesimally larger than the ones shown in Figure 1(a) cannot be packed. The proof is based on a simple recursive argument.

For the scenario with circular objects, Demaine, Fekete, and Lang [9] showed in 2010 that deciding whether a given set of disks can be packed into a unit square is NP-hard. Using a recursive procedure for partitioning the container into triangular pieces, Morr, Fekete and Scheffer [15, 24] proved that the critical packing density of disks in a square is $\pi/(3+2\sqrt{2}) \approx 0.539$.

More recently, Fekete, Keldenich and Scheffer [14] established the critical packing density of disks into a disk. Employing a number of algorithmic techniques in combination with some interval arithmetic and computer-assisted case checking, they proved that the critical packing density of disks in a disk is $1/2$; they also provide a video including an animated overview [6]. In a similar manner, Fekete et al. [13] established a closed-form description of the total disk area that is sometimes necessary and always sufficient to cover a rectangle depending on its aspect ratio.

Note that the main objective of this line of research is to compute tight worst-case bounds. For specific instances, a packing may still be possible, even if the density is higher; this also implies that proofs of infeasibility for specific instances may be trickier. However, the idea of using the total item volume for computing packing bounds can still be applied. See the work by Fekete and Schepers [11, 12], which shows how a *modified* volume for geometric objects can be computed, yielding good lower bounds for one- or higher-dimensional scenarios.

1.2 Related work: interval arithmetic and computer-assisted proofs

Establishing tight worst-case bounds for packing problems needs to overcome two main difficulties. The first is to deal with the need for accurate computation in the presence of potentially complicated coordinates; the second is the tremendous size of a full case analysis for a complete proof.

Developing methods for both of these challenges has a long tradition in mathematics. One of the first instances of interval arithmetic is Archimedes's classic proof [4] that $^{223}/_{71} \leq \pi \leq {}^{22}/_{7}$, establishing a narrow interval for the fundamental constant of geometry. This entails dealing with inaccurate computation not merely by giving a close approximation, but by establishing an interval for the correct value, which can be used for valid lower and upper bounds for subsequent computations.

Employing electronic devices (e.g., calculators or computer algebra) for mathematical arguments is a well-established method for eliminating tedious, error-prone manual calculations. A famous milestone for the role of computers in theorem proving itself is the confirmation of the Four Color Theorem, a tantalizing open problem for more than 100 years [29]. While the first pair of papers by Appel and Haken [2, 3] was still disputed, the universally accepted proof by Robertson et al. [26] still relies on extensive use of automated checking.

Another example is the resolution of the Kepler conjecture by Hales et al. [17]: While the first version of the proof [18] was still met with some skepticism, the revised and cleaned up variant [17] fits the mold of a more traditional proof, despite relying both on combinatorial results and computational case checking. Note that this proof uses a subdivision technique and interval arithmetic in a manner similar to the one used in this paper. As in this paper, the result of Hales et al. [17] is tight in the numerical sense, which means that due to the discretization error introduced by subdividing a space over $\mathbb{R}$ into finitely many pieces, parts of the proof must be carried out by other means.

Other instances of classic geometric problems that were resolved with the help of computer-assisted proofs are a tight bound for the Erdős-Szekeres problem for the existence of convex paths in planar sets of 17 points [27], a precursor by Hass and Schlafly [19] to the proof of the double bubble theorem by Hutchings et al. [20] (the shape that encloses and separates two given volumes and has the minimum possible surface area is a standard double bubble, i.e., three spherical surfaces meeting at angles of $^{2\pi}/_{3}$ on a common disk), or the proof of NP-hardness of finding a minimum-weight triangulation (MWT) of a planar point set by Mulzer and Rote [25].

Further examples in the context of packing and covering include a branch-and-bound approach for covering of polygons by not necessarily congruent disks with prescribed centers and a minimal sum of radii by Bánhelyi et al. [5].

2 A worst-case optimal algorithm

Now we describe Layer Packing, our worst-case optimal algorithm for packing squares into the unit disk $\mathcal{D}$. The basic idea is to combine refined variants of basic techniques (such as Shelf Packing) in several directions, subdividing the packing area into multiple geometric layers and components.

2.1 Outline of Layer Packing

By $s_1, \ldots, s_n$, we denote a sequence of squares and simultaneously their side lengths and assume that $s_1 \geq \cdots \geq s_n$ is a sorted sequence. Layer Packing distinguishes three cases that depend on the sizes of the first few, largest squares; see Figure 2 for illustrations.

(C 1) If $s_1 \leq 0.295$, we place a square of side length $\mathcal{X} = 1.388$ concentric into $\mathcal{D}$ and place one square of side length $\mathcal{X}_i = 0.295$, $i \in \{1, \ldots, 4\}$, to each side of $\mathcal{X}$, see Figure 2(a). The four largest squares $s_1, \ldots, s_4$ are placed in these containers $\mathcal{X}_1, \ldots, \mathcal{X}_4$. All other squares are packed into $\mathcal{X}$ using Shelf Packing.

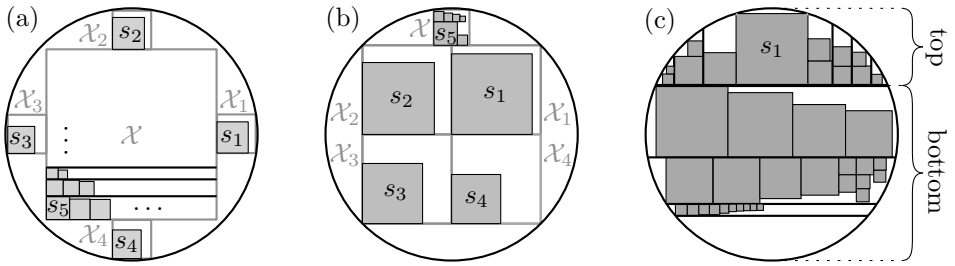


Figure 2 Illustration of the packings (a) in Case **(C1)**, (b) in Case **(C2)**, and (c) in Case **(C3)**.

(C2) If $s_1 \leq 1/\sqrt{2}$ and $s_1^2 + s_2^2 + s_3^2 + s_4^2 \geq 39/25$, let $\mathcal{X}_1, \ldots, \mathcal{X}_4$ be four squares of side length $1/\sqrt{2}$ that are placed into $\mathcal{D}$ as depicted in Figure 2(b). Furthermore, let $\mathcal{X}$ be a square of side length $\sqrt{2}/5$ that can be packed into $\mathcal{D}$ in addition to $\mathcal{X}_1, \ldots, \mathcal{X}_4$; see Figure 2(b). For $i \leq 4$, s_i is the only square packed into $\mathcal{X}_i$; this is possible because $s_i \leq s_1 \leq 1/\sqrt{2}$. All other squares are packed into $\mathcal{X}$ using Shelf Packing.

(C3) In the remaining cases, we make extensive use of a refined shelf packing approach. Specifically, the largest square s_1 is packed into $\mathcal{D}$ as high as possible, see Figures 2(c) and 3. The bottom side of s_1 induces a horizontal split of $\mathcal{D}$ into a *top* and a *bottom* part, which are then filled by two subroutines called Top Packing and Bottom Packing, described in Section 2.2. For each $i \geq 2$, we then

(C3a) use Top Packing to pack s_i if possible,

(C3b) else we use Bottom Packing to pack s_i.

In the remainder of the paper, we prove that Layer Packing only fails to pack a sequence of squares if its total area exceeds the critical bound of $8/5$.

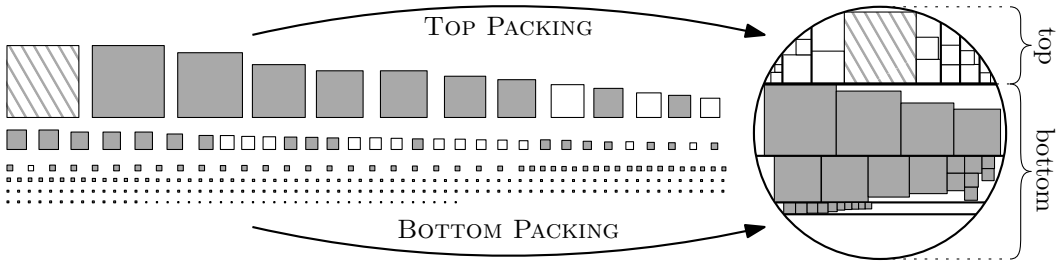


Figure 3 In case **(C3)**, the largest (hatched) square is packed topmost, inducing a top and a bottom part of $\mathcal{D}$. Subsequent (white) squares are packed into the pockets of the top part with *Top Packing* (using Refined Shelf Packing as a subroutine) if they fit; if they do not fit, they are shown in gray and packed into the bottom part with *Bottom Packing*, which uses horizontal SubContainer Slicing, and vertical Refined Shelf Packing within each slice.

2.2 Subroutines of Layer Packing

Layer Packing employs a number of different subroutines.

Refined Shelf Packing The greedy-type packing procedure Shelf Packing, employed by Moon and Moser [23], packs objects by decreasing size; see the top of Figure 1(a). At each stage, there is a (w.l.o.g. horizontal) straight cut that separates the unused portion of the container from a "shelf" into which the next square is packed. The height of a shelf is determined by the first packed object. Subsequent objects are packed next to

each other, until an object no longer fits into the current shelf; in this case, a new shelf is opened on top of the previous one. For Layer Packing, we use two modifications.
(1) Parts of the shelf boundaries may be circular arcs; however, we still have a supporting straight axis-parallel boundary and a second, orthogonal straight boundary.
(2) Our refined shelf packing uses the axis-parallel boundary line of a shelf as a support line for packing squares; in case of a collision with the circular boundary, we may move a square towards the middle of a shelf if this allows packing it. Note that this may only occur in shelves containing the horizontal diameter.

Top Packing The first and largest square s_1 is packed as high as possible into $\mathcal{D}$; see Figure 4(a). Then the horizontal line through the bottom of s_1 cuts the container into a *top part* that contains s_1, with two congruent empty pockets C_ℓ and C_r left and right of s_1; and a *bottom part*. Each pocket has two straight axis-parallel boundaries, b_x and b_y. By σ, we denote the largest square that fits into either pocket. For large s_1, the bottom side of σ does not lie on the same height as the bottom side of s_1; in that case, we ignore the parts of C_ℓ and C_r that lie below σ; see Figure 4(e). We use Refined Shelf Packing with shelves parallel to the shorter boundary among b_x and b_y, as shown in Figure 4(b) and (c). If a square does not fit into either pocket, it is packed into the bottom part.

Bottom Packing A square that does not fit into the top part of $\mathcal{D}$ is packed into the bottom part. For this purpose, we use (horizontal) SubContainer Slicing, and (vertical) SubContainer Packing within each subcontainer; see Figure 3 for the overall picture.

SubContainer Slicing For packing squares in the bottom part of $\mathcal{D}$, SubContainer Slicing subdivides $\mathcal{D}$ into smaller containers C_i, by using straight horizontal cuts; see Figure 4(a). The height of a subcontainer is determined by the first square packed into it.

SubContainer Packing Within each subcontainer, we use Refined Shelf Packing with vertical shelves. These shelves are packed from the longer of the two horizontal cuts, i.e., to pack C_i, we start from the boundary that is closer to the disk center; see Figure 4(d).

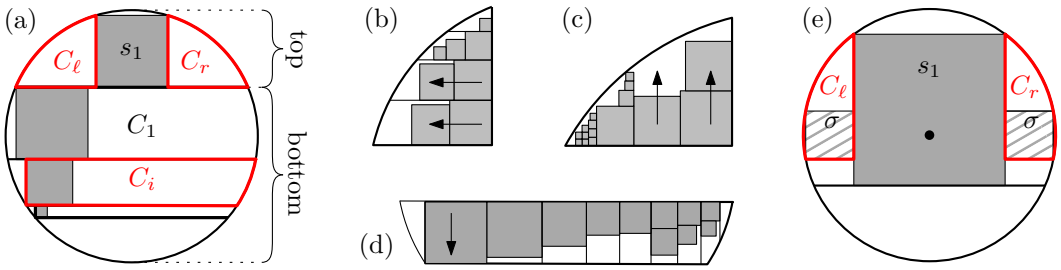


Figure 4 (a) Packing s_1 topmost into $\mathcal{D}$ yields the top part of $\mathcal{D}$ with pockets C_ℓ and C_r, and the bottom part of $\mathcal{D}$. The bottom part is partitioned by SubContainer Slicing into subcontainers C_i, with heights corresponding to the first packed square. (b) A pocket C_ℓ for which $b_x \leq b_y$ implies horizontal shelf packing. (c) A pocket C_ℓ for which $b_x > b_y$ implies vertical shelf packing. (d) Within each subcontainer C_i, SubContainer Packing places squares along vertical shelves, starting from the longer straight cut of C_i. (e) For large s_1, we disregard the parts of C_ℓ and C_r that lie below their inscribed square σ.

3 Proofs based on interval arithmetic

In interval arithmetic, operations like addition, multiplication or taking the square root are performed on real intervals $[a, b] \subset \mathbb{R}$ instead of real numbers. When applied to intervals, an operation $[a_1, b_1] \circ [a_2, b_2]$ results in the smallest interval that contains all possible values

of $x \circ y$ for $x \in [a_1, b_1], y \in [a_2, b_2]$. In a practical implementation on computers with finite precision, computing the smallest such interval is not always possible. However, using appropriate rounding modes or error bounds, it is still possible to compute an interval that over-approximates the resulting interval, i.e., contains all possible outcomes of the corresponding real operation. Predicates such as $[a_1, b_1] \leq [a_2, b_2]$ can also be evaluated on intervals. The result is a subset of $\{\texttt{false}, \texttt{true}\}$ containing all possible outcomes of $x \leq y$ for $x \in [a_1, b_1], y \in [a_2, b_2]$. This allows evaluating quantifier-free formulas on the Cartesian product of intervals in an over-approximative way.

In many cases throughout this paper, we want to prove that a given non-linear system of real constraints over a bounded k-dimensional space $\mathcal{R}$ is unsatisfiable. This space is typically spanned by a set of k real variables. Conceptually, to do this in an automatic fashion, we subdivide $\mathcal{R}$ into a sufficiently large number of k-dimensional cuboids. Each such cuboid is defined by an interval for each of the k real variables spanning $\mathcal{R}$. We then apply interval arithmetic to each such cuboid $\mathcal{C}$ to find a set S of constraints that together eliminate all points of $\mathcal{C}$, thus proving that no point in $\mathcal{C}$ satisfies all our constraints for a counterexample.

We use this simple technique because it scales relatively well with the complexity of the constraints and can handle non-polynomial constraints involving functions such as $\arccos(x)$ that occur in some of our proofs.

To improve the efficiency of this approach, our implementation of the basic concept is optimized in several ways. For instance, the subdivision proceeds in a tree-like fashion according to a fixed ordering of the k variables $v_1, \ldots, v_k$ spanning $\mathcal{R}$. If variables $v_1, \ldots, v_j$ suffice to exclude a part of $\mathcal{R}$, we do not split $v_{j+1}, \ldots, v_k$ on that part. Furthermore, we adaptively increase the local fineness of our subdivision if a coarser subdivision does not suffice for some part of $\mathcal{R}$.

Overall, this leads to a limited number of automated proofs[1]; for some of these, manual checking would also be feasible, but would involve many case distinctions and would be tedious and unsatisfying. Instead, we replace these proofs by the automatic procedure outlined above to have a clear structure instead of an otherwise overwhelming set of arguments. Combined, all automatic proofs required for this paper take less than 1.5 hours and less than 300 MB of memory on the 4 physical cores of the 2.3 GHz Intel i5-8259U CPU in one of the authors' laptops. The proofs involve up to $k = 9$ variables.

4 Analysis of subroutines

In the following, we establish a number of bounds for the subroutines from Section 2.2 that we use to prove the performance guarantee for Layer Packing.

4.1 Shelf Packing

In several places we make use of the following classic result regarding Shelf Packing.

▶ **Lemma 2** ([23]). *Shelf Packing packs every sequence $t_1 \geq \cdots \geq t_u$ of squares with a total area of at most $1/2 \cdot hw$ into an $h \times w$-rectangle with $t_1 \leq h \leq w$.*

If the side length of the largest square is small compared to the size of the container, one can guarantee a higher packing density.

[1] Source code available at `https://github.com/phillip-keldenich/squares-in-disk`.

▶ **Lemma 3** ([23]). *Any finite set of squares with largest square $x_1 < 1/2$ is packed by* Shelf Packing *into a unit square, provided its total area is at most $1/2 + 2(x_1 - 1/2)^2$.*

4.2 Top Packing

We prove the following lower bound on the square area packed by Top Packing, as long as at least one square fits into the left pocket C_ℓ. By $\sigma = \sigma(s_1)$, we denote the side length of the largest square that can be packed into C_ℓ or C_r; see Figure 5(c)–(e).

▶ **Lemma 4.** *Let $s_1 \geq \cdots \geq s_n$ be a sequence of squares for which* Layer Packing *fails to pack s_n. If $s_n \leq \sigma$, then* Top Packing *packs squares of total area at least $0.83\sigma^2$.*

Intuitively, the proof makes use of the Shelf Packing bound on the squares inscribed in C_ℓ and C_r, but additionally uses the gaps in C_r and C_ℓ to bound the square area packed into each of C_ℓ and C_r by $0.415\sigma^2$.

Before presenting its proof, we make some helpful observations. We assume the center of our unit disk $\mathcal{D}$ lies at the origin $(0, 0)$ of our coordinate system. Recall that Top Packing packs the largest square s_1 as high as possible into $\mathcal{D}$. This implies that the center of s_1 is on the vertical line $x = 0$. For some $u \in (-1, 1)$, we denote by $T(u)$ the side length of the largest square with center on $x = 0$ and bottom on $y = u$ that fits into $\mathcal{D}$; see Figure 5(a).

The inverse function $T^{-1}(s)$ of $T(u)$ describes the highest possible y-coordinate of the bottom side of a square of side length s; see Figure 5(b). Thus, Top Packing places the bottom-left corner of s_1 at $(-s_1/2, T^{-1}(s_1))$; note that this can be below or above the center of $\mathcal{D}$. Furthermore, recall that Top Packing packs the remaining disks into the pockets C_ℓ and C_r induced by placing s_1; see Figure 5(c).

Now, we present explicit formulas. Solving the equations in Figure 5(a)–(b), we get

$$T(u) = 2/5 \cdot \left(\sqrt{5 - u^2} - 2u\right) \quad \text{and} \quad T^{-1}(s) = \sqrt{1 - 1/4 \cdot s^2} - s.$$

To compute $\sigma(s_1)$, we observe the following. Below some threshold s_1^*, the bottom side of the inscribed square of C_ℓ lies on the horizontal line $y = T^{-1}(s_1)$ and its top left corner touches $\mathcal{D}$; see Figure 5(c). At the threshold $s_1^* = \sqrt{1/3 \cdot (2 + \sqrt{2})}$, both left corners of

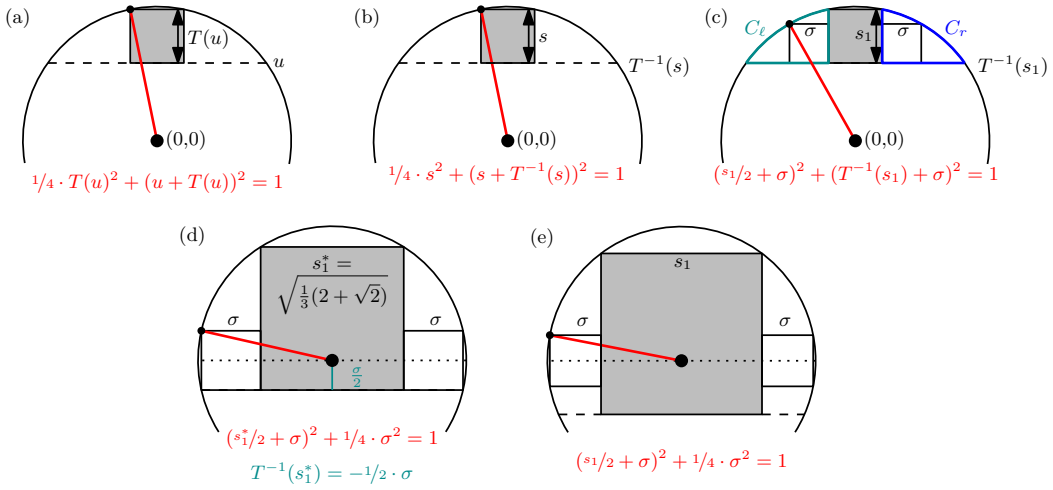


Figure 5 (a) The definition of $T(u)$ and its defining equation. (b) The definition of $T^{-1}(s)$ and its equation. (c) The pockets C_ℓ and C_r used by Top Packing with their inscribed square σ and its equation if $s_1 < s_1^*$, (d) $s_1 = s_1^*$ and (e) $s_1 > s_1^*$.

the inscribed square of C_ℓ touch the disk; see Figure 5(d). For $s_1 > s_1^*$, the center of C_ℓ's inscribed square lies on $y = 0$ and both left corners touch the disk; see Figure 5(e). Overall, this yields

$$\sigma = \begin{cases} 1/4 \cdot \left(-s_1 - 2T^{-1}(s_1) + \sqrt{8 - (s_1 - 2T^{-1}(s_1))^2}\right), & \text{if } s_1 \leq s_1^*, \\ 1/5 \cdot \left(\sqrt{20 - s_1^2} - 2s_1\right), & \text{otherwise.} \end{cases}$$

Now we are ready to present a proof of Lemma 4.

Proof of Lemma 4. We begin by observing that $s_n \leq \sigma$ would fit into either C_ℓ or C_r; as we fail to pack s_n, C_ℓ and C_r must contain other squares. In the following, we prove that Top Packing packs squares of area $A \geq 0.415\sigma^2$ into C_ℓ. An analogous argument works for C_r, implying an overall bound of $0.83\sigma^2$.

By $\ell_1 := \sqrt{1 - T^{-1}(s_1)^2} - s_1/2$, we denote the length of the bottom boundary of C_ℓ; see Figure 6(a). W.l.o.g., we assume $\ell_1 \leq s_1$; the other case is symmetric. In other words, we assume that the bottom boundary of C_ℓ is shorter than its right boundary, which means that we are using horizontal shelves that we fill from right to left as depicted in Figure 4(b).

Consider the subsequence $t_1, \ldots, t_u, t_{u+1}$ of $s_1, \ldots, s_n$, where $t_1, \ldots, t_u$ are the squares packed by Top Packing into C_ℓ before height σ is (strictly) exceeded, and t_{u+1} is the next square that we try to pack into C_ℓ. We observe that t_{u+1} may or may not be packed into C_ℓ by Top Packing, and that $u \geq 1$ by $t_1 \leq \sigma$, i.e., after placing the first square, height σ is not exceeded. We make use of the following lemma, proved by interval arithmetic.

▶ **Lemma 5** (Automatic Analysis for Top Packing). *Let $\ell_1 \leq s_1$, $x_{+1} = s_1/2 + \sigma/2\sqrt{2}$ and $x_{+2} = s_1/2 + 0.645\sigma$. Furthermore, let*

$$y_{+1} = \begin{cases} T^{-1}(s_1) + \sigma + \sigma/2\sqrt{2}, & \text{if } s_1 \leq s_1^*, \\ \sigma/2 + \sigma/2\sqrt{2}, & \text{otherwise,} \end{cases} \quad y_{+2} = \begin{cases} T^{-1}(s_1) + 2 \cdot 0.645\sigma, & \text{if } s_1 \leq s_1^*, \\ -\sigma/2 + 2 \cdot 0.645\sigma, & \text{otherwise;} \end{cases}$$

see Figure 6. Let $F_{TP_1}(s_1) := x_{+1}^2 + y_{+1}^2$ and $F_{TP_2}(s_1) := x_{+2}^2 + y_{+2}^2$. Then, for all $0.295 \leq s_1 \leq \sqrt{8/5}$, we have (1) $F_{TP_1}(s_1) \leq 1$ and (2) $F_{TP_2}(s_1) \leq 1$.

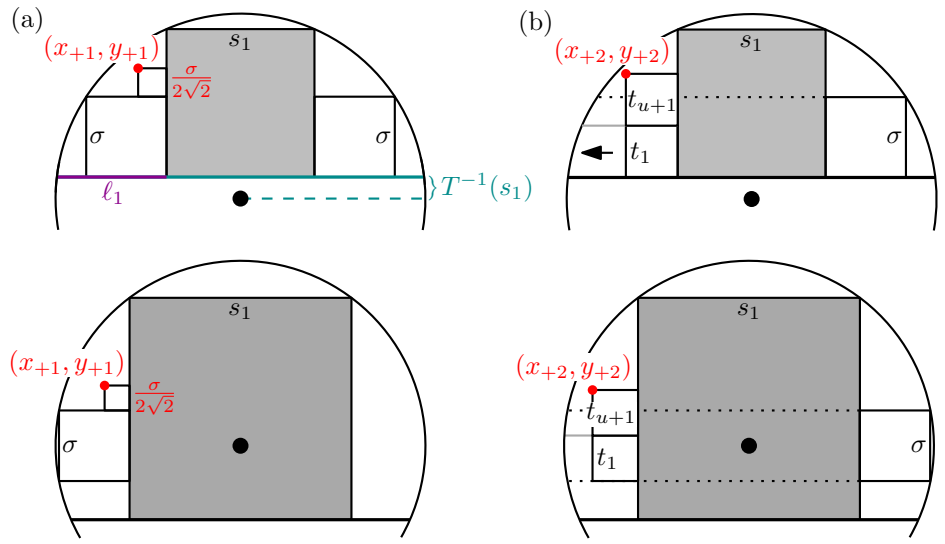


Figure 6 Illustration of the proof of Lemma 5. In both parts, the red point is always contained in the disk $\mathcal{D}$. (a) The values occurring in part (1). (b) The situation for t_1, t_{u+1} of maximum possible size in part (2).

If $0.645\sigma \le t_1$, the packed area inside C_ℓ is at least $t_1^2 \ge 0.645^2\sigma^2 > 0.415\sigma^2$. Thus, in the following, we assume $t_1 < 0.645\sigma$.

Furthermore, if $t_{u+1} \le \sigma/2\sqrt{2}$, we can apply Lemma 5 (1), showing that t_{u+1} can be packed into C_ℓ by Top Packing; see Figure 6(a). In particular, t_{u+1} can always be packed into C_ℓ such that its bottom side lies on height σ and its right side touches s_1. The total area packed by Top Packing into C_ℓ is at least the total area packed by Shelf Packing into the square of area σ; here we use the fact that the height of the bottom segment of a pocket and the bottom segment of the contained square σ coincide, see also Figure 4(e). Because packing t_{u+1} exceeds height σ, Lemma 3 implies that the total area of t_{u+1} and the squares already packed into C_ℓ exceeds $\sigma^2/2$. Thus, in the following, we assume $\sigma/2\sqrt{2} < t_{u+1} \le t_1 < 0.645\sigma$.

If $t_1 \le \sigma/2$, at least four squares are packed into C_ℓ by Refined Shelf Packing before height σ is exceeded. Consequently, the total packed area is at least $4\,(\sigma/2\sqrt{2})^2 = \sigma^2/2$. Thus, in the following we assume $t_1 > \sigma/2$.

Now let us assume that only one shelf is constructed before height σ is exceeded. That shelf has height t_1 and thus we must have $t_{u+1} > \sigma - t_1$. We use Lemma 5 (2) to prove that we can pack t_{u+1} on top of the first shelf, assuming that $t_1 = t_{u+1} = 0.645\sigma$ are as large as possible; see Figure 6(b). Thus, the total area packed into C_ℓ is at least $t_1^2 + t_{u+1}^2 \ge t_1^2 + (\sigma - t_1)^2 \ge \sigma^2/2$.

Otherwise, at least two shelves are constructed before height σ is exceeded. The first shelf has height t_1. The second shelf contains at least two squares because its height is at most $\sigma - t_1 \le \sigma/2$, and thus at most half of its width. Thus, the area packed into C_ℓ is at least $t_1^2 + 2t_{u+1}^2 \ge \sigma^2/4 + 2(\sigma/2\sqrt{2})^2 = \sigma^2/2$, concluding the proof of Lemma 4. ◀

4.3 Subcontainer Packing

For the analysis of SubContainer Packing, let $C_1, \dots, C_k$ be the subcontainers constructed by Bottom Packing and let $R_1, \dots, R_k$ be the maximal rectangles contained in $C_1, \dots, C_k$; see Figure 7.

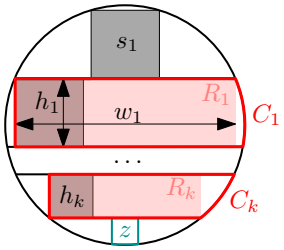


Figure 7 Subcontainers C_i, $i \le k$, produced by SubContainer Slicing.

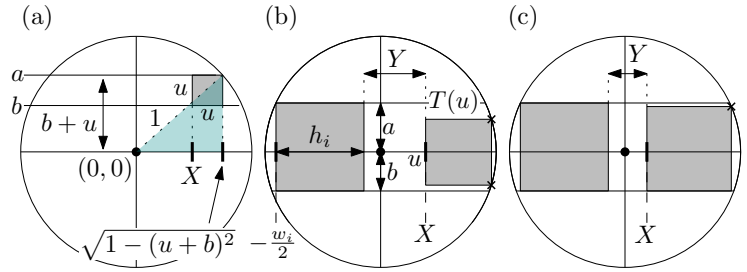


Figure 8 The computation of X for a square of side length u: (a) in case $b \ge 0$, which is symmetric to $a < 0$; (b) in case $a > 0 > b$ and $u \le 2c$; (c) in case $a > 0 > b$ and $u > 2c$.

For $i = 1, \dots, k$, let h_i and w_i denote the height and the width of R_i. Recall that h_i simultaneously denotes the height of C_i and the first square packed into C_i. Let z be the largest square that could be packed below C_k. We define $h_{k+1} := s_n$, so h_{i+1} always denotes the first square that did not fit into C_i. Furthermore, we denote the total area of squares packed into C_i by $\|C_i\|$. We establish several lower bounds on this area $\|C_i\|$. One such bound is derived from the following observation.

▶ **Observation 6.** *The total area packed by SubContainer Packing into C_i is at least the total area packed by Shelf Packing into R_i.*

If the width of R_i is at least twice its height, the following lemma improves on this bound. A proof can be found in the full version of this paper.

▶ **Lemma 7.** *For every sequence of squares $s_1, \dots, s_n$ for which* Layer Packing *constructs at least i subcontainers and fails to pack s_n, if $w_i \geq 2h_i$, we can bound the area packed into C_i by*

$$\|C_i\| \geq B_1(h_i, w_i, h_{i+1}) := \max \begin{cases} 1/2 \cdot h_i w_i + 1/4 \cdot h_i^2, \\ h_i^2 + (w_i - h_i - h_{i+1})h_{i+1}, \\ 1/2 \cdot h_i(w_i + h_i) - h_{i+1}^2. \end{cases}$$

We always pack at least the square h_i into C_i. As $h_{i+1} \leq h_i$, if $h_i + h_{i+1} \leq w_i$, we pack at least two squares into C_i: Let s_j be the second square we consider packing into C_i. If h_i and s_j do not fit into C_i, then $s_j = h_{i+1}$, as we would open a new subcontainer for s_j. This contradicts $h_i + h_{i+1} \leq w_i$, as h_i and h_{i+1} fit into R_i and thus into C_i. Summarizing, we can extend B_1 as follows.

▶ **Lemma 8.** *For every sequence of squares $s_1, \dots, s_n$ for which* Layer Packing *constructs at least i subcontainers and fails to pack s_n, we can bound the area packed into C_i by*

$$\|C_i\| \geq B_2(h_i, w_i, h_{i+1}) := \begin{cases} h_i^2 & \text{if } w_i < h_i + h_{i+1}, \\ h_i^2 + h_{i+1}^2 & \text{if } h_i + h_{i+1} \leq w_i < 2h_i, \\ B_1(h_i, w_i, h_{i+1}) & \text{if } 2h_i \leq w_i. \end{cases}$$

For the last lemma of this subsection, we introduce some useful notation. Let a, b be the y-coordinates of the upper and the lower side of C_i and let $c := c(a, b) = \min\{a, -b\}$. When C_i contains the center of the disk, i.e., $a > 0 > b$, c denotes the distance of the origin to the nearer side of C_i, see Figure 8(b) and (c). The maximal x-coordinate X of the left side of a square of side length u in C_i is determined by

$$X(a, b, u) := \left\{ \begin{array}{ll} \sqrt{1 - (u + b)^2} - u & \text{if } b \geq 0, \\ \sqrt{1 - (u - a)^2} - u & \text{if } a < 0, \\ T^{-1}(u) & \text{else if } u \leq 2c, \\ \sqrt{1 - (u - c)^2} - u & \text{otherwise} \end{array} \right\} = \begin{cases} T^{-1}(u) & \text{if } u \leq 2c, \\ \sqrt{1 - (u - c)^2} - u & \text{otherwise} \end{cases}$$

The x-coordinate of the right side of the first square h_i packed into subcontainer C_i is $-1/2 \cdot w_i + h_i$. Thus, as h_{i+1} did not fit into C_i, we can lower bound the total width of squares packed into C_i after h_i, see Figure 9(a), by

$$Y := Y(a, h_i, w_i, h_{i+1}) := 1/2 \cdot w_i - h_i + X(a, a - h_i, h_{i+1}).$$

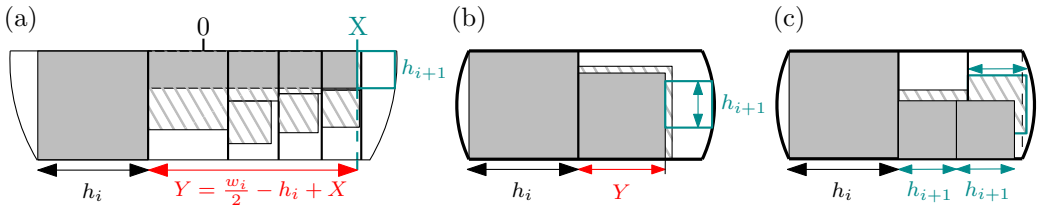


Figure 9 (a) Definition of Y. (b) The lower bound $B_3(a, h_i, w_i, h_{i+1})$ when two squares are packed into C_i. (c) The lower bound $B_3(a, h_i, w_i, h_{i+1})$ when at least three squares packed into C_i.

▶ **Lemma 9.** *For every sequence of squares $s_1, \dots, s_n$ for which* Layer Packing *constructs at least i subcontainers and fails to pack s_n, we can bound the area packed into C_i by*

$$\|C_i\| \geq B_3(a, h_i, w_i, h_{i+1}) := \max \begin{cases} h_i^2 + \max\{0, Y(a, h_i, w_i, h_{i+1})\} \cdot h_{i+1}, & (8.1) \\ h_i^2 + \min\left(\max^2(Y(a, h_i, w_i, h_{i+1}), 0), 2h_{i+1}^2\right). & (8.2) \end{cases}$$

Proof. If $Y \leq 0$, both bounds (8.1) and (8.2) simplify to h_i^2 and are valid, because h_i is packed into C_i. Thus, let us assume $Y > 0$. This implies that there are at least two squares packed into C_i; otherwise, h_{i+1} would fit into C_i. As Y is a lower bound on the total width of squares packed into C_i after h_i and h_{i+1} is a bound on their height, we obtain bound (8.1); see Figure 9(a).

Furthermore, if exactly two squares are packed into C_i, Y^2 can be used as lower bound on the area of the second square, see Figure 9(b). Otherwise, at least three squares are packed into C_i, and we can use $2h_{i+1}^2$ to bound their area, see Figure 9(c). Combining these two cases yields bound (8.2). ◀

We combine these previous bounds into a general lower bound for $\|C_i\|$.

▶ **Corollary 10.** *For every sequence of squares $s_1, \dots, s_n$ for which* Layer Packing *constructs at least i subcontainers and fails to pack s_n, we can bound the area packed into C_i by*

$$\|C_i\| \geq B_4(a, h_i, w_i, h_{i+1}) := \max \begin{cases} B_2(h_i, w_i, h_{i+1}), & \textit{(Lemma 8)} \\ B_3(a, h_i, w_i, h_{i+1}). & \textit{(Lemma 9)} \end{cases}$$

5 Analysis of the main algorithm

In this section, we prove our main result using the tools provided in Sections 3 and 4. On the highest level, the proof consists of three parts corresponding to the three cases that our algorithm distinguishes.

5.1 Analysis of (C1)

Recall that in case **(C 1)**, we place a container square $\mathcal{X}$ of side length 1.388 into $\mathcal{D}$, and pack the first four squares into pockets outside $\mathcal{X}$ and all remaining disks into $\mathcal{X}$ using Shelf Packing; see Figure 2(a).

▶ **Lemma 11.** *If* Layer Packing *fails to pack a sequence of squares $s_1, \dots, s_n$ with $s_1 \leq 0.295$, the total area of the squares exceeds $8/5$.*

Proof. Consider scaling down all side lengths by a factor of $1/1.388$, such that $\mathcal{X}$ is the unit square and $s_1 \leq 0.295/1.388 \approx 0.2125$. As s_5 is the first square packed by Shelf Packing into $\mathcal{X}$, Lemma 3 implies that the total area packed into the scaled $\mathcal{D}$ is at least $f(s_5) = 4s_5^2 + 1/2 + 2(s_5 - 1/2)^2$ with derivative $f'(s_5) = 12s_5 - 2$, which is minimized for $s_5 = 1/6$, where $f(1/6) = 5/6$. Thus, in the non-scaled configuration, the area packed is at least $5/6 \cdot 1.388^2 = 120409/75000 \approx 1.605 > 8/5$, concluding the proof. ◀

5.2 Analysis of (C2)

Recall that in case **(C 2)**, we pack the four largest squares into the squares $\mathcal{X}_1, \dots, \mathcal{X}_4$; all other squares are packed into a square container $\mathcal{X}$ on top of them; see Figure 2(b).

▶ **Lemma 12.** *If* Layer Packing *fails to pack a sequence $s_1, \dots, s_n$ of squares with $0.295 < s_1 \leq 1/\sqrt{2}$ and $s_1^2 + s_2^2 + s_3^2 + s_4^2 \geq 39/25$, the total area of the squares exceeds $8/5$.*

Proof. By assumption, the total area of the squares s_1, s_2, s_3, s_4 is at least $39/25 = 8/5 - 1/25$. As $\mathcal{X}$ has an area of $2/25$, Lemma 2 implies that Shelf Packing (and thus Layer Packing) only fails to pack all remaining squares into $\mathcal{X}$ if their area exceeds $1/25$. Consequently, the total area of the squares exceeds $8/5$, concluding the proof. ◀

5.3 Analysis of (C3)

Recall that z denotes the largest square that could be packed below the last subcontainer C_k constructed by Bottom Packing, as illustrated in Figure 7 or in Figure 10, where we have reflected the instance along the x-axis. We consider a sequence $s_1, \ldots, s_n$ of squares of total area S that Layer Packing fails to pack, and assume w.l.o.g. that s_{n-1} is packed. This implies $z < s_n$, where z denotes the side length of the largest square that could be packed below the last subcontainer C_k constructed by Bottom Packing; otherwise, a further subcontainer is constructed. We have $z = T(-T^{-1}(s_1) + \sum_{i=1}^{k} h_i)$; see Figure 10(a).

By $w(y_t, h) = 2\sqrt{\min\{1 - y_t^2, 1 - (y_t - h)^2\}}$, we denote the maximum width of a rectangle R that can be placed in $\mathcal{D}$ with top side at $y = y_t$ and height h; see Figure 10(b). Thus, we can express the width w_i of the rectangle R_i inscribed in some subcontainer C_i in terms of $s_1, h_1, \ldots, h_i$ as

$$w_i \coloneqq w\left(\left(T^{-1}(s_1) - \sum_{j=1}^{i-1} h_j\right), h_i\right).$$

Recall that $\sigma \coloneqq \sigma(s_1)$ denotes the side length of the largest squares that fits into the pockets C_ℓ and C_r as illustrated in Figure 5(c). In order to distinguish whether Top Packing has packed any squares into the pockets, we consider the function

$$E(s_1, s_n) \coloneqq \begin{cases} 0.83 \cdot \sigma(s_1)^2, & \text{if } s_n \leq \sigma, \\ 0, & \text{otherwise,} \end{cases}$$

which describes the total square area that Top Packing is guaranteed to pack due to Lemma 4.

For the analysis of Case **(C3)**, we distinguish cases depending on the number k of subcontainers constructed by Bottom Packing. Specifically, we consider the cases $k = 0, k = 1, k \in \{2, 3, 4\}$, and $k \geq 5$.

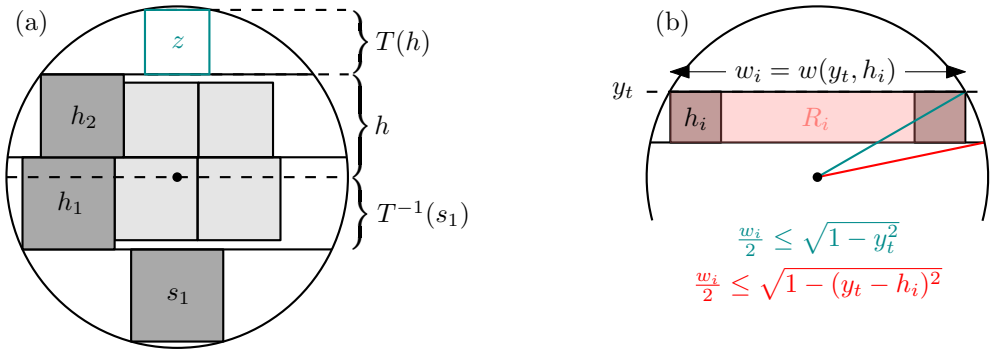


Figure 10 (a) Computing z for $k = 2$ using functions T and T^{-1}. (b) Computing the width $w_i = w(y_t, h)$ of rectangle R_i.

5.3.1 Analysis for no subcontainer

▶ **Lemma 13.** *If* Layer Packing *fails to pack a sequence $s_1, \dots, s_n$ of squares and* Bottom Packing *does not construct a subcontainer, the total area of the squares exceeds $8/5$.*

Proof. Because the algorithm fails to construct a first subcontainer in the bottom part, it follows that placing s_n as far to the bottom as possible yields an overlap with s_1. However, the minimum value for $s_1^2 + s_n^2$ for two overlapping squares packed into a disk is attained for $s_1 = s_n$. This corresponds to the worst-case configuration, implying that the total area of s_1 and s_n exceeds $8/5$. ◀

5.3.2 Analysis for one subcontainer

▶ **Lemma 14.** *If* Layer Packing *fails to pack a sequence $s_1, \dots, s_n$ of squares and* Bottom Packing *constructs exactly one subcontainer, the total area of the squares exceeds $8/5$.*

Proof. Combining Lemma 4 and Corollary 10 allows us to bound the area of $s_1, \dots, s_n$ by

$$S \geq F_{SC_1}(s_1, h_1, s_n) \coloneqq s_1^2 + B_4(T^{-1}(s_1), h_1, w_1, s_n) + s_n^2 + E(s_1, s_n),$$

where $E(s_1, s_n) = 0.83 \cdot \sigma(s_1)^2$ if $s_n \leq \sigma$, and $E(s_1, s_n) = 0$ if $s_n > \sigma$. Furthermore, we know $0 < z < s_n$, because Layer Packing fails to pack s_n. Moreover, we claim that at least one of the following conditions must hold: $s_1 > 1/\sqrt{2}$, $w_1 < 2h_1$, or $s_1^2 + h_1^2 + 2s_n^2 < 39/25$. Assume for contradiction that neither of these conditions hold. By $w_1 \geq 2h_1$, we know that at least two squares are packed into the first subcontainer. One of these squares has area h_1^2, and the other has area at least s_n^2. In particular, this implies that the algorithm packs s_1, s_2 and s_3. This implies $s_2 \geq h_1$ and $s_3, s_4 \geq s_n$. Thus we have $s_1^2 + s_2^2 + s_3^2 + s_4^2 \geq s_1^2 + h_1^2 + 2s_n^2 \geq 39/25$, which together with $s_1 \leq 1/\sqrt{2}$ implies that we are in Case **(C 2)** of our algorithm. This is a contradiction, because we only construct subcontainers in Case **(C 3)**. Thus the following lemma, proved automatically using interval arithmetic, proves that these conditions are sufficient to ensure $S \geq 8/5$.

▶ **Lemma 15** (One Subcontainer, Automatic Analysis for Lemma 14)**.** *Let $z \coloneqq T(T^{-1}(s_1) + h_1)$. For all s_1, h_1, s_n with $0 < z < s_n \leq h_1 \leq s_1$, $h_1 \leq T^{-1}(s_1) + 1$ and*

$$(s_1 > 1/\sqrt{2}) \vee (w_1 < 2h_1) \vee \left(s_1^2 + h_1^2 + 2s_n^2 < 39/25\right),$$

we have $F_{SC_1}(s_1, h_1, s_n) > 8/5$. ◀

5.3.3 Analysis for two to four subcontainers

▶ **Lemma 16.** *If* Layer Packing *fails to pack a sequence $s_1, \dots, s_n$ of squares and* Bottom Packing *constructs $k \in \{2, 3, 4\}$ subcontainers, the total area of the squares exceeds $8/5$.*

Proof. We use similar ideas as in the proof of Lemma 14. We bound the area packed by Top Packing by $E(s_1, s_n)$ using Lemma 4. Furthermore, we use Corollary 10 to bound the area packed into each of the $k \in \{2, 3, 4\}$ subcontainers by

$$\|C_i\| \geq B_4\left(\left(T^{-1}(s_1) - \sum_{j=1}^{i-1} h_i\right), h_i, w_i, h_{i+1}\right), 1 \leq i \leq k,$$

where $h_{k+1} \coloneqq s_n$. We can express $w_i = w(T^{-1}(s_1) - \sum_{j=1}^{i-1} h_j, h_i)$ in terms of s_1 and $h_j, 1 \leq j \leq i$. In total, for k subcontainers, this yields the bound

$$S \geq F_{SC_k}(s_1, h_1, \dots, h_k, s_n) \coloneqq s_1^2 + s_n^2 + E(s_1, s_n) + \sum_{i=1}^{k} B_4\left(T^{-1}(s_1) - \sum_{j=1}^{i-1} h_i, h_i, w_i, h_{i+1}\right).$$

Finally, we know $0 < z < s_n$ because the algorithm fails to pack s_n. Thus, the following lemma, proved automatically using interval arithmetic, suffices to complete the proof of Lemma 16.

▶ **Lemma 17** (Automatic Analysis for Lemma 16). *Let* $z_k = T(-T^{-1}(s_1) + \sum_{i=1}^{k} h_i)$.

($k = 2$) *For all* s_1, h_1, h_2, s_n *with* $0 < z_2 < s_n \leq h_2 \leq h_1 \leq s_1$ *and* $0.295 \leq s_1 \leq \sqrt{8/5}$ *and* $h_1 + h_2 \leq 1 + T^{-1}(s_1)$, *we have* $F_{SC_2} > 8/5$.

($k = 3$) *For all* s_1, h_1, h_2, h_3, s_n *with* $0 < z_3 < s_n \leq h_3 \leq h_2 \leq h_1 \leq s_1$ *and* $0.295 \leq s_1 \leq \sqrt{8/5}$ *and* $h_1 + h_2 + h_3 \leq 1 + T^{-1}(s_1)$, *we have* $F_{SC_3} > 8/5$.

($k = 4$) *For all* $s_1, h_1, h_2, h_3, h_4, s_n$ *with* $0 < z_4 < s_n \leq h_4 \leq h_3 \leq h_2 \leq h_1 \leq s_1$ *and* $0.295 \leq s_1 \leq \sqrt{8/5}$ *and* $h_1 + h_2 + h_3 + h_4 \leq 1 + T^{-1}(s_1)$, *we have* $F_{SC_4} > 8/5$. ◀

Due to space constraints, we omit the detailed analysis for five or more subcontainers, which can be found in the full version of the paper. This completes the analysis of **(C 3)** and thus the proof of our main result.

6 Conclusion

We have established the critical density for packing squares into a disk: Any set of squares of total area at most $8/5$ can be packed into a unit disk. As shown by our lower bound example, this guarantee is best-possible, i.e., it cannot be improved. The proof is based on an algorithm that subdivides the disk into horizontal subcontainers and uses a refined shelf packing scheme. The correctness of this algorithm is shown by careful manual analysis, complemented by a computer-assisted part that is based on interval arithmetic.

There is a variety of interesting directions for future research. Of particular interest is the critical density for packing squares of bounded size into a disk, which will result in a higher packing density; a more general problem concerns the critical packing density for packing other types of objects of bounded size into other types of containers. Other questions arise from considering questions in three- or even higher-dimensional space. We are optimistic that many of our techniques will be useful for settling these problems.

References

1. Mikkel Abrahamsen, Tillmann Miltzow, and Nadja Seiferth. Framework for ∃ℝ-completeness of two-dimensional packing problems. *IEEE Symposium on Foundations of Computer Science (FOCS)*, 2020. accepted. `arXiv:2004.07558`.
2. K. Appel and W. Haken. Every planar map is four colorable. Part I. Discharging. *Illinois Journal of Mathematics*, 21:429–490, 1977.
3. K. Appel and W. Haken. Every planar map is four colorable. Part II. Reducibility. *Illinois Journal of Mathematics*, 21:491–567, 1977.
4. Archimedes. Measurement of a circle, 250 B.C.
5. Balázs Bánhelyi, Endre Palatinus, and Balázs L. Lévai. Optimal circle covering problems and their applications. *Central European Journal of Operations Research*, 23:815–832, 2015.
6. Aaron T. Becker, Sándor P. Fekete, Phillip Keldenich, Sebastian Morr, and Christian Scheffer. Packing Geometric Objects with Optimal Worst-Case Density. In *Symposium on Computational Geometry (SoCG)*, pages 63:1–63:6, 2019. Video available at `https://www.ibr.cs.tu-bs.de/users/fekete/Videos/PackingCirclesInSquares.mp4`. `doi:10.4230/LIPIcs.SoCG.2019.63`.
7. Károly Böröczky Jr. *Finite packing and covering*, volume 154. Cambridge University Press, 2004.

8 Peter Brass, William O.J. Moser, and János Pach. *Research problems in discrete geometry*. Springer Science & Business Media, 2006.

9 E. D. Demaine, S.P. Fekete, and R. J. Lang. Circle packing for origami design is hard. In *Origami^5: 5th International Conference on Origami in Science, Mathematics and Education*, AK Peters/CRC Press, pages 609–626, 2011. arXiv:1105.0791.

10 Gábor Fejes Tóth. Recent progress on packing and covering. *Contemporary Mathematics*, 223:145–162, 1999.

11 S. P. Fekete and J. Schepers. New classes of fast lower bounds for bin packing problems. *Mathematical Programming*, 91(1):11–31, 2001.

12 S. P. Fekete and J. Schepers. A general framework for bounds for higher-dimensional orthogonal packing problems. *Mathematical Methods of Operations Research*, 60:311–329, 2004.

13 Sándor P. Fekete, Utkarsh Gupta, Phillip Keldenich, Christian Scheffer, and Sahil Shah. Worst-case optimal covering of rectangles by disks. In *Symposium on Computational Geometry (SoCG)*, pages 42:1–42:23, 2020.

14 Sándor P. Fekete, Phillip Keldenich, and Christian Scheffer. Packing Disks into Disks with Optimal Worst-Case Density. In *Symposium on Computational Geometry (SoCG)*, pages 35:1–35:19, 2019. doi:10.4230/LIPIcs.SoCG.2019.35.

15 Sándor P. Fekete, Sebastian Morr, and Christian Scheffer. Split packing: Algorithms for packing circles with optimal worst-case density. *Discrete & Computational Geometry*, 61(3):562–594, 2019.

16 Michael R. Garey and David S. Johnson. Complexity results for multiprocessor scheduling under resource constraints. *SIAM Journal on Computing*, 4(4):397–411, 1975.

17 Thomas Hales, Mark Adams, Gertrud Bauer, Tat Dat Dang, John Harrison, Le Truong Hoang, Cezary Kaliszyk, Victor Magron, Sean McLaughlin, Tat Thang Nguyen, Quang Truong Nguyen, Tobias Nipkow, Steven Obua, Joseph Pleso, Jason Rute, Alexey Solovyev, Thi Hoai An Ta, Nam Trung Tran, Thi Diep Trieu, Josef Urban, Ky Vu, and Roland Zumkeller. A formal proof of the Kepler conjecture. *Forum of Mathematics, Pi*, 5:e2, 2017. doi:10.1017/fmp.2017.1.

18 Thomas C. Hales. A proof of the Kepler conjecture. *Annals of Mathematics*, 162(3):1065–1185, 2005.

19 Joel Hass and Roger Schlafly. Double bubbles minimize. *Annals of Mathematics*, 151(2):459–515, 2000.

20 Michael Hutchings, Frank Morgan, Manuel Ritoré, and Antonio Ros. Proof of the double bubble conjecture. *Annals of Mathematics*, 155(2):459–489, 2002.

21 J. Y. T. Leung, T. W. Tam, C. S. Wong, G. H. Young, and F. Y. L. Chin. Packing squares into a square. *Journal of Parallel and Distributed Computing*, 10(3):271–275, 1990.

22 A. Lodi, S. Martello, and M. Monaci. Two-dimensional packing problems: A survey. *European Journal of Operational Research*, 141(2):241–252, 2002.

23 J. W. Moon and L. Moser. Some packing and covering theorems. In *Colloquium Mathematicae*, volume 17, pages 103–110. Institute of Mathematics, Polish Academy of Sciences, 1967.

24 S. Morr. Split packing: An algorithm for packing circles with optimal worst-case density. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 99–109, 2017.

25 Wolfgang Mulzer and Günter Rote. Minimum-weight triangulation is NP-hard. *Journal of the ACM*, 55(2):11:1–11:29, 2008.

26 N. Robertson, D. Sanders, P. Seymour, and R. Thomas. The four-colour theorem. *Journal of Combinatorial Theory Series B*, 70:2–44, 1997.

27 George Szekeres and Lindsay Peters. Computer solution to the 17-point Erdős-Szekeres problem. *The ANZIAM Journal*, 48(2):151–164, 2006.

28 Gábor Fejes Tóth. Packing and covering. In *Handbook of Discrete and Computational Geometry, Third Edition*, pages 27–66. Chapman and Hall/CRC, 2017.

29 R. Wilson. *Four colours suffice: How the map problem was solved*. Princeton University Press, 2013.