Tiling with Squares and Packing Dominos in Polynomial Time

Anders Aamand* Mikkel Abrahamsen* Thomas D. Ahle*
Peter M. R. Rasmussen*

November 9, 2020

Abstract

We consider planar tiling and packing problems with polyomino pieces and a polyomino container P. A polyomino is a polygonal region with axis parallel edges and corners of integral coordinates, which may have holes. We give two polynomial time algorithms, one for deciding if P can be tiled with 2×2 squares (that is, deciding if P is the union of a set of non-overlapping copies of the 2×2 square) and one for packing P with a maximum number of non-overlapping and axis-parallel 2×1 dominos, allowing rotations of $90°$. As packing is more general than tiling, the latter algorithm can also be used to decide if P can be tiled by 2×1 dominos.

These are classical problems with important applications in VLSI design, and the related problem of finding a maximum packing of 2×2 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* 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 $O(n \log n)$ algorithm for tiling with squares, and a more involved $O(n^4)$ algorithm for packing and tiling with dominos.

*Basic Algorithms Research Copenhagen (BARC), University of Copenhagen. BARC is supported by the VILLUM Foundation grant 16582.

1 Introduction

A chessboard had been mutilated by removing two diagonally opposite corners, leaving 62 squares. Philosopher Max Black asked in 1946 whether one could place 31 dominoes of size 1×2 so as to cover all of the remaining squares? Tiling problems of this sort are popular in recreational mathematics, such as the mathematical olympiads[1] and have been discussed by Golomb [11] and Gamow & Stern [10]. 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.

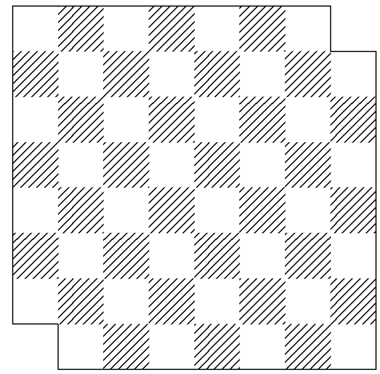


Figure 1: The chessboard polyomino envisioned by Max Black.

From an algorithmic point of view, it is natural to ask whether a given (large) polyomino P can be *tiled* 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° and 180°. 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 & Maass [13] gave the following motivation for their development of a polynomial time approximation scheme for packing 2×2 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. 2×2 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 2×2 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 [5, 7, 14, 17].

The most important result in tiling is perhaps the combinatorial group theory approach by Conway & Lagarias [6]. 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 [21] gives a nice introduction to the technique and shows how it can be used to decide if a polyomino without holes can be tiled by dominos. The running time is linear in the *area* of P.

The problem of *packing* a maximum number of dominos into a given polyomino P was apparently first analyzed by Berman, Leighton, & Snyder [4] 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 1×1 cell in P, and two vertices are connected if the two cells share a geometrical edge. The

[1] See e.g. the "hook problem" of the International Mathematical Olympiad 2004.

graph $G(P)$ is bipartite, so the problem can be solved in $O(n^{3/2})$ time using the Hopcroft–Karp algorithm, where n 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, & Robson [1] 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 & Yang [18] 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 be undecidable: Berger [2] proved in 1966 that deciding whether pieces from a given finite set of polyominos can tile the plane is Turing complete. For packing, Fowler, Paterson, & Tanimoto [9] showed already in the early 80s that deciding whether a given number of 3×3 squares can be packed into a polyomino (with holes) is NP-complete, and the result was strengthened to 2×2 squares by Berman, Johnson, Leighton, Shor, & Snyder [3].

As it turns out, for all of the above results, it was assumed that the container P is represented either as a sequence of the individual cells forming the interior of P or as a sequence of vertical and horizontal edges of length 1 forming the boundary of P. 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 relying on area or perimeter representations are insufficient. 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, & Van Omme [8] showed that even using a corner representation for a polymino P, the problem of deciding if k squares of size 2×2 can be packed into P is in NP. That was not clear before since the naive certificate specifies the placement of each of the k 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 2×2 squares into a polyomino P has thus been settled as NP-complete, it was left open whether the tiling problem was similarly hard. Tiling and packing are closely connected in this area of geometry, but their complexities can be drastically different. Indeed, we show in this paper that it can be decided in $O(n \log n)$ time by a surprisingly simple algorithm whether P can be tiled by 2×2 squares, where n is the number of corners of P.[2] With the area and perimeter representations, it is trivial to decide if P can be tiled

[2] We assume throughout the paper that we can make basic operations (additions, subtractions, comparisons) on the coordinates in $O(1)$ time. Otherwise, the time complexities of our two algorithms will be $O(nt \log n)$ and $O(n^3 t + n^4)$, respectively, where t is the time it takes to make one such operation.

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.

Furthermore, we provide an algorithm that can decide in $O(n^4)$ time if k dominos (i.e., rectangles of size 1×2 that can be rotated $90°$) can be packed in a given polyomino P. This algorithm is more complicated and we consider it our most important contribution. The algorithm implicitly constructs a maximum packing, and the same algorithm can be used to decide if P can be tiled by dominos. Table 1 summarises the known and new results.

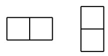
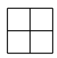
Shapes	Tiling	Packing
	$O(n^4)$ [This paper]	$O(n^4)$ [This paper]
	$O(n \log n)$ [This paper]	NP-complete [3, 8]

Table 1: Complexities of the four fundamental tiling and packing problems.

Further related work. The technique by Conway & Lagarias [6] has been adapted to obtain algorithms for tiling with other shapes than dominos: Kenyon & Kenyon [15] showed how to decide whether a given hole-free polyomino P can be tiled with translates of the rectangles $1 \times m$ and $k \times 1$ for fixed integers m and k. The running time is again linear in the area of P. They also described an algorithm to decide if a polyomino can be tiled by the rectangles $k \times m$ and $m \times k$ with running time quadratic in the area. Rémila [20] generalized the work by Kenyon and Kenyon and obtained a quadratic time algorithm for deciding whether a given hole-free polyomino can be tiled by translates of two fixed rectangles $k \times m$ and $k' \times m'$.

1.1 Our techniques

Tiling with 2×2 squares. We sort the corners of the given polyomino P by the x-coordinates and use a vertical sweep-line ℓ 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 ℓ if a tiling exists. As ℓ sweeps over P, we keep track of how the tiling pattern changes under ℓ. Each vertical edge of P that ℓ sweeps over causes changes to the tiling, and we must update our data structures accordingly.

Packing with dominos. Our basic approach is to reduce the polyomino P to a graph G^* of only $O(n^3)$ vertices and edges. We prove that a maximum matching in G^* corresponds to a maximum packing of dominos in P. The construction of G^* requires many techniques and the correctness relies on several structural results on domino packings and technical lemmas regarding the particular way we define the intermediate polyominos and graphs that are used to eventually arrive at G^*.

We first find the maximum subpolyomino $P_1 \subset P$ such that all corners of P_1 have even coordinates. We then use a hole-elimination technique: By carving channels in P_1 from the holes to the boundary, we obtain a hole-free subpolyomino $P_2 \subset P_1$. The particular way we choose the channels is important in order to ensure that the final graph G^* has size only $O(n^3)$. We now apply a technique of reducing P by removing everything far from the boundary of P_2: We consider the subpolyomino $Q \subset P_2$ of all cells with at least some distance $\Omega(n)$ to the boundary of P_2, and then we define $P_3 := P \setminus Q$ (note that Q is removed from P and not from P_2). The main insight is that any packing of dominos in P_3 can be extended to a packing of all of P that, restricted to Q, is a *tiling*. For this to hold, it turns out to be important that P_2 has no holes.

A crucial step is to prove that every cell in the polyomino P_3 has distance $O(n)$ to the boundary of P_3 and that P_3 has $O(n)$ corners. There may, however, still be an exponential number of cells in P_3 due to long *pipes* (corridors). We then develop a technique for contracting these long pipes. The contraction is not carried out geometrically, but in the incidence graph $G_3 := G(P_3)$ of the cells of P_3, by contracting long horizontal and vertical paths to single edges, and the resulting graph is G^*.

All vertices of G^* correspond to cells of P_3 with distance at most $O(n)$ from a corner of P_3, and since P_3 has $O(n)$ corners, we get that G^* has size $O(n^3)$. As we will see, it is easy to find a matching that covers all but $O(n)$ vertices of G^*. We then use Berge's lemma and make $O(n)$ breadth-first searches to find augmenting paths that extend the matching as much as possible. This results in a running time of $O(n^4)$.

2 Preliminaries

We define a *cell* to be a 1×1 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 (geometrical) edge. We say that P is *connected* if $G(P)$ is a connected graph. Figure 2 (a) illustrates a connected polyomino. For a simple closed curve $\gamma \subset \mathbb{R}^2$, we denote by $\operatorname{Int}\gamma$ the interior of γ. 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 $i \in \{1, \dots, h\}$, $\operatorname{Int}\gamma_i \subseteq \operatorname{Int}\gamma_0$, (3) for each distinct $i, j \in \{1, \dots, h\}$, $\operatorname{Int}\gamma_i \cap \operatorname{Int}\gamma_j = \emptyset$, 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 $P = \overline{\operatorname{Int}\gamma} \setminus (\bigcup_{i=1}^{h} \operatorname{Int}\gamma_i)$. It is standard to reduce our tiling and packing problems to corresponding tiling and packing problems for connected polyominos, so for simplicity we will assume that the input polyominos to our algorithms 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.

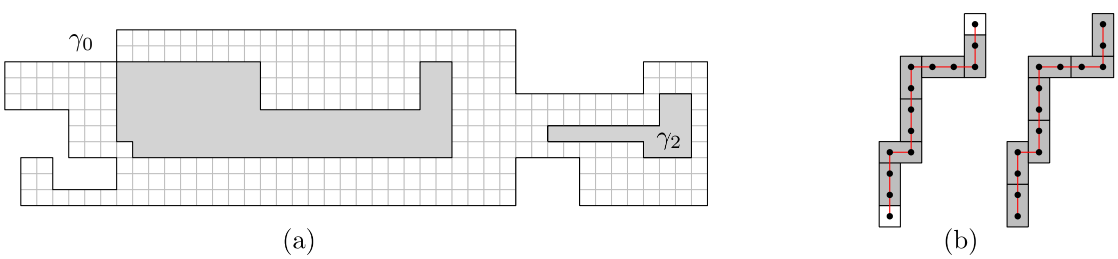


Figure 2: (a) A polyomino with two holes. (b) Extending a domino packing using an augmenting path in $G(P)$.

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

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

In our analysis, A and B will always be closed and bounded (they will in fact be polynomios), 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

$$B(A, r) := \left\{ x \in \mathbb{R}^2 \;\middle|\; \mathrm{dist}(x, A) \leq r \right\},$$

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

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 $(v_1, \ldots, v_{2k})$ 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 $\{v_1, \ldots, v_{2k}\}$ 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 2 (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 extending 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 2 by 2 squares

Naive algorithm. The naive algorithm to decide if P can be tiled with 2×2 tiles works as follows. Consider any convex corner c of P. A 2×2 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 2×2 squares. Otherwise, we recurse of 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.

Sweep line algorithm. Our algorithm for deciding if a given polyomino P can be tiled with 2×2 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 ℓ if a tiling exists. As ℓ sweeps over P, we keep track of how the tiling pattern changes under ℓ. Each vertical edge of P that ℓ 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 $T(P) \subset P$ 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 2×2 squares. If P is not tileable, we define $T(P) := \perp$.

Consider the situation where the sweep line is some vertical line ℓ with integral x-coordinate $x(\ell)$. The algorithm stores a set $\mathcal{I}$ of pairwise interior-disjoint closed intervals $\mathcal{I} = I_1, \ldots, I_m \subset \mathbb{R}$, ordered from below and up. Each interval I_i has endpoints at integers and represents the segment $I'_i := \{x(\ell)\} \times I_i$ on ℓ. 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 ℓ 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 3.

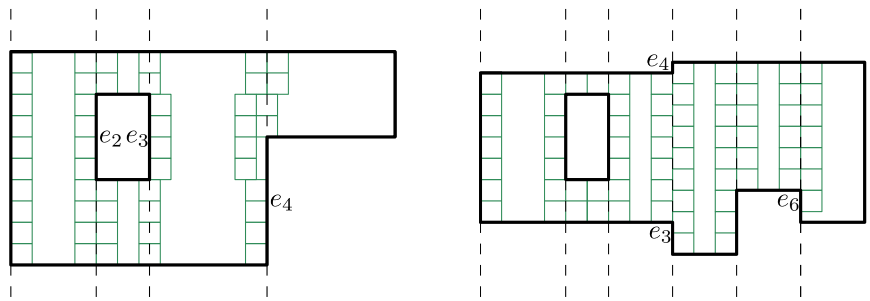


Figure 3: Two instances that cannot be tiled. Left: The edge e_2 splits the only interval in $\mathcal{I}$ into two smaller intervals. Then e_3 introduces a new interval with a different parity than the existing two. The edge e_4 makes the algorithm conclude that P cannot be tiled since e_4 overlaps 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 $I'_i \subset T(P)$, 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 3 shows the various events.

Consider the event that the sweep line ℓ reaches a vertical edge $e_j = \{x\} \times [y_0, y_1]$. If the interior of P is to the left of e_j, then $P \cap \ell$ shrinks. Each interval $I_i \in \mathcal{I}$ that overlaps $[y_0, y_1]$ must then also shrink, be split into two, or disappear from $\mathcal{I}$. This is handled by the for-loop at line 5. If the parity of one of these intervals I_i does not agree with the parity of e_j, we get from the parity invariant that P cannot be tiled, and hence the algorithm returns "no tiling" at line 7.

If on the other hand the interior of P is to the right of e_j, then $P \cap \ell$ expands and a new interval I must be added to $\mathcal{I}$. This is handled by the else-part at line 9. The new interval I may have one or two true neighbors in $\mathcal{I}$. If one or two such neighbors also have the same parity as I, we merge these intervals into one interval of $\mathcal{I}$. This ensures that the neighbor invariant is satisfied after e_j has been handled.

In line 13, we consider the case that we finished handling all vertical edges at some specific x-coordinate so that the sweep line will move to the right in order to handle the next edge e_{j+1} in the next iteration. If there is an interval I_i of odd length in $\mathcal{I}$, it follows from the parity invariant together with the neighbor invariant that P cannot be tiled, so the algorithm returns "no tiling" at line 14.

Algorithm 1:

```
1  Let e_1, ..., e_k be the vertical edges of P in sorted order.
2  for j = 1, ..., 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 ∈ 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 𝓘, let J := closure(I_i \ [y_0, y_1]), and if J ≠ ∅, add the interval(s) in J to 𝓘.
9      else
10         Make a new interval I := [y_0, y_1] with the parity p(I) := x(e_j) mod 2 and add I to 𝓘.
11         if I has one or two true neighbors in 𝓘 that also have the same parity as I
12             Merge those intervals in 𝓘.
13     if j < k and x(e_{j+1}) > x(e_j) and some I_i ∈ 𝓘 has odd length
14         return "no tiling"
15 return "tileable"
```

The above explanation of the algorithm argues that if the invariants hold before edge e_j is handled, they also hold after. It remains to argue that they also hold before the next edge e_{j+1} is handled in the case that the sweep line ℓ jumps to the right in order to sweep over e_{j+1}. In the open strip between the vertical lines containing e_j and e_{j+1}, there are no vertical segments of P. Hence, the pattern of the tiling $T(P)$ must continue as described by the parities $p(I_i)$ in between the edges e_j and e_{j+1}, so the parity invariant also holds before e_{j+1} is handled.

We already argued that if the algorithm returns "no tiling", then P is not tileable. Suppose on the other hand that the algorithm returns "tileable". In order to prove that P can then be tiled, we define for each $j \in [k]$ a polyomino $P_j \subset P$. We consider the situation where the sweep line ℓ contains e_j and e_j has just been handled by the algorithm. We then define P_j to be the union of

- the part of P to the left of ℓ, and
- the rectangle $[x(\ell), x(\ell)+1] \times I_i$ for each $I_i \in \mathcal{I}$ with a different parity than $x(\ell)$.

We first see that for each $j \in [k]$, we have $P_j \subset P$. To this end, we just have to check that the rectangles $[x(\ell), x(\ell)+1] \times I_i$ are in P. If one such interval was not in P, there would be an edge of P overlapping the segment $\{x(\ell)\} \times I_i$. Since I_i has a different parity than $x(\ell)$, this would make the algorithm report "no tiling" at line 7, contrary to our assumption.

We now prove by induction on j that each P_j can be tiled. Since $P = P_k$, this is sufficient. Along the way, we will also establish that $P_1 \subset P_2 \subset \ldots \subset P_k$. When $j = 1$, we see that P_j is empty, so the statement is trivial. Suppose now that P_j can be tiled and consider P_{j+1}. Note that if $x(e_j) = x(e_{j+1})$, so that ℓ does not move, then $P_j = P_{j+1}$, since all intervals that are created or modified when handling e_{j+1} have the same parity as $x(\ell)$, so in this case, P_{j+1} is tileable because P_j is.

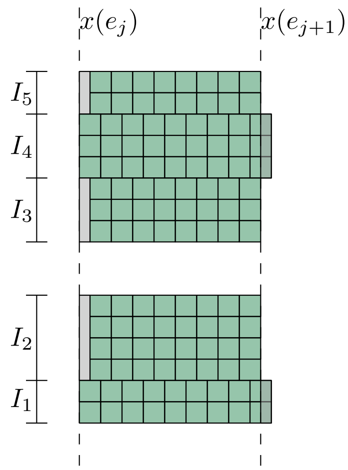


Figure 4: The polyomino P_j is the part of P to the left of the line $x = x(e_j)$ (this part of P_j is not shown) plus the grey rectangles along the line. Here, the difference $x(e_{j+1}) - x(e_j)$ is odd. The difference $\overline{P_{j+1} \setminus P_j}$ has been tiled with green 2×2 squares.

Consider now the case $x(e_j) < x(e_{j+1})$. Note that as $P_j \subset P$ and P_j is to the left of the vertical line $x = x(e_j) + 1$, we have $P_j \subset P_{j+1}$. We now consider the set $\overline{P_{j+1} \setminus P_j}$ and argue that it is tileable; see Figure 4. Let $I_1, \ldots, I_m$ be the intervals in $\mathcal{I}$ after e_j was handled. For each I_i, we add a rectangle $X \times I_i$ to P_j in order to obtain P_{j+1}, where $X \subset \mathbb{R}$ is an interval with lower endpoint $x(e_j)$ or $x(e_j) + 1$ and upper endpoint $x(e_{j+1})$ or $x(e_{j+1}) + 1$, and by the definition of P_j and P_{j+1}, it follows that X has even length. Since each I_i also has even length (otherwise, the algorithm would have returned "no tiling" at line 14 when e_j was handled), the difference $\overline{P_{j+1} \setminus P_j}$ is a union of rectangles with even edge lengths, so P_{j+1} is tileable since P_j is.

Runtime analysis. Assuming that we can compare two coordinates in $O(1)$ time, we sort the vertical edges by their x-coordinates in $O(n \log n)$ time. Since the intervals of $\mathcal{I}$ are pairwise interior-disjoint, we can implement $\mathcal{I}$ as a balanced binary search tree, where each leaf stores an interval I_i.

We now argue that each vertical edge e_j, with y-coordinates $[y_0, y_1]$, takes only $O(\log n)$ time to handle, since we need to make only $O(1)$ updates to $\mathcal{I}$. If the interior of P is to the left of e_j, then $[y_0, y_1] \subset \bigcup_{i \in [m]} I_i$. It then follows from the neighbor invariant that if $[y_0, y_1]$ overlaps more than one interval I_i, then the algorithm will return "no tiling". We therefore do at most $O(1)$ updates to $\mathcal{I}$, so it takes $O(\log n)$ time to handle e_j.

On the other hand, if the interior of P is to the right of e_j, we need to insert a new interval into $\mathcal{I}$ and possibly merge it with one or two neighbors in $\mathcal{I}$, so this also amounts to $O(1)$ changes to $\mathcal{I}$.

At line 13, we need to check the $O(1)$ intervals that were added or changed due to the edge e_j, so this can be done in $O(1)$ time. Hence, the algorithm has runtime $O(n \log n)$.

4 Packing dominos

In this section we will present our polynomial time algorithm for finding the maximum number of 1×2 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°. In any such packing, we can assume the pieces to have integral coordinates: if they do not, we can translate the pieces as far down and to the left as possible, and the corners will arrive at positions with integral coordinates. Again, we first describe a naive algorithm which runs in polynomial time in the area of the polyomino.

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 1×2 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.

Our goal is to find an algorithm running in polynomial time, even with the compact representation of P described in Section 2. In essence, we take the graph $G = G(P)$ above and construct from it a smaller graph, G^*, with $O(n^3)$ vertices. A maximum matching of G^* yields an (implicit) description of a maximum matching of G and we show that the maximum matching of G^* can be found in time $O(n^4)$.

4.1 Polynomial-time algorithm

We will next describe the steps of our algorithm for finding the maximum domino packing of a polyomino P. We first introduce the notion of a *pipe* (see Figure 5) and *consistent parity.*

Definition 1. Let P and Q be polyominos with $Q \subset P$. 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.

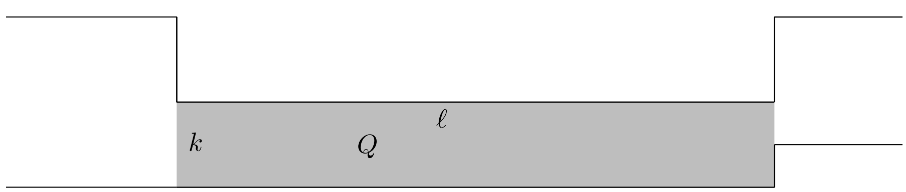


Figure 5: A pipe of width k and length ℓ.

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

Next we present the steps of the algorithm. Figures 6–8 demonstrates the steps on a concrete polyomino P.

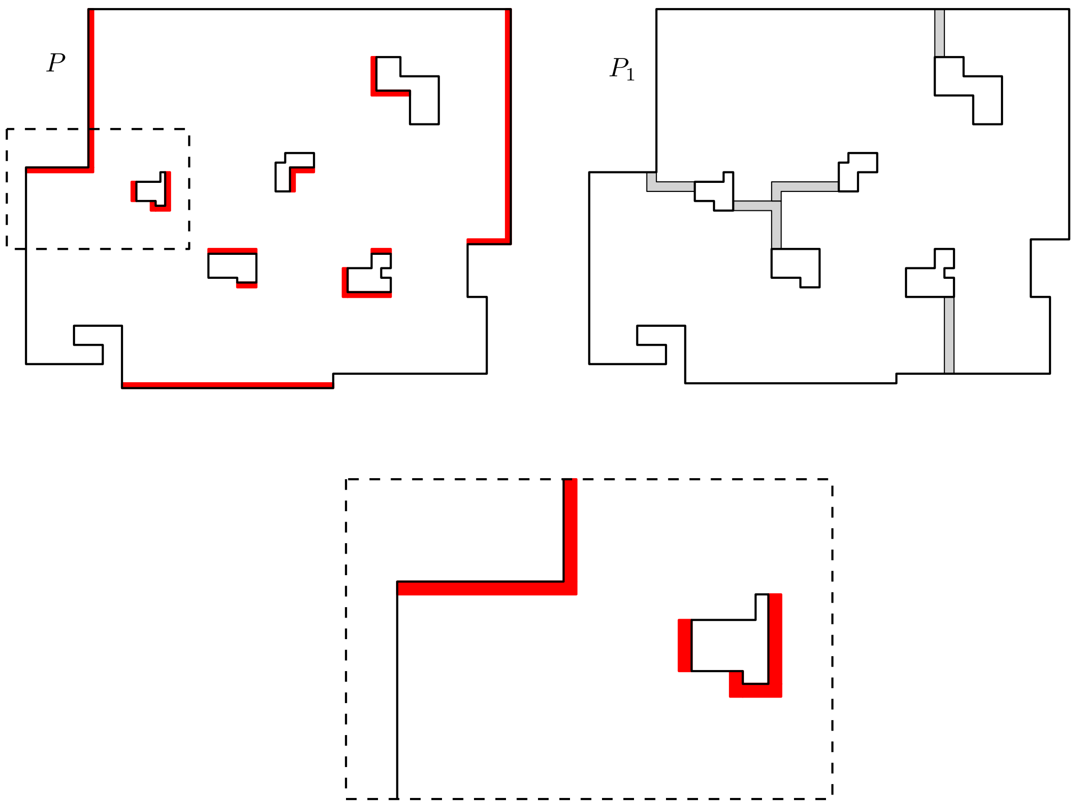


Figure 6: Steps 1 and 2 of the algorithm. Top left: Step 1, where the part $P \setminus P_1$, that is excluded from P_1 in order to make all coordinates even, is shown in red. Top right: Step 2, where the holes of a polyomino P_1 are connected to the outer boundary by the grey channels. Bottom: Closeup of the region in the dashed rectangle.

Step 1: Compute the unique maximal polyomino $P_1 \subset P$ with all coordinates even. We define P_1 to be the union of all 2×2 squares S of the form $S = [2i, 2i+2] \times [2j, 2j+2]$ with $i, j \in \mathbb{Z}$ and $S \subseteq P$. See the upper left and bottom part of Figure 6. It is readily checked that P_1 has at most n corners. As we will see, P_1 can be computed in time $O(n \log n)$.

Step 2: Compute a polyomino $P_2 \subset P_1$ with no holes and consistent parity by carving channels in P_1. Define $P'_0 := P_1$. For $i = 0, 1, \ldots$, we do the following. If there are holes in P'_i, we find a set of minimum size of 2×2 squares $S_1, \ldots, S_k$ contained in P'_i and with even corner coordinates that connects an edge of a hole to an edge of the outer boundary of P'_i. To be precise, an edge of S_1 should be contained in the boundary of a hole of P'_i, an edge of S_k should be contained in the outer boundary of P'_i, and for each $j \in \{1, \ldots, k-1\}$, S_j and S_{j+1} should share an edge.

We choose these squares such that they together form a $2 \times 2k$ or $2k \times 2$ rectangle or an L-shape, which is clearly always possible. We then define the polyomino $P'_{i+1} := P'_i \setminus \bigcup_{j=1}^{k} S_j$, which has less holes than P'_i. We stop when there are no more holes and define $P_2 := P'_i$ to be the resulting hole-free polyomino. Note that in iterations $i \geq 1$, the holes may get connected to holes that were eliminated in earlier iterations or to channels carved in earlier iterations. See the upper right part of Figure 6. We will later see that P_2 has strictly less than $3n$ corners and that it can be computed in time $O(n^3)$.

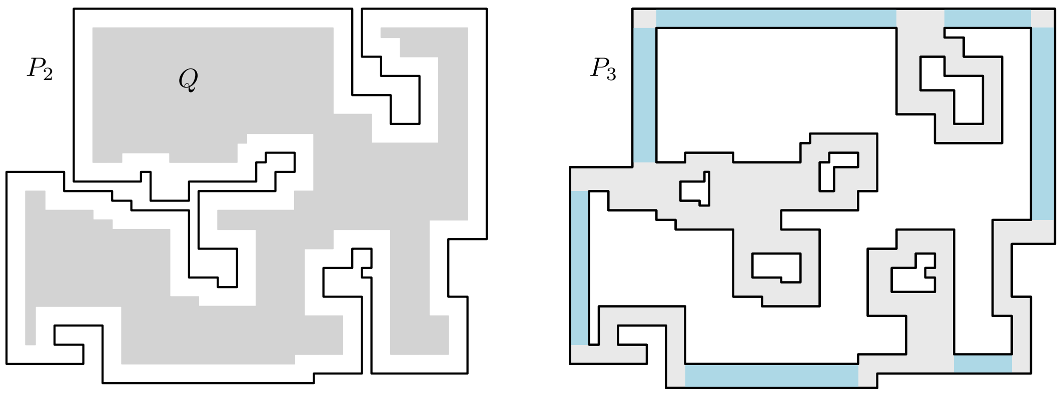


Figure 7: Steps 3 and 4 of the algorithm, performed on the instance from Figure 6. Left: Step 3, where the grey region Q is an offset of the hole-free polyomino P_2. In this example, Q is connected, but that is in general not the case. For pedagogical reasons, we offset by a smaller value than the algorithm would actually use. Right: Step 4, where the grey and blue areas are $P_3 := P \setminus Q$. The blue rectangles show the seven long pipes.

Step 3: Compute the offset $Q := B(P_2, -\lfloor 3n/2 \rfloor)$ and then $P_3 := P \setminus Q$. See the left part of Figure 7. Note that we remove Q from the original polyomino P in order to get P_3, and not from P_2. It is easy to check that Q has at most $3n$ corners and consistent parity. Hence $P_3 := P \setminus Q$ has at most $4n$ corners and, as we will see, P_3 has the property that for any $x \in P_3$, we have $\text{dist}(x, \partial P_3) = O(n)$. We will show how this step can be carried out in time $O(n \log n)$.

Step 4: Find the long pipes of P_3. Find all maximal long pipes $T_1, \ldots, T_r$ in P_3 (recall that a pipe is long if its length is at least 3 times its width). See the right part of Figure 7. As we will see, there are at most $O(n)$ such pipes, they are disjoint, and they each have width $O(n)$. Later we will show how the pipes can be found in time $O(n \log n)$.

Step 5: Shorten the pipes and compute the associated graph G^*. Define $G_3 := G(P_3)$. We modify G_3 by performing the following shortening step for each $1 \leq i \leq r$; see Figure 8. Assume with no loss of generality that the pipe T_i is of the form $T_i = [0, \ell] \times [0, k]$ where ℓ is the length and $k \leq \ell/3$ is the width. If $\ell \leq 6$, we do nothing. Otherwise, for each $j \in \{0, \ldots, k-1\}$, we let $S_j = [k+2, r] \times [j, j+1]$, where $r := 2\lceil \ell/2 \rceil - k - 2$, so that $G(S_j)$ is a horizontal path in G_3 consisting of an even number of vertices. For each $j \in \{1, \ldots, k-1\}$, we proceed by deleting the vertices of S_j and their incident edges from G_3, and instead, we add an edge from the cell

$[k+1, k+2] \times [j, j+1]$ to the cell $[r, r+1] \times [j, j+1]$ (i.e., we connect the cells to the left and right of S_j with each other).

We denote the graph obtained after iterating over all i by G^*. Note that in G^*, there are only $O(k^2) = O(n^2)$ vertices corresponding to cells in each pipe T_i, since each pipe has width $k = O(n)$. We show below that G^* has $O(n^3)$ vertices and can be computed in time $O(n^3)$.

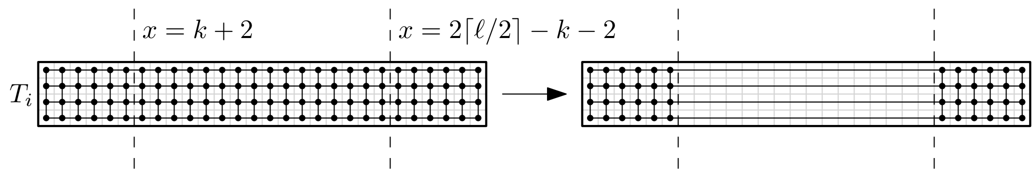


Figure 8: Step 5 of the algorithm. The part of the graph $G(T_i)$ in between the dashed vertical lines is substituted for long horizontal edges.

Step 6: Find the size of a maximum domino packing of P. We finally run a maximum matching algorithm with G^* as input and let the size of the resulting matching be α. Let N_0 be the area of P and N_2 be the number of vertices of G^*. The algorithm outputs $\alpha + (N_0 - N_2)/2$ as the value of a maximum domino packing of P. We show below that this step can be performed in time $O(n^4)$.

This completes the description of the algorithm. In Section 4.2, we will provide some structural results on domino packings and polyominos. In Section 4.3, we will use these results to argue that the algorithm works correctly. In Section 4.4, we will show that the reduced graph G^* has $O(n^3)$ vertices and edges. Finally, in Section 4.5, we will use this to argue how the steps of the algorithm can be implemented with the claimed running times.

4.2 Structural results on polyominos and domino packings

Building up to our structural results on domino packings, we require a definition and a few simple lemmas. Variations of the following lemma is well-known. We present a proof for completeness.

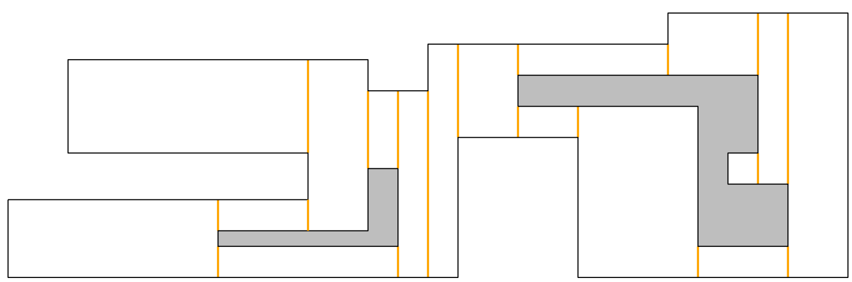


Figure 9: A partition of a polyomino (with two holes marked in grey) into rectangles using vertical line segments (orange).

Lemma 2. *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 polyominos too.*

Proof. For each concave corner of the polygon we add a vertical line segment starting at that corner and going upwards or downwards depending on which of the four possible rotations the corner has. 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 3. *The number of rectangular pieces in Lemma 2 is at most* $\frac{3}{4}n - 2$.

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

Lemma 4. *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 2×2 square as in Definition 2. 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 5. *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 6 and Lemma 7.

Lemma 6. *Let P and P_0 be polyominos such that $P_0 \subseteq P$, P_0 has no wholes, 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 1×2 dominos which restricts to a tiling of Q.*

Remark. Let us briefly pause to explain the importance of Lemma 6. Suppose that P contains a region Q as described. Then Lemma 6 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. Another tool, namely to contract long pipes, will be described in Lemma 7 below, and in Section 4.4, we will conclude that these two tools used carefully together reduce the packing problem to that of finding a maximum matching in a graph G^* of size $O(n^3)$.

Proof. It follows from Lemma 4 that Q has consistent parity, and it can thus be tiled with 2×2 squares and hence with dominos. Let $\mathcal{Q}$ be a tiling of Q.

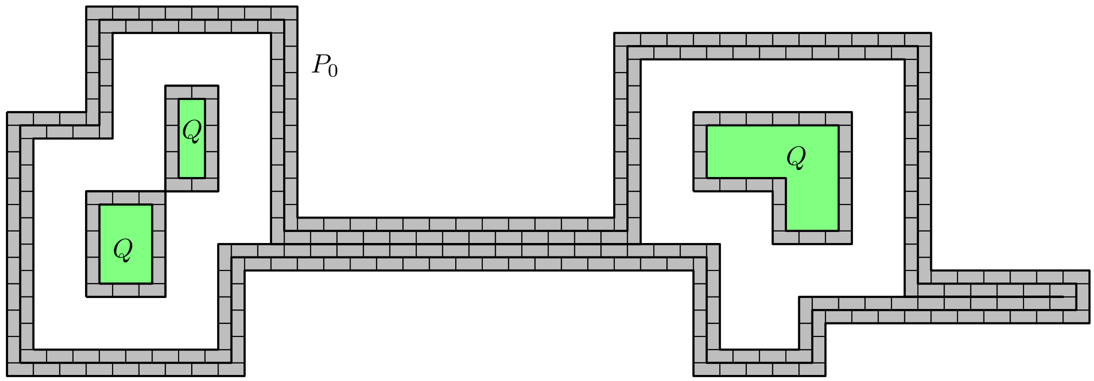


Figure 10: The polyomino P_0 and the offset Q (shown in green). The figure also illustrates the 'layers' A_i and their domino tilings, $\mathcal{A}_i$.

Define $R = P \setminus P_0$ and note that R has at most n corners. It follows from Corollary 3 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 $i \in \{1, \ldots, r\}$ be fixed. As P_0 has consistent parity, it follows from Lemma 4 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 5 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 $(v_1, \ldots, v_{2k})$ is the sequence of cells corresponding to such a cycle, $\{v_1 \cup v_2, v_3 \cup v_4, \ldots, v_{2k-1} \cup v_{2k}\}$ 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 $\mathcal{A}_1, \ldots, \mathcal{A}_r$ 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. Suppose that the matching M can be extended to a matching of size $|M|+k$ using a sequence $C_1, \ldots, C_k$ of $k \leq r$ augmenting paths one after the other. These paths can be chosen such that for each $i \in \{1, \ldots, k\}$, we have $C_i \subset R \cup \bigcup_{j=1}^{i} A_j$.

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 by at most $r = \lfloor \frac{3}{8}n \rfloor$ augmenting paths. It then follows immediately from the claim that these 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 $G(Q)$ corresponds to the tiling $\mathcal{Q}$.

We proceed to prove the claim by induction on k. The statement is trivial for $k = 0$, so consider a sequence $C_1, \ldots, C_k$ of augmenting paths and suppose inductively that $C_1, \ldots, C_{k-1}$ can be chosen such that for each $i \in \{1, \ldots, k-1\}$, we have $C_i \subset R \cup \bigcup_{j=1}^{i} A_j$. We need to prove that we can then choose C_k such that $C_k \subset R \cup \bigcup_{j=1}^{k} A_j$. Suppose that C_k is an augmenting path from u to v such that $C_k : u = u_1, u_2, \ldots, u_{2\ell} = v$. Let D be a Hamiltonian cycle of one of the connected

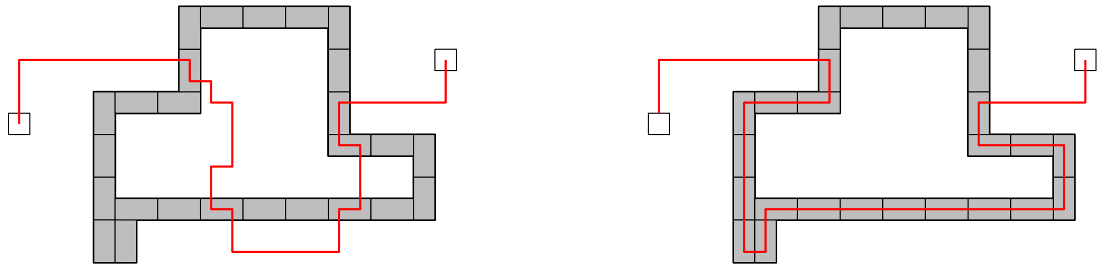


Figure 11: Left: An alternating path between two unmatched vertices which enters a connected component of $G(A_k)$. Right: Modifying the alternating path using the the Hamiltonian cycle of the connected component.

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, resulting in a path C'_k. 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 $G(A_k)$. 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 $R \cup \bigcup_{j=1}^{k} A_j$.

□

As it turns out, Lemma 6 is not in itself sufficient to yield a polyomino with area $n^{O(1)}$. For example, P may contain exponentially long and narrow ‘pipes’, say of width $n/10$, which will remain when Q is removed, as Q is obtained by offsetting $P_0 \subset P$. Surprisingly, it turns out that such narrow ’pipes’ are the only obstacles that prevent us from reducing to an instance of size polynomial in n. This is what motivates the following lemma which intuitively yields a reduction for shortening long narrow pipes.

Lemma 7. *Let $k, \ell \in \mathbb{N}$ with ℓ even. Let $L \subseteq [-1, 0] \times [0, k]$, $R \subseteq [\ell, \ell+1] \times [0, k]$ be polyominos 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 $b - w$.*

Proof. As each domino covers one black and one white cell, any packing will leave at least $b - w$ cells uncovered. We thus need to demonstrate the existence of a packing with exactly $b - w$ uncovered cells. To see that such a packing exists, it is very illustrative to consider Fig. 12. An example of a polyomino, P, is illustrated in Fig. 12(a). We first tile as many cells of L and R as possible, such that no two uncovered cells of L and R share an edge. We will call these uncovered cells *notches*. We next show how we can alter the configuration of notches by only adding a layer of width 2. First, we note that a notch can be shifted an even number of cells downwards or upwards using the construction in Fig. 12 (b). In case we have two notches of different colours in the chessboard coloring and with no other notches between them, we can use the construction in Fig. 12 (c) to

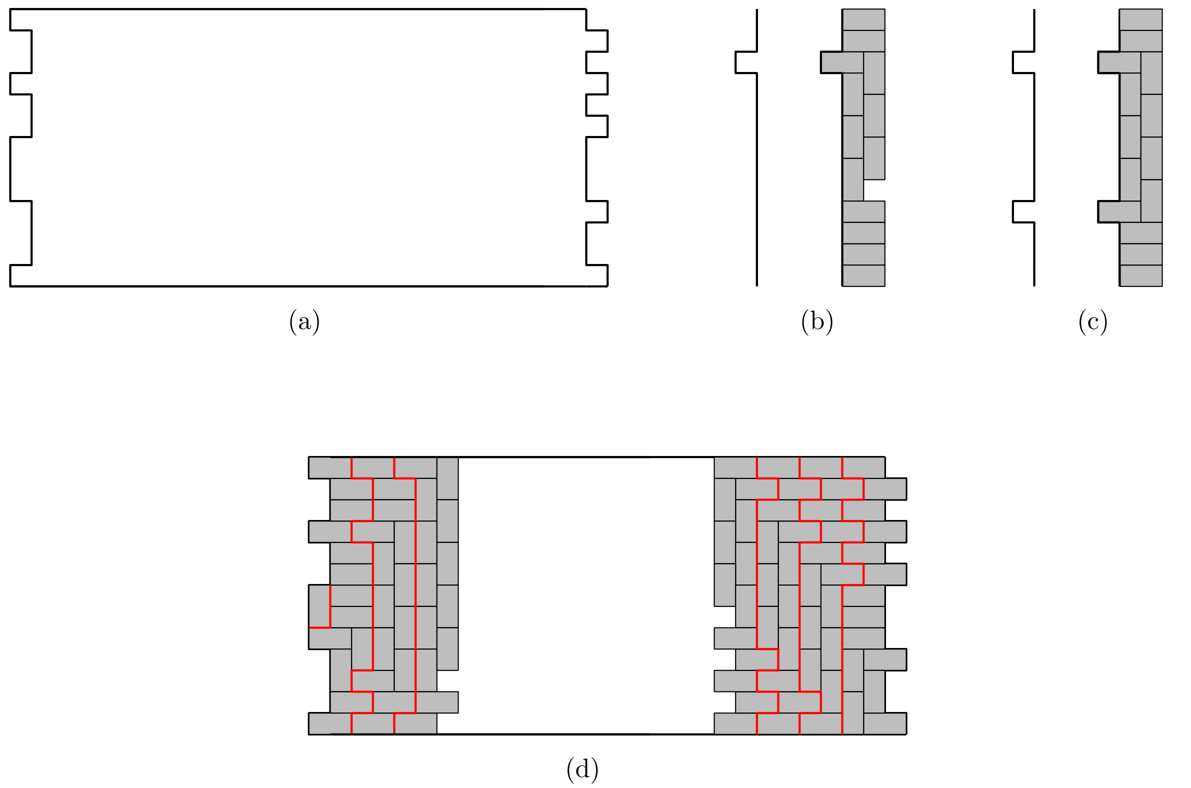


Figure 12: (a) The polyomino P. (b) Shifting a notch. (c) Cancelling two notches. (d) The partial packing obtsained after shifting notches downwards and cancelling notches when possible.

cancel these two notches from the configuration of notches. Our goal is to use the constructions of (b) and (c) to shift the notches of L and R downwards, cancelling notches if possible. Going through the notches of L from bottom to top, we shift them down as far as possible using the construction in (b). In case a notch has a different color than the nearest notch below it, we use construction (c) to cancel them. We further add horizontal dominos such that the configuration of notches is preserved at all other positions than where the shifting or cancelling occurs. We do a similar thing for R. The process from start to end is illustrated in Fig. 12(d) which also shows the resulting partial tiling. The red lines in the figure separates the steps of the process.

Note that each added layer of the process has thickness 2. Initially, there are at most k notches in L and R (after the first step which isolates the notches). The assumption that $l \geq 2k$ thus guarantees that we are able to finish this partial packing. Let b' and w' be the number of black and white cells uncovered by this partial packing. Then $b' - w' = b - w$. It is also easy to check that we can complete the packing of P using only horizontal dominos and leaving exactly $b' - w' = b - w$ cells uncovered. This completes the proof. □

Lemma 7 allows us to 'shorten' long and narrow pipes when searching for the maximum domino packing. It follows from the Lemma that going from G_3 to G^* in the shortening step 5 of our algorithm does not alter the number of unmatched vertices in a maximum matching. We will return to this in Section 4.3. Note that, unlike in Lemma 6, we cannot simply remove (part of) the pipe from the polyomino. The following lemma allows us to upper bound the size of a set of overlapping pipes no two of which are contained in the same larger pipe.

Lemma 8. *Let $G = (V, E)$ be a graph of order $n \geq 2$ with no self-loops but potential multiple edges. Suppose that G has a planar embedding such that for any pair of multiple edges (e_1, e_2), the Jordan curve formed by e_1 and e_2 in the planar embedding of G contains a vertex of G in its interior. Then the number of edges of G is upper bounded by $3n - 5$.*

Proof. In what follows, we will use the classic result that the number of edges of a simple planar graph of order n is upper bounded by $3n - 6$. We prove the result by strong induction on n. For $n = 2$ the result is trivial so let $n > 2$ be given and suppose the bound holds for smaller values of n. Let $\mathcal{E}$ be a planar embedding of G. Let (e_1, e_2) be a pair of distinct multiple edges that is minimal in the sense that no other such pair (e_1', e_2') exists with the following property: If γ and γ' are the Jordan curves formed by (e_1, e_2) and (e_1', e_2') in $\mathcal{E}$, then $\operatorname{Int}\gamma' \subset \operatorname{Int}\gamma$. Assume that e_1 and e_2 connect vertices u and v. Let V' be the set of vertices of G that are contained in $\operatorname{Int}\gamma$ under $\mathcal{E}$ and let $k = |V'|$. Then, $1 \leq k \leq n - 2$. Let $G_1 = (V_1, E_1)$ where $V_1 = V' \cup \{u, v\}$ and E_1 is formed by e_1 together with all edges of G that are incident to a vertex in V'. Let $G_2 = (V_2, E_2)$ where $V_2 = V \setminus V'$ and $E_2 = E \setminus E_1$. Clearly G_1 is a simple planar graph on $k + 2$ vertices. Moreover, it is readily checked that G_2 is a planar graph on $n - k$ vertices which satisfies the assumptions of the lemma. Note that $2 \leq n - k < n$. It thus follows from the inductive hypothesis that the number of edges of G is upper bounded by

$$3(n - k) - 5 + 3(k + 2) - 6 = 3n - 5.$$

This completes the proof. □

Lemma 9. *Let P be a polyomino with n corners. Let $Q_1, \ldots, Q_r$ be pairwise disjoint pipes of P, no two of which are contained in a larger pipe of P, i.e., for no two distinct i, j does there exist a pipe Q with $Q_i \cup Q_j \subseteq Q$. Then $r \leq 3n - 5$.*

Proof. We construct a graph $G = (V, E)$ as follows. V is the set of (geometric) edges of P. For each $i \in \{1, \ldots, r\}$ we let $u_i, v_i \in V$ be the two parallel edges of P which contain two opposite sides of Q_i and we add the edge (u_i, v_i) to E. G is thus a graph of order n with exactly r edges. We note that G may have multiple edges but it has a natural planar embedding, $\mathcal{E}$, such that for each pair of multiple edges (e_1, e_2) the Jordan curve formed by e_1 and e_2 under $\mathcal{E}$ contains a vertex of G in its interior (see Figure 13). Here we used that for any two i, j with $1 \leq i < j \leq r$, the two pipes Q_i and Q_j are not contained in a larger pipe of P. It thus follows from Lemma 8 that $r \leq 3n - 5$. □

We need to argue that the algorithm outputs the correct value and that the different steps can be implemented to obtain the stated running times.

4.3 Correctness of the domino packing algorithm

We now refer the reader back to the description of our domino packing algorithm from Section 4.1 and show that it correctly finds the size of a maximum domino packing. To show this, it suffices to show that maximum matchings of G_0 and G^*, leave the same number of unmatched vertices. First note that P_1 has at most n corners. Further, a polyomino with n corners can have at most

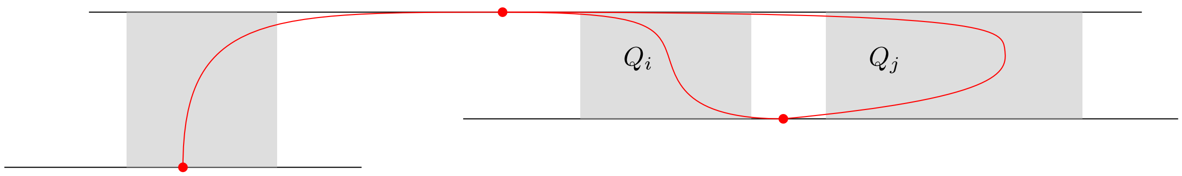


Figure 13: The construction of the planar graph G. Pipes are shown in grey. The white rectangle between, Q_i and Q_j must contain some nonempty subset of ∂P in its interior — otherwise this rectangle could joined with Q_i and Q_j forming a larger pipe of P containing both Q_i and Q_j.

$(n-4)/4$ holes and since we remove a hole and add at most 6 new corners in going from P_i' to P_{i+1}', P_2 has at most $5n/2 < 3n$ corners. It follows that also $Q = B(P_2, \lfloor 3n/2 \rfloor)$ has at most $3n$ corners. Letting n_1 denote the number of corners of $P_3 = P \setminus Q$ it finally follows that $n_1 \leq 4n$. Now $\text{dist}(\partial P, Q) \geq \lfloor 3n/2 \rfloor \geq \lfloor \frac{3}{8} n_1 \rfloor$ and moreover, Q has consistent parity and no holes, so Lemma 6 applies, giving that G_0 has a maximum matching, which restricts to a perfect matching of $G(Q)$ and to a maximum matching of G_3. In particular, maximum matchings of G_0 and G_1 leave the same number of vertices unmatched.

Next, we argue that maximum matchings of G_3 and G^* again leave the same number of vertices unmatched. It is easy to see that a maximum matching of G^* can be extended to a matching of G_3 with the same number of unmatched vertices by simply inserting more horizontal dominos in the horizontal pipes and vertical dominos in the vertical pipes (here we use that the S_j's as defined in Step 3, each consists of an even number of cells). Conversely, let M_1 be a maximum matching of G_3. We show that G^* has a matching, M_2, with the same number of uncovered cells. For this we consider the pipes $(T_i)_{i=1}^r$ found in step 4 of the algorithm. For each $1 \leq i \leq r$, we let T_i' be the pipe obtained from T_i by shortening T_i by one layer of cells in each end. The length of T_i' is thus two shorter than that of T_i. Let further $L_i \supseteq T_i'$ consist of all cells of P which are covered by a domino which cover at least one cell of T_i'. The sets $(L_i)_{i=1}^r$ are pairwise disjoint and they are each of the form of the set L in Lemma 7 (up to a 90 degree rotation). Moreover, the maximum matching M_1 restricts to a maximum matchings of M_1' of $G(P_3 \setminus \bigcup_{i=1}^r L_i)$ and a maximum matching $M_1^{(i)}$ of $G(L_i)$ for $1 \leq i \leq r$. For $1 \leq i \leq r$, we let $G_3^{(i)} = G(L_i)$ and $G_2^{(i)}$ be the corresponding subgraph of G_2. We define M_2 to be M_1' combined with any maximum matchings of the $G_2^{(i)}$, $1 \leq i \leq r$. By applying Lemma 7 to each L_i, it follows that the maximum matchings of $G_3^{(i)}$ and $G_2^{(i)}$ leave the same number of unmatched vertices. It thus follows that M_2 and M_1 leave the same number of unmatched vertices. This finishes the argument that the algorithm works corrrectly.

4.4 Bounding the size of the reduced instance

In determining the running time of our algorithm, it is crucial to bound the size of the reduced instance G^*. In this section we show that G^* has $O(n^3)$ vertices. In the next section, we will see how this leads to an $O(n^4)$ time algorithm for finding a maximum matching in G^*.

We start out by proving the following lemma.

Lemma 10. *The polyomino P_3 contains no $63n \times 63n$ square subpolyomino.*

Proof. Let $n' = \lfloor 3n/2 \rfloor$. We show that P_3 contains no $41n' \times 41n'$ square as a subpolyomino and the desired result will follow. Suppose for contradiction that $S \subseteq P_3$ is such a subpolyomino. Note that

Q consists of exactly those points of P_1 of distance at least n' to all the channels of $C := P_1 \setminus P_2$ and to ∂P_1. Thus any point $x \in P_3$ has distance at most n' to C or to ∂P_1. In particular S contains a $39n' \times 39n'$ square subpolyomino, $S_1 \subseteq P_1$, all points of which are of distance at least n' to ∂P_1 and thus, of distance at most n' to C.

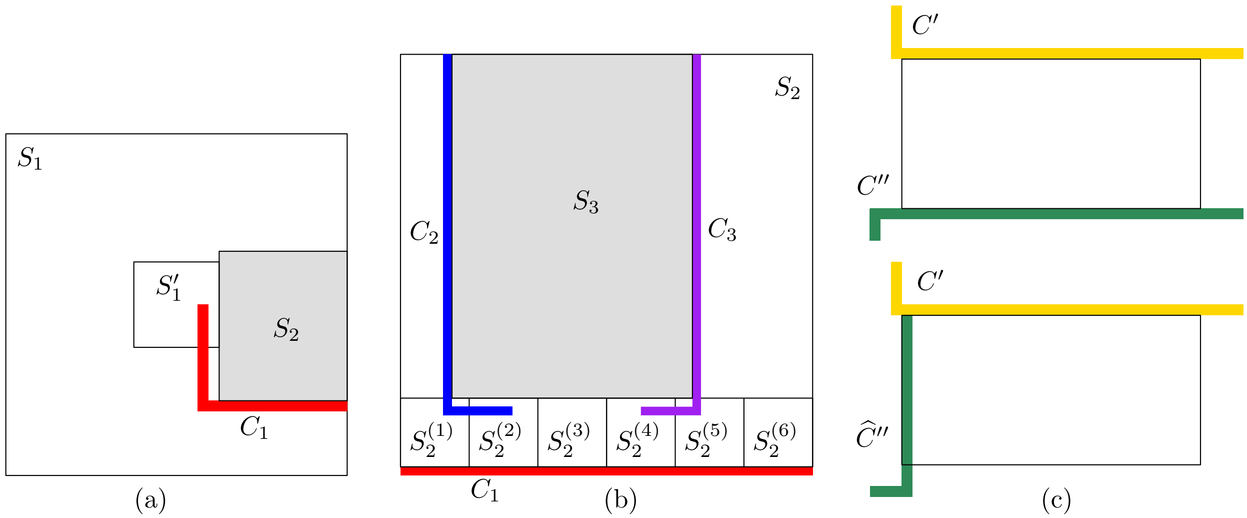


Figure 14: Situations in the proof of Lemma 10.

By the way we chose the channels, each channel connects a hole of P_1 with either the boundary of P_1 or with a channel already carved in an earlier iteration. Since $\partial P_1 \cap S_1 = \emptyset$, it follows that any channel intersecting S_1 has an end outside S_1 and thus leaves S_1 through an edge of S_1.

Write S_1' for the central $3n' \times 3n'$ square polyomino of S_1; see Figure 14) (a). Since any point of S_1 is of distance at most n' to C, S_1' must intersect a channel $C_1 \subset C$ (depicted in red in the figure). We know that C_1 leaves S_1. It is simple to check that this leads to the existence of an $18n' \times 19n'$ rectangular polyomino $S_2 \subseteq S_1$ having along one of its sides a straight part of the channel C_1 of length $18n'$; see Figure 14 (b). Assume with no loss of generality that $S_2 = [0, 18n'] \times [0, 19n']$ and that the channel C_1 runs along the base of the rectangle S_2 as in the figure. For $1 \leq i \leq 6$ we define $S_2^{(i)}$ to be the square polyomino $[3(i-1)n', 3in'] \times [0, 3n']$. By the same reasoning as above, each of these squares must intersect the set of channels C non-trivially. As each channel turns at most once by construction, the squares $S_2^{(i)}$ are disjoint from C_1. To finish the proof, we require the following claim.

Claim. Let $B = [0, k] \times [0, \ell]$, $k, \ell \in \mathbb{N}$ be a $k \times \ell$ square polyomino. Suppose that $[0, k] \times [-1, 0]$ is contained in some channel, C', and that $[0, k] \times [\ell, \ell + 1]$ is contained in some other channel, C''. Then $k \leq \ell + 2$.

Proof of Claim. See Figure 14 (c). Suppose without loss of generality that C' was carved in iteration i and C'' was carved in iteration j in the process of generating P_2 in step 2 of the algorithm, and that $i < j$. The channel C'' was chosen to connect a yet unconnected hole H of P'_{j-1} with the outer boundary of P'_{j-1} along a shortest path in the L_∞-norm. At the time C'' was carved, C' had already been carved and thus the edges of C' (except the two "ends" of length 2) is part of the outer face of P'_{j-1}. Under the assumption $k > \ell + 2$, we can find a shorter path connecting H to $\partial P'_{j-1}$;

see the bottom part of Figure 14 (c). This shorter path shows that we could have picked a shorter channel $\widehat{C}''$ in place of C''. This is a contradiction, so we conclude that $k \le \ell + 2$. □

Let us now finish the proof of the lemma. We know that the two squares $S_2^{(2)}$ and $S_2^{(5)}$ each intersect channels of C. Let us denote these not necessarily distinct channels respectively C_2 and C_3. If these channels are the same, the channel C_2 passes straight through $S_2^{(3)}$. But then the two channels C_1 and C_2 run in parallel for a length of at least $3n' + 4$ and they have distance at most $3n'$ which gives a contradiction with the claim. Thus C_2 and C_3 are different channels. By the same reasoning as for C_1, the channel C_2 must leave S_2. If it does so in a direction parallel to C_1, we similarly obtain a contradiction with the claim. Thus, it most leave S_2 in a direction perpendicular to C_1. The same logic applies to C_3; see Figure 14 (b). Now the two channels C_2 and C_3 provide a contradiction to the claim. Indeed, their straight segments span a box, S_3, of dimensions $\ell \times 18n'$ where $\ell \le 18n' - 4$. With this contradiction, we conclude that P_3 contains no $41n' \times 41n'$ square as a subpolyomino and the proof is complete. □

Remark. No serious effort has been made to optimize the constants in Lemma 10.

Corollary 11. *For any $x \in P_3$, we have dist$(x, \partial P_3) \le 32n$.*

Proof. If not, P_3 contains a $63n \times 63n$ square, a contradiction. □

Corollary 11 shows that each point of P_3 is of distance $O(n)$ to the boundary of P_3. In particular, this shows that the long pipes, $T_1, \dots, T_r$, found in step 4 of the algorithm all have width at most $O(n)$. By Lemma 9, $r = O(n)$, so when performing the shortening reduction in step 5 of our algorithm, the part of G^* contained in contracted pipes gets size $O(n^3)$. We finish this section by showing that $P_3 \setminus \bigcup_{i=1}^r T_i$ also consists of $O(n^3)$ cells. From this it will follow that G^* is of order $O(n^3)$ which is what we require. We state the result as a lemma.

Lemma 12. *The reduced instance G^* found by our algorithm has $O(n^3)$ vertices and edges.*

Proof. For technical reasons to be made clear shortly we define T_i' to be the pipe obtained from T_i by shortening T_i by a layer of cells in each end. The length of T_i' is thus exactly the length of T_i minus 2. Let R be the polyomino $P_3 \setminus \bigcup_{i=1}^r T_i'$. Consider an edge, e, in the set $\partial R \setminus \partial P_3$ which by definition is the edge forming an end of a shortened pipe T_i'. It then follows that the endpoints of e are corners of R and that e is not contained in a longer edge of R (this would not necessarily be the case if we had not shortened the pipes when defining R).

As discussed, it suffices to show that R has $O(n^3)$ cells. Note that R has $O(n)$ corners: Indeed, R is obtained from P_3, which has $O(n)$ corners, by removing $O(n)$ channels, each of which adds only 4 corners. We show that any point $x \in R$ is of distance $O(n)$ from a corner. Since each corner can have at most $O(n^2)$ cells within distance $O(n)$, this will show that R has $O(n^3)$ cells.

So let $x \in R$ be arbitrary. Also let $c := 32$. By Corollary 11, any point of P_3 is of distance at most cn to ∂P_3. It follows that, similarly, any point of R is of distance at most cn to ∂R. Let S be the $6cn \times 6cn$ square centered at x and suppose that S contains no corner of R. Then $\partial R \cap S$ is a collection of horizontal and vertical straight line segments. Moreover, they are either all horizontal or all vertical as otherwise, they would intersect in a corner of R inside S. Assume without loss of generality that they are all horizontal. Using that any point of R is of distance at most cn to ∂R, it follows that there exists two such parallel segments of distance at most $2cn$, one being above x and one being below. Take a closest pair of such segments. Together they form a pipe, T, of R

of length $6cn$ and width at most $2cn$, i.e., a pipe of a length at least three times its width. Now T is disjoint from the pipes $T_1, \dots, T_r$ (since $T \subset R$ and each T'_i is disjoint from R) and we claim that T is in fact also a pipe of P_3. To see this, we simply observe that by Corollary 11, the pipes found in step 4 of our algorithm have width at most $2cn$. In particular, the edges of $\partial R \setminus \partial P$ have length at most $2cn$. However, the pipe T has length $6cn$. The long sides of T are contained in edges ∂R. However, no edge of $\partial R \setminus \partial P_3$ are longer than $2cn$ and we saw that no edge of $\partial R \setminus \partial P_3$ are contained in longer edges of ∂P_3. It follows that the long edges of T are in fact edges of ∂P_3, so T is a pipe of P_3. This contradicts the maximality of the set of pipes $T_1, \dots, T_r$. We thus conclude that S does contain a corner of ∂R. Since $x \in R$ was arbitrary, this shows that any point in R is of distance at most $3cn = O(n)$ to a corner of R and the proof is complete. □

4.5 Implementation of the individual steps

We next describe how the different step of our domino tiling algorithm can be implemented.

Step 1: Compute the unique maximal polyomino $P_1 \subset P$ with all coordinates even. We first compute the set $P_1 \subset P$ with consistent parity. To obtain P_1, we move all corners of P to the interior of P to the closest points with even coordinates as shown in Figure 6.

Moving the corners may cause some corridors of P to collapse (namely the corridors of P of thickness 1), so that P_1 has overlapping edges corresponding to degenerate corridors. The degenerate vertical corridors can be filtered out in $O(n \log n)$ time as follows (and the degenerate horizontal ones are handled analogously). We sort the vertical edges after their x-coordinates and thus partition the edges into groups with identical x-coordinates. For each group of vertical edges with the same x-coordinate, we sort them according to the y-coordinates of their lower endpoints. Let $e_1, \dots, e_k$ be one such sorted group. We run through the edges $e_1, \dots, e_k$ in this order. For each edge e_i, we run through the succeeding edges until we get to an edge e_j which is completely above e_i. For each of the overlapping edges e_k, $k \in \{i+1, \dots, j-1\}$, we remove the corresponding degenerate corridor of P_1 created by e_i and e_k. Since no triple of edges can be pairwise overlapping, there are $O(n)$ overlapping pairs in total, so this process is dominated by sorting, which takes $O(n \log n)$ time.

Step 2: Compute a polyomino $P_2 \subset P_1$ with no holes and consistent parity by carving channels in P_1. Remember that we need to find a set of minimum size of 2×2 squares $S_1, \dots, S_k$ contained in P'_i and with even coordinates that connects an edge of a hole to an edge of the outer boundary of P'_i. For each pair of an edge of a hole of P'_i and an edge of the outer boundary of P'_i, we can compute the size of the smallest set of squares connecting those two edges in $O(1)$ time, so by checking all pairs, we find the overall smallest set in $O(n^2)$ time. Note that no edge of a middle square S_j, $2 \le j \le k-1$, is contained in the boundary $\partial P'_i$, since otherwise, there would be a smaller set of squares with the desired properties. Therefore, constructing $P'_{i+1} := P'_i \setminus \bigcup_{j=1}^{k} S_j$ can then be done in $O(1)$ time once the squares S_j have been found. Since there are initially $O(n)$ holes to eliminate, the process takes $O(n^3)$ time in total.

Step 3: Compute the offset $Q := B(P_2, -\lfloor 3n/2 \rfloor)$ and then $P_3 := P \setminus Q$. We now explain how to compute the set $Q \subset P_2$ defined by $Q := B(P_2, -\lceil 3n/4 \rceil)$. The boundary of Q can be computed from the L_∞ Voronoi diagram $VD := VD(P_2)$ of the edges of P_2 by a well-known technique described by Held, Lukács, & Andor [12], as follows. The Voronoi diagram VD is a plane

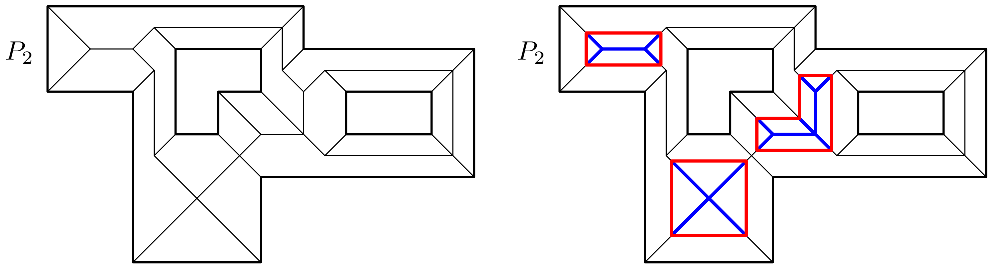


Figure 15: Left: The L_∞ Voronoi diagram $VD = VD(P_2)$ of the edges of a polyomino P_2. Right: The blue parts of VD are the subgraphs that have some sufficient distance d to the boundary ∂P_2. The red cycles enclose the regions of P_2 with at least distance d to ∂P_2.

graph contained in P_2 that partitions P_2 into one region $R(e_i)$ for each edge e_i of P_2, such that if $x \in R(e_i)$ then $\text{dist}(x, e_i) = \text{dist}(x, \partial P_2)$, and VD consists of horizontal and vertical line segments and line segments that make 45° angles with the x-axis; see Figure 8 (left).

We find all maximal subgraphs of VD consisting of points with distance at least $\lceil 3n/4 \rceil$ to ∂P_2. The leafs of each subgraph G lie on a cycle in P_2 where the distance to ∂P_2 is constantly $\lceil 3n/4 \rceil$, and the cycles can be found by traversing the leafs of G in clockwise order; see [12] for the details and Figure 8 (right) for a demonstration.

We can compute VD in $O(n \log n)$ time using the sweep-line algorithm of Papadopoulou & Lee [19]. In our special case where all edges are horizontal or vertical, the algorithm becomes particularly simple as described by Martínez, Vigo, Pla-García, & Ayala [16]. Once we have VD, it takes $O(n)$ time to compute ∂Q.

One can avoid the computation of VD by offsetting the boundary into the interior by distance 1 repeatedly $\lceil 3n/4 \rceil$ times. After each offset, we remove collapsed corridors as described in step 1. This would take in total $O(n^2 \log n)$ time.

The representation of $P_3 := P \setminus Q$ is obtained by simply adding the cycles representing the boundary of Q to the representation of P.

Step 4: Find the long pipes of P_3. Recall that each long pipe $T_i \subset P_3$ is a maximal rectangle with a pair of edges contained in ∂P_3 which are at least 3 times as long as the other pair of edges.

To find the pipes, we compute the L_∞ Voronoi diagram $VD_1 := VD(P_3)$ of the edges of P_3 in $O(n \log n)$ time, as described in step 3. Consider a long pipe T_i. Assume without loss of generality that $T_i = [0, \ell] \times [0, k]$ where ℓ is the length and $k \leq \ell/3$ is the width. We now observe that the segment $e := [k/2, \ell - k/2] \times k/2$ in the horizontal symmetry axis of T_i is contained in an edge of VD_1, since for any point p in e, the edges of P_3 closest to p are the horizontal edges of T_i.

Each horizontal or vertical edge e of VD_1 separates the regions of points that are closest to a pair s_1, s_2 of horizontal or vertical edges of P_3. It is easy to check whether s_1, s_2 define a long pipe containing (a part of) e. Hence, all pipes can be identified by traversing the edges of VD_1. As VD_1 has complexity $O(n)$, this step takes $O(n \log n)$ time in total.

Step 5: Shorten the pipes and compute the associated graph G^*. Recall that we define $G_3 := G(P_3)$ and obtain the final graph G^* by replacing long horizontal (resp. vertical) paths in pipes with long horizontal (resp. vertical) edges. Once P_3 and the long pipes have been computed, it is straightforward to construct G^* in $O(n^3)$ time, since the size of G^* is $O(n^3)$ by Lemma 2.

Step 6: Find the size of a maximum domino packing of P. Each vertex of G^* corresponds to a cell of P_3. Let $R \subset P_3$ be the polyomino formed by these cells (the polyomino R will in general be disconnected). Then R has $O(n)$ corners, since P_3 has $O(n)$ corners and we introduce at most 4 corners by removing the central cells of each long pipe. Hence, by Lemma 2, R can be partitioned into $O(n)$ rectangles $R_1, \ldots, R_k$. We now consider a maximum domino packing of each rectangle R_i, which will leave at most one cell uncovered in each R_i. This packing corresponds to a matching in G^* with at most $O(n)$ unmatched vertices. We then run $O(n)$ breadth-first searches in G^* to augment the matching as much as possible using augmenting paths. Since G^* has size $O(n^3)$ by Lemma 12, this takes in total $O(n^4)$ time.

Remark. We note that we can also find an (implicit) description of a maximum domino packing of P. We simply extend the matching of G^* by inserting horizontal dominos in the horizontal pipes and vertical dominos in the vertical pipes. We further decide to give Q any standard tiling, e.g., the one that uses only horizontal dominos. It follows from the correctness of the algorithm that this gives a maximum domino packing of P.

References

[1] Danièle Beauquier, Maurice Nivat, Eric Remila, and Mike Robson. Tiling figures of the plane with two bars. *Computational Geometry*, 5(1):1–25, 1995.

[2] Robert Berger. The undecidability of the domino problem. *Memoirs of the American Mathematical Society*, 1(66), 1966.

[3] Fran Berman, David Johnson, Tom Leighton, Peter W. Shor, and Larry Snyder. Generalized planar matching. *Journal of Algorithms*, 11(2):153–184, 1990.

[4] 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`.

[5] 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.

[6] 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.

[7] Dirk K. de Vries. Investigation of gross die per wafer formulas. *IEEE Transactions on Semiconductor Manufacturing*, 18(1):136–139, 2005.

[8] Dania El-Khechen, Muriel Dulieu, John Iacono, and Nikolaj Van Omme. Packing 2×2 unit squares into grid polygons is NP-complete. In *Proceedings of the 21st Canadian Conference on Computational Geometry (CCCG 2009)*, pages 33–36, 2009.

[9] 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.

[10] George Gamow and Marvin Stern. *Puzzle-math.* Macmillan, 1958.

[11] S. W. Golomb. Checker boards and polyominoes. *The American Mathematical Monthly*, 61(10):675–682, 1954.

[12] Martin Held, Gábor Lukács, and László Andor. Pocket machining based on contour-parallel tool paths generated by means of proximity maps. *Computer-Aided Design*, 26(3):189–203, 1994.

[13] Dorit S. Hochbaum and Wolfgang Maass. Approximation schemes for covering and packing problems in image processing and VLSI. *Journal of the ACM (JACM)*, 32(1):130–136, 1985.

[14] 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.

[15] 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.

[16] J. Martínez, M. Vigo, N. Pla-García, and D. Ayala. Skeleton computation of an image using a geometric approach. In *31st Eurographics (EG 2010)*, 2010.

[17] 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.

[18] Igor Pak and Jed Yang. Tiling simply connected regions with rectangles. *Journal of Combinatorial Theory, Series A*, 120(7):1804 – 1816, 2013.

[19] Evanthia Papadopoulou and D.T. Lee. The L_∞ Voronoi diagram of segments and VLSI applications. *International Journal of Computational Geometry & Applications*, 11(05):503–528, 2001.

[20] Eric Rémila. Tiling a polygon with two kinds of rectangles. *Discrete Comput. Geom.*, 34(2):313–330, 2005.

[21] William P. Thurston. Conway's tiling groups. *The American Mathematical Monthly*, 97(8):757–773, 1990.