

# Tiling with Squares and Packing Dominos in Polynomial Time

Anders Aamand  

MIT, Cambridge, MA, US

Mikkel Abrahamsen  

BARC, University of Copenhagen, Denmark

Thomas Ahle  

BARC, University of Copenhagen, Denmark

Peter M. R. Rasmussen  

BARC, University of Copenhagen, Denmark

---

## Abstract

A polyomino is a polygonal region with axis-parallel edges and corners of integral coordinates, which may have holes. In this paper, we consider planar tiling and packing problems with polyomino pieces and a polyomino container  $P$ . We give polynomial-time algorithms for deciding if  $P$  can be tiled with  $k \times k$  squares for any fixed  $k$  which can be part of the input (that is, deciding if  $P$  is the union of a set of non-overlapping  $k \times k$  squares) and for packing  $P$  with a maximum number of non-overlapping and axis-parallel  $2 \times 1$  dominos, allowing rotations by  $90^\circ$ . As packing is more general than tiling, the latter algorithm can also be used to decide if  $P$  can be tiled by  $2 \times 1$  dominos.

These are classical problems with important applications in VLSI design, and the related problem of finding a maximum packing of  $2 \times 2$  squares is known to be NP-hard [J. Algorithms 1990]. For our three problems there are known pseudo-polynomial-time algorithms, that is, algorithms with running times polynomial in the *area* or *perimeter* of  $P$ . However, the standard, compact way to represent a polygon is by listing the coordinates of the corners in binary. We use this representation, and thus present the first polynomial-time algorithms for the problems. Concretely, we give a simple  $O(n \log n)$ -time algorithm for tiling with squares, where  $n$  is the number of corners of  $P$ . We then give a more involved algorithm that reduces the problems of packing and tiling with dominos to finding a maximum and perfect matching in a graph with  $O(n^3)$  vertices. This leads to algorithms with running times  $O(n^3 \frac{\log^3 n}{\log^2 \log n})$  and  $O(n^3 \frac{\log^2 n}{\log \log n})$ , respectively.

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Design and analysis of algorithms

**Keywords and phrases** packing, tiling, polyominoes

**Digital Object Identifier** 10.4230/LIPIcs.SoCG.2022.1

**Related Version** *Full Version*: <https://arxiv.org/abs/2011.10983>

**Funding** *Anders Aamand*: Supported by a DFF-International Postdoc Grant from the Independent Research Fund Denmark.

*Mikkel Abrahamsen*: Supported by Starting Grant 1054-00032B from the Independent Research Fund Denmark under the Sapere Aude research career programme. BARC is supported by the VILLUM Foundation grant 16582.

*Thomas Ahle*: BARC is supported by the VILLUM Foundation grant 16582.

*Peter M. R. Rasmussen*: BARC is supported by the VILLUM Foundation grant 16582.

## 1 Introduction

A chessboard has been mutilated by removing two diagonally opposite corners, leaving 62 squares. Philosopher Max Black asked in 1946 whether one can place 31 dominos of size  $1 \times 2$  so as to cover all of the remaining squares? Tiling problems of this sort are popular in



© Anders Aamand, Mikkel Abrahamsen, Thomas Ahle, and Peter M. R. Rasmussen; licensed under Creative Commons License CC-BY 4.0

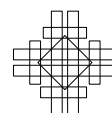
38th International Symposium on Computational Geometry (SoCG 2022).

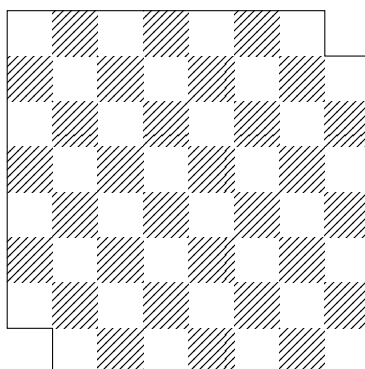
Editors: Xavier Goaoc and Michael Kerber; Article No. 1; pp. 1:1–1:17

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany





■ **Figure 1** The chessboard polyomino envisioned by Max Black.

recreational mathematics, such as the mathematical olympiads<sup>1</sup> and have been discussed by Golomb [14] and Gamow and Stern [12]. The mutilated chessboard and the dominos are examples of the type of polygon called a *polyomino*, which is a polygonal region of the plane with axis-parallel edges and corners of integral coordinates. We allow polyominos to have holes.

From an algorithmic point of view, it is natural to ask whether a given (large) polyomino  $P$  can be *tilled* by copies of another fixed (small) polyomino  $Q$ , which means that  $P$  is the union of non-overlapping copies of  $Q$  that may or may not be rotated by  $90^\circ$  and  $180^\circ$ . As the answer is often a boring *no*, one can ask more generally for the largest number of copies of  $Q$  that can be *packed* into the given container  $P$  without overlapping. Algorithms answering this question (for various  $Q$ ) turn out to have important applications in very large scale integration (VLSI) circuit technology. As a concrete example, Hochbaum and Maass [15] gave the following motivation for their development of a polynomial-time approximation scheme for packing  $2 \times 2$  squares into a given polyomino  $P$  (using the area representation of  $P$ , to be defined later).

“For example, 64K RAM chips, some of which may be defective, are available on a rectilinear grid placed on a silicon wafer.  $2 \times 2$  arrays of such nondefective chips could be wired together to produce 256K RAM chips. In order to maximize yield, we want to pack a maximal number of such  $2 \times 2$  arrays into the array of working chips on a wafer.”

Although the mentioned amounts of memory are small compared to those of present day technology, the basic principles behind the production of computer memory are largely unchanged, and methods for circumventing defective cells of wafers (the cells are also known as *dies* in this context) is still an active area of research in semiconductor manufacturing [7, 9, 17, 19].

The most important result in tiling is perhaps the combinatorial group theory approach by Conway and Lagarias [8]. Their algorithmic technique is used to decide whether a given finite region consisting of cells in a regular lattice (triangular, square, or hexagonal) can be tiled by pieces drawn from a finite set of tile shapes. Thurston [25] gives a nice introduction to the technique and shows how it can be used to decide if a polyomino without holes can

<sup>1</sup> See e.g. the “hook problem” of the International Mathematical Olympiad 2004.

be tiled by dominos. The running time is  $O(a \log a)$ , where  $a$  is the *area* of  $P$ . Pak, Sheffer, and Tassy [22] described an algorithm with running time  $O(p \log p)$ , where  $p$  is the *perimeter* of  $P$ .

The problem of *packing* a maximum number of dominos into a given polyomino  $P$  was apparently first analyzed by Berman, Leighton, and Snyder [5] who observed that this problem can be reduced to finding a maximum matching of the incidence graph  $G(P)$  of the cells in  $P$ : There is a vertex for each  $1 \times 1$  cell in  $P$ , and two vertices are connected by an edge if the two cells share a geometrical edge. The graph  $G(P)$  is bipartite, so the problem can be solved in  $O(a^{3/2})$  time using the Hopcroft–Karp algorithm, where  $a$  is the number of cells (i.e., the area of  $P$ ).

On the flip-side, a number of hardness results have been obtained for simple tiling and packing problems: Beauquier, Nivat, Remila, and Robson [2] showed that if  $P$  can have holes, the problem of deciding if  $P$  can be tiled by translates of two rectangles  $1 \times m$  and  $k \times 1$  is NP-complete as soon as  $\max\{m, k\} \geq 3$  and  $\min\{m, k\} \geq 2$ . Pak and Yang [23] showed that there exists a set of at most  $10^6$  rectangles such that deciding whether a given *hole-free* polyomino can be tiled with translates from the set is NP-complete. Other generalizations have even turned out to be undecidable: Berger [3] proved in 1966 that deciding whether pieces from a given finite set of polyominoes can tile the plane is Turing-complete (interestingly, Wijshoff and van Leeuwen [26] and Beauquier and Nivat [1] gave algorithms for deciding whether a single polyomino tiles the plane). For packing, Fowler, Paterson, and Tanimoto [11] showed already in the early 80s that deciding whether a given number of  $3 \times 3$  squares can be packed into a polyomino (with holes) is NP-complete, and the result was strengthened to  $2 \times 2$  squares by Berman, Johnson, Leighton, Shor, and Snyder [4].

As it turns out, for all of the above results, it is assumed that the container  $P$  is represented either as a list of the individual cells forming the interior of  $P$  or as a list of the boundary cells. We shall call these representations the *area representation* and *perimeter representation*, respectively. The area and perimeter representations correspond to a unary rather than binary representation of integers and the running times of the existing algorithms are thus only pseudo-polynomial. It is much more efficient and compact to represent  $P$  by the coordinates of the corners, where the coordinates are represented as binary numbers. This is the way one would usually represent polygons (with holes) in computational geometry: The corners are given in cyclic order as they appear on the boundary of  $P$ , one cycle for the outer boundary and one for each of the holes of  $P$ . We shall call such a representation a *corner representation*. With a corner representation, the area and perimeter can be exponential in the input size, so the known algorithms which rely on an area or perimeter representation to be polynomial, are in fact exponential when using this more efficient encoding of the input. Problems that are NP-complete in the area or perimeter representation are also NP-hard in the corner representation, but NP-membership does not necessarily follow. In our practical example of semiconductor manufacturing, the corner representation also seems to be the natural setting for the problem: Hopefully, there are only few defective cells to be avoided when grouping the chips, so the total number of corners of the usable region is much smaller than its area.

El-Khechen, Dulieu, Iacono, and Van Omme [10] showed that even using a corner representation for a polyomino  $P$ , the problem of deciding if  $m$  squares of size  $2 \times 2$  can be packed into  $P$  is in NP. That was not clear before since the naive certificate specifies the placement of each of the  $m$  squares, and so, would have exponential length. Beyond this, we know of no other work using the corner representation for polyomino tiling or packing problems.



**Our contribution**

While the complexity of the problem of packing  $2 \times 2$  squares into a polyomino  $P$  has thus been settled as NP-complete, the complexity of the tiling problem was left unsettled. Tiling and packing are closely connected in this area of geometry, but their complexities can be drastically different. Indeed, we show in Section 3 that it can be decided in  $O(n \log n)$  time by a surprisingly simple algorithm whether  $P$  can be tiled by  $k \times k$  squares for any fixed  $k \in \mathbb{N}$  which can even be part of the input. Here,  $n$  is the number of corners of  $P$ .<sup>2</sup> With the area and perimeter representations, it is trivial to decide if  $P$  can be tiled in polynomial time (see Section 3), but as noted above, using the corner representation, it is not even immediately obvious that the problem is in NP.

In Section 4, we provide and analyze a simple algorithm, which we denote **simple-packer**, that can decide if  $m$  dominos (i.e., rectangles of size  $1 \times 2$  that can be rotated  $90^\circ$ ) can be packed in a given polyomino  $P$ . The algorithm works by truncating long edges of  $P$ , so that the resulting polyomino  $P''$  has area  $O(n^4)$ . The graph  $G(P'')$  induced by the unit square cells constituting  $P''$  can likewise be constructed in time  $O(n^4)$ . We then use a multiple-sink multiple-source maximum flow algorithm as a black box [6, 13] to find a maximum matching in  $G(P'')$ , which results in a running time of  $O(n^4 \frac{\log^3 n}{\log^2 \log n})$ . In order to decide if  $P$  can be tiled with dominos, we can instead use a single-source shortest path algorithm [21], with which one can find perfect matchings in bipartite planar graphs [20], and we obtain a slightly better running time of  $O(n^4 \frac{\log^2 n}{\log \log n})$ . Although the truncation process of reducing the size to  $O(n^4)$  is simple, the proof of correctness is nontrivial and requires some structural lemmas on domino packings.

In the full version of this paper, we manage to reduce the domino packing and tiling problems to finding a maximum and perfect matching in a bipartite planar graph  $G^*$  with  $O(n^3)$  vertices, instead of  $O(n^4)$  as for **simple-packer**. We denote this algorithm **fast-packer**. The actual graph  $G^*$  can also be constructed in time  $O(n^3)$ . This reduction relies on the same structural results as are needed for **simple-packer**, but it is however quite a bit more complicated, and many techniques and technical lemmas are required to prove correctness and bound the size of  $G^*$ . We obtain running times of  $O(n^3 \frac{\log^3 n}{\log^2 \log n})$  and  $O(n^3 \frac{\log^2 n}{\log \log n})$  for packing and tiling, respectively. Table 1 summarises the known and new results.

■ **Table 1** Complexities of the four fundamental tiling and packing problems. Here,  $n$  is the number of corners of the container  $P$ . The algorithm for tiling with squares works for any size  $k \times k$ .

Shapes	Tiling	Packing
	$O(n^3 \frac{\log^2 n}{\log \log n})$ [This paper]	$O(n^3 \frac{\log^3 n}{\log^2 \log n})$ [This paper]
	$O(n \log n)$ [This paper]	NP-complete [4, 10]

<sup>2</sup> We assume throughout the paper that we can make basic operations (additions, subtractions, comparisons) on the coordinates in  $O(1)$  time. Otherwise, the time complexity of our square tiling algorithm will be  $O(nt \log n)$  and the domino packing algorithm will have complexity  $O(n^3 t + n^3 \frac{\log^3 n}{\log^2 \log n})$ , where  $t$  is the time it takes to make one such operation.

## Open problems

Many interesting questions remain for slightly more complex shapes than studied in this paper. For instance, polynomial-time algorithms are known for tiling polyominoes with larger rectangles in the area representation [18, 24]. Are there also algorithms in the corner representation? For the problems that are NP-complete in the area representation [2, 16, 23], which are also contained in NP in the corner representation?

Another interesting problem is to design domino tiling and packing algorithms with better running times than our  $\tilde{O}(n^3)$  in the corner representation, e.g., near-linear time algorithms. It seems conceivable that the techniques of [22] can lead to such improvements for tiling *simply connected* polyominoes with dominos. Specifically, it is shown that to decide tilability of a simply connected polyomino  $P$ , it suffices to check a certain Lipschitz condition on a *height function* defined on the boundary  $\partial P$  of  $P$ , and that (essentially) this check can be carried out by considering only  $O(p)$  pairs of boundary points, where  $p$  is the perimeter of  $P$ . It is plausible that one could obtain a similar bound on the number of pairs to be checked in terms of  $n$ , which would lead to a faster domino tiling algorithm for hole-free polyominoes.

## 1.1 Our techniques

### Tiling with $k \times k$ squares

We sort the corners of the given polyomino  $P$  by the  $x$ -coordinates and use a vertical sweep-line  $\ell$  that sweeps over  $P$  from left to right. The intuition is that the algorithm keeps track of how the tiling looks in the region of  $P$  to the left of  $\ell$  if a tiling exists. As  $\ell$  sweeps over  $P$ , we keep track of how the tiling pattern changes under  $\ell$ . Each vertical edge of  $P$  that  $\ell$  sweeps over causes changes to the tiling, and we must update our data structure accordingly.

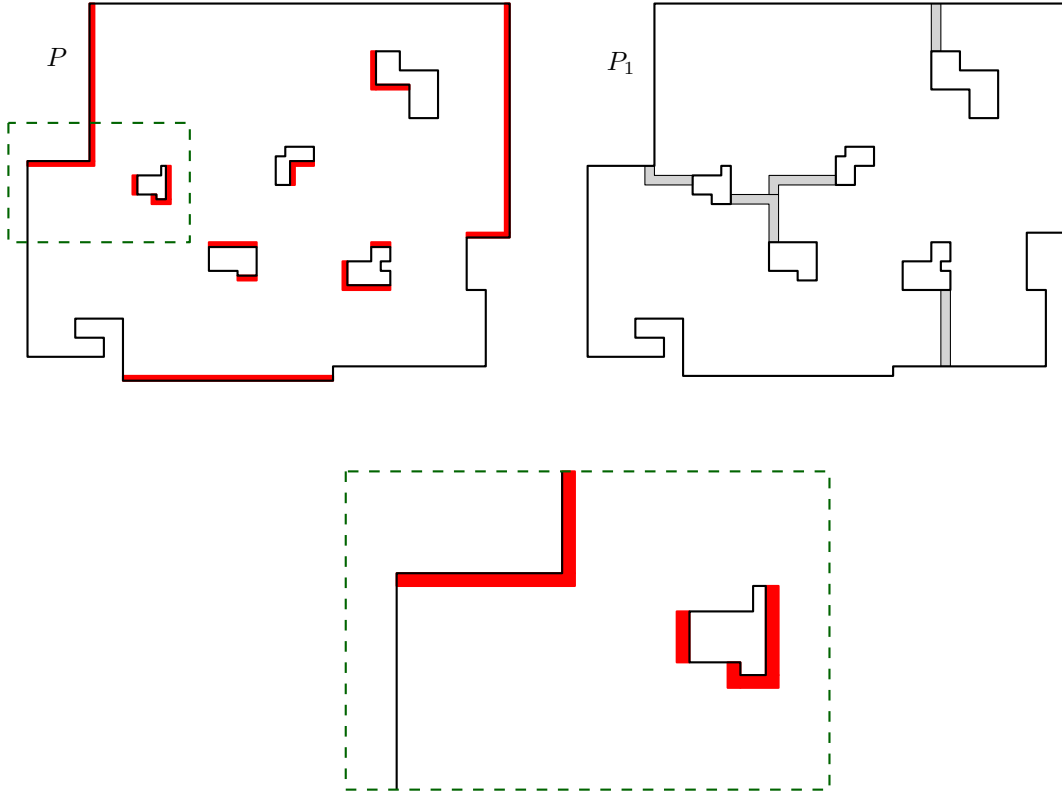
### Packing dominos

The basic approach of both the **simple-packer** and the **fast-packer** algorithm is to reduce the packing problem in the polyomino  $P$  (with  $n$  corners) to a maximum matching problem in a graph  $G^*$  with only polynomially many vertices and edges. We prove that a maximum matching in  $G^*$  corresponds to a maximum packing of dominos in  $P$ . The construction of  $G^*$  relies on some non-trivial structural results on domino packings.

The algorithm **simple-packer** first sorts the corners by  $x$ -coordinates and considers the corners in this order  $c_1, \dots, c_n$ . When  $x(c_{i+1}) - x(c_i) > 9n$ , we move all the corners  $c_{i+1}, \dots, c_n$  to the left by a distance of  $\approx x(c_{i+1}) - x(c_i) - 6n$ , so that the new distance is  $\approx 6n$ . We then do a similar truncation of vertical edges, and the resulting polyomino  $P''$  has area  $O(n^4)$ . We define  $G^*$  as the induced graph  $G^* := G(P'')$  and then compute a maximum matching  $M$  in  $G^*$  using a multiple-source multiple-sink maximum flow algorithm [6, 13]. The structural lemmas are used to ensure that the number of uncovered cells in maximum domino packings of the original polyomino  $P$  and the reduced  $P''$  are the same, so it follows that a maximum domino packing in  $P$  has size  $|M| + \frac{\text{area}(P) - \text{area}(P'')}{2}$ .

The algorithm **fast-packer** works by reducing the packing problem to finding a maximum matching in a bipartite planar graph  $G^*$  with  $O(n^3)$  vertices. The number of dominos in a maximum packing in the original polyomino  $P$  is then  $|M| + \frac{\text{area}(P) - |V(G^*)|}{2}$ , where  $M$  is a maximum matching in  $G^*$  and  $V(G^*)$  is the set of vertices of  $G^*$ . The construction of  $G^*$  requires many techniques and technical lemmas regarding the particular way we define intermediate polyominoes and graphs that are used to eventually arrive at  $G^*$ . The process

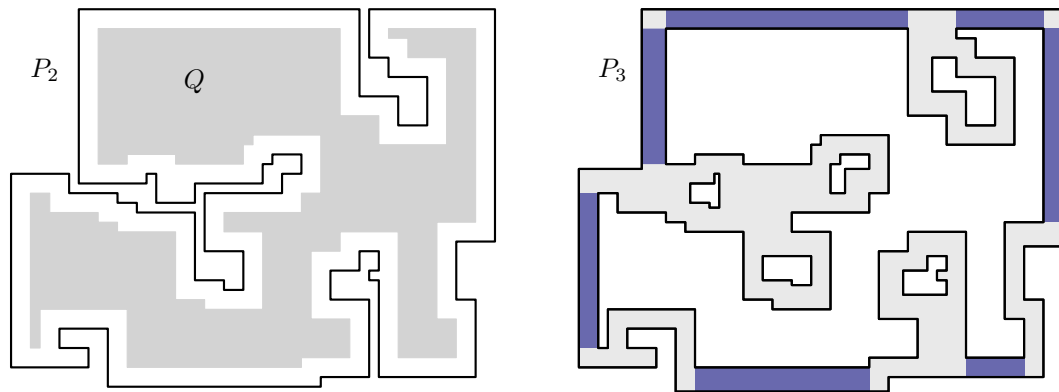
consists of five steps, and they are illustrated and described informally in Figures 2–4. We refer the reader to the full version for a detailed description of the steps and proofs that the algorithm works as claimed.



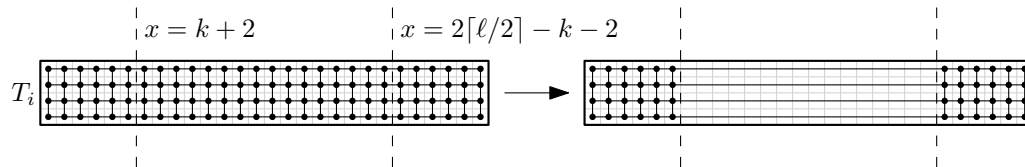
■ **Figure 2** Steps 1 and 2 made by the `fast-packer` algorithm. Top left: In step 1, we define  $P_1 \subseteq P$  to be the maximum subpolyomino with all corner coordinates even. The removed part  $P \setminus P_1$  is shown in red along the edges. Top right: In step 2, we carve channels from the holes of  $P_1$  to the outer boundary, to get a hole-free polyomino  $P_2$ . Bottom: Closeup of the region in the dashed rectangle.

## 2 Preliminaries

We define a *cell* to be a  $1 \times 1$  square of the form  $[i, i + 1] \times [j, j + 1]$ ,  $i, j \in \mathbb{Z}$ . A subset  $P \subseteq \mathbb{R}^2$  is called a *polyomino* if it is a finite union of cells. For a polyomino  $P$ , we define  $G(P)$  to be the graph which has the cells in  $P$  as vertices and an edge between two cells if they share a (geometric) edge. We say that  $P$  is *connected* if  $G(P)$  is a connected graph. Figure 5 (a) illustrates a connected polyomino. For a simple closed curve  $\gamma \subseteq \mathbb{R}^2$ , we denote by  $\text{Int } \gamma$  the interior of  $\gamma$ . An alternative way to represent a connected polyomino is by a sequence of simple closed curves  $(\gamma_0, \gamma_1, \dots, \gamma_h)$  such that (1) each of the curves follows the horizontal and vertical lines of the integral grid  $\mathbb{Z}^2$ , (2) for each  $i \in \{1, \dots, h\}$ ,  $\text{Int } \gamma_i \subseteq \text{Int } \gamma_0$ , (3) for each distinct  $i, j \in \{1, \dots, h\}$ ,  $\text{Int } \gamma_i \cap \text{Int } \gamma_j = \emptyset$ , and (4) for distinct  $i, j \in \{0, \dots, h\}$ ,  $\gamma_i \cap \gamma_j \subseteq \mathbb{Z}^2$ . For a connected polyomino  $P$ , there exists a unique such sequence (up to permutations of  $\gamma_1, \dots, \gamma_h$ ) with  $P = \overline{\text{Int } \gamma_0} \setminus (\bigcup_{i=1}^h \text{Int } \gamma_i)$ . It is standard to reduce our tiling and packing problems to corresponding tiling and packing problems for connected polyominoes, so for simplicity we will assume that the input polyominoes to our algorithms

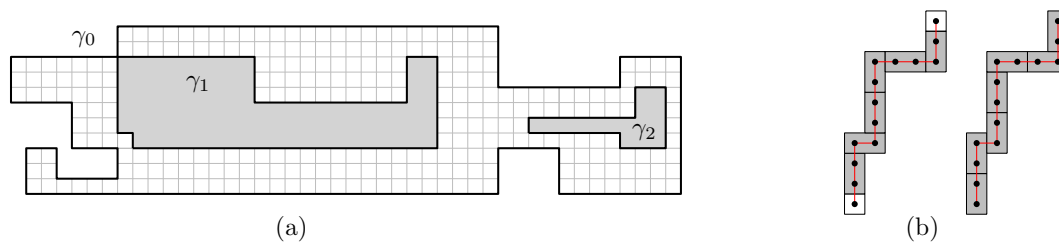


■ **Figure 3** Steps 3 and 4 made by the **fast-packer** algorithm, performed on the instance from Figure 2. Left: In step 3, we compute the grey region  $Q$ , which is obtained by offsetting the boundary of the hole-free polyomino  $P_2$  inwards by a distance of  $\Theta(n)$ . In this example,  $Q$  is connected, but that is in general not the case. Right: In step 4, we consider the polyomino  $P_3 := P \setminus Q$  in which all cells have distance  $O(n)$  to the boundary. We identify long pipes (the dark, blue rectangles). These may be exponentially long and must therefore be reduced.



■ **Figure 4** In step 5, the algorithm **fast-packer** reduces each pipe  $T_i$  of  $P_3$ , here of size  $\ell \times k$ , to a graph of polynomial size. The part of the induced graph  $G(T_i)$  in between the dashed vertical lines is replaced by long horizontal edges.

are connected. The *corners* of a polyomino  $P$  (specified by a sequence  $(\gamma_0, \gamma_1, \dots, \gamma_h)$ ), are the corners of the curves  $\gamma_0, \dots, \gamma_h$ . We assume that an input polyomino with  $n$  corners is represented using  $O(n)$  words of memory by describing the corners of each of the curves  $\gamma_0, \dots, \gamma_h$  in cyclic order.



■ **Figure 5** (a) A polyomino with two holes. (b) Extending a domino packing using an augmenting path in  $G(P)$ .

In this paper we will exclusively work with the  $L_\infty$ -norm when measuring distances. For two points  $a, b \in \mathbb{R}^2$  we define  $\text{dist}(a, b) = \|a - b\|_\infty$ . For two subsets  $A, B \subseteq \mathbb{R}^2$  we define

$$\text{dist}(A, B) = \inf_{(a,b) \in A \times B} \text{dist}(a, b).$$

In our analysis,  $A$  and  $B$  will always be closed and bounded (they will in fact be polyomios), and then the inf can be replaced by a min. Finally, we need the notion of the *offset*  $B(A, r)$  of a set  $A \subseteq \mathbb{R}^2$  by a value  $r \in \mathbb{R}$ . If  $r \geq 0$ , we define

$$B(A, r) := \{x \in \mathbb{R}^2 \mid \text{dist}(x, A) \leq r\},$$

and otherwise, we define  $B(A, r) := B(A^c, -r)^c$ . Note that if  $r \geq 0$ , we have  $A \subseteq B(A, r)$  and otherwise, we have  $B(A, r) \subseteq A$ .

Note that a domino packing of  $P$  naturally corresponds to a matching of  $G(P)$  and we will often take this viewpoint. We therefore require some basic matching terminology and a result on how to extend matchings. Let  $G$  be a graph and  $M$  a matching of  $G$ . A path  $(v_1, \dots, v_{2k})$  of  $G$  is said to be an *augmenting path* if  $v_1$  and  $v_{2k}$  are unmatched in  $M$  and for each  $1 \leq i \leq k-1$ ,  $v_{2i}$  and  $v_{2i+1}$  are matched to each other in  $M$ . Modifying  $M$  restricted to  $\{v_1, \dots, v_{2k}\}$  by instead matching  $(v_{2i-1}, v_{2i})$  for  $1 \leq i \leq k$ , we obtain a larger matching which now includes the two vertices  $v_1$  and  $v_{2k}$ . See Figure 5 (b) for an illustration in the context of domino packings. We require the following basic result by Berge which guarantees that any non-maximum matching of  $G$  can always be extended to a larger matching using an augmenting path as above.

► **Lemma 1** (Berge). *Let  $G$  be a graph and  $M$  a matching of  $G$  which is not maximum. Then there exists an augmenting path between two unmatched vertices  $G$ .*

### 3 Tiling with squares

#### 3.1 Naive algorithm

The naive algorithm to decide if  $P$  can be tiled with  $k \times k$  tiles works as follows. Consider any convex corner  $c$  of  $P$ . A  $k \times k$  square  $S$  must be placed with a corner at  $c$ . If  $S$  is not contained in  $P$ , we conclude that  $P$  cannot be tiled with  $k \times k$  squares. Otherwise, we recurse on the uncovered part  $P \setminus S$ . When nothing is left, we conclude that  $P$  can be tiled. This algorithm runs in time polynomial in the area of  $P$  and also shows that if  $P$  can be tiled, there is a unique way to do it.

#### 3.2 Sweep-line algorithm

For the ease of presentation, we focus on the case of deciding tileability using  $2 \times 2$  squares. It is straightforward to adapt the algorithm to decide tileability by  $k \times k$  squares for any fixed  $k \in \mathbb{N}$ , as explained in the full version.

Our algorithm for deciding if a given polyomino  $P$  can be tiled with  $2 \times 2$  squares uses a vertical sweep line that sweeps over  $P$  from left to right. The intuition is that the algorithm keeps track of how the tiling looks in the region of  $P$  to the left of  $\ell$  if a tiling exists. As  $\ell$  sweeps over  $P$ , we keep track of how the tiling pattern changes under  $\ell$ . Each vertical edge of  $P$  that  $\ell$  sweeps over causes changes to the tiling, and we must update our data structures accordingly.

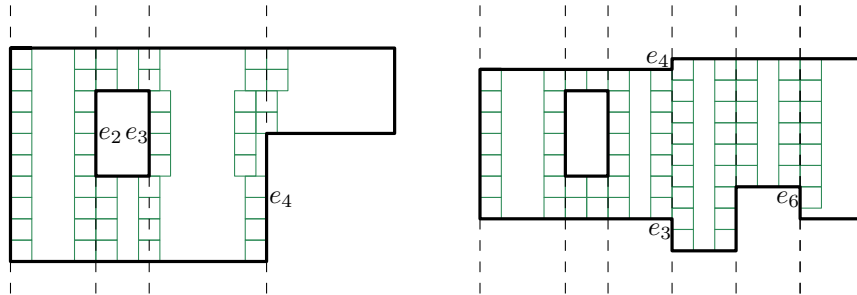
Recall that if  $P$  is tileable, then the tiling is unique. We define  $T(P) \subseteq P$  to be the union of the boundaries of the tiles in the tiling of  $P$ , i.e., such that  $P \setminus T(P)$  is a set of open  $2 \times 2$  squares. If  $P$  is not tileable, we define  $T(P) := \perp$ .

Consider the situation where the sweep line is some vertical line  $\ell$  with integral  $x$ -coordinate  $x(\ell)$ . The algorithm stores a set  $\mathcal{I}$  of pairwise interior-disjoint closed intervals  $\mathcal{I} = I_1, \dots, I_m \subseteq \mathbb{R}$ , ordered from bottom to top. Each interval  $I_i$  has endpoints at integers



and represents the segment  $I'_i := \{x(\ell)\} \times I_i$  on  $\ell$ . In the simple case that no vertical edge of  $P$  has  $x$ -coordinate  $x(\ell)$  (so that no change to the set  $P \cap \ell$  happens at this point), the intervals  $\mathcal{I}$  together represent the part of  $\ell$  in  $P$ , i.e., we have  $P \cap \ell = \bigcup_{i \in [m]} I'_i$ . If one or more vertical edges of  $P$  have  $x$ -coordinate  $x(\ell)$ , then  $P \cap \ell$  changes at this point and the intervals  $\mathcal{I}$  must be updated accordingly.

For each interval  $I_i$  we store a *parity*  $p(I_i) \in \{0, 1\}$ , which encodes how the tiling must be at  $I'_i$  if  $P$  is tileable. To make this precise, we state the following *parity invariant* of the algorithm under the assumption that  $P$  is tileable; see also Figure 6.



■ **Figure 6** Two instances that cannot be tiled. Left: The edge  $e_2$  splits the only interval in  $\mathcal{I}$  into two smaller intervals. Then  $e_3$  introduces a new interval with a different parity than the existing two. The edge  $e_4$  makes the algorithm conclude that  $P$  cannot be tiled since  $e_4$  overlaps an interval with the wrong parity. Right: The edges  $e_3$  and  $e_4$  introduce new intervals that are merged with the existing one. Edge  $e_6$  introduces an interval which is merged with the existing interval and the result has odd length, so the algorithm concludes that  $P$  cannot be tiled.

- If  $p(I_i)$  and  $x(\ell)$  have the same parity, then  $I'_i \subseteq T(P)$ , i.e.,  $I'_i$  follows the boundaries of some tiles and does not pass through the middle of any tile.
- Otherwise,  $I'_i \cap T(P)$  consists of isolated points, i.e.,  $I'_i$  passes through the middle of some of the tiles and does not follow the boundary of any tile.

We say that two neighboring intervals  $I_i, I_{i+1}$  of  $\mathcal{I}$  are *true neighbors* if  $I_i$  and  $I_{i+1}$  share an endpoint. In addition to the parity invariant, we require  $\mathcal{I}$  to satisfy the following *neighbor invariant*: Any pair of true neighbors of  $\mathcal{I}$  have different parity.

The pseudocode of the algorithm is shown in Algorithm 1. Initially, we sort all vertical edges after their  $x$ -coordinates and break ties arbitrarily. We then run through the edges in this order. Each edge makes a change to the set  $P \cap \ell$ , and we need to update the intervals  $\mathcal{I}$  accordingly so that the parity and the neighbor invariants are satisfied after each edge has been handled. Figure 6 shows examples of the two cases where the algorithm concludes that there is not tiling.

Using the parity and neighbour invariants, it is proven in the full version that the algorithm returns “tileable” if and only if  $P$  is tileable. Moreover, it is shown that the algorithm can be implemented to run in  $O(n \log n)$  time.

#### 4 Simple domino packing algorithm

In this section we will present our polynomial-time algorithm `simple-packer` for finding the maximum number of  $1 \times 2$  dominos that can be packed in a polyomino  $P$ . We assume that the dominos must be placed with axis-parallel edges, but they can be rotated by  $90^\circ$ . In any such packing, we can assume the pieces to have integral coordinates: if they do not, we

■ **Algorithm 1** Our simple sweep line algorithm for deciding if a polyomino (that may have wholes) can be tiled with  $2 \times 2$  square polyominos.

---

```

1 Let  $e_1, \dots, e_k$  be the vertical edges of  $P$  in sorted order.
2 for  $j = 1, \dots, k$  do
3   Let  $[y_0, y_1]$  be the interval of  $y$ -coordinates of  $e_j$ .
4   if the interior of  $P$  is to the left of  $e_j$ 
5     for each  $I_i \in \mathcal{I}$  that overlaps  $[y_0, y_1]$  do
6       if  $I_i$  and  $x(e_j)$  have different parity
7         return "no tiling"
8       Remove  $I_i$  from  $\mathcal{I}$ , let  $J := I_i \setminus [y_0, y_1]$ , and if  $J \neq \emptyset$ , add the interval(s) in
9          $J$  to  $\mathcal{I}$ .
9   else
10    Make a new interval  $I := [y_0, y_1]$  with the parity  $p(I) := x(e_j) \bmod 2$  and add
11     $I$  to  $\mathcal{I}$ .
12    if  $I$  has one or two true neighbors in  $\mathcal{I}$  that also have the same parity as  $I$ 
13      Merge those intervals in  $\mathcal{I}$ .
13  if  $j < k$  and  $x(e_{j+1}) > x(e_j)$  and some  $I_i \in \mathcal{I}$  has odd length
14    return "no tiling"
15 return "tileable"

```

---

can translate the pieces as far down and to the left as possible, and the corners will arrive at positions with integral coordinates. We first describe a naive algorithm which runs in polynomial time in the area of the polyomino.

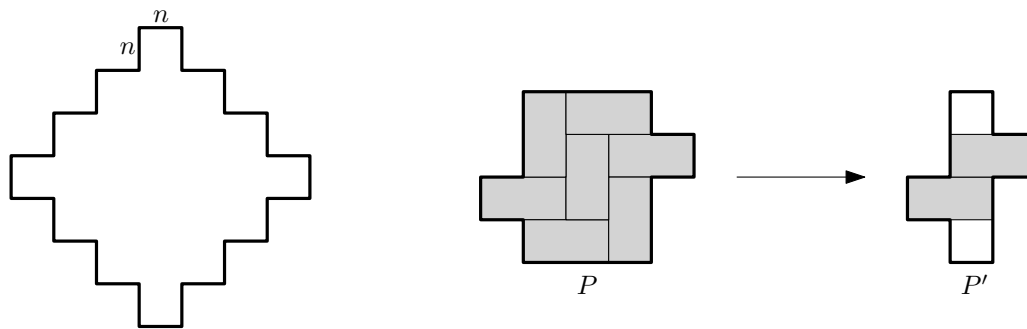
#### 4.1 Naive algorithm

The naive algorithm considers the graph  $G(P) = (V, E)$  where  $V$  is the set of cells of  $P$  and  $e = (u, v) \in E$  if and only if the two cells  $u$  and  $v$  have a (geometrical) edge in common. The maximum number of  $1 \times 2$  dominos that can be packed in  $P$  is exactly the size of a maximum matching of  $G$  and it is well known that such a maximum matching can be found in polynomial time in  $|V|$ , i.e., in the area of  $P$ .

#### 4.2 Simple polynomial-time algorithm

Our polynomial-time algorithm, **simple-packer**, first sorts the corners of  $P$  by  $x$ -coordinates and consider the corners in this order  $c_1, \dots, c_n$ . When  $x(c_{i+1}) - x(c_i) > 9n$ , we move all the corners  $c_{i+1}, \dots, c_n$  to the left by a distance of  $2 \lfloor \frac{x(c_{i+1}) - x(c_i)}{2} \rfloor - 6n$ . We call this operation a *contraction*. The result after all of the contractions is a polyomino  $P'$  with the parities of the  $x$ -coordinates unchanged and with the difference between the  $x$ -coordinates of any two consecutive corners at most  $6n$ . We then consider the corners in order according to  $y$ -coordinates and do a similar truncation of the long vertical edges. We have now reduced the container  $P$  to an orthogonal polygon  $P''$  of area at most  $O(n^4)$ , since the span of the  $x$ -coordinates is  $O(n^2)$ , as is the span of the  $y$ -coordinates. We proceed by finding maximum or perfect matchings in  $G(P'')$ , as described in the introduction.

For some containers  $P$ , the graph  $G(P'')$  really has  $\Omega(n^4)$  vertices, so **simple-packer** is indeed slower than **fast-packer**. For instance when the boundary of  $P$  consists of four "staircases", each consisting of  $n/4$  vertices, where each step has width and height  $n$ ; see Figure 7 (left). Here, **fast-packer** will remove most of the interior, leaving a layer of cells of thickness  $O(n)$  around the boundary, but **simple-packer** will not make any contractions.



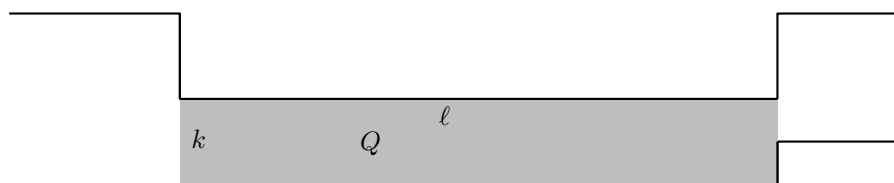
■ **Figure 7** Left: A polyomino with area  $\Omega(n^4)$  that `simple-packer` will not reduce. Right: If we truncate edges so that consecutive  $x$ -coordinates have difference either 1 or 2 (keeping the parities invariant), then there may be more uncovered cells in a maximum packing of the reduced instance than in the original.

One might be tempted to think that we can even truncate the edges so that the difference between consecutive  $x$ - and  $y$ -coordinates is either 1 or 2, keeping the parity of all coordinates. However, this does not work, as seen in Figure 7 (right). Two dominos can be packed in the reduced container  $P'$ , and the reduction decreases the area by eighth cells, so the formula would give that the original container  $P$  has room for six dominos, but there is actually room for seven.

### 4.3 Structural results on polyominoes and domino packings

Building up to our structural results on domino packings, we require a few definition and simple lemmas. We first introduce the notion of a *pipe* (see Figure 8) and *consistent parity*.

► **Definition 2.** Let  $P$  and  $Q$  be polyominoes with  $Q \subseteq P$ . We say that  $Q$  is a pipe of  $P$  if  $Q$  is rectangular and both vertical edges of  $Q$  or both horizontal edges of  $Q$  are contained in edges of  $P$ . The width of the pipe is the distance between this pair of edges. The length of the pipe is the distance between the other pair of edges. We say that a pipe is long if its length is at least 3 times its width.

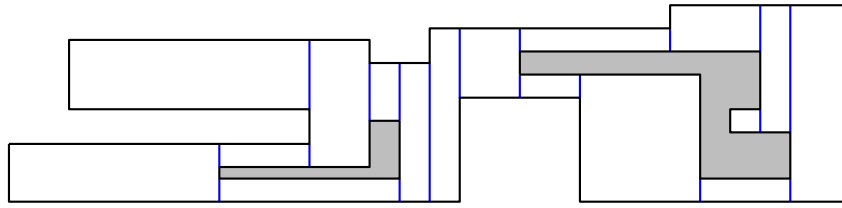


■ **Figure 8** A pipe  $Q$  of width  $k$  and length  $\ell$ .

► **Definition 3.** We say that a polyomino  $P$  has consistent parity if all first coordinates of the corners of  $P$  have the same parity and likewise for the second coordinates. Equivalently,  $P$  has consistent parity if there exists an open  $2 \times 2$  square,  $S$ , such that for all choices of integers  $i, j$  and  $S' = S + (2i, 2j)$ , either  $S' \subseteq P$  or  $S' \cap P = \emptyset$ .

Variations of the following lemma are well-known. We present a proof for completeness.

► **Lemma 4.** Let  $P$  be an orthogonal polygon with  $n$  corners and  $h$  holes.  $P$  can be divided into at most  $n/2 + h - 1$  rectangular pieces by adding only vertical line segments to the interior of  $P$ . If  $P$  is a polyomino, the rectangular pieces can be chosen to be polyominoes too.



■ **Figure 9** A partition of a polyomino with two holes into rectangles using vertical line segments (blue).

**Proof.** For each concave corner of the polygon we add a vertical line segment in the interior of the polygon starting from that corner and going upwards or downwards (depending on the rotation of the given corner). This is illustrated in Figure 9. Let  $s$  be the number of line segments added. It is easy to check that this gives a partition of  $P$  into exactly  $s - h + 1$  rectangles. With  $h$  holes, the number of concave corners is  $n/2 + 2(h - 1)$ , so also  $s \leq n/2 + 2(h - 1)$  and the result follows. ◀

Note that for a polygon with  $n$  corners,  $h \leq (n - 4)/4$ , so we have the following trivial corollary.

► **Corollary 5.** *The number of rectangular pieces in Lemma 4 is at most  $\frac{3}{4}n - 2$ .*

We next show that the property of consistent parity is preserved under integral offsets.

► **Lemma 6.** *Let  $P$  be a polyomino. If  $P$  has consistent parity, then  $B(P, 1)$  and  $B(P, -1)$  have consistent parity.*

**Proof.** Suppose  $P$  has consistent parity. Let  $S$  be a  $2 \times 2$  square as in Definition 3. Define  $S_1 = S + (1, 1)$ . It is easy to check that for all choices of integers  $i, j$  and  $S'_1 := S_1 + (2i, 2j)$ , either  $S'_1 \subseteq B(P, 1)$  or  $S'_1 \cap B(P, 1) = \emptyset$ . Thus  $B(P, 1)$  has consistent parity. The argument that  $B(P, -1)$  has consistent parity is similar. ◀

► **Lemma 7.** *Let  $P$  be a connected polyomino of consistent parity and without holes. Define  $L_1 = B(P, 1) \setminus P$  and  $L_{-1} = P \setminus B(P, -1)$ . Then  $G(L_1)$  and  $G(L_{-1})$  both have a Hamiltonian cycle of even length.*

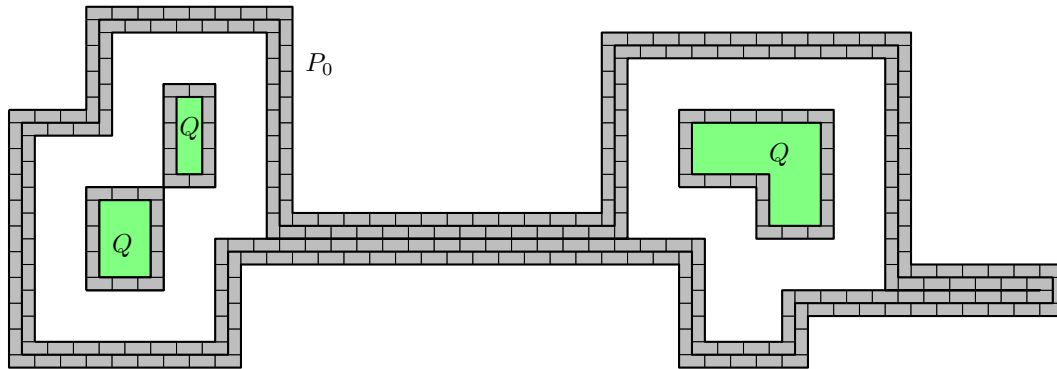
**Proof.** To obtain a Hamiltonian cycle of  $G(L_1)$ , we can simply trace  $P$  around the outside of its boundary, visiting all cells of  $L_1$  in a cyclic order. The corresponding closed trail of  $G(L_1)$  visits each vertex at least once. The assumption of consistent parity is easily seen to imply that we in fact visit each vertex exactly once, so the obtained trail is a Hamiltonian cycle. The graph  $G(L_1)$  is bipartite, so the cycle has even length. The argument that  $G(L_{-1})$  has a Hamiltonian cycle of even length is similar. ◀

With the above in hand, we are ready to state and prove our main structural results on domino packings. They are presented in Lemma 8 and Lemma 10.

► **Lemma 8.** *Let  $P$  and  $P_0$  be polyominoes such that  $P_0 \subseteq P$ ,  $P_0$  has no holes, and  $P_0$  has consistent parity. Let the total number of corners of  $P$  and  $P_0$  be  $n$ . Define  $r = \lfloor \frac{3}{8}n \rfloor$  and  $Q = B(P_0, -r)$ . There exists a maximum packing of  $P$  with  $1 \times 2$  dominos which restricts to a tiling of  $Q$ .*

Let us briefly pause to explain the importance of Lemma 8. Suppose that  $P$  contains a region  $Q$  as described. Then Lemma 8 tells us that *any* domino tiling of  $Q$  can be extended to a maximum domino packing of  $P$ . We can thus disregard  $Q$  and focus on finding a maximum packing of  $P \setminus Q$ , thus reducing the problem to a smaller instance. This is one of our key tools for reducing the size of the original polyomino  $P$  to a matching problem of polynomial size.

**Proof.** It follows from Lemma 6 that  $Q$  has consistent parity, and it can thus be tiled with  $2 \times 2$  squares and hence with dominos. Let  $\mathcal{Q}$  be a tiling of  $Q$ .



■ **Figure 10** The polyomino  $P_0$  and the offset  $Q$  (shown in green). The figure also illustrates the “layers”  $A_i$  and their domino tilings,  $\mathcal{A}_i$ .

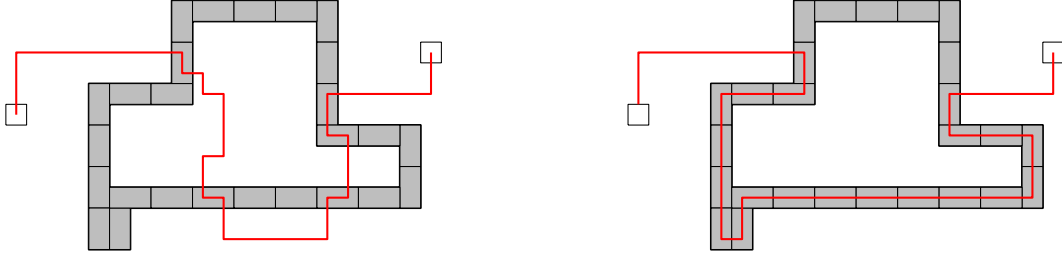
Define  $R = P \setminus P_0$  and note that  $R$  has at most  $n$  corners. It follows from Corollary 5 that  $R$  can be partitioned into less than  $\frac{3}{4}n$  rectangular polyominos. Each of these rectangles has a domino packing with at most one uncovered cell (which happens when the total number of cells in the rectangle is odd). Fix such a packing  $\mathcal{R}$  of the rectangles of  $R$  with dominos.

We next describe a tiling of  $P_0 \setminus Q$  as follows. For integers  $1 \leq i \leq r$  we define,  $A_i = B(P_0, -i + 1) \setminus B(P_0, -i)$ . Intuitively, we can construct  $Q$  from  $P_0$  by peeling off the “layers”  $A_i$  of  $P_0$  one at a time. Let  $i \in \{1, \dots, r\}$  be fixed. As  $P_0$  has consistent parity, it follows from Lemma 6 that  $B(P_0, -i + 1)$  has consistent parity. It is also easy to check that  $B(P_0, -i + 1)$  has no holes either, and it then follows from Lemma 7 that each connected component of  $G(A_i)$  has a Hamiltonian cycle of even length. These cycles give rise to a natural tiling of  $A_i$ ; if  $(v_1, \dots, v_{2k})$  is the sequence of cells corresponding to such a cycle, then  $\{v_1 \cup v_2, v_3 \cup v_4, \dots, v_{2k-1} \cup v_{2k}\}$  is a tiling of the cells of the cycle, and the union of such tilings over all connected components in  $G(A_i)$  gives a tiling of  $A_i$  with dominos. Denote this tiling by  $\mathcal{A}_i$ . See Figure 10 for an illustration of this construction.

Combining the tilings  $\mathcal{A}_1, \dots, \mathcal{A}_r$  and  $\mathcal{Q}$  with the packing  $\mathcal{R}$ , we obtain a domino packing,  $\mathcal{P}$ , of  $P$  where at most  $\frac{3}{4}n$  cells of  $P$  are uncovered. We now wish to extend this packing to a maximum packing in a way where we do not alter the tiling  $\mathcal{Q}$  of  $Q$ . If we can do this, the result will follow. Let  $M$  be the matching corresponding to  $\mathcal{P}$  in  $G(P)$ . We make the following claim.

▷ **Claim 9.** Let  $k \leq r$ . Suppose that the matching  $M$  can be extended to a matching of size  $|M| + k$ . Then this extension can be made using a sequence  $C_1, \dots, C_k$  of  $k$  augmenting paths one after the other (that is,  $C_i$  is an augmenting path *after* the matching has been extended using  $C_1, \dots, C_{i-1}$ ) such that for each  $i \in \{1, \dots, k\}$ , we have that  $C_i$  only uses vertices of  $G(R \cup \bigcup_{j=1}^i A_j)$ .

Before proving this claim, we first argue how the result follows. Since there are less than  $\frac{3}{4}n$  unmatched vertices in  $M$ , we can extend  $M$  to a maximum matching using at most  $r = \lfloor \frac{3}{8}n \rfloor$  augmenting paths. By the claim, these paths can be chosen so that they avoid the vertices of  $G(Q)$ . In particular, we never alter the matching of  $G(Q)$ , so the final maximum matching restricted to  $G(Q)$  is just the tiling  $\mathcal{Q}$ .

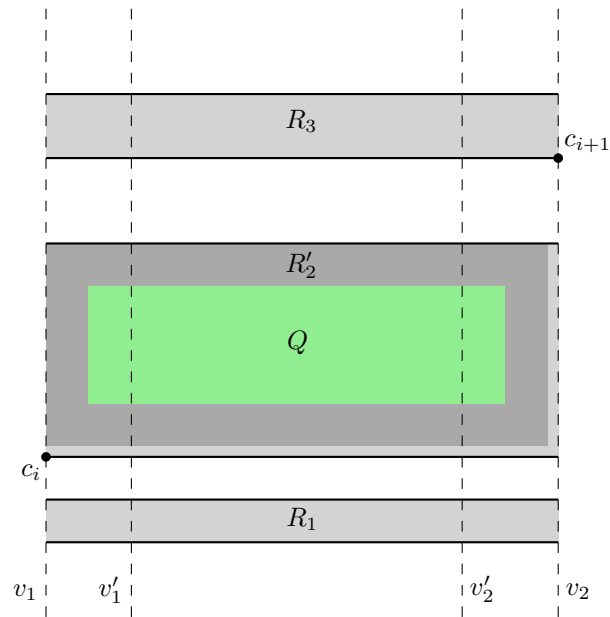


■ **Figure 11** Left: An alternating path between two unmatched vertices which enters a connected component of  $G(A_k)$ . Right: Modifying the alternating path using the the Hamiltonian cycle of the connected component.

We proceed to prove the claim by induction on  $k$ . The statement is trivial for  $k = 0$ , so let  $1 \leq k \leq r$  satisfy the assumptions of the claim and suppose inductively that  $C_1, \dots, C_{k-1}$  can be chosen such that for each  $i \in \{1, \dots, k-1\}$ , we have that  $C_i$  only uses vertices of  $G(R \cup \bigcup_{j=1}^i A_j)$ . After augmenting the matching using  $C_1, \dots, C_{k-1}$ , we have only modified the matching restricted to  $G(R \cup \bigcup_{j=1}^k A_j)$ . By Lemma 1, we can find an augmenting path  $C'_k$  connecting two unmatched vertices  $u, v$  of  $G(P)$ . We will modify  $C'_k$  to a path  $C_k$  with  $C_k \subseteq R \cup \bigcup_{j=1}^k A_j$ . Write  $C'_k : u = u_1, u_2, \dots, u_{2\ell} = v$ . Let  $D$  be a Hamiltonian cycle of one of the connected components of  $G(A_k)$ ; see Figure 11. If the path  $C'_k$  ever enters the vertices of  $D$ , we let  $i$  be minimal such that  $u_i \in D$  and  $j$  be maximal such that  $u_j \in D$ . We can now replace the subpath  $u_i, u_{i+1}, \dots, u_j$  of  $C'_k$  with part of the Hamiltonian cycle  $D$ . Whether we go clockwise or counterclockwise along  $D$  depends on whether  $u_i$  is matched with  $u_{i+1}$  in a clockwise or counterclockwise fashion in  $D$ . We do the same modification for every Hamiltonian cycle  $D$  corresponding to a connected component of  $G(A_k)$  that  $C'_k$  intersects. Note that each cycle  $D$  partitions the vertices  $G(P) \setminus D$  into an interior and an exterior part. Since  $P_0$  has no holes and  $u, v \in R$ , the original path  $C'_k$  enters  $D$  from the exterior at  $u_i$  and likewise leaves  $D$  into the exterior at  $u_j$ . Also note that  $Q$  is contained in the interior parts of the cycles of  $G(A_k)$ . It then follows that the final resulting path  $C_k$  avoids  $Q$  and  $A_j$  for  $j > k$ , so it is contained in  $R \cup \bigcup_{j=1}^k A_j$ . ◀

Lemma 8 allows us to “ignore” parts of the polyomino  $P$  with distance  $\Omega(n)$  to the boundary. In order to argue that the answer output by `simple-packer` is correct, we also need to argue that we can ignore long pipes (see Definition 2). This is what motivates the following lemma which intuitively yields a reduction for shortening long pipes. We defer the proof to the full version.

► **Lemma 10.** *Let  $k, \ell \in \mathbb{N}$  with  $\ell$  even. Let  $L \subseteq [-1, 0] \times [0, k]$ ,  $R \subseteq [\ell, \ell + 1] \times [0, k]$  be polyominoes and define  $P = L \cup R \cup ([0, \ell] \times [0, k])$ . Color the cells of the plane in a chessboard like fashion and let  $b$  and  $w$  be respectively the number of black and white cells contained in  $P$ . Assume without loss of generality that  $b \geq w$ . If  $\ell \geq 2k$ , then the number of uncovered cells in a maximum domino packing of  $P$  is exactly  $b - w$ . Moreover, there exists a maximum domino packing such that the rectangle  $[k + 1, \ell - k - 1] \times [0, k]$  is completely covered and all dominos intersecting the rectangle are horizontal.*



■ **Figure 12** A contraction made by the algorithm `simple-packer` with one fat and two skinny rectangles. The algorithm moves all corners  $c_{i+1}, \dots, c_n$  to the left, essentially contracting the area between the vertical lines  $v'_1$  and  $v'_2$  to nothing.

#### 4.4 Correctness of the algorithm

We now verify that the number of uncovered cells in maximum packings is invariant under a single contraction, and the correctness of the algorithm hence follows. To this end, suppose that  $x(c_{i+1}) - x(c_i) > 9n$ , so that we move the corners  $c_{i+1}, \dots, c_n$  to the left; see Figure 12. It is clear that a domino packing with exactly  $\ell$  uncovered cells after the contraction gives rise to a domino packing with exactly  $\ell$  uncovered cells before the contraction, simply by inserting extra horizontal dominos in the rectangles that were contracted. For the converse, let  $v_1$  and  $v_2$  be vertical lines with  $x$ -coordinates  $x(c_i)$  and  $x(c_{i+1})$ , respectively, and let  $V$  be the vertical strip bounded by  $v_1$  and  $v_2$ . The intersection  $P \cap V$  is a collection of disjoint rectangles  $R_1, \dots, R_k$  of width  $x(c_{i+1}) - x(c_i)$  and various heights. We define a rectangle  $R_i$  to be *fat* if its height is more than  $3n$ , and otherwise  $R_i$  is *skinny*. We now define a polyomino  $P_0$  in order to apply Lemma 8. For each fat rectangle  $R_i$ , we let  $R'_i \subseteq R_i$  be the maximum rectangle with even coordinates and add  $R'_i$  to  $P_0$ . As each rectangle  $R_i$  corresponds to exactly two horizontal edges, the number of rectangles  $k$  is at most  $n/4$  and in particular, the number of corners of  $P \setminus P_0$  is at most  $2n$ . Letting  $Q := B(P_0, -\lfloor 3n/2 \rfloor)$ , we get from Lemma 8 that there exists a maximum packing of  $P$  that restricted to  $Q$  is a tiling.

We define  $P_1 := P \setminus Q$  and observe that the contraction corresponds to contracting a set of long pipes in  $P_1$ . These pipes are the skinny rectangles  $R_i$  and the parts of the fat rectangles vertically above and below the removed part  $Q$ . We therefore get from Lemma 10 that a maximum packing before the contraction having  $\ell$  uncovered cells, gives rise to a packing of the contracted polyomino with exactly  $\ell$  uncovered cells.

## References

- 1 Danièle Beauquier and Maurice Nivat. On translating one polyomino to tile the plane. *Discrete & Computational Geometry*, 6:575–592, 1991. doi:10.1007/BF02574705.
- 2 Danièle Beauquier, Maurice Nivat, Eric Remila, and Mike Robson. Tiling figures of the plane with two bars. *Computational Geometry*, 5(1):1–25, 1995. doi:10.1016/0925-7721(94)00015-N.
- 3 Robert Berger. The undecidability of the domino problem. *Memoirs of the American Mathematical Society*, 1(66), 1966. doi:10.1090/memo/0066.
- 4 Fran Berman, David Johnson, Tom Leighton, Peter W. Shor, and Larry Snyder. Generalized planar matching. *Journal of Algorithms*, 11(2):153–184, 1990. doi:10.1016/0196-6774(90)90001-U.
- 5 Francine Berman, Frank Thomson Leighton, and Lawrence Snyder. Optimal tile salvage, 1982. Technical report, Purdue University, Department of Computer Sciences, <https://docs.lib.purdue.edu/cgi/viewcontent.cgi?article=1321&context=cstech>.
- 6 Glencora Borradaile, Philip N. Klein, Shay Mozes, Yahav Nussbaum, and Christian Wulff-Nilsen. Multiple-source multiple-sink maximum flow in directed planar graphs in near-linear time. *SIAM Journal on Computing*, 46(4):1280–1303, 2017. doi:10.1137/15M1042929.
- 7 Chen-Fu Chien, Shao-Chung Hsu, and Jing-Feng Deng. A cutting algorithm for optimizing the wafer exposure pattern. *IEEE Transactions on Semiconductor Manufacturing*, 14(2):157–162, 2001.
- 8 J.H Conway and J.C Lagarias. Tiling with polyominoes and combinatorial group theory. *Journal of Combinatorial Theory, Series A*, 53(2):183–208, 1990. doi:10.1016/0097-3165(90)90057-4.
- 9 Dirk K. de Vries. Investigation of gross die per wafer formulas. *IEEE Transactions on Semiconductor Manufacturing*, 18(1):136–139, 2005.
- 10 Dania El-Khechen, Muriel Dulieu, John Iacono, and Nikolaj Van Omme. Packing  $2 \times 2$  unit squares into grid polygons is NP-complete. In *Proceedings of the 21st Canadian Conference on Computational Geometry (CCCG 2009)*, pages 33–36, 2009.
- 11 Robert J. Fowler, Michael S. Paterson, and Steven L. Tanimoto. Optimal packing and covering in the plane are NP-complete. *Information processing letters*, 12(3):133–137, 1981.
- 12 George Gamow and Marvin Stern. *Puzzle-math*. Macmillan, 1958.
- 13 Pawel Gawrychowski and Adam Karczmarsz. Improved bounds for shortest paths in dense distance graphs. In *45th International Colloquium on Automata, Languages, and Programming (ICALP 2018)*, pages 61:1–61:15, 2018. doi:10.4230/LIPIcs.ICALP.2018.61.
- 14 S. W. Golomb. Checker boards and polyominoes. *The American Mathematical Monthly*, 61(10):675–682, 1954. doi:10.1080/00029890.1954.11988548.
- 15 Dorit S. Hochbaum and Wolfgang Maass. Approximation schemes for covering and packing problems in image processing and VLSI. *Journal of the ACM*, 32(1):130–136, 1985.
- 16 Takashi Horiyama, Takehiro Ito, Keita Nakatsuka, Akira Suzuki, and Ryuhei Uehara. Complexity of tiling a polygon with trominoes or bars. *Discret. Comput. Geom.*, 58(3):686–704, 2017. doi:10.1007/s00454-017-9884-9.
- 17 S. Jang, J. Kim, T. Kim, H. Lee, and S. Ko. A wafer map yield prediction based on machine learning for productivity enhancement. *IEEE Transactions on Semiconductor Manufacturing*, 32(4):400–407, 2019.
- 18 C. Kenyon and R. Kenyon. Tiling a polygon with rectangles. In *Proceedings of the 33rd Annual Symposium on Foundations of Computer Science (FOCS 1992)*, pages 610–619, 1992.
- 19 Hanno Melzner and Alexander Olbrich. Maximization of good chips per wafer by optimization of memory redundancy. *IEEE Transactions on Semiconductor Manufacturing*, 20(2):68–76, 2007.
- 20 Gary L. Miller and Joseph Naor. Flow in planar graphs with multiple sources and sinks. *SIAM Journal on Computing*, 24(5):1002–1017, 1995. doi:10.1137/S0097539789162997.



- 21 Shay Mozes and Christian Wulff-Nilsen. Shortest paths in planar graphs with real lengths in  $O(n \log^2 n / \log \log n)$  time. In *16th Annual European Symposium on Algorithms (ESA 2010)*, 2010. doi:10.1007/978-3-642-15781-3\_18.
- 22 Igor Pak, Adam Sheffer, and Martin Tassy. Fast domino tileability. *Discrete & Computational Geometry*, 56(2):377–394, 2016. doi:10.1007/s00454-016-9807-1.
- 23 Igor Pak and Jed Yang. Tiling simply connected regions with rectangles. *Journal of Combinatorial Theory, Series A*, 120(7):1804–1816, 2013. doi:10.1016/j.jcta.2013.06.008.
- 24 Eric Rémila. Tiling a polygon with two kinds of rectangles. *Discrete Comput. Geom.*, 34(2):313–330, 2005. doi:10.1007/s00454-005-1173-3.
- 25 William P. Thurston. Conway’s tiling groups. *The American Mathematical Monthly*, 97(8):757–773, 1990.
- 26 H.A.G. Wijshoff and J. van Leeuwen. Arbitrary versus periodic storage schemes and tessellations of the plane using one type of polyomino. *Information and Control*, 62(1):1–25, 1984. doi:10.1016/S0019-9958(84)80007-8.