Packing Squares into a Disk with Optimal Worst-Case Density

Sándor P. Fekete[1], Vijaykrishna Gurunathan[2], Kushagra Juneja[2], Phillip Keldenich[1], Linda Kleist[1], and Christian Scheffer[1]

1 **Department of Computer Science, TU Braunschweig, Germany**
{s.fektete, p.keldenich, l.kleist, c.scheffer}@tu-bs.de
2 **Department of Computer Science & Engineering, IIT Bombay, India**
krishnavijay1999@gmail.com, kuku12320@gmail.com

Abstract

We provide a tight result for a fundamental problem arising from packing squares into a circular container: The critical density of packing squares in 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 unit disk; in contrast, for any $\varepsilon > 0$ there are sets of squares of area $8/5 + \varepsilon$ that cannot be packed. This settles the last case of packing circular or square objects into a circular or square container, as the critical densities for squares in a square ($1/2$), circles in a square ($\pi/3+\sqrt{2} \approx 0.539$) and circles in a circle ($1/2$) have already been established. The proof uses a careful manual analysis, complemented by a minor automatic 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.

1 Introduction

Problems of geometric packing and covering arise in a wide range of natural applications. They also have a long history of spawning many extremely 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; 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 volume often does not suffice to decide feasibility of a set, making it necessary to provide an explicit packing or covering.

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 $\delta_p \leq 1$, such that any set S of squares with a total volume of at most δ_p can always be packed into a disk C of area 1, regardless of the individual sizes of the elements in S? We show that the correct answer is $\delta_p = 8/5\pi$: Any set of squares of total area at most $8/5$ can be packed into a unit disk (with radius 1), and for any value $A > 8/5$, there are sets that cannot be packed. This quantity is of mathematical importance, as it settles an open problem, as well as of algorithmic interest, because it provides a simple criterion for feasibility. It also settles the last remaining case of packing circular or square objects into a circular or square container, see Figure 1 for an overview.

1.1 Related Work

Problems of square packing have been studied for a long time. The decision problem whether it is possible to pack a given set of squares into the unit square was shown to be strongly

36th European Workshop on Computational Geometry, Würzburg, Germany, March 16–18, 2020.
This is an extended abstract of a presentation given at EuroCG'20. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

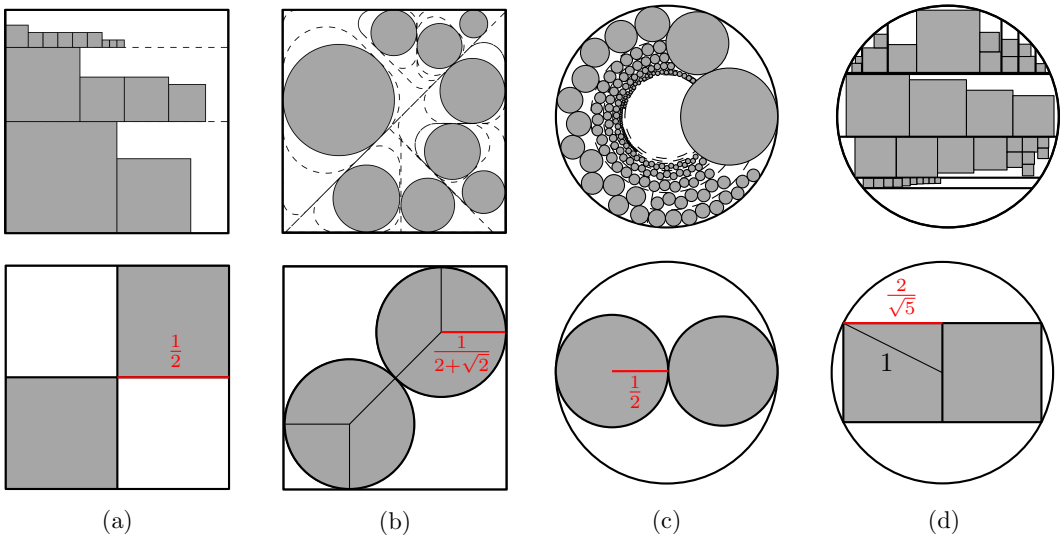

Figure 1 Illustration of the worst-case optimal approaches and worst case instances for packing (a) squares into a square with Shelf Packing by Moon and Moser [7]. (b) disks into a square by Fekete et al. [5]. (c) disks into a disk by Fekete et al. [4] (d) squares into a disk [this paper].

NP-complete by Leung et al. [6]. Already in 1967, Moon and Moser [7] proved that the critical packing density for squares into a square is $1/2$. As illustrated in Figure 1(a), this is best possible. Demaine, Fekete, and Lang [1] showed in 2010 that deciding whether a given set of disks can be packed into a unit square is NP-hard. Consequently, there is (most likely) no deterministic polynomial-time algorithm to decide whether a given set of disks can be packed into a given container. The problem of establishing the critical packing density for disks in a square was posed by Demaine, Fekete, and Lang [1] and resolved by Morr, Fekete and Scheffer [5, 8]. Using a recursive procedure for partitioning the container into triangular pieces, they proved that the critical packing density of disks in a square is $\pi/(3+2\sqrt{2})$. More recently, Fekete et al. [4] established the critical packing density of $1/2$ for packing disks into a disk by employing a number of algorithmic techniques in combination with interval arithmetic. 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 [2, 3], which shows how a *modified* volume for geometric objects can be computed, yielding good lower bounds for one- or higher-dimensional scenarios.

2 A Worst-Case Optimal Algorithm

The main result is a worst-case optimal algorithm for packing squares into a unit disk.

▶ **Theorem 2.1.** *Every set of squares with a total area of at most $8/5$ can be packed into a disk with radius 1. This is worst-case optimal, i.e., for every $\lambda > 8/5$ there exists a set of squares with a total area of λ that cannot be packed into the unit disk.*

A proof of Theorem 2.1 consists of (i) a class of instances that provide the upper bound of $8/5$ and (ii) an algorithm that achieves the lower bound by packing any set of squares with a total area of at most $8/5$ into the unit disk.

The upper bound is implied by any two squares with a side length of $\sqrt{4/5} + \varepsilon$, for arbitrary $\varepsilon > 0$, see Figure 1(d): When placed in the unit disk, either of them must contain the disk center in its interior, so both cannot be packed simultaneously.

In the following, we sketch a constructive proof for the lower bound by describing an algorithm that can pack any instance with total area $8/5$. Because our proof is constructive, it yields a constant-factor approximation algorithm for the smallest disk in which a given set of squares can be packed.

2.1 Description of the Algorithm

In the following, we consider a set of given squares with side lengths $s_1, \ldots, s_n$. We pack them in sequential order by decreasing size into the unit disk $\mathcal{D}$, and assume without loss of generality that $s_1 \geq \cdots \geq s_n$. Our algorithm distinguishes three types of instances:

1. All squares are small, i.e., $s_1 \leq 0.295$.
2. The first four squares are fairly large, i.e., $s_1 \leq \frac{1}{\sqrt{2}}$ and $s_1^2 + s_2^2 + s_3^2 + s_4^2 \geq \frac{8}{5} - \frac{1}{25}$.
3. All other cases.

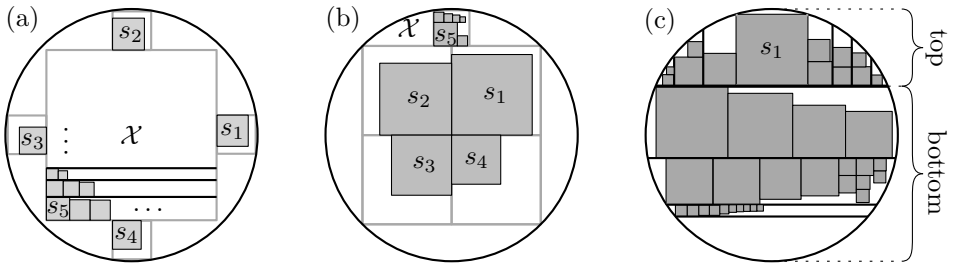

Figure 2 (a) Packing in case 1. (b) Packing in case 2. (c) The packing in the remaining cases is a combination of Top Packing (top) and Bottom Packing (bottom).

In the first case, we pack all but the first four squares into a large square container by Shelf Packing and each of the first four squares adjacent to one of the four sides as illustrated in Figure 2(a). In the second case, we pack the first four squares into a central square container, achieving high enough packed area that it suffices to pack the remaining squares into a smaller subsquare with the worst-case packing density of squares into a square. In the third case, we make extensive use of a refined shelf packing. Specifically, the largest square in the third case is packed into $\mathcal{D}$ as high as possible, see Figure 2(c) and Figure 3 for an illustration. The bottom of this square induces a horizontal split of disk into a *top* and a *bottom* part, which are then packed by two subroutines called Top Packing and Bottom Packing as described in Section 2.2. This yields the following description.

1. If $s_1 \leq 0.295$, place a square of side length $\mathcal{X} = 1.388$ concentric into $\mathcal{D}$ and place one square of side length $\mathcal{X}_i = 0.295$ to each side of $\mathcal{X}$, see Figure 2(a).
   - For $i = 1, 2, 3, 4$, pack each s_i into one of the squares of side length $\mathcal{X}_i = 0.295$.
   - For $i \geq 5$, use Shelf Packing for packing s_i into $\mathcal{X}$.
2. If $s_1 \leq \frac{1}{\sqrt{2}}$ and $s_1^2 + s_2^2 + s_3^2 + s_4^2 \geq \frac{39}{25}$, let $\mathcal{X}_1, \ldots, \mathcal{X}_4$ be the four equally sized maximal squares that fit into $\mathcal{D}$ and let be $\mathcal{X}$ the largest square that can be additionally packed into $\mathcal{D}$, see Figure 2(b).
   - For $i = 1, 2, 3, 4$, pack each s_i into one of the squares of side length $\mathcal{X}_i$.
   - For $i \geq 5$, use Shelf Packing for packing s_i into $\mathcal{X}$.

3. Otherwise
   - Pack s_1 as far as possible to the top into $\mathcal{D}$.
   - For $i \geq 2$,
     (3.1) if possible, use Top Packing for packing s_i,
     (3.2) otherwise, use Bottom Packing for packing s_i.

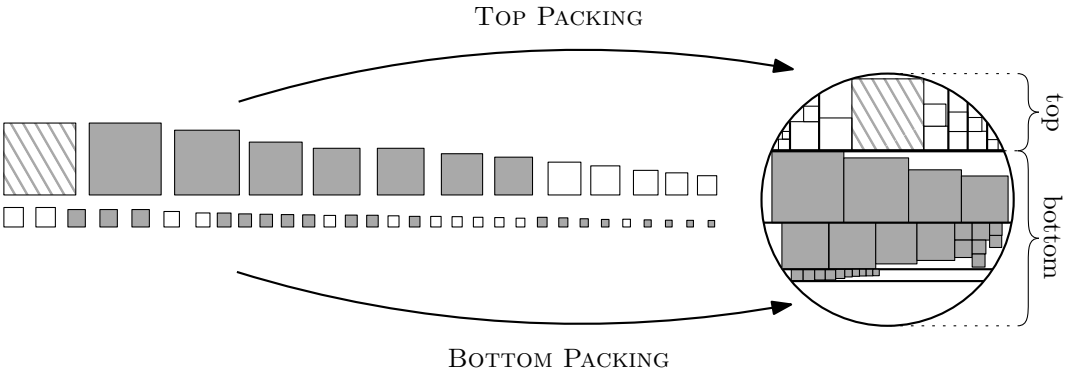

Figure 3 Our algorithm packs squares in decreasing order. The largest (hatched) square is packed as far as possible to the top, inducing a top and a bottom part, with the empty top space consisting of two congruent pockets. Subsequent (white) squares are packed into these top pockets with *Top Packing* (which uses 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 shelf packing within each slice.

2.2 Subroutines of Our Algorithm

In the following, we briefly describe the subroutines of our algorithm.

Refined Shelf Packing. In the classic shelf packing procedure by Moon and Moser [7], the objects are packed in the greedy manner by decreasing size in rectangular subcontainers called *shelves*; see top of Figure 1 (a). When an object does not fit in the current shelf, a new shelf is opened; the height of a shelf is determined by the first object that it accommodates. 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.

Top Packing. The first square s_1 is packed as high as possible into the disk $\mathcal{D}$, see Figure 4 (a). Then the horizontal line ℓ_1 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; each such pocket has two straight axis-parallel boundaries, b_x and b_y. We use refined shelf packing with shelves parallel to the shorter straight boundary, as shown in Figure 4 (c) and (d). If a square s_i does not fit into either pocket, it is packed into the part below ℓ_1.

Bottom Packing. For packing a squares in the bottom part of $\mathcal{D}$, SubContainer Slicing subdivides the unused portion of the container disk into smaller pieces, by using straight horizontal cuts analougous to shelf packing; see Figure 5 (Left). The height of a subcontainer is determined by the first packed square. Within each subcontainer, (vertical) Refined Shelf Packing is used; see Figure 3 for the overall picture. These shelves are

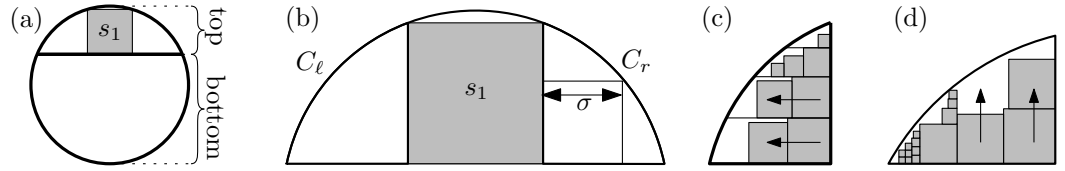

Figure 4 (a) Packing s_1 topmost into $\mathcal{D}$. (b) The top part of $\mathcal{D}$ with the pockets C_ℓ and C_r, and the size σ of the largest inscribed square. (c) A pocket C_ℓ where $b_x \leq b_y$, resulting in horizontal shelf packing. (d) A pocket C_ℓ where $b_x > b_y$, resulting in vertical shelf packing.

packed from the longer of the two horizontal cuts, i.e., away from the boundary that is closer to the disk center; see Figure 5 (Right) for packing the subcontainer.

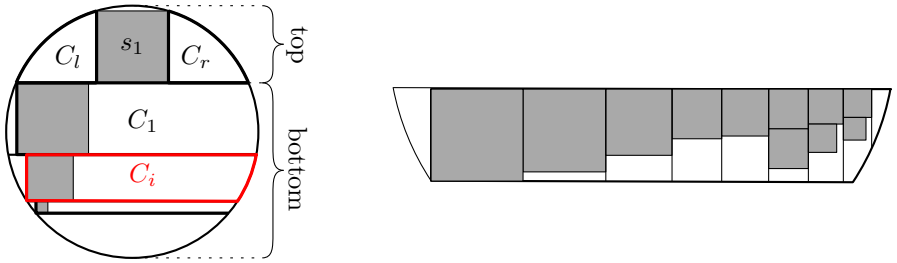

Figure 5 (Left) SubContainer Slicing partitions the lower part of $\mathcal{D}$ into subcontainers C_i, with the height corresponding to the first packed square. (Right) Within each subcontainer, SubContainer Packing places squares into C_i along vertical shelves, starting from the longer straight cut of the subcontainer.

2.3 Correctness of the Algorithm

Similar to the argument by Moon and Moser for squares packed into a square container, we use careful bookkeeping to prove that this algorithm only fails to pack a square in the decreasing if the total area of all squares exceeds the critical bound. The analysis uses an intricate combination of manual analysis and an automated analysis based on interval arithmetic. Details are omitted due to lack of space.

3 Complexity

We present the idea of an hardness proof for packing squares into a disk.

▶ **Theorem 3.1.** *It is* NP*-hard to decide whether a given set of squares fits into a disk.*

The proof uses a reduction from 3-Partition; it is somewhat similar to the one by Leung et al. [6] for deciding whether a given set of squares fits into a given square container, and the one by Demaine, Fekete, and Lang in 2010 [1] for deciding whether a give set of disks fits into a given square container; see Figure 6 for an illustration.

Eight (gray) *framing* squares can only be packed by leaving a central rectangular pocket P and some outside gaps. The numbers of the 3-Partition instance are mapped to a set of (red) *number* squares of almost equal size, with small modifications of size ε_i, such that a triple (i, j, k) of (red) number squares fits into P if and only if $\varepsilon_i + \varepsilon_j + \varepsilon_k \leq 0$, i.e., if there is a feasible 3-Partition. For filling the gaps outside the framing squares, a set of (yellow and

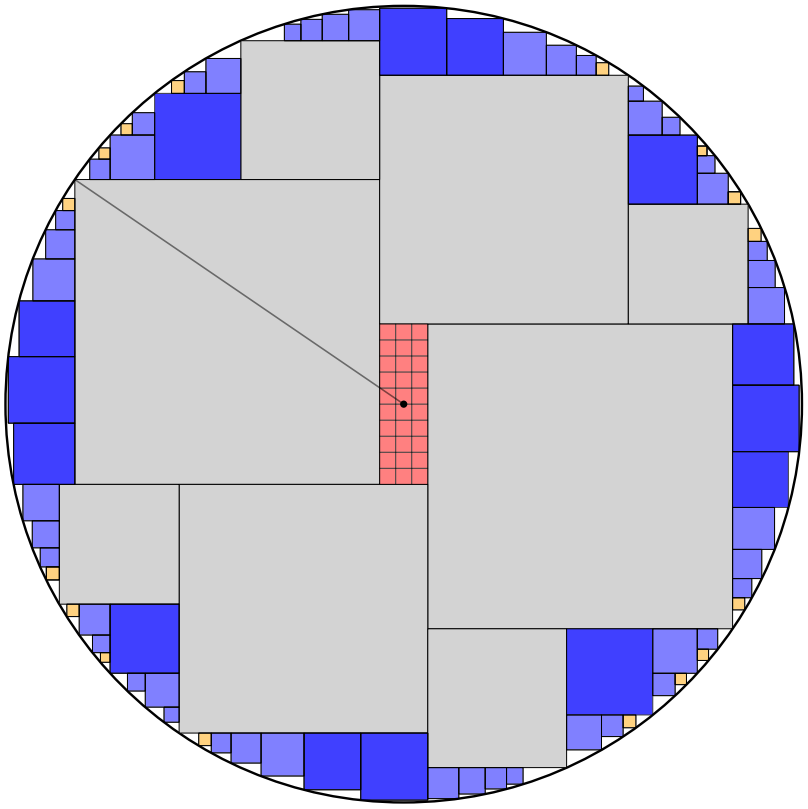

Figure 6 Illustration of the 3-PARTITION reduction.

blue) *filler* squares are constructed, so that no (red) number square can be packed outside P if all filler squares are packed outside P. A detailed proof establishes the following claims.

1. Up to symmetries, the framing squares can only be packed in one canonical way, leaving a central pocket P.
2. The filler squares fight tightly when packed in the canonical manner outside P.
3. When all filler squares are packed outside P, the number squares can only be packed into P. This is possible if and only if there is a feasible 3-partition.
4. Packing a filler square inside P forces an unpackable gap preventing a feasible packing.
5. The overall construction can be realized with squares of sufficiently approximated edge lengths of polynomial description size.

We omit details due to limited space, and the fact that the hardness proof is neither surprising nor central to this paper.

4 Conclusions

We have established the critical density for packing squares into a disk, based on a number of advanced techniques that are more involved than the ones used for packing squares or disks into a square. Numerous questions remain open, in particular the critical density for packing squares of bounded size into a disk. We are optimistic that our techniques will be useful.

References

1 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.

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

3 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.

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

5 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.

6 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.

7 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.

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