

Optimal Polyomino Fences

Alexis Langlois-Rémillard, Mia N. Müßig, Helmut Podhaisky, Erika Roldan, Günter Rote
Maximilian Brömme*, Matthes Norden*, Maximilian Prietzel*, Daniel Yu*, Mykhailo Lyader†

October 24, 2026

Abstract

The n -omino fence problem asks how many cells can be surrounded by placing every n -omino exactly once in the grid. We present a new dynamic programming approach for this problem, allowing us to prove optimality for the pentomino and hexomino fence and to classify all possible pentomino orders by optimal area. For the first time we also provide strong heuristic solutions and upper bounds for all fences up to $n = 15$, after which we even approximate the 500-omino fence with computational experiments. Finally, we generalize the problem to 3D, where we prove optimality for pentacubes and derive heuristic solutions for hexacubes and heptacubes.

*Part of the student team at Freie Universität Berlin that worked on the 3D walls

†Worked with Helmut Podhaisky as part of a school project

A13
A14
A15
A16
A17
A18
A19
A20
A21
A22
A23
A24
A25
A26
A27
A28
A29
A30
A31
A32
A33
A34
A35
A36
A37
A38
A39
A40
A41
A42
A43
A44
A45
A46
A47
A48
A49
A50
A51
A52
A53
A54
A55

Contents	
1 Introduction	3
2 Precise definition of the problem	3
2.1 Fences and their area	3
2.2 The boundary around the largest region	5
3 Pentomino fences	7
3.1 Edge boundary	7
3.2 Dynamic programming with fixed order	9
3.2.1 Algorithm	9
3.2.2 Solving the min-max problem	12
3.3 Dynamic programming with unrestricted order	13
3.4 Identification of all optimal pentomino fences	15
4 Hexomino fences	17
4.1 Optimizations for dynamic programming	18
4.1.1 Reference vectors	18
4.1.2 Eight-way partitioning	20
4.1.3 Using a prescribed target area	22
4.2 Proof of optimality with DP after Branch-and-Bound (B&B) preprocessing	23
4.2.1 Step 0: Computing the reference vector repertoire of a hexomino	23
4.2.2 An upper bound via Minkowski averages	23
4.2.3 Step 1: Branch-and-Bound over vector sequences	25
4.2.4 Step 2: Taking into account the rotation and the parity	27
4.2.5 Step 3: Dynamic programming with fixed vectors	27
4.3 Step 4: Identification of all optimal solutions	27
4.4 Overlapping pieces and multiple shared edges	28
5 Heptomino and octomino fences	30
5.1 Lower bound using pruned dynamic programming	30
5.2 Upper bounds with Branch-and-Bound	31
6 Larger fences	31
6.1 $F_{\max} + D_{\max}$ upper bound	33
6.2 Heuristic greedy solver	33
6.3 The limit shape	34
7 Three-dimensional walls	37
7.1 Definitions	37
7.2 An isoperimetric upper bound	38
7.3 Pentacubes	40
7.4 Hexacubes and heptacubes	41
8 Independent confirmation of hexomino optimality with integer programming	42
9 Conclusion	43
– References	44
A Optimizing the total enclosed area	45
B Hexomino area distribution of random orders	50
C Expected length of reference vectors	50

1 Introduction

We consider the square grid $\mathbb{Z}^2$, whose cells are unit squares. Two cells are *4-adjacent* if they share an edge, and *8-adjacent* if they share an edge or a vertex. A set of cells is *4-connected* (resp. *8-connected*) if its cells form a connected graph under 4-adjacency (resp. 8-adjacency). The *free n -ominoes* are the 4-connected sets of n cells, where two sets are considered the same if one can be obtained from the other by translation, rotation, and reflection. There are 12, 35, 108, and 369 free n -ominoes for $n = 5, 6, 7, 8$, respectively.

The *n -omino fence problem* asks: what is the largest area (number of cells) that can be enclosed by placing every n -omino exactly once in the grid? The pieces must not overlap, and they must touch along full edges, not only at corners. We give a precise definition of the problem in Section 2.

Victor Feser first presented such fence-like puzzles with pentominoes in 1968 [5]. One of them, which asks for the largest area enclosed by a fence of the 12 pentominoes, became widely known when it appeared in Martin Gardner’s *Mathematical Games* column in the *Scientific American* in 1973 [6]. Gardner presented his best attempt, enclosing 127 squares, and reported that Donald E. Knuth had found a fence of area 128, together with an informal proof that 128 cannot be exceeded. In 1978, Takakazu Shimauchi published a more rigorous argument for 128 being optimal [22]. In a 2025 paper by a subset of the authors [16] this result was reconfirmed with an integer linear-programming (ILP) model, which was solved by the solver Gurobi. They also found an improved solution of area 1586 (reported originally as 1588 because of a counting error) for the hexomino fence.

In this paper, we continue our work on the problem. After the problem definition in Section 2, we start with the pentomino fence in Section 3. We have taken great care to make the presented dynamic programming (DP) approach accessible to a general mathematically interested audience, with detailed explanations and extensive visualizations. Additionally, the code verifying pentomino optimality does not rely on external solvers and runs in a few minutes even on low-budget hardware.

In Section 4, we look at hexomino fences, whose maximum area was unknown until now. We use an optimized version of the dynamic programming approach to prove that 1597 is the maximum area, and we identify all optimal hexomino fences. We proceed to heptominoes and octominoes, without reaching optimality (Section 5), and then far beyond (Section 6). Finally, in Section 7, we turn to three dimensions: we determine the maximum enclosable volume with pentacubes and find heuristic solutions for hexacubes and heptacubes.

Repository with programs and results. The programs and result files of our calculations are published on GitHub [15]. We also provide an interactive exploration of the main results [14]. The workflow of the computations, including the programs and the files that they exchange, is described in the README file of the repository.

2 Precise definition of the problem

The informal description in the introduction leaves some questions open that must be settled before a computer search can claim to have found an optimal fence: Which cells count as enclosed? What if the enclosed cells fall apart into several parts? What happens with pieces that lie inside the enclosed area? In this section, we give precise definitions. Figure 1 shows an arrangement of pieces of various sizes that exhibits many of the fine points. It is far from optimal, and because of the free-floating triomino T , it is not even a valid fence. Nevertheless, we will refer to it throughout this section.

2.1 Fences and their area

Recall from the introduction that two cells are *4-adjacent* if they share an edge, and *8-adjacent* if they share an edge or a vertex. We are given a finite collection of polyominoes, the *pieces*. In the n -omino fence problem, these are the free n -ominoes, each of them exactly once, but the definitions make sense for any collection of polyominoes, as in Figure 1. A *placement* puts every piece into the grid, after an arbitrary rotation, reflection, and translation.

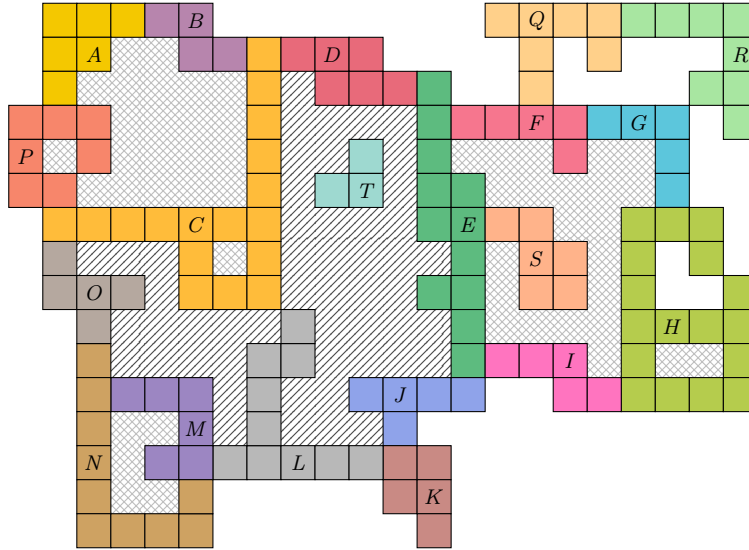


Figure 1: An arrangement of 20 polyomino pieces of different sizes, labeled *A* to *T*, which illustrates the definitions of Section 2. The largest single region, which determines the area (51 cells), is hatched. The other five regions (21, 21, 5, 2, and 1 cells) are cross-hatched; they do not count. The white cells surrounded by *F*, *Q*, *R*, and *G*, and the three white cells in the upper part of *H* are not enclosed, since they are 8-connected to the outside. Because of the free-floating triomino *T*, the arrangement is not a valid fence.

Definition 1. A placement is a fence if the placed pieces are pairwise disjoint and their union Φ is 4-connected. The cells of Φ are the fence cells, and all other cells are free. Since Φ is finite, exactly one 8-connected component of free cells is infinite; we call it the outside. The free cells that do not belong to the outside are enclosed, and the 8-connected components of the set of enclosed cells are the regions of the fence.

We define the area of the fence as the number of cells of a largest region, or 0 if there is no region.

The *n*-omino fence problem asks for the largest area of a fence built by using each free *n*-omino exactly once.

Notions of connectivity. Free cells are regarded as connected even if they touch only at a corner, whereas pieces must share an edge in order to be connected. These two conventions fit together, by a basic fact of digital topology [20, 13]. Enclosing an 8-connected set *X* of cells and cutting it off from other 8-connected regions requires a 4-connected barrier (and vice versa). Thus, the pieces of such a wall must be joined along edges. Figure 1 shows the consequences:

- The eight white cells surrounded by *F*, *Q*, *R*, and *G* look as if they were enclosed, but they are 8-connected to the outside through the diagonal gap where *G* and *R* touch at a corner. For the same reason, the three white cells in the upper part of *H* are not enclosed: They reach the outside through a diagonal gap between two cells of the piece *H* itself.
- Conversely, enclosed cells that touch only at a corner belong to the same region. The hatched region consists of a part of 15 cells on the left and a part of 36 cells in the middle, which are connected only through the diagonal gap between *C* and *L*. Together, they form a single region of 51 cells. Similarly, the cell that is surrounded by *P* on all four sides belongs to the region at the upper left, since it touches this region at a corner.

Connected area. In our definition we only count the largest region as the area of the fence. The arrangement in Figure 1 has six regions, with 51, 21, 21, 5, 2, and 1 cells, respectively, and its area according to our definition is 51. The smaller regions are cross-hatched: the regions of 21 cells at the upper left and at the right, the region of 5 cells between *M* and *N*, and the holes inside the pieces

C and H . Counting only the largest region captures the intention of the original puzzle, which asks for a single field of a “pentomino farm” [5]. In particular, a hole inside a single piece will hardly ever determine the area. This becomes relevant when we consider the octominoes in Section 5.1. We discuss the alternative of counting the total enclosed area in Appendix A.

Connectivity of the fence, and unused pieces. The free-floating triomino T of Figure 1 violates the requirement that Φ must be 4-connected. It lies inside the hatched region, but its cells are not free and do not count: The hatched region has 51 cells, although its outer boundary encloses 54 cells. Moving T to the outside would enlarge the region to 54 cells.

The requirement of 4-connectivity does not change the maximum area. To see this, consider any placement of pairwise disjoint pieces, not necessarily 4-connected, with a largest region $\mathcal{R}$. By the fact quoted above, the fence cells that are 8-adjacent to $\mathcal{R}$ and lie in the unbounded 4-connected component of the complement of $\mathcal{R}$ form a 4-connected set. Hence they belong to a single 4-connected component K of Φ , which is a union of pieces. We remove all pieces that do not belong to K and attach them, one after the other, to the outside of the fence, for example directly on top of the currently highest fence cell. An attached piece lies completely in the outside, and therefore it cannot destroy or shrink a region. The cells of the removed pieces that lay inside $\mathcal{R}$, like T , become free and join the region, because they are surrounded by $\mathcal{R}$ and K . The result is a fence whose largest region contains $\mathcal{R}$.

By the same procedure, pieces that play no role in enclosing the largest region can always be moved to the outside. In the n -omino fence problem, where every piece must be used, such pieces are thus harmless: effectively, the problem allows using an arbitrary subset of the pieces.

2.2 The boundary around the largest region

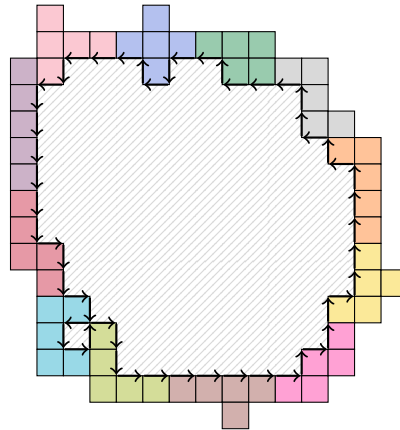


Figure 2: The counterclockwise boundary around the largest region is indicated by the arrows. Note that the pentomino fence has area 119, since the single extra cell in the bottom-left corner is 8-connected to the other cells.

Let $\mathcal{R}$ be a region of a fence. Its *boundary* consists of the unit edges between a cell of $\mathcal{R}$ and a cell not in $\mathcal{R}$. The latter is always a fence cell, because a free cell that is adjacent to $\mathcal{R}$ would belong to $\mathcal{R}$. Every 4-connected component of the complement of $\mathcal{R}$ contains such fence cells, and since Φ is 4-connected and disjoint from $\mathcal{R}$, it lies in a single 4-connected component. Hence the complement of $\mathcal{R}$ is 4-connected; in other words, no piece can lie inside $\mathcal{R}$. It follows that the boundary of $\mathcal{R}$ forms a single closed walk, which we orient counterclockwise around $\mathcal{R}$, so that $\mathcal{R}$ lies on the left side and the fence on the right side. At a vertex where two cells of $\mathcal{R}$ touch diagonally, the walk passes twice. There, we connect the edges such that the walk turns right and continues along the same fence cell. Examples are the corner between C and L in Figure 1 and the bottom-left corner in Figure 2. The area enclosed by this closed walk, as computed by the Shoelace formula in Section 3.1, is exactly $|\mathcal{R}|$.

Walking along the boundary of $\mathcal{R}$, we record the fence cells that we pass: for each edge, the fence cell on its right side, and at each left turn, where the walk goes around a convex corner of $\mathcal{R}$, additionally the cell that is diagonally opposite to the corner cell of $\mathcal{R}$. This cell is a fence cell as well, since it is

8-adjacent to $\mathcal{R}$. In this way, we record exactly the fence cells that are 8-adjacent to $\mathcal{R}$. Consecutive recorded cells are 4-adjacent: After a straight step, the two cells on the right side are 4-adjacent. After a right turn, the cell on the right side is the same. At a left turn, the diagonally opposite cell is 4-adjacent to the cells on the right side before and after the turn. Replacing every recorded cell by the piece containing it and merging consecutive repetitions, we obtain a cyclic sequence of pieces, the *pieces around $\mathcal{R}$* . By construction, consecutive pieces share an edge.

For the largest region in Figure 1, the pieces around the region, in counterclockwise order starting at the bottom, are $L, K, J, E, D, C, O, N, M$. The piece K touches the region only at a vertex, and it is recorded only at a left turn. Nevertheless, it is indispensable: J and L touch only at a corner, and without K , the region would leak to the outside through this diagonal gap. For the region at the right, the sequence is I, H, G, F, E, S, E . The piece E occurs twice, because S is attached to E and protrudes into the region. Removing S would enlarge this region. The following lemma shows that this cannot happen for a largest region of an optimal fence.

Lemma 1. *Let $\mathcal{R}$ be a largest region of a fence of maximum area. Then every piece occurs at most once in the cyclic sequence of pieces around $\mathcal{R}$. Consequently, for every piece P , the edges of the boundary of $\mathcal{R}$ that belong to P form a single connected (possibly empty) interval along the boundary of P .*

Proof. Suppose that some piece P occurs twice, so that the cyclic sequence has the form P, σ_1, P, σ_2 with nonempty sequences σ_1 and σ_2 . Let a and c be cells of P that were recorded at the two occurrences, and let η be a 4-connected path from a to c within P . The path η runs outside of $\mathcal{R}$ and connects two points on the boundary of $\mathcal{R}$. By the Jordan curve theorem, it cuts off a bounded *pocket* from the complement of $\mathcal{R}$. The pocket is bounded by η and by the part of the boundary walk between the two occurrences of P on one side, say, on the side of σ_1 . Every piece of σ_1 lies in the pocket, because it is 4-connected and can neither cross P nor $\mathcal{R}$.

We remove all pieces that lie in the pocket and attach them to the outside of the fence, as described in Section 2.1. The remaining pieces are still 4-connected, because every 4-connected path in Φ that passes through the pocket must enter and leave it through P . The cells of the pocket become free. They are surrounded by P and $\mathcal{R}$, and $\mathcal{R}$ is still surrounded by P and the pieces of σ_2 . Hence the freed cells are enclosed. Since the recorded cells of σ_1 are 8-adjacent to $\mathcal{R}$, the freed cells together with $\mathcal{R}$ belong to a region that is strictly larger than $\mathcal{R}$, contradicting the optimality of the fence.

The idea is illustrated in Figure 3a: there, P is the I-pentomino, σ_1 consists of the X-pentomino, and the X-pentomino lies in the pocket cut off by the two occurrences of P . A piece of the pocket may of course also be adjacent to a different enclosed region, as in Figure 3b; but then it occurs in the boundary walk of that region and not in that of $\mathcal{R}$, so each boundary walk still contains it only once.

For the second statement, consider two consecutive edges of the boundary walk that both have a cell of P on their right side. Between them, the walk either goes straight, or it turns right around a corner of the same cell, or it turns left around a vertex whose diagonally opposite cell belongs to P as well (since otherwise another piece would be recorded between them). In each case, the two edges are consecutive on the boundary of P , traversed clockwise around P . $\square$

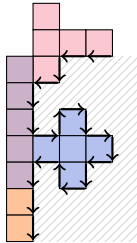


Figure 3a: Here the I-pentomino occurs twice in the boundary, but we can simply remove the *pocket* formed by the X-pentomino, which only increases the area.

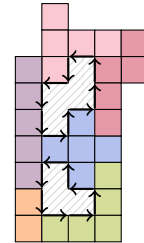


Figure 3b: If a piece in the *pocket* is used elsewhere in the fence, we must have two separate regions of enclosed cells and in their separate boundaries each pentomino occurs only once.

Lemma 1 justifies the model that we use in the rest of the paper. The pieces around the largest region $\mathcal{R}$ of an optimal fence form a cyclic sequence $\pi_1, \dots, \pi_k$ of distinct pieces, in which consecutive

pieces share an edge. Every piece π_i contributes a single consecutive section of its boundary to the boundary walk of $\mathcal{R}$. By the same local analysis as in the proof, the edge of π_i immediately before this section is shared with π_{i-1} , and the edge immediately after it is shared with π_{i+1} ; these will be the *incoming* and *outgoing attachment edges* in Section 3. The section can be empty if the piece touches $\mathcal{R}$ only at a vertex, like K in Figure 1. The area $|\mathcal{R}|$ is obtained by applying the Shoelace formula to the concatenation of these sections. Pieces that do not occur around $\mathcal{R}$ play no role. A piece may border several regions, like C in Figure 1, but only its contact with the largest region matters.

3 Pentomino fences

3.1 Edge boundary

As already mentioned in Section 2.2, instead of counting the number of cells, the algorithm will focus on the counterclockwise boundary shown in Figure 2. So we are given a closed edge boundary with counterclockwise edges $f_1, \dots, f_t$, where edge f_i starts from coordinates $(x(s_i), y(s_i))$ and ends at $(x(e_i), y(e_i))$. Then the Shoelace formula, first described by Albrecht Meister in 1769 [17], gives us the area as:

$$A = \frac{1}{2} \left[\sum_{i=1}^t x(s_i)y(e_i) - x(e_i)y(s_i) \right]. \quad (1)$$

Note that since the edges are connected we have $x(e_i) = x(s_{i+1})$ and $y(e_i) = y(s_{i+1})$, with the wrap-around $e_t = s_1$.

To compute the Shoelace formula more efficiently during the search algorithm, we use a lookup table and precompute all possibilities for how a pentomino can be part of the edge boundary. This means that, for each pentomino, we go over its up to 8 distinct orientations, obtained by rotation and reflection, and then look at all consecutive edge sections $\{f_p, \dots, f_q\}$ of the boundary of the pentomino itself. As the example in Figure 4 shows, these edges are oriented counterclockwise around the edge boundary, but clockwise with respect to the pentomino's own boundary. Then we calculate

$$\Delta A_{p,q} := \sum_{i=p}^q x(s_i)y(e_i) - x(e_i)y(s_i),$$

while assuming that the start of the first boundary edge, i.e., $(x(s_p), y(s_p))$, is shifted to $(0, 0)$. Note that, as seen with piece K in Figure 1, the incoming and outgoing attachment edges (shown in red in Figure 4) can also be directly adjacent. In that case, we just have $\{f_p, \dots, f_q\} = \emptyset$ and $\Delta A_{p,q} = 0$. During the search, we use the following lemma.

Lemma 2. *When we place the pentomino with the start of the first boundary edge at $(\Delta x, \Delta y)$, the contribution of its edges to the Shoelace formula is*

$$\sum_{i=p}^q [x(s_i)y(e_i) - x(e_i)y(s_i)] = \Delta A_{p,q} + \Delta x d_y - \Delta y d_x, \quad (2)$$

where we define (d_x, d_y) as the difference between the end e_q and start vertex s_p of the edge section $\{f_p, \dots, f_q\}$.

Proof. Write $x_0(\cdot), y_0(\cdot)$ for the precomputed coordinates used to obtain $\Delta A_{p,q}$, i.e., those satisfying $(x_0(s_p), y_0(s_p)) = (0, 0)$ and $(x_0(e_q), y_0(e_q)) = (d_x, d_y)$. This means

$$x(s_i) = x_0(s_i) + \Delta x, \quad y(s_i) = y_0(s_i) + \Delta y,$$

and so for a single Shoelace term we have

$$\begin{aligned} x(s_i)y(e_i) - x(e_i)y(s_i) &= (x_0(s_i) + \Delta x)(y_0(e_i) + \Delta y) - (x_0(e_i) + \Delta x)(y_0(s_i) + \Delta y) \\ &= [x_0(s_i)y_0(e_i) - x_0(e_i)y_0(s_i)] + \Delta x[y_0(e_i) - y_0(s_i)] + \Delta y[x_0(s_i) - x_0(e_i)]. \end{aligned}$$

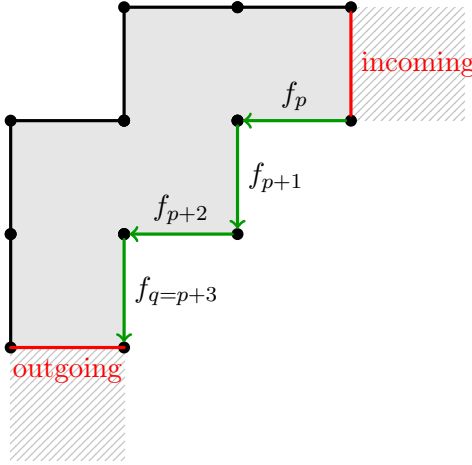


Figure 4: One possibility of using the W-pentomino. The green directed edges form part of the boundary of the enclosed region. The red edges are the attachment edges, i.e., the edges shared with the neighboring pentominoes, whose cells are shaded. They will become important later for ensuring edge-to-edge connectivity.

Summing over $i = p, \dots, q$, the first bracket sums to $\Delta A_{p,q}$ by definition. For the other two sums, connectivity gives $x_0(e_i) = x_0(s_{i+1})$ and $y_0(e_i) = y_0(s_{i+1})$, so both telescope to

$$\sum_{i=p}^q [y_0(e_i) - y_0(s_i)] = y_0(e_q) - y_0(s_p), \quad \sum_{i=p}^q [x_0(s_i) - x_0(e_i)] = x_0(s_p) - x_0(e_q).$$

Since $(x_0(s_p), y_0(s_p)) = (0, 0)$ and $(x_0(e_q), y_0(e_q)) = (d_x, d_y)$, these reduce to d_y and $-d_x$, respectively, and the formula is proven. $\square$

Before proceeding, we want to discuss a nice geometric interpretation of $\Delta A_{p,q}$. All steps of the dynamic programming search for the pentominoes can be understood without it, but it will be very useful for the hexomino fence.

For the consecutive edge section $\{f_p, \dots, f_q\}$, we connect its starting vertex $(\Delta x, \Delta y)$ with its ending vertex $(\Delta x + d_x, \Delta y + d_y)$ and the origin $(0, 0)$ to form a triangle, as seen in Figure 5a. By the Shoelace formula, its area is $A = \frac{1}{2} \left[\sum_{i \in \{a,b,c\}} x(s_i)y(e_i) - x(e_i)y(s_i) \right]$, and the summand corresponding to edge c is:

$$x(s_c)y(e_c) - x(e_c)y(s_c) = \Delta x \cdot (\Delta y + d_y) - \Delta y \cdot (\Delta x + d_x) = \Delta x d_y - \Delta y d_x.$$

If we instead look at the polygon formed by edges a , b and $\{f_p, \dots, f_q\}$, the summands for a and b stay the same and by Equation (2) the remaining edges contribute $\Delta A_{p,q} + \Delta x d_y - \Delta y d_x$. Thus, the only differing term $\Delta A_{p,q}$ describes exactly what is added when we replace the direct vector c with the polyomino boundary $\{f_p, \dots, f_q\}$.

In the example shown in Figure 5a, this corresponds to the area of the two green triangles. They have total area 1, and since the Shoelace formula multiplies every contribution by $\frac{1}{2}$, the value of $\Delta A_{p,q}$ is exactly twice that, as a simple calculation confirms.

In some cases, like the P-pentomino in Figure 5b, we can also lose some area by switching to the polyomino boundary instead of the direct vector (shown with red dots). So we have to subtract the areas that are now lost from the areas gained, which in this specific case comes out to 0, but can also sometimes be a negative value.

When we later work with such vectors directly in Section 4.1.1, the analogous quantity will be called the *correction* (Definition 3); there, the vectors connect the midpoints of the attachment edges instead of the vertices s_p and e_q .

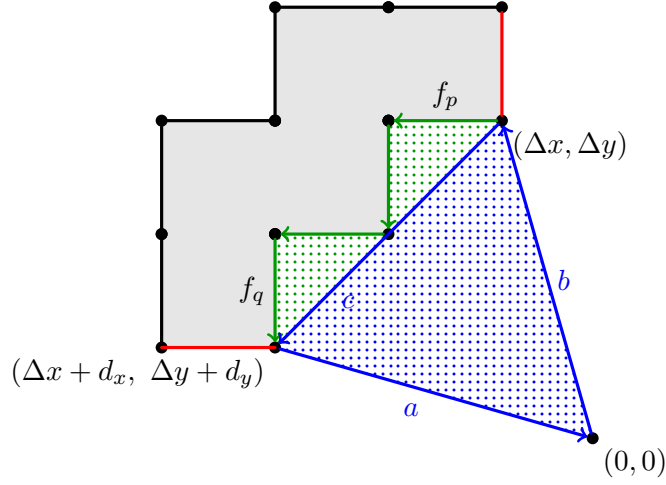


Figure 5a: The difference $\Delta A_{p,q}$ describes exactly the difference between the area of the blue triangle and the polygon formed from edges a , b and $\{f_p, \dots, f_q\}$. Here, this difference is positive and drawn as two green triangles.

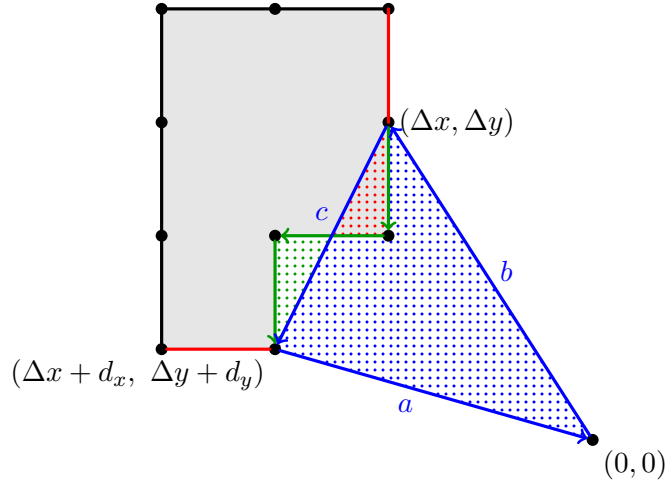


Figure 5b: Here the difference between the area of the blue triangle and the polygon formed from edges a , b and $\{f_p, \dots, f_q\}$ consists of gained area (green triangle) and lost area (red triangle).

3.2 Dynamic programming with fixed order

3.2.1 Algorithm

We begin with the restricted case in which the pentominoes are placed in a fixed order $\pi = (\pi_1, \dots, \pi_{12})$, and discuss the full search over all orders in Section 3.3.

The key idea is to build the edge boundary one pentomino at a time and track the running Shoelace sum along the way. Crucially, since the Shoelace formula (1) is a simple sum over individual edges, we can evaluate it incrementally: each time we append a new pentomino's edge segment to the boundary, we add the corresponding summands directly, without needing the boundary to be closed yet. This allows us to reason about partial boundaries and define an exact objective for each intermediate state of the construction.

We fix the starting grid vertex at the origin $(0,0)$ and place pentominoes one by one, each time extending the open boundary chain by one pentomino's edge segment. After placing the first i pentominoes, the partial boundary is a connected open chain running from $(0,0)$ to some endpoint $(x,y) \in \mathbb{Z}^2$.

Definition 2. $T_i(x,y)$ is the maximum accumulated Shoelace sum over all ways to place $\pi_1, \dots, \pi_i$ such that the partial boundary ends at (x,y) .

The answer to the fixed-order problem is then $\frac{1}{2}T_{12}(0,0)$: the boundary must return to the origin to close, giving the area of the fence with the $\frac{1}{2}$ coming from the Shoelace formula (1). The base case

is $T_0(0,0) = 0$, since the empty boundary starts and ends at the origin with no Shoelace summands.

For the recurrence, recall from Section 3.1 that we precompute a set of all possibilities of using pentomino π_i in one of its 8 orientations. Each possibility is a triple $(\Delta A, d_x, d_y)$, where ΔA is the Shoelace contribution of the pentomino's edge segment when its starting vertex is placed at the origin, and (d_x, d_y) is the displacement from that starting vertex to the endpoint of the segment. We denote the set of all possibilities for using pentomino π_i , over all its orientations, by Poss_i .

To compute $T_i(x, y)$, we consider all ways of reaching (x, y) by appending π_i to a partial boundary ending at the predecessor position $(x - d_x, y - d_y)$. Applying (2) with $(\Delta x, \Delta y) = (x - d_x, y - d_y)$ gives the Shoelace gain of this step:

$$\Delta A + (x - d_x) d_y - (y - d_y) d_x.$$

The recurrence is therefore:

$$T_i(x, y) = \max_{(\Delta A, d_x, d_y) \in \text{Poss}_i} \Delta A + (x - d_x) d_y - (y - d_y) d_x + T_{i-1}(x - d_x, y - d_y). \quad (3)$$

Reintroducing the edge-to-edge connectivity constraint. So far, the recurrence relation ignores the requirement that consecutive pentominoes must share a full grid edge and not merely touch at a corner: As the example in Figure 6 shows, we only check that the boundary edges of the previous pentomino end where the boundary edges of the next one start.

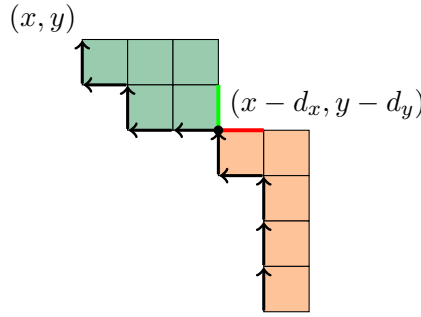


Figure 6: One example demonstrating that the simple recurrence equation does not enforce the edge-to-edge connectivity constraint. The green edge of the P-pentomino does not connect to the red edge of the L-pentomino.

For this we now look at the clockwise edge boundary of the pentomino itself, and define the *incoming attachment edge* as the edge immediately before f_p (the first edge of the section on the inner boundary of the fence) and the *outgoing attachment edge* as the edge immediately after f_q (the last edge of this section). We again refer to Figure 4 for a visualization; in Figure 6, the outgoing attachment edge of the L-pentomino is drawn in red and the incoming attachment edge of the P-pentomino in green.

To ensure that the pentominoes are connected edge to edge, we simply introduce the additional constraint that the new incoming attachment edge must lie in the same cardinal direction $\tau \in \{N, E, S, W\}$ with respect to the vertex $(x - d_x, y - d_y)$ as the previous outgoing attachment edge.

Accordingly, we augment the DP state to $\psi = ((x, y), \tau)$, where (x, y) is the current endpoint of the partial boundary and τ is the required incoming attachment edge orientation for the next pentomino. The precomputed possibilities are indexed by the incoming orientation, so $\text{Poss}[\pi_i, \tau]$ contains the tuples $(\tau', \Delta A, d_x, d_y)$ valid when π_i is attached from direction τ , and τ' is the resulting outgoing attachment edge orientation. The initial state is $\psi_0 = ((0,0), \tau_0)$ for a fixed starting orientation τ_0 , and a completed fence corresponds to returning to ψ_0 after all 12 pentominoes have been placed.

Algorithm `OPTIMALORDEREDFENCE` implements this procedure. The DP table T_i is a dictionary mapping states $\psi = ((x, y), \tau)$ to the best Shoelace sum achieved so far by any partial boundary of length i that reaches ψ . At each step i , we iterate over all states in T_{i-1} and all valid possibilities for π_i given the current orientation. The translation-corrected gain A_{add} is computed via (2) using the predecessor position (note that now (x, y) refers to the predecessor coordinates, i.e., (d_x, d_y) is already subtracted), and the new state ψ' records the updated position and outgoing orientation. On the

A320
A321
A322

final step $i = 12$, only transitions that return to ψ_0 are permitted, enforcing closure of the boundary. The algorithm tracks the best closed-boundary Shoelace sum (twice the area) in $\hat{A}_{\max}$ and returns $A_{\max} := \frac{1}{2}\hat{A}_{\max}$ as the optimal area.

Algorithm 1 OPTIMALORDEREDFENCE(π)

```

1: Input: order  $\pi = (\pi_1, \dots, \pi_{12})$ , precomputed possibilities Poss
2:  $\psi_0 \leftarrow ((0, 0), \tau_0)$  ▷ fixed start position and orientation
3:  $T_0[\psi_0] \leftarrow 0$  ▷ initialize the dynamic program
4:  $\hat{A}_{\max} \leftarrow 0$ 
5: for  $i = 1$  to 12 do
6:    $T_i \leftarrow \emptyset$ 
7:   for each state  $((x, y), \tau)$  with value  $A$  in  $T_{i-1}$  do
8:     for each  $(\tau', \Delta A, (d_x, d_y)) \in \text{Poss}[\pi_i, \tau]$  do
9:        $A_{\text{add}} \leftarrow \Delta A + xd_y - yd_x$  ▷ translation correction in the Shoelace sum
10:       $\psi' \leftarrow ((x + d_x, y + d_y), \tau')$ 
11:      if  $i = 12$  and  $\psi' \neq \psi_0$  then
12:        continue ▷ enforce boundary closure on the final step
13:      end if
14:       $T_i[\psi'] \leftarrow \max\{T_i[\psi'], A + A_{\text{add}}\}$ 
15:      if  $\psi' = \psi_0$  then ▷ record the best closed boundary found so far
16:         $\hat{A}_{\max} \leftarrow \max\{\hat{A}_{\max}, A + A_{\text{add}}\}$ 
17:      end if
18:    end for
19:  end for
20: end for
21: return  $\frac{1}{2}\hat{A}_{\max}$ 

```

A323
A324
A325
A326
A327

Multiple shared edges. Note that our search method does not prevent consecutive pentominoes from sharing more than one edge, as shown in Figure 7. Depending on which edge is chosen as the respective incoming/outgoing attachment edge, the remaining shared edges appear as back-and-forth appendices on the boundary. However, these appendices contribute zero to the Shoelace sum: each such edge is traversed once in each direction, so its two contributions cancel exactly.

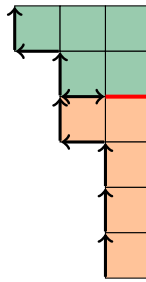


Figure 7: If consecutive pentominoes share several edges, then, depending on the choice of the attachment edge (drawn in red), the inner boundary can contain appendices (here the double-arrow edge). They cancel out in the Shoelace formula and contribute zero area, so they can be safely ignored.

A328
A329
A330
A331
A332
A333

Solution reconstruction and self-intersection. Each DP table entry stores only the current state $\psi = ((x, y), \tau)$, discarding all information about which orientations and attachment edges were used for the pentominoes placed so far. This compression is what makes the search tractable, but it means OPTIMALORDEREDFENCE cannot directly output a concrete pentomino placement, nor can it detect whether any candidate placement is *self-intersecting*, i.e., whether two pentominoes overlap on the same grid cell.

Self-intersection is a genuine concern because the Shoelace formula remains well-defined even for self-crossing boundaries: it computes area weighted by the winding number, so any doubly-covered region is counted twice. An example is shown in Figure 8, where overlapping pentominoes cause the boundary to cross itself and the Shoelace value exceeds the true enclosed area. This means the update

$$T_i[\psi'] \leftarrow \max\{T_i[\psi'], A + A_{\text{add}}\}$$

could in principle promote a value originating from an invalid self-intersecting placement, making $\frac{1}{2}\hat{A}_{\text{max}}$ technically an upper bound rather than a confirmed fence area. Additionally, the enclosed cells could consist of multiple regions and by our definition we would only count the largest of them as the fence's area.

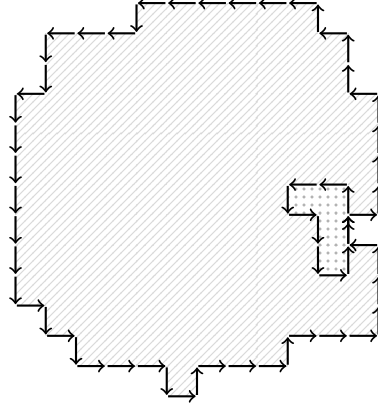


Figure 8: Example of a self-intersecting boundary that can arise when placed pentominoes overlap. The doubly-covered cells (dotted region) cause the boundary to cross itself; the Shoelace formula counts the area of this region twice via its winding number.

The absence of a concrete solution and the self-intersection ambiguity can both be resolved by solution reconstruction. We maintain a second table Pred alongside T : whenever a transition with $A + A_{\text{add}} > T_i[\psi']$ causes an update, we record $\text{Pred}_i[\psi'] \leftarrow (\psi, p)$, where ψ is the predecessor state and p is the possibility used for π_i . After the search, the optimal placement is recovered by tracing back through Pred : starting from $\text{Pred}_{12}[\psi_0]$ gives the state ψ_{11} and how π_{12} was placed; then $\text{Pred}_{11}[\psi_{11}]$ gives ψ_{10} and how π_{11} was placed; and so on. With the explicit placement in hand, we can check directly whether any two pentominoes overlap. If no overlap is found, which was always the case in practice, the self-intersection concern is eliminated. After checking that all enclosed cells belong to the same region with a breadth-first search (BFS), we could always confirm $\frac{1}{2}\hat{A}_{\text{max}}$ as the true optimal fence area for the order π .

To enumerate *all* optimal solutions, one instead collects all predecessor pairs (ψ, p) with $A + A_{\text{add}} = T_i[\psi']$ and runs standard backtracking through Pred , verifying each reconstructed solution individually. It should be noted that when adjacent pentominoes share multiple edges, two different paths could correspond to the same pentomino fence. Although this never occurred in practice for optimal fences, one has to verify, strictly speaking, that all the produced solutions are actually distinct.

3.2.2 Solving the min-max problem

The *min-max problem* asks: over all possible orders π of the 12 pentominoes, what is the minimum value of the maximum fence area achievable under that order? We solve this by running the algorithm `OPTIMALORDEREDFENCE` exhaustively over all orders.

The number of distinct orders to consider is $\frac{11!}{2} = 19\,958\,400$, since by symmetry we can fix the first piece and the direction in which the pentominoes are placed. One nice optimization that speeds up the search is prefix reuse: if two consecutive orders in the enumeration share the same first t pentominoes, the DP tables $T_1, \dots, T_t$ and $\text{Pred}_1, \dots, \text{Pred}_t$ are identical and can be carried over without recomputation. To maximize the applicability of this optimization, we process the orders in lexicographic order.

A368
A369
A370
A371
A372
A373

We ran the full search multi-threaded on a machine with an AMD Ryzen 7 7735HS CPU and 20 GB of DDR5 memory, which took around 2 days to complete.

The distribution of optimal areas across all orders is shown in Figure 9. The minimum of the distribution, and so the answer to the min-max problem, is 111, achieved by 313 orders. One such min-max solution is shown in Figure 10. For hexominoes, we have only sampled random orders (Appendix B).

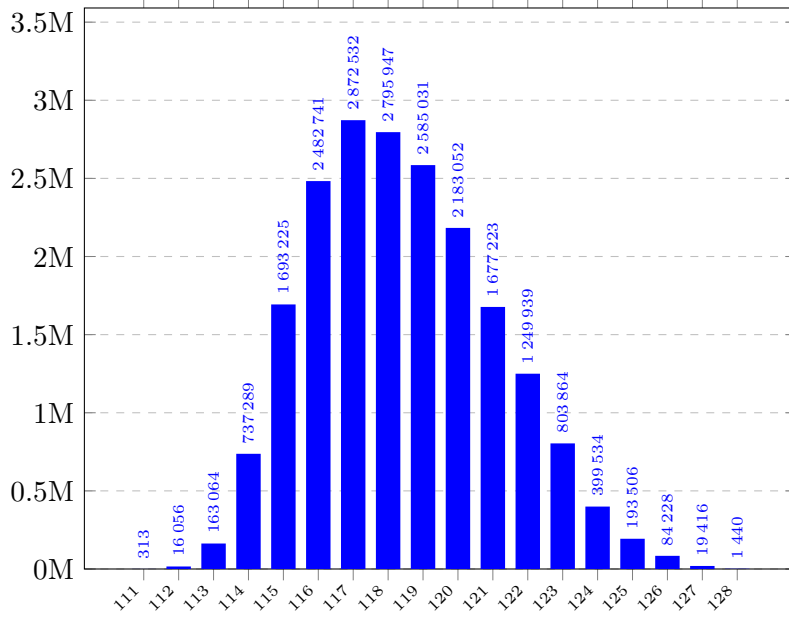


Figure 9: Distribution of the maximum achievable fence area across all $\frac{11!}{2}$ pentomino orders. The minimum value of 111 is achieved by 313 orders.

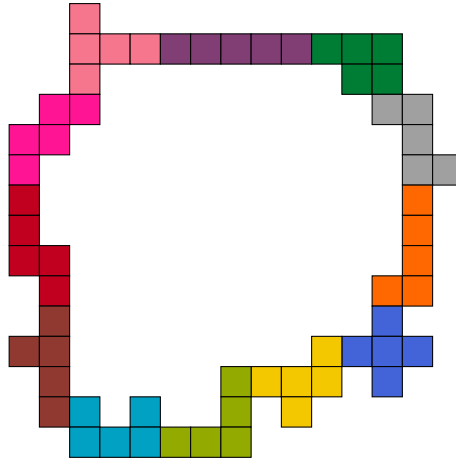


Figure 10: One of the 313 min-max pentomino fence orders, enclosing a maximum area of 111 grid cells.

A374
A375

Interactive lab. In the *pentomino DP lab* (<https://erikaroldanroa.github.io/fence-challenge/pentomino-dp-lab/>) the reader can follow the dynamic program step by step for any order.

A376

3.3 Dynamic programming with unrestricted order

A377
A378
A379

To solve the original pentomino fence problem, i.e., finding the largest area achievable by any valid placement, regardless of the order, we extend OPTIMALORDEREDFENCE with one additional loop. Rather than advancing to a fixed next pentomino π_i at each step, we iterate over all pentominoes

not yet used in the current partial boundary and try each one as the next piece. The DP state is augmented to record the set of used pentominoes alongside the position and orientation, and the recurrence otherwise remains identical. Thus, we can keep everything else in Algorithm 1.

Crucially, the state only needs to track *which* pentominoes have been used, not the order in which they were placed. This is valid because the Shoelace sum accumulated so far is already fully captured by the value stored in the table: two partial boundaries that used the same set of pentominoes and arrive at the same state $\psi = ((x, y), \tau)$ with the same accumulated sum are entirely interchangeable from the perspective of any future extension, regardless of the order in which those pentominoes were placed.

State space size. We can easily bound the size of each table T_i by the total number of possible state attribute combinations. There are always just 4 possibilities for the orientation, and $\binom{12}{i}$ many ways of choosing the i used pentominoes. For the current coordinates (x, y) we can have all grid points reachable so far, which can be roughly estimated as $50i^2$, the approximate size of a Manhattan ball of radius $5i$ (since each pentomino contributes 5 cells in Manhattan distance). In Table 1, we illustrate that this estimate (third column) matches the actual number of reachable positions during the algorithm (fourth column) well.

Then the upper bound for $|T_i|$ is obtained by multiplying the 4 orientations with the $\binom{12}{i}$ used pentomino sets and finally the number of reachable positions. This means that the fifth column of Table 1 is the product of the second and fourth columns times 4. It shows that no $|T_i|$ ever contains more than a few million entries, which is also confirmed by the actual observed $|T_i|$ in the last column. The large discrepancy between actual size and upper bound for small i is caused by the fact that some pentomino subsets can reach far fewer coordinates.

i	$\binom{12}{i}$	$50i^2$	actual $ \{(x, y)\} $	upper bound $ T_i $	actual $ T_i $
1	12	50	57	2 736	730
2	66	200	212	55 968	29 519
3	220	450	456	401 280	245 919
4	495	800	784	1 552 320	1 028 126
5	792	1 250	1 172	3 712 896	2 621 789
6	924	1 800	1 628	6 017 088	4 451 545
7	792	2 450	2 092	6 627 456	5 230 563
8	495	3 200	2 604	5 155 920	4 292 220
9	220	4 050	3 172	2 791 360	2 424 160
10	66	5 000	3 796	1 002 144	900 744
11	12	6 050	4 476	214 848	198 688

Table 1: The upper bound and actual size of the DP table T_i at each step.

Note that the actual $|T_i|$ values reported in the last column are measured with the following symmetry breaking: After numbering the pentominoes, we require that the second placed pentomino has the lowest id of any pentomino used (to eliminate rotational symmetry) and that the first pentomino has a lower id than the third pentomino (to eliminate mirror symmetry).

This all means that except for the final check that the fence is closed at $i = 12$, we do not need any pruning. If instead we kept track of the order of pentominoes, the $\binom{12}{i}$ would turn into an $i!$ factor and make $|T_i|$ far too large to keep in memory.

Solution reconstruction. Solution reconstruction carries over unchanged in structure. The back-pointer table Pred now additionally records which pentomino was chosen at each step, so that the traceback recovers both the sequence of pentominoes and their placements. The self-intersection argument requires slightly more care: in the fixed-order case, an inflated Shoelace value from an overlapping placement can only displace a correct value for the same order. Here, it could in principle alter which pentomino appears to have been placed at which step, reordering the reconstructed sequence. However, the same verification argument holds: if no overlaps are detected in the reconstructed placement, then

the placement is geometrically valid regardless of order, and if BFS shows that all enclosed cells belong to the same region, $\frac{1}{2}\hat{A}_{\max}$ is confirmed as the true optimum. If overlaps or multiple regions are found, $\frac{1}{2}\hat{A}_{\max}$ remains an upper bound.

Running the search single-threaded on the same Ryzen 7 machine mentioned in Section 3.2.2, it only takes 2.5 minutes to complete. Note that when updating $\hat{A}_{\max}$ (line 15), we only check if the fence is closed and not if all pentominoes are already placed. Thus, each time we are finished building a given T_i and arrive at line 19, $\frac{1}{2}\hat{A}_{\max}$ corresponds to the largest area of a fence achievable by using any i of the 12 pentominoes. We present those intermediate values in Table 2. As already discussed at the end of Section 2.1, we observe that it is never advantageous to leave out a pentomino.

size i	1	2	3	4	5	6	7	8	9	10	11	12
max. fence area $A_{\max}(i)$	0	1	6	13	24	34	47	62	78	94	111	128
$A_{\max}(i) - A_{\max}(i - 1)$	–	1	5	7	11	10	13	15	16	16	17	17

Table 2: Largest enclosed area that can be obtained with i pentominoes.

3.4 Identification of all optimal pentomino fences

As discussed before, solution reconstruction can also be run in the all-solutions mode, enumerating every optimal placement. This yields exactly 5760 different solutions, which we can quickly verify to be valid and distinct. We call two solutions distinct if no combination of translation, rotation, and reflection maps one onto the other with every pentomino covering the same cells. Figure 11a shows one optimal solution.

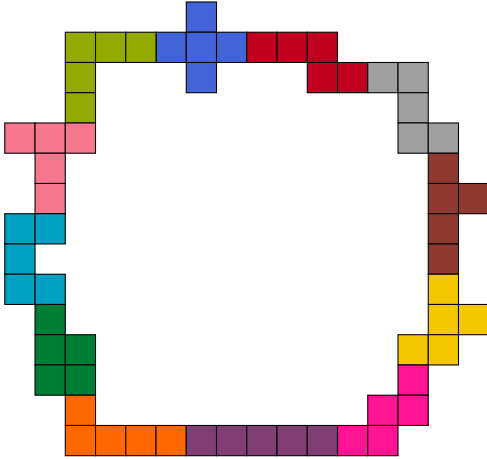


Figure 11a: One of the 5760 optimal pentomino fence solutions, enclosing an area of 128 grid cells. Note that if we reflect the gray Z-pentomino in its diagonal and/or the brown Y-pentomino in its horizontal axis, we get 3 further optimal solutions with the same order of pentominoes.

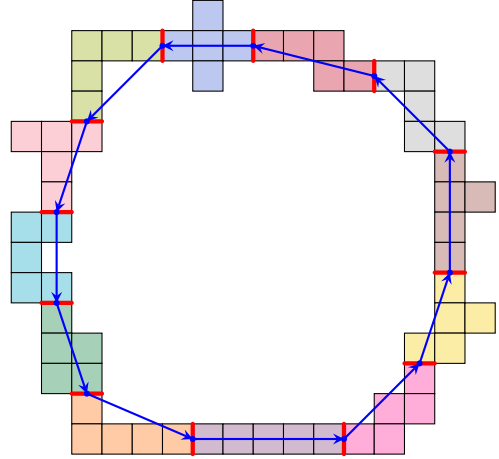


Figure 11b: The attachment edges (red) and the reference vectors (blue), which connect the red edges between neighboring pentominoes in counterclockwise direction.

Despite this large number, it is possible to get a systematic overview of the solutions. For this purpose, it is useful to abstract from the actual shapes of the pieces and consider only their *reference vectors*, which run between the midpoints of consecutive attachment edges, see Figure 11b. (This concept will play a prominent role when we consider hexominoes in Section 4, see Definition 3.) The reference vectors of a fence, taken together, form the *reference polygon*.

It turns out that every optimal fence has exactly four *corner pieces*, where the attachment edges change direction between horizontal and vertical. The remaining pieces are used in a “straight” way. This property is not surprising; it is also true for all optimal hexomino fences, and most likely also for n -omino fences with larger n .

Figure 12 shows how the twelve pentominoes are used in an optimal fence. They can be classified by their reference vectors into groups I–VI of sizes 1, 1, 2, 2, 3, 3. Each pentomino of groups IV and V

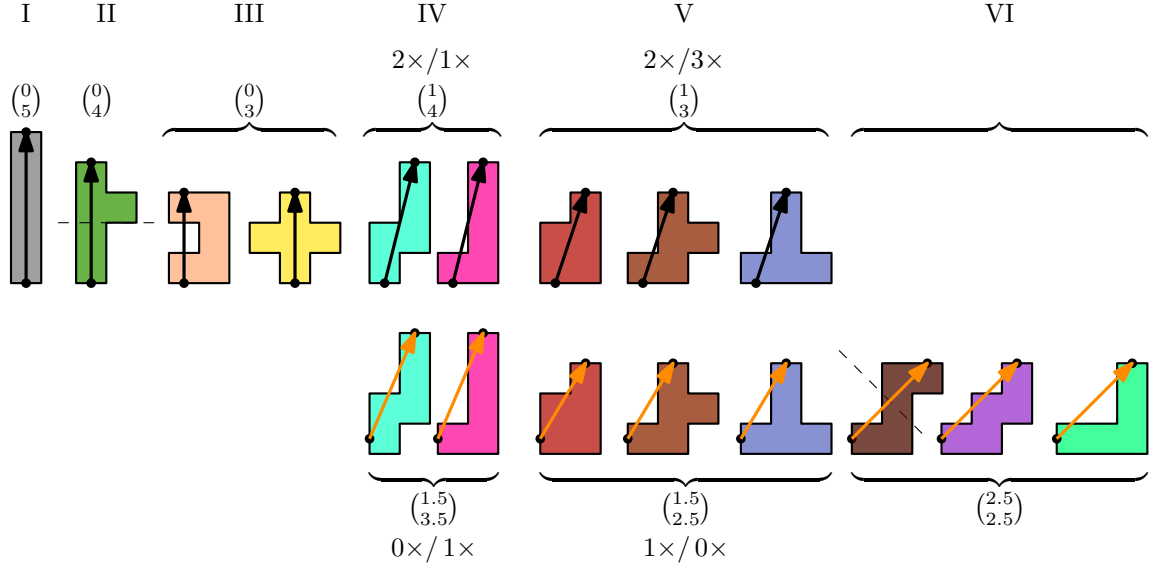


Figure 12: Usage of the parts in optimal pentomino fences. The lower row shows the ways in which a pentomino can serve as one of the four corner pieces, and the upper row shows the straight pieces.

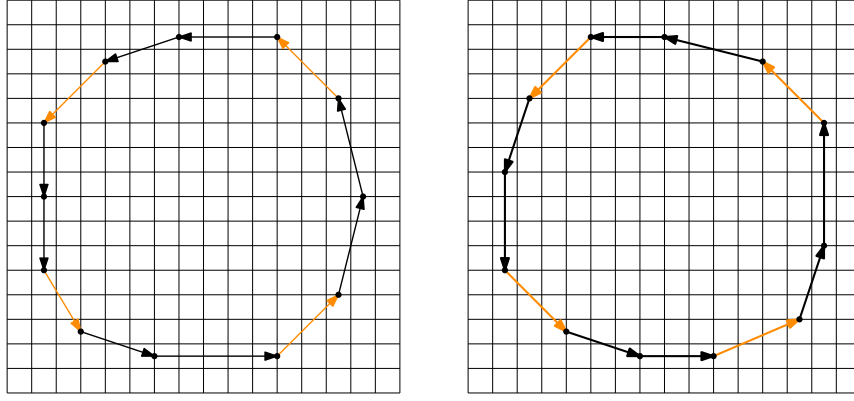


Figure 13: Two optimal reference polygons out of ten. The four corner vectors are highlighted in orange.

can be used in two possible ways, either as a corner piece or as a straight piece. All pieces of the same group are exchangeable.

The three pieces of group VI, with reference vector $(\frac{2.5}{2.5})$, can only be used as corner pieces. The fourth corner piece is either of the form $(\frac{1.5}{2.5})$ from group V, or $(\frac{1.5}{3.5})$ from group IV. (When we say “of the form $(\frac{1.5}{3.5})$ ”, we mean any of the eight rotated or reflected vectors $(\pm\frac{1.5}{\pm 3.5})$ or $(\pm\frac{3.5}{\pm 1.5})$.)

If we just look at the reference polygons, it turns out that there are just ten different reference polygons that occur in optimal solutions, two of which are shown in Figure 13. They are of two types:

1. There are six reference polygons that contain two reference vectors of the form $(\frac{1}{4})$ and two reference vectors of the form $(\frac{1}{3})$, see Figure 13 (left) for an example. The fourth corner vector is of the form $(\frac{1.5}{2.5})$.
2. There are four reference polygons that contain one reference vector of the form $(\frac{1}{4})$ and three reference vectors of the form $(\frac{1}{3})$, see Figure 13 (right) for an example. The fourth corner vector is of the form $(\frac{1.5}{3.5})$.

For a fixed reference polygon, we have to assign the pieces to the reference vectors. The pieces of each group can be arbitrarily permuted among the positions for which they fit. This leads to $2! \cdot 2! \cdot 3! \cdot 3! = 144$ possibilities, for a total of $10 \times 144 = 1440$ different optimal pentomino orders, which is in accordance with the data reported in Figure 9.

In addition, the Z-pentomino can always be reflected in its diagonal, and the Y-pentomino along its shorter axis, without changing the enclosed area, as indicated by the dotted axis lines in Figure 12. This leads to the total of $1440 \times 2 \times 2 = 5760$ optimal fences.

4 Hexomino fences

For the 35 hexominoes, the dynamic program of Section 3.3 is no longer feasible: tracking the set of used pieces alone introduces a factor of up to $\binom{35}{17} \approx 4.5 \cdot 10^9$, and the number of reachable positions grows as well. In this section we therefore exploit the geometric structure of good fences to reduce the search space drastically. The main result is the following.

Theorem 1. *The largest area that can be enclosed by a fence of the 35 hexominoes is 1597.*

A fence enclosing 1597 cells is shown in Figure 14. It was found with a stochastically adapted linear programming approach [15], improving the previously best known area of 1586 [16]. The remainder of this section proves that no fence encloses 1598 or more cells and then determines all optimal fences. In Section 4.1, we begin with a simpler problem for the dynamic program: We assume that it has already been decided *how* each hexomino is used, i.e., which pair of attachment edges is used, and we only search for the optimal order of the hexominoes. We develop various optimizations to make this simplified problem feasible. In Section 4.2, we describe a Branch-and-Bound preprocessing step that, for a given target area, narrows down the possible ways of using the hexominoes to a short list of candidates that might reach this area. For these candidates, we run the dynamic programming search of Section 4.1 to confirm that for none of them can the hexominoes be ordered to achieve an area of at least 1598.

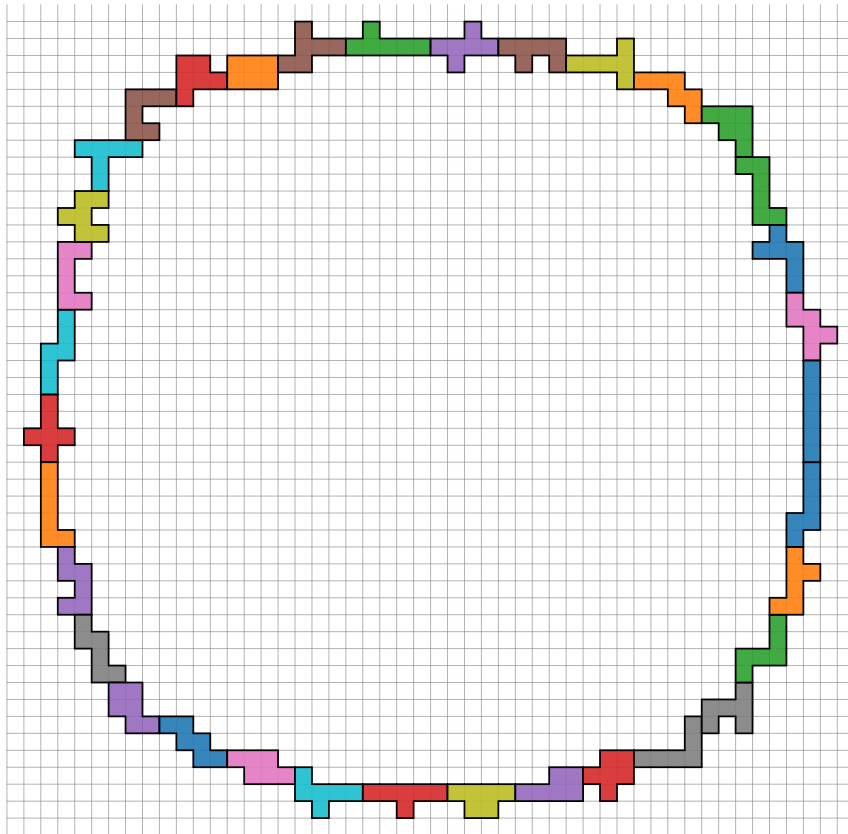


Figure 14: A hexomino fence enclosing the maximum possible area of 1597 cells.

A similar dynamic programming approach has been recently used by Har-Peled, Harb, and He [7] for a problem of a different flavor: enclosing a given number of grid points by the shortest rope, or more precisely, finding a set of grid points whose convex hull has the smallest possible perimeter.

Interactive lab. In the *hexomino lab* (<https://erikaroldanroa.github.io/fence-challenge/hexomino-lab/>) one can try building a hexomino fence and interact with one optimal solution.

4.1 Optimizations for dynamic programming

4.1.1 Reference vectors

By Lemma 1, consecutive pieces around the largest region share an attachment edge, and each piece has an *incoming* and an *outgoing* attachment edge. We orient every attachment edge from the enclosed region towards the outside. The midpoints of the attachment edges are the key to a more compact description of a fence, see Figure 11b. The following definition builds on the geometric observations at the end of Section 3.1, and Figure 15 illustrates it with a few examples.

Definition 3. Consider a piece that is used with an incoming attachment edge with midpoint m_{in} and an outgoing attachment edge with midpoint m_{out} , and let $f_p, \dots, f_q$ be the edges of the piece that lie on the boundary of the enclosed region between the two attachment edges, oriented clockwise around the piece as in Section 3.1.

1. The reference vector of the piece is $v := m_{\text{out}} - m_{\text{in}}$.
2. The correction Δ of the piece is the signed area enclosed by the closed path that starts at m_{in} , follows the incoming attachment edge inwards to the starting point s_p of f_p , then the boundary edges $f_p, \dots, f_q$, then the outgoing attachment edge outwards to m_{out} , and finally returns to m_{in} along $-v$.
3. The rotation D of the piece is the signed turning angle (counterclockwise turns counted positively) accumulated when walking from the incoming attachment edge along $f_p, \dots, f_q$ to the outgoing attachment edge, plus 180° .
4. The parity bit B is the Manhattan distance between the start points of the two attachment edges (that is, their inner endpoints), modulo 2.

Concatenating the reference vectors of the pieces $\pi_1, \dots, \pi_k$ around the largest region of a fence gives a closed polygon, the reference polygon. It is a (not necessarily simple) k -gon, which we traverse counterclockwise.

The quantities v , Δ , D , and B are invariant under translation, so they can be precomputed for every piece and every choice of attachment edges, just like the Shoelace contributions $\Delta A_{p,q}$ in Section 3.1. Since the vectors are attached to the midpoints of the attachment edges, which have half-integer coordinates, all corrections are multiples of $\frac{1}{8}$.

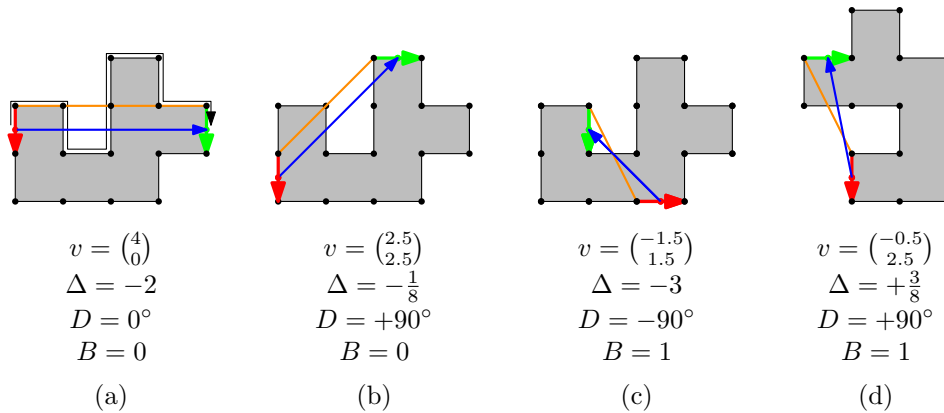


Figure 15: Some pairs of attachment edges, incoming in red and outgoing in green, together with the corresponding reference vectors v , corrections Δ , rotations D , and parity bits B .

The area of the reference polygon is a good approximation of the area of the fence, and the corrections account exactly for the difference:

Lemma 3. Let A_{ref} be the signed area of the reference polygon of a fence with pieces $\pi_1, \dots, \pi_k$ around the largest region and corresponding corrections $\Delta_1, \dots, \Delta_k$. Then the area of the fence is

$$A = A_{\text{ref}} + \Delta_1 + \Delta_2 + \dots + \Delta_k. \quad (4)$$

Proof. The signed area of a closed path is half its Shoelace sum, and the Shoelace sum is additive over the edges of the path. We add up the closed paths of Definition 3 for the pieces $\pi_1, \dots, \pi_k$. The reversed reference vectors contribute $-A_{\text{ref}}$. Every attachment edge is shared by two consecutive pieces and has the same inner endpoint for both: the first piece traverses the half edge outwards to the midpoint, and the second piece traverses it inwards, so these contributions cancel. What remains are the sections $f_p, \dots, f_q$ of all pieces, which together form the boundary walk of the largest region $\mathcal{R}$, by Lemma 1 and the discussion after it. By the Shoelace formula, their contribution is the area enclosed by this walk, which is $|\mathcal{R}| = A$, because no piece lies inside $\mathcal{R}$ (Section 2.2). Hence $\Delta_1 + \dots + \Delta_k = A - A_{\text{ref}}$. $\square$

Note that the lemma is a statement about signed areas and does not require the reference polygon to be simple. As Figure 16b shows, a valid fence can have a self-intersecting reference polygon. Conversely, as in Section 3, joining pieces at common attachment edges may produce overlapping placements (Figure 16a), for which formula (4) still yields a well-defined, but meaningless value.

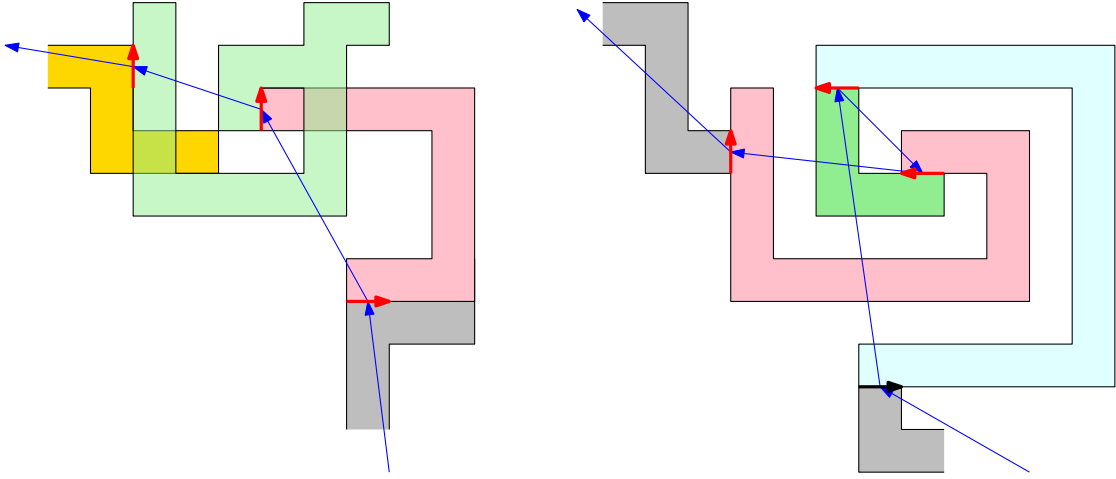


Figure 16: (a) Joining pieces at common attachment edges can lead to overlaps. (b) On the other hand, a reference polygon can intersect itself and still correspond to a valid fence.

The correction is usually negative. For example, a straight piece whose reference vector runs parallel to the inner boundary at distance $\frac{1}{2}$ loses a strip of width $\frac{1}{2}$, which gives $\Delta = -2$ in Figure 15a.

The rotation. The rotation D is a multiple of 90° . Most pieces in a good fence have parallel attachment edges and rotation $D = 0^\circ$, while the *corner pieces* have $D = +90^\circ$. Unusual pieces can also have $D = 180^\circ$ or $D = -90^\circ$ (an inward-turning piece), although it is hard to imagine them in an optimal fence. Swapping the incoming and outgoing edge changes the sign of D . We do not exclude any possibility a priori but consider all pairs of edges as potential attachment edges, and for hexominoes this results in rotations between -360° and $+360^\circ$. Note that D is determined as a signed angle, not only modulo 360° . At each attachment edge, the walk along the boundary makes a U-turn from the outgoing edge of one piece to the incoming edge of the next, which the added 180° in the definition of the rotation counts as a left turn. With this convention the turning angles along the edge boundary add up correctly, and since the edge boundary is a closed curve traversed counterclockwise, the rotations of all pieces of a fence satisfy

$$D_1 + D_2 + \dots + D_k = 360^\circ. \quad (5)$$

The parity bit. The vectors v'_i between the start points (inner endpoints) of the attachment edges form a closed polygon. For a closed integer k -gon with edge vectors $\begin{pmatrix} x_i \\ y_i \end{pmatrix}$, $\sum_{i=1}^k (x_i + y_i)$ is zero, and

therefore even. We allow the pieces to be reflected and rotated, which may swap x_i and y_i and negate them. Negating an integer does not change its parity, and swapping them does not change the parity of $x_i + y_i$. We can therefore conclude: The sum of the parity bits must be even, no matter how the pieces are reflected or rotated:

$$B_1 + B_2 + \dots + B_k \equiv 0 \pmod{2}. \quad (6)$$

This constraint will allow us to exclude combinations of reference vectors, because they cannot form a closed reference polygon.

The repertoire of reference quadruples. Algorithm 2 computes, for a single polyomino, all possible quadruples (v, Δ, D, B) . It enumerates the pairs of attachment edges exactly as the precomputation in Section 3.1; here, $w_1, \dots, w_t$ are the vertices of the polyomino boundary in clockwise order. For hexominoes, the boundary is always a simple cycle, so this vertex sequence is well defined. When this is no longer the case, which can happen from heptominoes on, we use a “right-hand rule”, which means that we always choose the rightmost edge from the current vertex. Starting with octominoes, we can have n -ominoes that have one or more separate boundaries on the inside, e.g., the 3×3 square with a missing center. In such cases we only use the outer boundary, since it makes no sense to put pieces into the holes of other pieces.

Algorithm 2 REFERENCEPERTOIRE(p)

```

1: Input: polyomino  $p$ 
2:  $\mathcal{V} \leftarrow \emptyset$ 
3: for each distinct orientation  $P$  of  $p$  do
4:    $(w_1, \dots, w_t) \leftarrow$  boundary vertices of  $P$  in clockwise order, indices taken modulo  $t$ 
5:   for  $s = 1$  to  $t$  do ▷ incoming attachment edge  $(w_s, w_{s+1})$ 
6:     for  $e = s + 1$  to  $s + t - 1$  do ▷ outgoing attachment edge  $(w_e, w_{e+1})$ 
7:        $m_{\text{in}} \leftarrow \frac{1}{2}(w_s + w_{s+1})$  and  $m_{\text{out}} \leftarrow \frac{1}{2}(w_e + w_{e+1})$ 
8:        $v \leftarrow m_{\text{out}} - m_{\text{in}}$ 
9:        $(u_0, u_1, \dots, u_r) \leftarrow (m_{\text{in}}, w_{s+1}, w_{s+2}, \dots, w_e, m_{\text{out}})$  and  $u_{r+1} \leftarrow u_0$  ▷ closed path
10:       $\Delta \leftarrow \frac{1}{2} \sum_{k=0}^r \det(u_k, u_{k+1})$  ▷ Shoelace formula
11:       $D \leftarrow \sum_{k=s}^{e-1} \text{turn}(w_{k+1} - w_k, w_{k+2} - w_{k+1}) + 180^\circ$  ▷ rotation
12:       $\begin{pmatrix} x' \\ y' \end{pmatrix} \leftarrow w_{s+1} - w_e$ ;  $B \leftarrow (x' + y') \bmod 2$  ▷ parity bit
13:      add  $(v, \Delta, D, B)$  to  $\mathcal{V}$ 
14:     end for
15:   end for
16: end for
17: return  $\mathcal{V}$ 

```

Here $\text{turn}(f, f')$ for two unit grid vectors f and f' is $+90^\circ$ if $\det(f, f') = 1$, -90° if $\det(f, f') = -1$ and 0° if f and f' point in the same direction. The case that they point in opposite directions is excluded. For the straight piece in Figure 15a, the walk turns right twice, so the sum of the turns is -180° , and $D = 0^\circ$.

The sums in lines 10 and 11 need not be computed from scratch as written; for better efficiency, they could, for example, be evaluated incrementally, although the computational effort of this preprocessing step is negligible in the context of the whole process.

4.1.2 Eight-way partitioning

Lemma 3 separates the area of a fence into two parts. The sum of the corrections does not depend on the order in which the pieces are joined, only on how each piece is used. The area of the reference polygon, on the other hand, depends on the order, but only through the reference vectors. If we have chosen a reference vector of every piece, it is then enough to arrange these vectors into a polygon of maximum area, and at the end we simply add the fixed sum of the corrections.

This is promising because optimal fences have a very regular structure. There are four corner pieces, and all other pieces have parallel attachment edges. Moreover, the attachment edges of almost all pieces are chosen so that the reference vector is as long as possible: For a “roundish” polygon such as the reference polygon that we are trying to build, the area grows quadratically with its perimeter. Thus, a slightly longer reference vector usually gains more than enough area to compensate a deterioration through a different correction. This is not a strict rule; the U-pentomino, for example, is not used with its longest reference vector in optimal pentomino fences, see Figure 12. Hexominoes are a similar case, see Section 4.4. When all reference vectors are fixed up to rotation and reflection, we arrive at the following auxiliary problem.

Problem MAXPOLYGON: the largest polygon from prescribed edge vectors, up to the 8 grid symmetries. Given vectors $(a_1), \dots, (a_N)$, each vector may be used in any of its up to eight rotated and reflected variants $(c_i) \in \{(\pm a_i), (\pm b_i)\}$. Form a closed N -gon Ω with these edge vectors that encloses the largest possible area.

Once a variant (c_i) is selected for every vector, the problem is easy: the polygon closes if and only if $\sum_{i=1}^N (c_i) = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$, and among all polygons with a given multiset of edge vectors, the convex one obtained by sorting the vectors by direction has the largest signed area. We write $F(v_1, \dots, v_N)$ for the area of this convex polygon, with $F(v_1, \dots, v_N) = -\infty$ if $v_1 + \dots + v_N \neq \begin{pmatrix} 0 \\ 0 \end{pmatrix}$. Since there are 8^N possible choices of variants, the difficulty lies in choosing them.

We solve MAXPOLYGON with dynamic programming. The optimal polygon is convex, so we can build it by adding the vectors one at a time in the order from the “steepest” (closest to a $\pm 45^\circ$ diagonal) to the “flattest” (closest to a coordinate axis). The polygon then grows in four separate convex chains, one around each of the four diagonal directions, see Figure 17. Following the dominant direction of their edges, we call them the NE, NW, SW and SE chains, and we denote the vectors from the first to the last vertex of each chain by $x_{NE}, x_{NW}, x_{SW}, x_{SE}$, and call them the *sum vectors* of the chains.

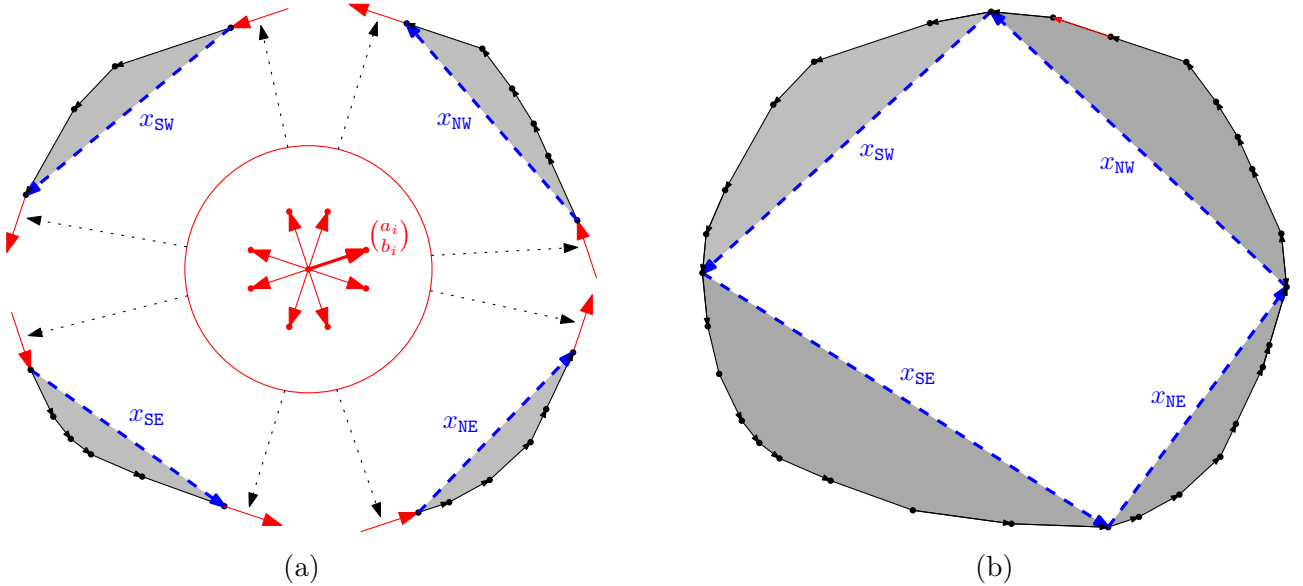


Figure 17: (a) The i -th vector can extend the partial solution in one of eight possible ways. (b) At the end, the four chains are joined into a closed polygon.

The eight rotated and reflected variants of a vector correspond to the eight places where it can be attached: at the beginning or at the end of one of the four chains. If we assume $0 \leq b_i \leq a_i$, then the variant (a_i) is prepended to the NE chain and (b_i) is appended to it, and the other three chains are obtained by rotating by multiples of 90° : the NW chain receives $(-b_i)$ at its beginning and $(-a_i)$ at its end, the SW chain $(-a_i)$ and $(-b_i)$, and the SE chain (b_i) and (a_i) . Since the vectors are processed in order of decreasing steepness, each chain stays convex and sorted by direction.

Each chain, together with the segment between its endpoints, encloses a certain area (shaded gray in Figure 17). The algorithm stores, for every combination of sum vectors, the largest total area

$$\text{OPT}_i[x_{\text{NE}}, x_{\text{NW}}, x_{\text{SW}}, x_{\text{SE}}]$$

of the four chains that can be obtained from the first i vectors such that these four vectors are the resulting sum vectors. When adding the variant $e = \begin{pmatrix} c_i \\ d_i \end{pmatrix}$ to a chain with sum vector x , the area of that chain grows by the triangle spanned by x and e , i.e., by $\frac{1}{2} \det(x, e)$ if e is appended at the end and by $\frac{1}{2} \det(e, x)$ if it is prepended at the beginning, while the sum vector becomes $x + e$. Thus, each entry of OPT_{i-1} produces eight candidate values for different entries of OPT_i , of which we keep only the best value per entry. As in Section 3, the crucial point is that we do not need to know the orientations and the order of the first $i - 1$ vectors to extend the solution; the four sum vectors and the corresponding area are all the information we need.

After all N vectors have been added, we consider the entries of OPT_N with $x_{\text{NE}} + x_{\text{NW}} + x_{\text{SW}} + x_{\text{SE}} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$. These are exactly the ones where the four chains close up to a polygon (Figure 17b). To the stored value we add the area of the quadrilateral formed by the four sum vectors,

$$\frac{1}{2} [\det(x_{\text{NE}}, x_{\text{NW}}) + \det(x_{\text{NE}} + x_{\text{NW}}, x_{\text{SW}})],$$

to obtain the area of the whole polygon, and the largest value obtained in this way is the optimum.

Memory requirement. In our applications all vectors have integer or half-integer coordinates. With $\ell := \sum_{i=1}^N (|a_i| + |b_i|)$, there are at most $O(\ell^8)$ possible quadruples of sum vectors, so the running time and the space requirement are polynomially bounded by $O(N\ell^8)$. If only the optimal value is needed and not the solution itself, the tables of previous steps can be discarded, and $O(\ell^8)$ memory suffices. For hexominoes this is still far too large, and we need to restrict the search space further.

4.1.3 Using a prescribed target area

The size of the tables is the main bottleneck. In practice, most of their entries cannot lead to an optimal solution. We can discard entries if we specify in advance a target A_{target} that we want to reach.

For a table entry $\text{OPT}_i[x_{\text{NE}}, x_{\text{NW}}, x_{\text{SW}}, x_{\text{SE}}]$, the final polygon consists of the four chains built so far and the $N - i$ remaining vectors. Replacing each chain by its sum vector removes exactly the stored area, so the area of any completion equals the stored value plus the area of a polygon whose edges are the four sum vectors and the remaining vectors. Ignoring the directions and only considering the lengths of these $N - i + 4$ edges, we can bound the latter area by the largest area of any polygon with the given side lengths. It is well known (see, for example, [18, Theorem 12.5a]) that this maximum is attained if the polygon is inscribed in a circle. Its radius R can be determined as the solution of a nonlinear equation in one variable, namely $\sum_k 2 \arcsin(\ell_k / (2R)) = 2\pi$ for given side lengths ℓ_k (with an appropriate modification if the center of the circle lies outside the polygon), and once R is determined, the area can be easily computed.

Adding the stored value gives an upper bound for the area that can be reached from this partial solution. If this bound, plus the known sum of the corrections, is smaller than the target area A_{target} , we do not need to store the entry. This reduces the size of the tables and, as a consequence, also the running time. To prove that no fence is better than the best known one with area A^* , we can choose $A_{\text{target}} = A^* + 1$, since fence areas are integers; in our computations we use the slightly smaller threshold $A^* + 0.999$ to guard against rounding errors.

We determine the radius R by a simple binary-search procedure, narrowing down the interval in which R is contained. The sum of the angles (which should equal 2π in the above formula) is a decreasing function of R , whereas the sum of the areas of the triangles spanned by the chords of length ℓ_k is an increasing function of R . Thus, if R is too big for the angle condition, but the area is smaller than the threshold, we can safely discard the entry. On the other hand, if R is too small for the angle condition, but the area is already bigger than the threshold, we already know that the entry will pass the test. We hardly ever need to determine the radius R very precisely. On average, a

pruning test takes less than 8 binary search iterations. The formulas involve arcsine and square-root computations, plus the addition of N numbers. There is no propagation of floating-point rounding errors over several levels, and the safety buffer of 0.001 is generous in comparison to these errors.

To give the reader an impression of how effective this pruning technique is: we can confirm in 150 minutes of computing time and with 130 GB of memory that the reference vectors of the fence in Figure 14 cannot be arranged into a fence of area larger than 1597. (By itself, this does not prove optimality, because other choices of reference vectors might give better results.) For heptominoes and octominoes, this pruning procedure is not sufficient, and we will additionally restrict the search space heuristically in Section 5.

4.2 Proof of optimality with DP after Branch-and-Bound (B&B) preprocessing

The dynamic program of Section 4.1.2 assumes that a fixed reference vector has been chosen for every piece. To prove Theorem 1, we therefore first reduce the enormous number of possible choices of reference vectors to a handful of candidates. We apply the Branch-and-Bound technique, using an upper bound based on Minkowski averages of polygons. The proof consists of four steps.

4.2.1 Step 0: Computing the reference vector repertoire of a hexomino

For each hexomino $i = 1, \dots, 35$, Algorithm 2 determines the *repertoire*

$$\mathcal{V}_i = \{(v_i^1, \Delta_i^1, D_i^1, B_i^1), (v_i^2, \Delta_i^2, D_i^2, B_i^2), \dots, (v_i^{l_i}, \Delta_i^{l_i}, D_i^{l_i}, B_i^{l_i})\}$$

of all possible reference vectors with their corrections, rotations, and parity bits, where all rotated and reflected variants are listed explicitly. A fence selects one quadruple $(v_i, \Delta_i, D_i, B_i) \in \mathcal{V}_i$ for every piece and arranges the pieces in a suitable order. To allow a piece to be unused, which is harmless by Section 2.1, we add the null option $(v, \Delta, D, B) = (0, 0, 0, 0)$ to the repertoire of every piece. Adding this option does not change the outcome of the computations below: Step 1 returns exactly the same candidate sequences with and without it, both for the threshold 1597.999 and for the threshold 1596.999 of Section 4.3. By Lemma 3, the maximality of the convex arrangement, and conditions (5) and (6), every fence satisfies

$$\text{Area} \leq S_0 := \max \left\{ F(v_1, \dots, v_{35}) + \Delta_1 + \dots + \Delta_{35} \mid (v_i, \Delta_i, D_i, B_i) \in \mathcal{V}_i, \right. \\ \left. D_1 + \dots + D_{35} = 360^\circ, B_1 + B_2 + \dots + B_{35} \equiv 0 \pmod{2} \right\}. \quad (7)$$

This is already a rather precise formulation of the fence problem: The only condition that is not checked is that the resulting fence is free of overlaps.

4.2.2 An upper bound via Minkowski averages

We can compute an easy upper bound for the problem MAXPOLYGON by the technique of Minkowski averages. The *Minkowski average* of two polygons Ω_1 and Ω_2 is

$$\Omega' = \frac{1}{2}[\Omega_1 \oplus \Omega_2] := \left\{ \frac{1}{2}(u_1 + u_2) \mid u_1 \in \Omega_1, u_2 \in \Omega_2 \right\},$$

where the term in brackets is the *Minkowski sum*, the element-wise sum of the two sets. For two convex polygons, the Minkowski average is easy to construct, see Figure 18: we merge the counter-clockwise directed edge vectors of Ω_1 and Ω_2 into one list sorted by direction, scale all vectors by $\frac{1}{2}$ and concatenate them to form Ω' .

The famous Brunn–Minkowski inequality¹ implies that, for convex sets Ω_1 and Ω_2 of equal area, the area of Ω' is at least as large as the area of Ω_1 . This also holds for averages of more than two sets, and we apply it to the eight rotated and reflected copies of a hypothetical optimal solution Ω of MAXPOLYGON, see Figure 19. Let $\rho_1, \dots, \rho_8$ be the eight symmetries of the grid that fix the origin, i.e., the rotations by multiples of 90° and the reflections in the axes and the diagonals, and consider

$$\Omega' = \frac{1}{8}[\rho_1\Omega \oplus \rho_2\Omega \oplus \dots \oplus \rho_8\Omega].$$

¹See, for example, https://en.wikipedia.org/wiki/Brunn-Minkowski_theorem.

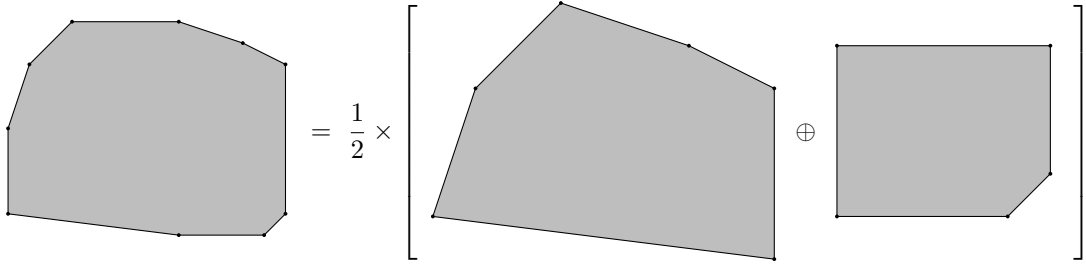


Figure 18: The Minkowski average of two polygons.

The nice property of this expression is that we know exactly what Ω' looks like, even though we do not know Ω : every edge vector $\begin{pmatrix} a_i \\ b_i \end{pmatrix}$ of Ω appears in Ω' scaled by $\frac{1}{8}$ in each of its eight variants. Since all polygons $\rho_k \Omega$ have the same area as Ω , the area of Ω' cannot be smaller than that of Ω . We thus obtain an easily computable upper bound for MAXPOLYGON.

We expect the optimum solution to be rather round and not as spiky as in Figure 19. Thus, the Minkowski average Ω' will be just a slightly better rounded version of Ω and the true area of Ω will be close to the bound. (If every given vector occurs with a multiplicity divisible by 8, then Ω' consists of whole vectors and is itself an optimal solution.)

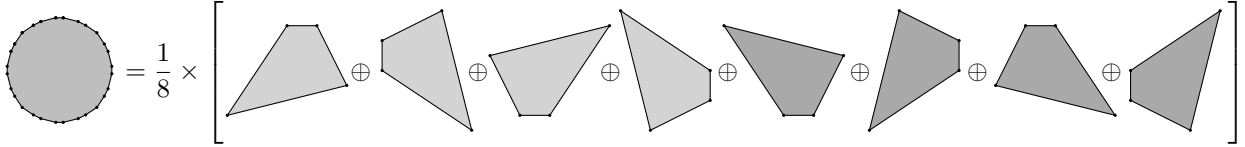


Figure 19: The Minkowski average of the eight rotated and reflected copies of a polygon.

The area of Ω' can be expressed by a simple formula:

$$F_M := \frac{1}{16} \sum_{i=1}^N \sum_{j=1}^N C \left[\begin{pmatrix} a_i \\ b_i \end{pmatrix}, \begin{pmatrix} a_j \\ b_j \end{pmatrix} \right] \quad (8)$$

with

$$C \left[\begin{pmatrix} a_i \\ b_i \end{pmatrix}, \begin{pmatrix} a_j \\ b_j \end{pmatrix} \right] := \max \{ \max(|a_i|, |b_i|) \cdot (|a_j| + |b_j|), (|a_i| + |b_i|) \cdot \max(|a_j|, |b_j|) \}.$$

So F_M is a sum over all pairwise products of vector lengths, where one vector is measured in the maximum norm $\max(|a|, |b|)$ and the other in the 1-norm $|a| + |b|$, choosing the combination that gives the larger product. If all vectors are transformed into the first octant and sorted by slope, i.e., $0 \leq b_i \leq a_i$ and $\frac{b_1}{a_1} \leq \frac{b_2}{a_2} \leq \dots \leq \frac{b_N}{a_N}$, the formula simplifies to

$$F_M = \frac{1}{8} \left(\sum_{1 \leq i < j \leq N} a_i(a_j + b_j) + \frac{1}{2} \sum_{i=1}^N a_i(a_i + b_i) \right), \quad (9)$$

since the flatter vector $\begin{pmatrix} a_i \\ b_i \end{pmatrix}$ is always measured in the maximum norm a_i and the steeper vector $\begin{pmatrix} a_j \\ b_j \end{pmatrix}$ in the 1-norm $a_j + b_j$.

Theorem 2. *The optimum area for the problem MAXPOLYGON is less than or equal to F_M from formula (9).*

Proof. We have already argued why the area of the convex polygon Ω' is an upper bound. It remains to prove that this area equals (9).

The polygon Ω' is centrally symmetric. A centrally symmetric convex polygon with edge vectors $\pm w_1, \pm w_2, \dots, \pm w_h$ is a translate of a *zonogon*, a Minkowski sum $\bigoplus_k [0, w_k]$ of line segments. The area of such a zonogon is $\sum_{1 \leq k < l \leq h} |\det(w_k, w_l)|$. (This follows easily from the fact that the zonogon

can be tiled by the parallelograms spanned by the pairs w_k, w_l^2 , see [21, (56–57)].) Every vector $\begin{pmatrix} a_i \\ b_i \end{pmatrix}$ contributes to Ω' the four pairs of opposite vectors $\pm w_i^{(1)}, \dots, \pm w_i^{(4)}$ with $w_i^{(1)} = \frac{1}{8} \begin{pmatrix} a_i \\ b_i \end{pmatrix}$, $w_i^{(2)} = \frac{1}{8} \begin{pmatrix} b_i \\ a_i \end{pmatrix}$, $w_i^{(3)} = \frac{1}{8} \begin{pmatrix} -b_i \\ a_i \end{pmatrix}$ and $w_i^{(4)} = \frac{1}{8} \begin{pmatrix} -a_i \\ b_i \end{pmatrix}$ in the upper half-plane. Consider a pair $i \leq j$. Since the whole configuration is symmetric under the eight grid symmetries, and $|\det|$ is invariant under them, it suffices to compute the contribution of $w_i^{(1)}$ and multiply it by 4. As $\frac{b_i}{a_i} \leq \frac{b_j}{a_j}$, all four determinants $\det(w_i^{(1)}, w_j^{(k)})$ are nonnegative (see Figure 20), so we can combine them:

$$\sum_{k=1}^4 \det(w_i^{(1)}, w_j^{(k)}) = \det\left(w_i^{(1)}, \sum_{k=1}^4 w_j^{(k)}\right) = \begin{vmatrix} a_i/8 & 0 \\ b_i/8 & (a_j + b_j)/4 \end{vmatrix} = \frac{a_i(a_j + b_j)}{32}.$$

Multiplying by 4 gives the term $\frac{1}{8}a_i(a_j + b_j)$ in (9). For $i = j$ this counts every determinant $|\det(w_i^{(k)}, w_i^{(k')})|$ with $k \neq k'$ twice (and for $k = k'$ the determinant is 0), so in this case the result is weighted by the factor $\frac{1}{2}$. $\square$

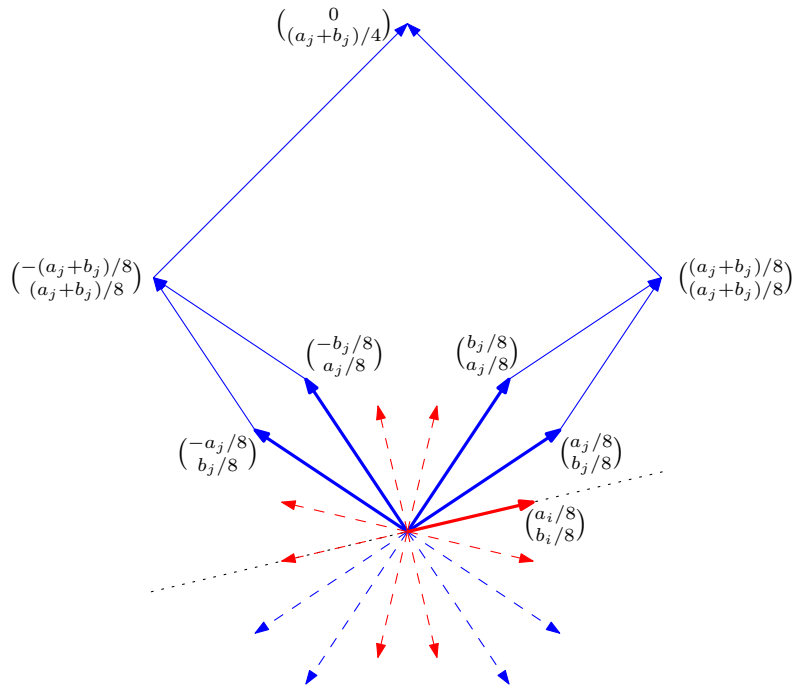


Figure 20: Combining $\frac{1}{8} \begin{pmatrix} a_i \\ b_i \end{pmatrix}$ with all rotated and reflected copies of $\frac{1}{8} \begin{pmatrix} a_j \\ b_j \end{pmatrix}$.

This bound is often remarkably sharp. For example, the reference polygon of the optimal hexomino fence in Figure 14 has area 1663, while the bound gives 1666.5625.

4.2.3 Step 1: Branch-and-Bound over vector sequences

For now, we ignore the rotation condition $D_1 + \dots + D_{35} = 360^\circ$ and the parity condition $B_1 + \dots + B_{35} \equiv 0 \pmod{2}$, and we also no longer distinguish between rotated and reflected reference vectors. This leads to the reduced repertoire

$$\mathcal{W}_i = \{(\bar{v}_i^1, \Delta_i^1), (\bar{v}_i^2, \Delta_i^2), \dots, (\bar{v}_i^{m_i}, \Delta_i^{m_i})\},$$

where $\bar{v}_i^j$ stands for an equivalence class of up to eight rotated and reflected vectors, and Δ_i^j is the largest correction of any copy of this vector in $\mathcal{V}_i$. For the hexominoes, the sets $\mathcal{W}_i$ contain between 12 and 22 elements. Our goal is to find a small list of vector sequences $(\bar{v}_1, \dots, \bar{v}_{35})$ with $\bar{v}_i \in \mathcal{W}_i$

²<https://en.wikipedia.org/wiki/Zonogon>

that contains all optimal ones. Writing $\bar{F}(\bar{v}_1, \dots, \bar{v}_{35})$ for the optimal value of MAXPOLYGON with the given vector classes, we have

$$S_0 \leq S_1 := \max \{ \bar{F}(\bar{v}_1, \dots, \bar{v}_{35}) + \Delta_1 + \dots + \Delta_{35} \mid (\bar{v}_i, \Delta_i) \in \mathcal{W}_i \}, \quad (10)$$

and by Section 4.2.2,

$$\bar{F}(\bar{v}_1, \dots, \bar{v}_{35}) \leq F_M(\bar{v}_1, \dots, \bar{v}_{35}) = \frac{1}{8} \sum_{1 \leq i < j \leq 35} C[\bar{v}_i, \bar{v}_j] + \frac{1}{16} \sum_{i=1}^{35} C[\bar{v}_i, \bar{v}_i]. \quad (11)$$

We distribute each correction Δ_i evenly, with a share of $\frac{1}{35}$, over the 35 terms of (11) in which $\bar{v}_i$ occurs. Maximizing every term separately over the available choices then gives the bound

$$\begin{aligned} \mathcal{F}(\mathcal{W}_1, \dots, \mathcal{W}_{35}) := & \sum_{1 \leq i < j \leq 35} \max \{ \frac{1}{8} C[\bar{v}_i, \bar{v}_j] + \frac{\Delta_i + \Delta_j}{35} \mid (\bar{v}_i, \Delta_i) \in \mathcal{W}_i, (\bar{v}_j, \Delta_j) \in \mathcal{W}_j \} \\ & + \sum_{i=1}^{35} \max \{ \frac{1}{16} C[\bar{v}_i, \bar{v}_i] + \frac{\Delta_i}{35} \mid (\bar{v}_i, \Delta_i) \in \mathcal{W}_i \} \geq S_1, \end{aligned} \quad (12)$$

which can be computed easily by combining all pairs of candidates in each term. If every set $\mathcal{W}_i$ contains a single element, it coincides with $F_M + \sum_i \Delta_i$. For larger sets the bound is weaker, since in each of the 35 terms containing $\bar{v}_i$ the vector can be chosen independently, whereas a solution uses the same $\bar{v}_i$ in all terms. For the original sets $\mathcal{W}_i$, the bound (12) has the value 1674.094.

Conceptually, we now enumerate all vector sequences in 35 nested loops, selecting $(\bar{v}_1, \Delta_1) \in \mathcal{W}_1$, then $(\bar{v}_2, \Delta_2) \in \mathcal{W}_2$ and so on, and output every sequence with $F_M(\bar{v}_1, \dots, \bar{v}_{35}) + \Delta_1 + \dots + \Delta_{35} \geq 1597.999$. Done naively, this would take astronomically long. Instead, on level i , when $(\bar{v}_1, \Delta_1), \dots, (\bar{v}_i, \Delta_i)$ are already fixed, we evaluate the bound

$$\mathcal{F}(\{(\bar{v}_1, \Delta_1)\}, \dots, \{(\bar{v}_i, \Delta_i)\}, \mathcal{W}_{i+1}, \dots, \mathcal{W}_{35}),$$

and if it is already smaller than 1597.999, the threshold cannot be exceeded anywhere in this branch of the enumeration tree and we prune it. As explained above, the bound becomes more accurate the more the choices are restricted, which makes the pruning very effective, see Algorithm 3.

Algorithm 3 BRANCHANDBOUND($i, \mathcal{W}_1, \dots, \mathcal{W}_{35}$)

```

1: Input: level  $i$ , sets  $\mathcal{W}_1, \dots, \mathcal{W}_{i-1}$  already reduced to single elements
2: if  $\mathcal{F}(\mathcal{W}_1, \dots, \mathcal{W}_{35}) < 1597.999$  then
3:   return ▷ prune this branch
4: end if
5: if  $i > 35$  then
6:   output the candidate sequence  $(\bar{v}_1, \dots, \bar{v}_{35})$  ▷ all sets are singletons
7:   return
8: end if
9: for each  $(\bar{v}_i, \Delta_i) \in \mathcal{W}_i$  do
10:  BRANCHANDBOUND( $i + 1, \mathcal{W}_1, \dots, \mathcal{W}_{i-1}, \{(\bar{v}_i, \Delta_i)\}, \mathcal{W}_{i+1}, \dots, \mathcal{W}_{35}$ )
11: end for

```

Starting with BRANCHANDBOUND(1, $\mathcal{W}_1, \dots, \mathcal{W}_{35}$), this procedure visits about 30 million nodes of the enumeration tree and finds 7128 candidate sequences. In our implementation, the runtime was roughly two hours.³ The candidate sequences are passed on to the next step.

³The running time could be reduced further by eliminating vectors beforehand: If fixing a single $(\bar{v}_i, \Delta_i)$ while keeping all other sets $\mathcal{W}_j$ variable already gives a bound below 1597.999, this vector can be removed once and for all. For simplicity, we did not do this. We remark that, with the threshold 1594.999 instead of 1597.999, the search produces 136 543 candidates in 5 hours; among these there are also solutions containing pieces with rotation -90° . Fortunately, no such solutions occur with the original bound 1597.999, because the subsequent steps would not work for them.

4.2.4 Step 2: Taking into account the rotation and the parity

Next, we subject each of the 7128 vector sequences from Step 1 to a stricter test that also takes the rotation condition (5) and the parity condition (6) into account. Consider a fixed vector sequence $((\bar{v}_1, \Delta_1), \dots, (\bar{v}_{35}, \Delta_{35}))$. For each i , let $\mathcal{U}_i$ be the set of triples $(\Delta_i^j, D_i^j, B_i^j)$ of those quadruples $(v_i^j, \Delta_i^j, D_i^j, B_i^j) \in \mathcal{V}_i$ whose vector v_i^j coincides with the prescribed class $\bar{v}_i$ after a suitable rotation or reflection. Then

$$\Delta^{\text{rot}} := \max \{ \Delta_1 + \dots + \Delta_{35} \mid (\Delta_i, D_i, B_i) \in \mathcal{U}_i, \\ D_1 + \dots + D_{35} = 360^\circ, B_1 + B_2 + \dots + B_{35} \equiv 0 \pmod{2} \}$$

can be computed very quickly by dynamic programming over the partial sums of the rotations and the parities. If $F_M(\bar{v}_1, \dots, \bar{v}_{35}) + \Delta^{\text{rot}} < 1597.999$, the sequence can be excluded.

From now on, only the *multiset* $\{\bar{v}_1, \dots, \bar{v}_{35}\}$ of vectors matters, and eight multisets remain. Two of them can be excluded by the following trick: We let the reference vectors start and end at the *inner* endpoints of the attachment edges instead of their midpoints. This changes the reference vectors of the corner pieces, and all the corrections have to be adjusted accordingly. For the exact area, this makes no difference because these two changes cancel, but the bounds become tighter.

Each of the remaining six multisets consists of four corner pieces with rotation $+90^\circ$ and otherwise only pieces with rotation 0° . The program also checks whether there are alternatives with rotations other than 0° and $+90^\circ$, but for these the sum of the corrections was always too small. This is important for the next step: it tells us that the four corner pieces point in four different directions, so the tables of the dynamic program can be initialized accordingly. Step 2 takes only a few seconds.

4.2.5 Step 3: Dynamic programming with fixed vectors

Finally, we run the dynamic program from Section 4.1.2 with the pruning from Section 4.1.3 on each of the six multisets, with each chain initialized with one of the corner pieces. As in Step 2, the reference vectors are attached to the inner endpoints of the attachment edges. This makes the reference vectors of the corner pieces integral, while the reference vectors of all other pieces remain unchanged, since their two attachment edges are parallel. The main effect is that the work of the dynamic programming algorithm is reduced.

There is one exception, in which this would make a corner piece flatter than a “straight” piece with $D = 0^\circ$: the reference vector $\begin{pmatrix} 1.5 \\ 2.5 \end{pmatrix}$ would become $\begin{pmatrix} 1 \\ 2 \end{pmatrix}$, which has a smaller slope than $\begin{pmatrix} 2 \\ 3 \end{pmatrix}$. This would interfere with the order of processing the vectors, so in this case we went in the opposite direction: we attached the reference vectors to the *outer* endpoints of the attachment edges instead, which gives the corner piece the reference vector $\begin{pmatrix} 2 \\ 3 \end{pmatrix}$.

This computation took about 16 hours for all six multisets, with a peak memory usage of about 130 GB on a machine with 300 GB of main memory. No solution with an area of at least 1597.999 was found, which completes the proof of Theorem 1. Note that, unlike in Section 3, no check for overlaps is needed here: every step only relaxes the problem, so the computations yield a valid upper bound, and the matching lower bound is given by the explicit fence in Figure 14. Section 8 confirms the upper bound 1597 independently with integer programming, under an additional assumption on how the hexominoes are used.

4.3 Step 4: Identification of all optimal solutions

We ran Steps 1–3 with the weaker bound 1596.999, and we added another step that finds all solutions with at least this area, i.e., all optimal solutions, by backtracking through the dynamic-programming table. Step 1 now produces 21 283 candidate sequences (instead of 7128), Step 2 reduces them to 11 multisets of reference vector classes (instead of 6), and Step 3 reduces this number to a single multiset

$$1 \times \begin{pmatrix} 6 \\ 0 \end{pmatrix}, 2 \times \begin{pmatrix} 5 \\ 0 \end{pmatrix}, 3 \times \begin{pmatrix} 4 \\ 0 \end{pmatrix}, 3 \times \begin{pmatrix} 5 \\ 1 \end{pmatrix}, 11 \times \begin{pmatrix} 4 \\ 1 \end{pmatrix}, 4 \times \begin{pmatrix} 3 \\ 1 \end{pmatrix}, 6 \times \begin{pmatrix} 4 \\ 2 \end{pmatrix}, 1 \times \begin{pmatrix} 3 \\ 2 \end{pmatrix}, 4 \times \begin{pmatrix} 2.5 \\ 2.5 \end{pmatrix},$$

running for about two days. For this multiset, Step 4 produces 244 different optimal reference polygons up to symmetry. As in the case of pentominoes (Section 3.4), hexominoes that realize the same reference

vector are interchangeable. For the last two classes $1 \times \binom{3}{2}$ and $4 \times \binom{2.5}{2.5}$ (the four corner pieces), the same set of five hexominoes can be used, which gives a factor of $5!$. Apart from these five pieces, no piece can be used for two different reference vectors. Thus, we have

$$2! \cdot 3! \cdot 3! \cdot 11! \cdot 4! \cdot 6! \cdot 5! \times 244 = 1\,454\,129\,298\,800\,640\,000$$

different optimal hexomino orders. Finally, we determine all optimal solutions, which, as defined in Section 3.4, are distinct if they differ in their cells. Similarly to the Y- and Z-pentominoes, now 10 of the hexominoes can be rotated or reflected without changing their reference vector and correction. Thus, there are

$$2^{10} \times 2! \cdot 3! \cdot 3! \cdot 11! \cdot 4! \cdot 6! \cdot 5! \times 244 = 1\,489\,028\,401\,971\,855\,360\,000 \approx 1.49 \cdot 10^{21}$$

optimal solutions to the hexomino fence problem.

The above calculation rests on the assumption that different reference polygons give rise to different fences. It is conceivable that two pieces share more than one edge, as in Figure 7, and then the fence would not have a unique reference polygon. The following section excludes this possibility.

4.4 Overlapping pieces and multiple shared edges

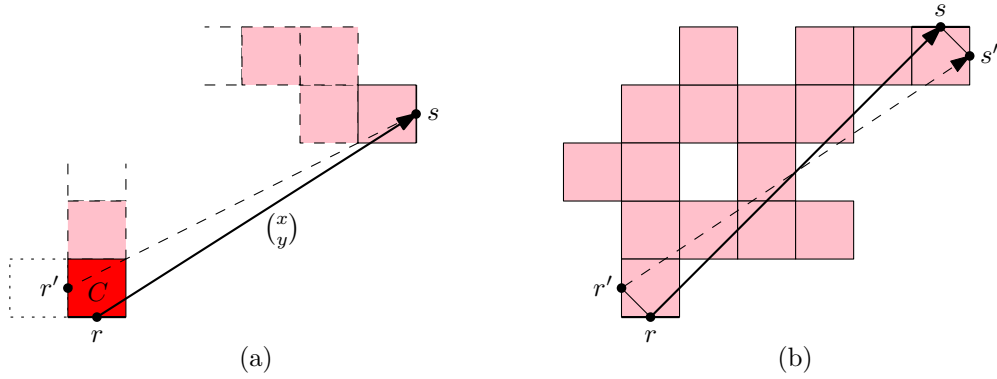


Figure 21: (a) The largest distance between two edge midpoints r and s . (b) When a reference vector of slope ± 1 is available, there is always a longer vector.

We need an elementary lemma about longest reference vectors.

Lemma 4. *Let P be a finite, nonempty set of unit cells of the square grid. Then a longest vector between two edge midpoints of the boundary of P lies in $\mathbb{Z}^2$, and the two edges on which these midpoints lie are parallel.*

Proof. A boundary edge is vertical or horizontal; we call its midpoint a V -point in $\mathbb{Z} \times (\mathbb{Z} + \frac{1}{2})$ or an H -point in $(\mathbb{Z} + \frac{1}{2}) \times \mathbb{Z}$, respectively. The difference of two points of the same type lies in $\mathbb{Z}^2$, whereas the difference of a V -point and an H -point lies in $(\mathbb{Z} + \frac{1}{2})^2$. Suppose that r, s are of different types, and consider the vector $s - r = \begin{pmatrix} x \\ y \end{pmatrix}$ with $x, y \in \mathbb{Z} + \frac{1}{2}$. We will show that r, s cannot have the largest distance.

Without loss of generality, we assume that $0 < y \leq x$, r is an H -point, and s is a V -point, as in Figure 21a. (The other cases are symmetric.) Consider the red cell c_0 on whose boundary r lies. Assume first that the cell to the left of c_0 (the dotted cell in Figure 21a) does not belong to P . Then we take the midpoint r' of the left boundary edge of c_0 , and

$$\|s - r'\|^2 = (x + \frac{1}{2})^2 + (y - \frac{1}{2})^2 = x^2 + y^2 + (x - y) + \frac{1}{2} \geq x^2 + y^2 + \frac{1}{2} > x^2 + y^2 = \|s - r\|^2,$$

that is, r, s cannot have the maximum distance. If the dotted cell to the left of c_0 belongs to P , we walk left until we hit a vertical boundary edge of P . The midpoint of that edge will have an even larger distance from s than r' . Hence the two endpoints of a longest vector are of the same type: they lie on parallel edges, and their difference lies in $\mathbb{Z}^2$. $\square$

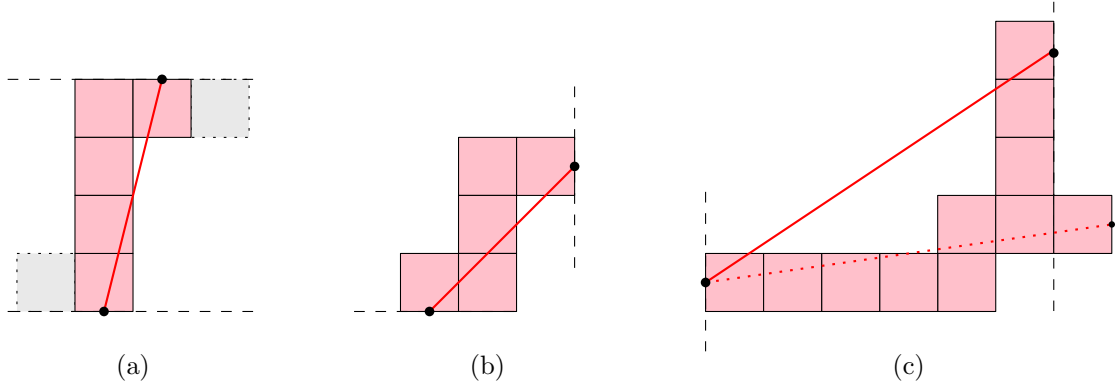


Figure 22: (a) A separated piece with two parallel attachment edges. If the red reference vector is the longest possible reference vector, the two shaded cells cannot be part of the piece. (b) A separated corner piece. (c) The longest reference vector, of length $\sqrt{6^2 + 4^2} = \sqrt{52}$, does not make the piece separated: The alternative choice of reference vector, which would lead to a separated piece, has length $\sqrt{7^2 + 1^2} = \sqrt{50}$.

We call a piece P with reference vector v *separated* if P does not cross the lines through the edges at the endpoints of v , see Figure 22.

Lemma 5. *Suppose that:*

1. *The reference polygon is convex.*
2. *It contains exactly four half-integral “corner vectors” in $(\mathbb{Z} + \frac{1}{2})^2$, one pointing into each of the four quadrants.*
3. *It contains at least one axis-parallel vector in each of the four directions.*
4. *Every piece is separated.*
5. *For each corner piece, the reference vector is the longest possible vector between midpoints of a horizontal and a vertical boundary edge.*
6. *For each of the remaining pieces, the reference vector is the longest possible vector between two boundary-edge midpoints.*

Then consecutive pieces do not overlap, and their common attachment edge is their only common edge.

Proof. By assumption 2, the four corner vectors divide the cyclic sequence of reference vectors into four runs of consecutive non-corner vectors. Since the polygon is convex, the vectors are ordered by direction. Hence the run between the corner vectors pointing into the first and into the second quadrant consists of the vectors whose directions lie strictly between the directions of these two corner vectors. We call it the *right run*. Its vectors $\begin{pmatrix} x \\ y \end{pmatrix}$ have directions in the interval $(0^\circ, 180^\circ)$, and so $y > 0$. We concentrate on the right run; the other three runs are treated similarly.

By assumption 3, the right run contains a vector $\begin{pmatrix} 0 \\ y \end{pmatrix}$; in particular, no run is empty, and no two corner pieces are consecutive.

By Lemma 4, every piece in the run has parallel attachment edges, and hence all attachment edges in the right run are parallel. In fact, since the run contains a vertical vector, they are horizontal. The horizontal lines through the attachment edges cut the plane into strips, and since the pieces are separated, they are confined to their strips and cannot overlap.

We now show that two pieces cannot share more than one edge. Consider the initial part of the right run, up to the first vertical vector. These vectors are of the form $\begin{pmatrix} x \\ y \end{pmatrix}$ with $x \geq 0$ (and $y > 0$), see Figure 22a. If P is such a piece, no cell of P can touch the bottom supporting line to the left of the lower endpoint of the vector, and no cell can touch the top supporting line to the right of the upper endpoint, because otherwise there would be a longer reference vector (Figure 22a). The next piece P'

above P has the same properties. Thus, on the line separating them, P touches only at or to the left of the chosen attachment edge, and P' only at or to the right of it, and so that edge is the unique common edge of P and P' . The only case where this argument fails is when a right-leaning vector $\begin{pmatrix} x \\ y \end{pmatrix}$ with $x > 0$ is followed by a left-leaning vector $\begin{pmatrix} x' \\ y' \end{pmatrix}$ with $x' < 0$, but since there is at least one vertical vector with $x = 0$, this case does not arise.

The argument extends to the two corner pieces at the lower and upper end of the right run: The maximality condition 5 applies analogously, and we conclude that the whole run including the corner pieces is overlap-free, and consecutive pieces intersect only in one edge. $\square$

For hexominoes, the assumptions are unfortunately not fulfilled for one exceptional piece: The cyan T-shaped piece in the upper left side of Figure 14 is not treated as required: Its longest reference vector is $\begin{pmatrix} 4 \\ 0 \end{pmatrix}$, as opposed to the reference vector $\begin{pmatrix} 2 \\ 3 \end{pmatrix}$ of length $\sqrt{13}$ with which it is used. It can also be used with reference vector $\begin{pmatrix} 5/2 \\ 5/2 \end{pmatrix}$ as a corner piece; in fact, it is one of the five hexominoes mentioned above that can be corner pieces.

Nevertheless, it can be checked that in the position in which it is used in Figure 14, it has the properties needed in the proof: It is separated, and besides the attachment edge, it touches each separating line only on the side permitted in the proof. However, the T-piece can also be used in a reflected position where the notch points upwards and not to the left. In that position it is not separated. Here we have to make an ad-hoc argument: The piece P' above cannot intersect the notch, because otherwise it would have to touch the separating line on the wrong side of the attachment edge. It cannot even touch the notch: It cannot touch the notch from the right (or from the left), because then it would touch the separating line. P' must touch the notch from the top. On the other hand, its reference vector $\begin{pmatrix} x' \\ y' \end{pmatrix}$ must satisfy $\frac{x'}{y'} \geq \frac{2}{3}$, and then one can argue that the top edge of the notch would give rise to a longer reference vector.

When the T-piece is used as a corner piece, the assumptions of the lemma are fulfilled. All other pieces are used with their longest possible reference vectors, and they are separated. All the remaining assumptions of the lemma are fulfilled as well, for all 244 reference polygons.

We conclude that the reference polygon is unique for all solutions. We also conclude that none of the pieces in the right run can overlap, or in any other run. What we have proved does not exclude the possibility that, say, pieces from the right run and the bottom run intersect. However, these runs obviously move away from each other, so we refrain from proving it.

Larger polyominoes. Figure 22c is an example where the longest possible reference vector does not lead to separation. It can be checked that this is a smallest example in terms of the size of the polyomino: one needs at least 11 cells for this phenomenon to occur with a strict maximum. One can construct an example with a tie in length between $\begin{pmatrix} 4 \\ 3 \end{pmatrix}$ and $\begin{pmatrix} 5 \\ 0 \end{pmatrix}$ with eight cells, but not with seven. For heptomino fences, Lemma 5 can be applied if longest reference vectors are chosen.

5 Heptomino and octomino fences

5.1 Lower bound using pruned dynamic programming

The target-area test of Section 4.1.3 is relatively expensive: It requires iteratively solving a nonlinear equation to find the radius R of the circumcircle. Therefore, we apply this test once per generation, and not for each insertion. During the i -th generation, an entry in OPT_i may receive several insertions, and its value may be improved several times. After the whole table OPT_i has been filled, we apply the test to every entry and keep only the entries that pass.

However, filling the complete table without any pruning becomes infeasible for heptominoes and octominoes. Therefore, we apply some extra heuristic tests. We expect the four sum vectors $x_{\text{NE}}, x_{\text{NW}}, x_{\text{SW}}, x_{\text{SE}}$ to be balanced: They should be directed roughly in their respective $\pm 45^\circ$ direction, and they should approximately have the same length. Therefore, at each table insertion, we test the following two conditions.

1. *Balance.* After folding all four sum vectors into the first quadrant, we discard an entry if any two vectors have L^1 -distance more than a fixed threshold W , betting that an entry with highly unbalanced vectors is unlikely to be completed to a good polygon.
2. *Reach.* In addition, we look at the sum $\begin{pmatrix} s_x \\ s_y \end{pmatrix} = x_{NE} + x_{NW} + x_{SW} + x_{SE}$, the vector still missing to close the polygon. We discard an entry if $\max(|s_x|, |s_y|)$ exceeds a second threshold W' .

By trying various values for W and W' for a given n -omino fence, the dynamic programming search is able to find very good heuristic solutions in a reasonable amount of time. Figure 23 shows a heptomino fence with area 19 733 which we found in one hour and an octomino fence with area 279 134 that took seven hours of runtime.

Note that there is one octomino (a 3×3 square with a missing center) whose center hole is not 8-connected to the outside. By our definition in Section 2.1, this enclosed cell forms a separate region, which does not count towards the area.

5.2 Upper bounds with Branch-and-Bound

For an upper bound we can simply reuse the Branch-and-Bound method of Section 4.2.3. If Step 1 is run with a bound of c instead of 1597.999 and returns no feasible candidate sequence, we know that no fence can achieve an area $\geq c$.

This method is already quite powerful for the hexominoes: $c = 1601.999$ is the smallest value for which all possibilities can be ruled out by Step 1. Thus, an upper bound of 1601 for the hexomino fence, which exceeds the true optimum by only 4, can be proved in around ten minutes (measured on an AMD Ryzen 7 7735HS CPU and 20 GB of DDR5 memory).

On the same hardware, we can prove an upper bound of 19 830 for heptominoes and 280 389 for octominoes in around 2 days of runtime each. We compare them to the lower bounds from Section 5.1 in Table 3, and can observe that the relative difference between the bounds is less than 0.5%. Note that the upper bounds are just the best possible ones within that time limit, and are likely not the best bounds achievable by this method. We stopped since the runtime of the B&B appears to grow significantly even for small decreases in c ; for example, proving the heptomino bound 19 850 takes just one hour.

n	best DP solution	B&B upper bound	relative difference
7	19 733	19 830	0.49%
8	279 134	280 389	0.45%

Table 3: The best lower and upper bounds that we have found for the heptomino and octomino fences. The relative difference is taken with respect to the lower bound.

6 Larger fences

We now investigate n -omino fences for $n \geq 9$, where our dynamic programming approach is infeasible. We denote the number of free n -ominoes by $A'(n)$; its exact values are known up to $n = 59$ [19]. For larger n , the following estimate exists:

$$A'(n) \approx \frac{1}{8} \frac{c \cdot \lambda^n}{n} \quad (13)$$

where $\lambda = 4.0626$ and $c = 0.3169$ [9]. The original formulation in [9] estimates the number of fixed n -ominoes (i.e., rotated and reflected copies are counted as distinct), but since almost no large n -ominoes have any symmetries, we can just divide that estimate by 8. Note that this estimate has a significant uncertainty: it is only proven that $4.0025 \leq \lambda \leq 4.5238$ [2, 4].

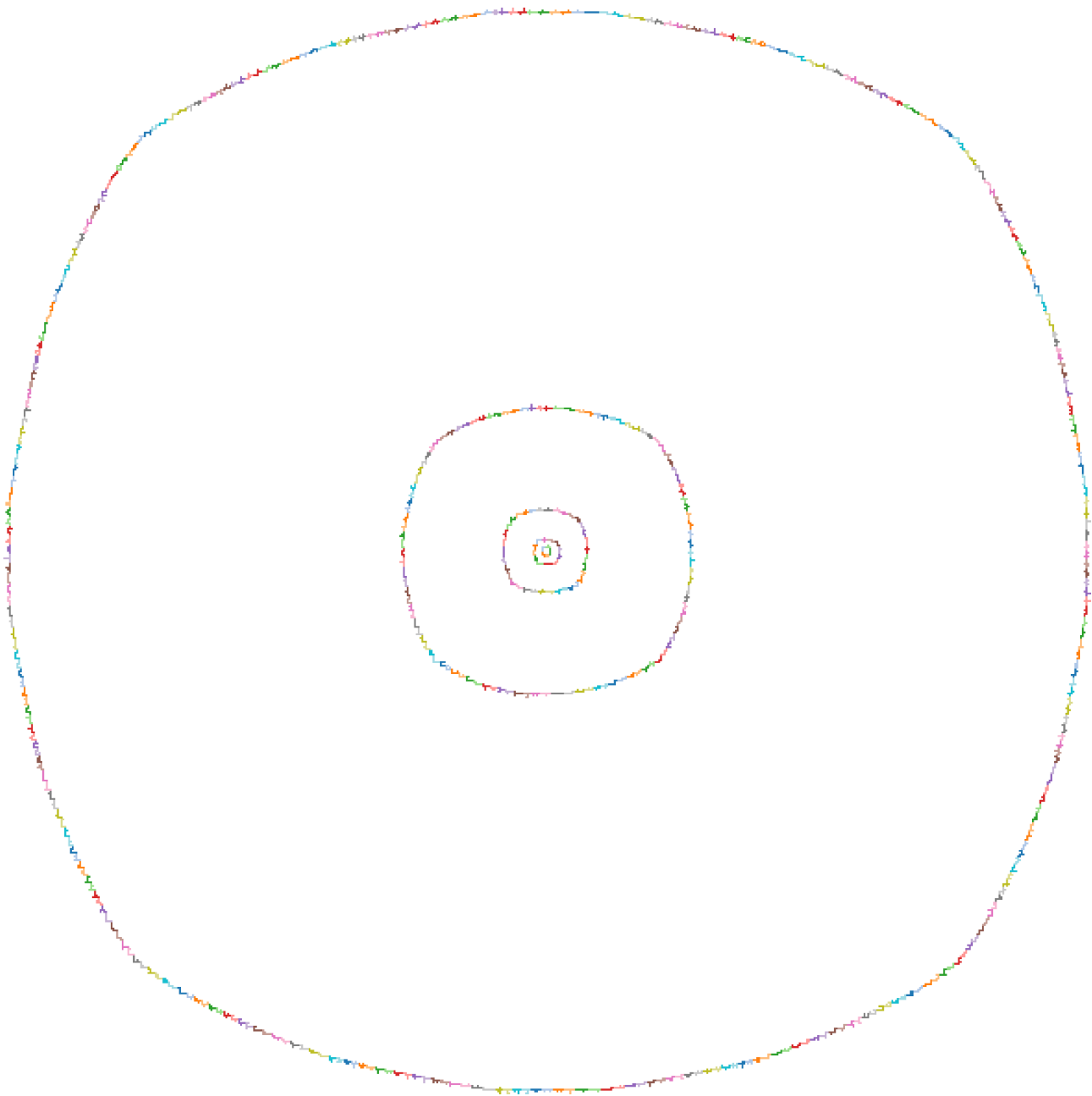


Figure 23: Our best solutions for the heptomino and octomino fences, alongside an optimal tetromino, pentomino, and hexomino fence.

6.1 $F_{\max} + \Delta_{\max}$ upper bound

We relax the bound S_1 from (10) by optimizing over the choice of $\bar{v}_i$ and Δ_i separately:

$$\begin{aligned} \text{Area} &\leq S_1 = \max \left\{ \bar{F}(\bar{v}_1^{a_1}, \bar{v}_2^{a_2}, \dots, \bar{v}_{A'(n)}^{a_{A'(n)}}) + \Delta_1^{a_1} + \Delta_2^{a_2} + \dots + \Delta_{A'(n)}^{a_{A'(n)}} \mid 1 \leq a_i \leq m_i \right\} \\ &\leq \max \left\{ \bar{F}(\bar{v}_1^{a_1}, \bar{v}_2^{a_2}, \dots, \bar{v}_{A'(n)}^{a_{A'(n)}}) \mid 1 \leq a_i \leq m_i \right\} + \max \left\{ \Delta_1^{a_1} + \dots + \Delta_{A'(n)}^{a_{A'(n)}} \mid 1 \leq a_i \leq m_i \right\} \end{aligned} \quad (14)$$

Recall that $\bar{F}(\bar{v}_1, \bar{v}_2, \dots, \bar{v}_{A'(n)})$ is the maximum area of a convex $A'(n)$ -gon obtained by rotating and reflecting the vectors $\bar{v}_1, \bar{v}_2, \dots, \bar{v}_{A'(n)}$ and concatenating them in a suitable order. For this we use as an easy upper bound the area of the largest $A'(n)$ -gon with a given perimeter. It is well known that the k -gon of maximum area with given perimeter u is the regular k -gon [18, Theorem 12.4a], and its area is

$$G_k(u) = \frac{u^2}{4k} \cot \frac{\pi}{k}.$$

Thus,

$$\bar{F}(\bar{v}_1, \bar{v}_2, \dots, \bar{v}_{A'(n)}) \leq G_{A'(n)}(|\bar{v}_1| + |\bar{v}_2| + \dots + |\bar{v}_{A'(n)}|),$$

and we can bound the first term in (14) by

$$F_{\max} := G_{A'(n)}(|\bar{v}_1^{\max}| + |\bar{v}_2^{\max}| + \dots + |\bar{v}_{A'(n)}^{\max}|)$$

where $\bar{v}_i^{\max}$ denotes a longest vector among $\bar{v}_i^1, \dots, \bar{v}_i^{m_i}$. We denote the maximum in the second expression in (14) by $\Delta_{\max}$. (This is the maximum without the rotation and parity conditions, in contrast to Δ^{rot} of Step 2 in Section 4.2.4.) Since the area is an integer, we have proven the upper bound

$$\text{Area} \leq \lfloor F_{\max} + \Delta_{\max} \rfloor.$$

This is easy to compute. For each n -omino we only need to determine the longest reference vector and the maximal correction separately. The bounds up to $n = 15$ are given in Table 4. For $n = 7$ and 8, this bound is considerably weaker than the more expensive bound of Table 3.

6.2 Heuristic greedy solver

For the lower bound, we heuristically attempt to construct an $A'(n)$ -gon that is as regular as possible. As a start, we simply choose the longest reference vector for each n -omino. When there is a tie, we use the correction as a secondary criterion. We then try to solve the MAXPOLYGON problem of Section 4.1.2 for these vectors. A solution can only exist if the parity condition (6) is satisfied. Therefore, if necessary, we flip the parity by changing the reference vector of the longest L-piece so that it is used like a stick of $n - 1$ cells.

The four vectors closest to the 45° direction are selected beforehand as the polygon's four corners. More precisely, these are n -ominoes in which the vector between two diagonally opposite corners is $\begin{pmatrix} \lceil n/2 \rceil \\ \lceil n/2 \rceil \end{pmatrix}$. For $n \geq 6$, there are always sufficiently many pieces of this kind. (This changes four reference vectors, but leaves the parity condition intact.) In contrast to the dynamic programming approach of Section 4.1.2, we process the remaining vectors in a greedy way, in decreasing order of length: Each vector is assigned to whichever of the eight arcs keeps the partial vector sums of the arcs as balanced as possible, breaking ties in favor of keeping the running total closest to zero. Most of the time, the greedy balancing will not result in an exact total of $(0, 0)$, i.e., a closed loop. We therefore make a small number of corrective swaps at the end: a residual of at least 2 in one coordinate is removed by exchanging two vectors that have the same other coordinate and whose coordinates in the residual direction add up to ± 1 : each of them is replaced by the mirror image of the other. This does not change the lengths of the vectors used, but reduces the residual by 2 in that coordinate. A final residual of $(\pm 1, \pm 1)$ is eliminated by reflecting one suitable vector at the diagonal (that is, by swapping the absolute values of its coordinates). All chosen vectors, corners included, are then sorted once by slope into the cyclic order of the final $A'(n)$ -gon. The area is then computed via the Shoelace formula.

We remark that a parity discrepancy arises also for $n = 5$ and $n = 6$. For pentominoes, it is the U-pentomino that is treated exceptionally: it is not used with its longest reference vector in the optimal fences (Figure 12). For hexominoes, it is the T-shaped piece discussed in Section 4.4.

A Python implementation can be found in `best-fence-greedy.py` [15]. Finding the greedy solutions itself is very quick, but generating all n -ominoes and determining their longest reference vectors can take a while for larger n . After a solution is constructed, our implementation optionally validates that all n -ominoes are placed exactly once and without overlap. Additionally, we verify, either with the Shoelace formula applied to the edge boundary of Section 3.1 or with an interval sweep, that every region other than the largest one is a hole inside a single n -omino.

As discussed after Lemma 5, this lemma implies that there is no overlap between consecutive pieces in the case of heptomino fences. For larger polyominoes, one can easily argue that consecutive pieces with parallel reference vectors cannot overlap. The argument can be extended to reference vectors with small angles, such as the angles that arise in the greedy construction, but we do not include this argument here.

We computed heuristic solutions for $n \in \{7, \dots, 15\}$, which, including the polyomino generation and all verifications, took around 90 minutes on an AMD Ryzen 7 7735HS CPU. The greedy construction itself takes just 11 seconds even for $n = 15$, once the longest reference vectors are calculated. Besides the 20 GB of DDR5 memory, we needed additional swap space for $n = 14$ and 15, which probably slowed down the verification further.

The areas of these solutions, which directly give a lower bound $L(n)$ on the optimum, are shown in Table 4. One can observe that the relative difference to the upper bound decreases clearly for $n \geq 9$.

n	$A'(n)$	$U(n) := \lfloor F_{\max} + \Delta_{\max} \rfloor$	$L(n)$	$(U(n) - L(n))/L(n)$
7	108	20 385	19 718	3.38%
8	369	284 170	279 105	1.81%
9	1 285	4 052 974	4 003 811	1.23%
10	4 655	61 484 424	60 832 848	1.07%
11	17 073	942 469 010	934 201 230	0.89%
12	63 600	14 736 083 080	14 625 233 629	0.76%
13	238 591	231 340 860 898	229 830 309 620	0.66%
14	901 971	3 657 193 899 930	3 635 789 660 265	0.59%
15	3 426 576	57 965 740 117 847	57 656 581 776 211	0.54%

Table 4: For the maximum area of each n -omino fence for $n \leq 15$, we present the upper bound $U(n) := \lfloor F_{\max} + \Delta_{\max} \rfloor$. The areas of the heuristic solutions are themselves a lower bound $L(n)$, which we present alongside the relative gap $(U(n) - L(n))/L(n)$. For comparison, we also include the bounds for $n \leq 8$, for which we know better bounds or even the exact values. We also have $U(5) = 154$ and $U(6) = 1 731$.

We present the reference polygons for the greedy solutions for $n = 9, 11, 13, 15$ in Figure 24. For larger n the shape becomes rounder, but even with over 3 million vectors at $n = 15$ the “corners” at the corner pieces are still distinctly visible. Of course, our heuristic solver explicitly uses the eight arcs, but the same rounded octagon shape can be observed in the optimal hexomino fence and in the dynamic programming solutions for $n = 7$ and 8. Additionally, the $F_{\max} + \Delta_{\max}$ upper bound shows that the true optimum cannot be much larger.

6.3 The limit shape

Looking at the results for $n \leq 15$ in Figure 24, it remains unclear whether the four diagonal dents in the reference polygon will persist for larger n or if the reference polygon will just become circular. To investigate this problem, we choose $n = 500$, which is a significant leap compared to the heuristic results but still leads to an acceptable runtime for the following sampling procedure. By formula (13), we estimate $A'(500)$, the number of free 500-ominoes, as $2.0 \cdot 10^{300}$. However, due to the uncertainty in the growth constant λ mentioned above, the value of $A'(500)$ could be only around 10^{297} or roughly as high as 10^{323} .

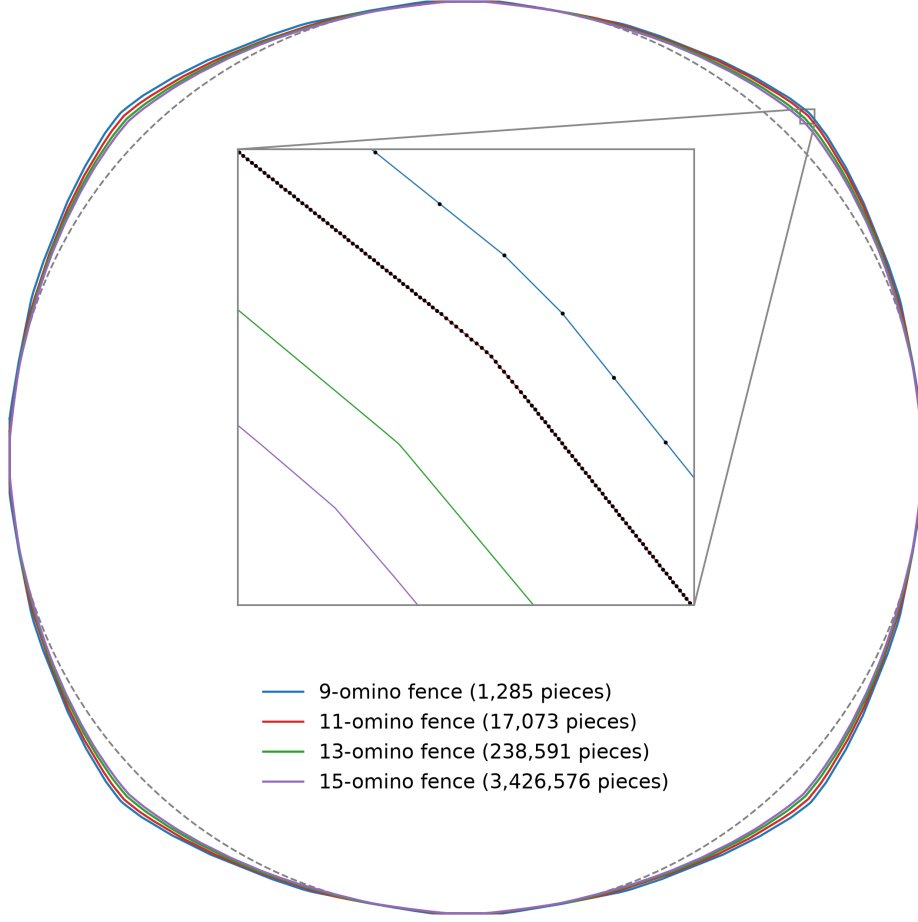


Figure 24: The reference polygons for $n = 9, 11, 13, 15$ found by the greedy approach. The polygons are scaled to fit in the same square box, alongside a gray reference circle. The enlarged inset shows the situation near the Northeast corner. Vertices for $n = 9, 11$ appear as small black dots. For larger n , the vertices lie so densely on the edges that they could not be distinguished.

We use the Metropolis algorithm [8] to sample random 500-ominoes. A step of the chain removes a uniformly random cell x of the current polyomino A and adds a uniformly random cell y that is 4-adjacent to $A \setminus \{x\}$ (possibly $y = x$); if the result is connected, it becomes the new state, and otherwise the move is rejected and A is kept. For our experiments, we made $4n^2 = 10^6$ steps of the Markov chain for every sample; this number is a heuristic choice for the mixing time. Figure 25 shows three random 500-ominoes. Our algorithm samples from the uniform distribution of *fixed* polyominoes, where rotated and reflected copies are distinct. Since the overwhelming majority of polyominoes have no symmetries, this is a good approximation for the free polyominoes, which are used for our fences.

In this way, we generated 13.5 million random 500-ominoes, using about 200 CPU days. For each of them, we determine the longest reference vector in the straightforward way: among all midpoints of boundary edges of the n -omino, we take the pair with the largest Euclidean distance and use the vector connecting the two midpoints. Up to reflections and rotations by 90° , we normalize this vector to (x, y) with $x \geq y \geq 0$. If multiple vectors share the same length, we record all of them. By Lemma 4, these vectors have integer coordinates.

Figure 26 shows a histogram of the frequency and the average vector length of the generated reference vectors, after distributing them into buckets of width 0.2° according to their angle $\alpha = \arctan \frac{y}{x}$. The average number of vectors per bucket is 60 000. In case of a tie between k reference vectors of equal length, we have counted each of these vectors with weight $\frac{1}{k}$.

Two observations stand out: Vectors around the main axes (0°) are rare, and exact diagonal lines

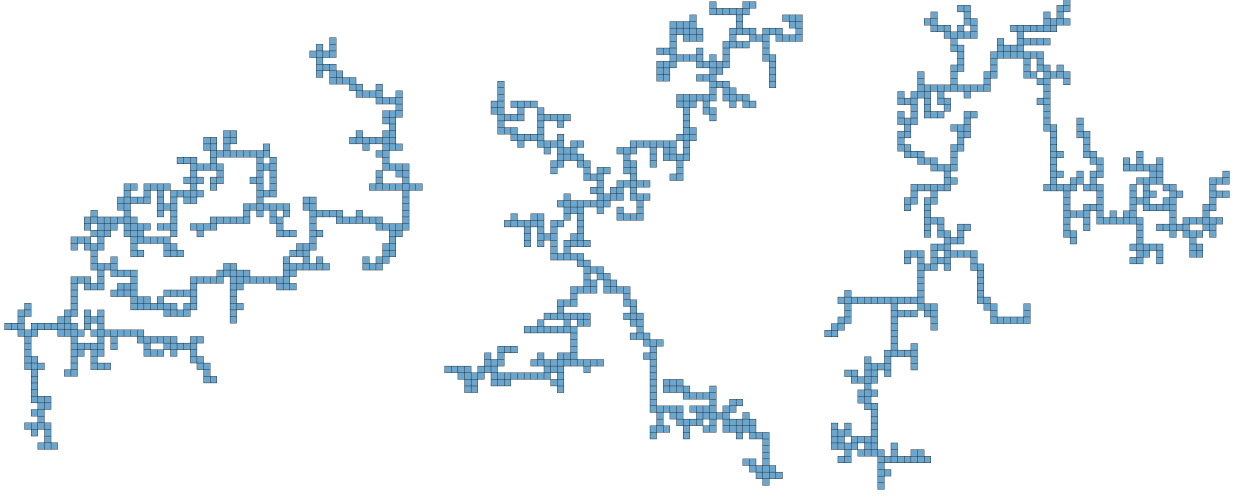


Figure 25: Three random 500-ominoes

(slope 1, i.e., 45°) are missing completely. Additionally, we observe fewer but longer reference vectors slightly above slope $\frac{1}{2}$ (26.57°), and a weaker version of this effect near slope $\frac{1}{3}$ (18.43°).

The missing 45° directions are explained by the fact that we select longest distances between *edge midpoints*, see Figure 21. A reference vector in diagonal direction can always be improved by “nudging” it off the diagonal. In fact, the closest vector to a diagonal direction in our sample is $\binom{95}{94}$, with angle 44.70° .

The irregularities near the “simple” slopes $0, \frac{1}{2}, \frac{1}{3}$, etc. are due to the properties of the rational numbers and have nothing to do with polyominoes per se. The fractions $\frac{p}{q}$ with p, q bounded by some constant are not distributed evenly, but they leave gaps near simple fractions. For example, the bucket $0^\circ\text{--}0.2^\circ$ contains all the axis-parallel vectors $\binom{1}{0}, \binom{2}{0}, \binom{3}{0}, \dots$, but the shortest vector that is not of this form is the vector $\binom{287}{1}$. The shortest vector in the bucket $0.2^\circ\text{--}0.4^\circ$ is $\binom{144}{1}$. The average length of the reference vectors is around 80. There exist 500-ominoes with much larger diameters, but they are extremely rare, and they do not appear in our sample. The vector with the smallest angle in our sample is $\binom{140}{1}$ with angle 0.41° . Accordingly, the vectors in the bucket $0.4^\circ\text{--}0.6^\circ$ are longer than average.

Let us look near the slope $\frac{1}{2}$: The shortest vector in the range $26.4^\circ\text{--}26.6^\circ$ that is not a multiple of $\binom{2}{1}$ is $\binom{141}{70}$. The shortest vectors in the adjacent range $26.6^\circ\text{--}26.8^\circ$ are $\binom{99}{50}, \binom{101}{51}, \dots$, and there are few samples of this sort. The irregularity near slope $\frac{1}{3}$ is less pronounced but still visible. Gaps near less simple fractions like $\frac{2}{3}, \frac{1}{4}$, etc., are smaller; they would show up in a histogram with narrower buckets.

Let us investigate the shape of the 500-omino fence. For this, we sort our 13.5 million samples by slope, concatenate them, and compare them with an arc of 45° of a circle, as shown in Figure 27. Since differences are hard to see at this scale, we also plot the relative difference of the arc radius at each angle in Figure 28. It is perhaps surprising that the curve runs inside the circle for small ϕ . One can also see some irregularity near $\arctan \frac{1}{2} \approx 26.57^\circ$.

We see in Figure 28 that the relative deviation from the circle decreases slowly with increasing n ; thus it seems that the shape converges to a circle after all.

In comparison to the huge number of 15-ominoes, the number of true corners, which are marked by dots, is remarkably small. The reason is that, for n -ominoes, the vectors $\binom{x}{y}$ that appear are restricted by the condition $|x| + |y| \leq n$, and practically all of these vectors appear, because one can easily construct an n -omino with a given longest reference vector. The only exceptions are vectors $\binom{x}{y}$ that are shorter than about $2\sqrt{n/\pi}$, i.e., with $x^2 + y^2 \lesssim 4n/\pi$, because of the isodiametric inequality. (The reader may try to guess what the less frequent directions are, those that appear between close consecutive dots. (Some dots are so close that they are not distinguishable in the plot.)

Finally, we estimate the area of the 500-omino fence. To this end, we combine 8 copies of the chain formed by the 13.5 million samples in Figure 27 and calculate the area of the resulting closed loop. Then we scale it up as if the loop consisted of all $2.0 \cdot 10^{300}$ free 500-ominoes instead of just

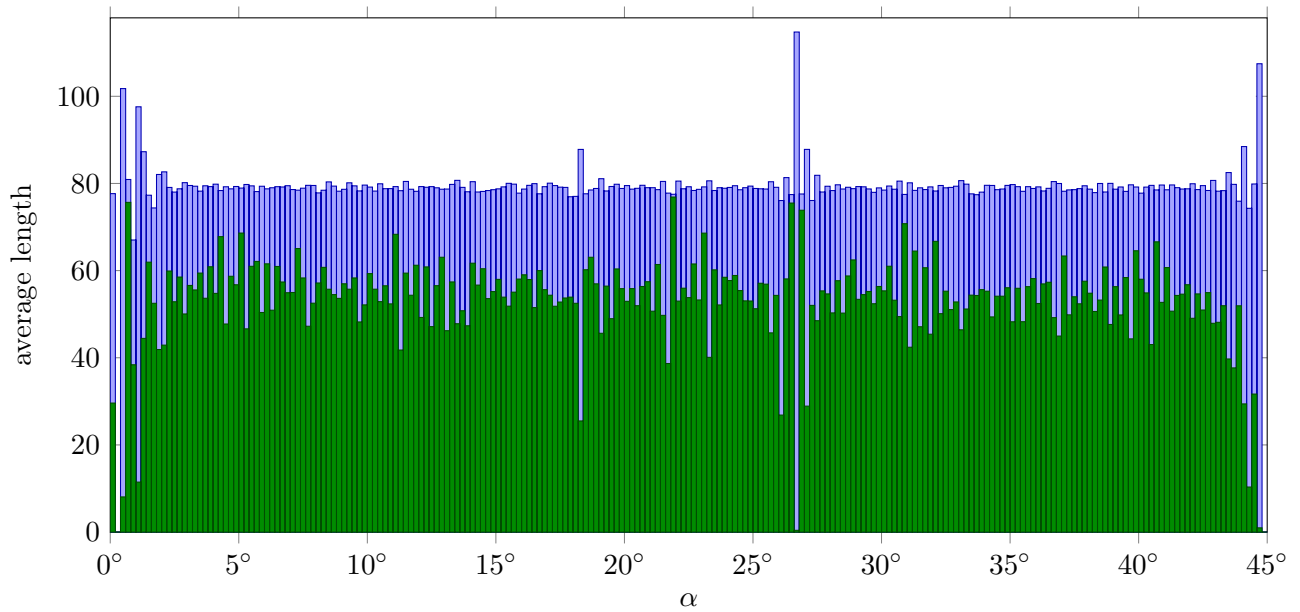


Figure 26: A histogram of the frequency (green) and the average length (blue) of the 13.5 million random reference vectors of 500-ominoes, distributed into buckets of width 0.2° . The buckets 0.2° – 0.4° and 44.8° – 45° are empty. The bucket 26.6° – 26.8° contains only 465 vectors, and these are atypically long.

108 million vectors. Doing this for our data, we arrive at a reference polygon area of $2.0 \cdot 10^{603}$. The estimate for the actual 500-omino fence is the same, because for large n , the sum of the corrections is negligible. More concretely, since the correction for any n -omino can be trivially bounded by n^2 , the sum of all corrections is at most $n^2 A'(n) \approx 5.0 \cdot 10^{305}$, i.e., around 300 orders of magnitude smaller than our estimate. However, due to the enormous uncertainty in the true number of free 500-ominoes mentioned at the beginning of this subsection, this estimate should only be taken as a rough order of magnitude.

Appendix C studies how the expected length of the longest reference vector of a random n -omino grows with n .

Interactive lab. In the *limit-shape lab* (<https://erikaroldanroa.github.io/fence-challenge/limit-shape-lab/>) one can see the reference polygons of the optimal fences for $n = 4, 5, 6$ and of the greedy solutions for $n = 7, \dots, 15$ (Figure 24) next to a circle, and experiment with reference vector samples for $n = 500$.

7 Three-dimensional walls

In three dimensions, polyominoes become polycubes, fences become *walls*, and the enclosed area becomes an enclosed *volume*.

7.1 Definitions

The definitions of Section 2.1 carry over, with the pair $(8, 4)$ of connectivities replaced by $(26, 6)$ [13]. Two cells of $\mathbb{Z}^3$ are *6-adjacent* if they share a face, and *26-adjacent* if they share at least a vertex. As in the plane, free cells are regarded as connected even if they touch only at a vertex or an edge, so the pieces of a wall must be joined face to face.

Definition 4. A placement of polycubes is a wall if the pieces are pairwise disjoint and their union Φ is 6-connected. The cells of Φ are the wall cells, and all other cells are free. The outside is the unique infinite 26-connected component of free cells; the other free cells are enclosed. The volume of the wall is the number of cells of the largest 26-connected component of enclosed cells.

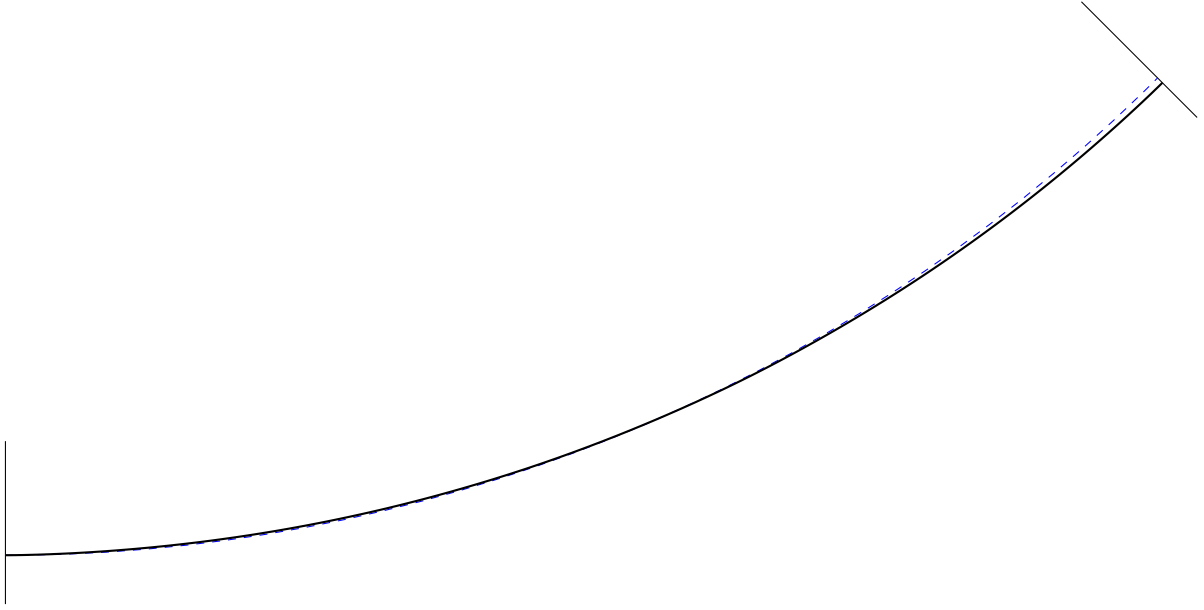


Figure 27: The reference vectors of 13.5 million random 500-ominoes, sorted by slope and concatenated into a convex polygonal chain from $(0, 0)$ to (X, Y) (solid). The dashed blue curve is the arc of the circle of radius $R = X + Y$ around $K = (0, X + Y)$, which starts at $(0, 0)$ with horizontal tangent. The two straight segments lie on the rays from K through the start and the end point of the chain; they enclose an angle of 45° .

The *n*-polycube wall problem asks for the largest volume of a wall that uses each *n*-polycube exactly once.

Orientation. For polycubes we use a different convention than for polyominoes: mirror images count as *distinct* pieces. Thus there are 29 pentacubes (23 if mirror images were identified), 166 hexacubes, and 1023 heptacubes. We did not choose this convention deliberately when we started to work on walls, but it can be defended: a polyomino can be flipped over in the three-dimensional world we live in, whereas a polycube cannot be mirrored. In any case, the wall problem is a legitimate question for both interpretations.

7.2 An isoperimetric upper bound

Extending the dynamic programming approach, which was very successful in 2D, is not feasible: there are more *n*-polycubes than *n*-ominoes, each polycube has more orientations, and there are drastically more combinations of pairwise adjacencies (instead of the single cyclic sequence of Lemma 1). Instead, we relax the problem to an isoperimetric one: we drop the requirement that the wall is composed of *n*-polycubes and only count its unit cubes.

Since a free cell may escape through 26-connected paths, a wall that encloses a set S of cells must contain every cell that shares at least a vertex with S , that is, the set $N(S) \setminus S$. Here $N(S)$ is the closed neighborhood of S in the graph on $\mathbb{Z}^3$ in which two cells are adjacent if their ℓ_∞ -distance is 1 (i.e., they share a vertex). Hence a wall built from *n*-polycubes with a total of B cubes encloses a volume of at most

$$V^*(B) = \max\{|S| : |N(S) \setminus S| \leq B\}.$$

This is the *vertex-isoperimetric problem*: for a given number of cells, find the set that can be enclosed by the fewest cubes.

In the plane, the extremal sets are squares and almost-squares; in three dimensions they are cubes and *almost-cubes* [1]. Formally, an almost-cube Q of dimension d and order $k \in \mathbb{N}$ ($k \geq 2$) is defined inductively: it consists of a primary d -dimensional cuboid

$$C = \{1, \dots, k_1\} \times \{1, \dots, k_2\} \times \dots \times \{1, \dots, k_d\}$$

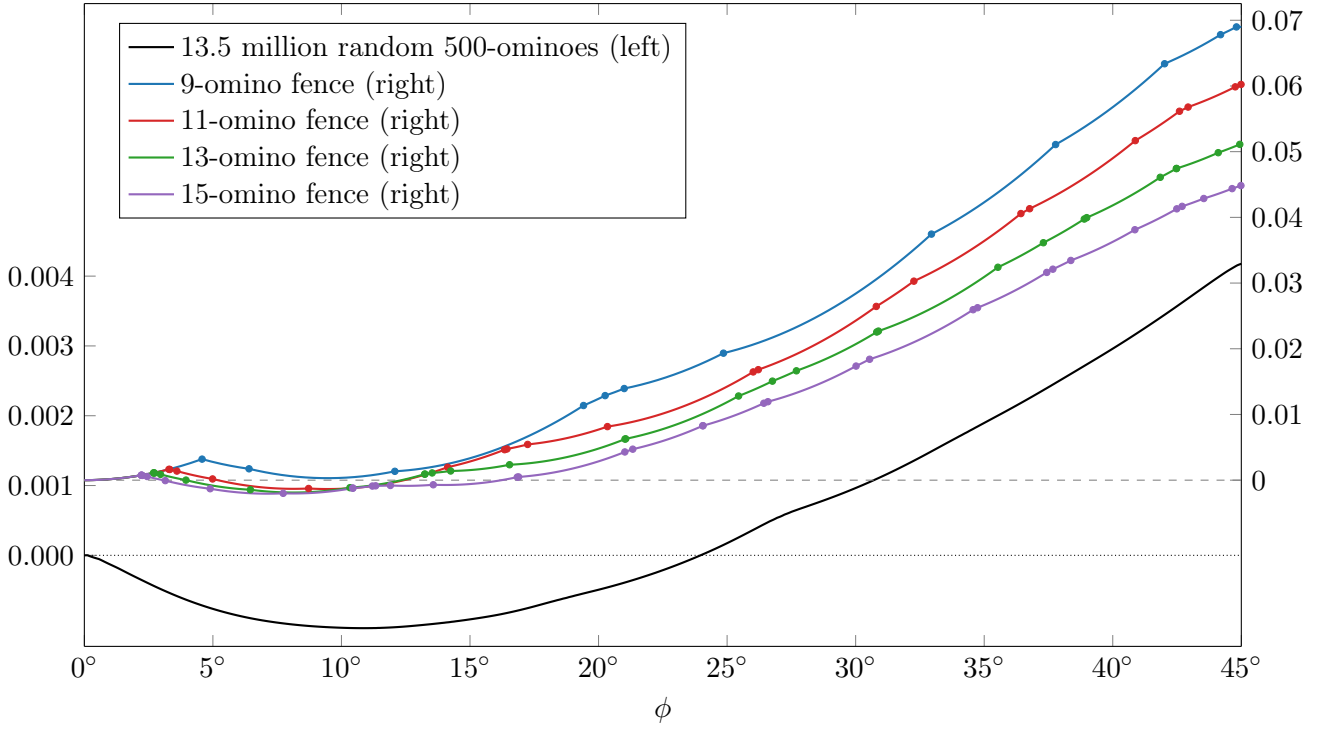


Figure 28: The deviation of the chain from the circle in Figure 27, relative to the radius R of the circle. The parameter ϕ is the angle as seen from the center of the circle, starting from the downward direction. For comparison, the colored curves show the result from the same S-to-SE $1/8$ -sector of the polygons of Figure 24, drawn with a larger vertical scale that is shown on the right edge. The dots mark the true vertices, where a change of direction occurs. The straight polygon edges appear curved because the plot compares them with a circle.

with edge lengths $k_i \in \{k, k+1\}$, and a lower-dimensional almost-cube Q' of dimension $d-1$ and order $k' \leq k$ that is attached to exactly one boundary face of C . Veomett and Radcliffe [23] solved the vertex-isoperimetric problem exactly: for every number of cells, an initial segment of an explicit ordering of $\mathbb{Z}^3$ minimizes $|N(S)|$ among all sets with this number of cells, and these sets are almost-cubes. Consequently, $V^*(B)$ is attained by an almost-cube S , which we call the *target region*; the set $W = N(S) \setminus S$ is the *target wall*.

The maximum is attained by an almost-cube, but not only by almost-cubes. This already happens in the plane: the 2×5 rectangle needs a fence of 18 cells, the same as the almost-square with 10 cells (a 3×3 square with one extra cell). Taking the budget viewpoint, however, the same 18 cells enclose a 3×4 rectangle with 12 cells; in the plane, the optimal solutions are squares or rectangles whose side lengths differ by 1. In three dimensions, a budget of 146 cubes (one more than the 29 pentacubes supply) allows the maximum volume 54, which is attained by a $3 \times 3 \times 6$ box (its wall has $5 \cdot 5 \cdot 8 - 54 = 146$ cubes) as well as by the almost-cube formed by a $3 \times 4 \times 4$ box with a 2×3 layer attached to one of its sides.

Surface area. For the related problem of minimizing the *surface area* of n unit cubes in $\mathbb{Z}^d$, the optimum is also known. It follows from a theorem of Bollobás and Leader [3] on the maximum number of edges induced by n vertices of the grid: initial segments of their cube order (which is the order later used by Veomett and Radcliffe, up to relabeling the coordinates) are optimal, and they are special almost-cubes. As above, not every optimal set is an almost-cube. Alonso and Cerf [1, Corollaries 2.5 and 3.4], who were not aware of this work, gave an explicit formula in dimensions two and three. They also described the minimizing shapes, completely in two dimensions and partially in three (their “canonical shapes” are exactly the initial segments of Bollobás and Leader).⁴ Empirically, surface area

⁴Theorem 3.5 in [1], which gives conditions under which only almost-cubes are optimizers, is subject to exceptions: one family are the $j \times j \times (j+3)$ boxes, another family are the $j \times (j+1) \times (j+1)$ boxes from which $2j$ cubes

and wall size $|N(S) \setminus S|$ seem to be minimized by the same configurations for a given number $|S|$ of cells, but this has not been proved.

Exact cover problem. If the target wall W of an optimal target region can be covered by our n -polycubes, the wall has volume $V^*(B)$, and we have found an optimal solution of the 3D wall problem. Often this is not possible. In that case we try smaller almost-cubes and also allow some polycubes to protrude from the target wall to the outside (but never into S), so that the method provides only an upper and a lower bound.

Thus we have reduced the placement of polycubes to an exact cover problem on the target wall. For small instances such as the pentacubes, exact solvers, including Knuth’s Dancing Links algorithm [10, 11] and integer linear programming, are computationally feasible. For larger polycubes, where the search space explodes, we use a greedy placement heuristic that positions new pieces adjacent to the boundaries of previously placed pieces.

7.3 Pentacubes

The 29 pentacubes consist of 145 unit cubes. The minimum wall size is always even; it is 144 for $|S| = 52$ and 146 for $|S| = 53$. Hence 145 cubes can enclose at most 52 cells. The optimal target region S is the almost-cube formed by a $3 \times 4 \times 4$ box with a 2×2 layer attached to one of its sides, with $|S| = 3 \cdot 4 \cdot 4 + 2 \cdot 2 = 52$. Its closed neighborhood $N(S)$ is the almost-cube formed by a $5 \times 6 \times 6$ box and a $4 \times 4 \times 1$ layer (Figure 29), and the target wall $W = N(S) \setminus S$ has 144 cells.

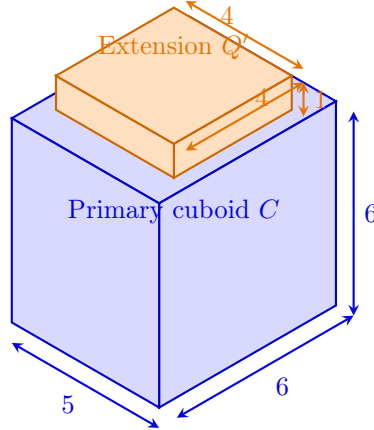


Figure 29: The closed neighborhood $N(S)$ of the optimal target region S for the pentacubes: an almost-cube with a $5 \times 6 \times 6$ primary cuboid C and a $4 \times 4 \times 1$ extension Q' . The target region S (a $3 \times 4 \times 4$ box with a 2×2 layer) lies inside; the target wall is $W = N(S) \setminus S$.

We encode the covering of W as an exact cover problem. Let $\mathcal{P}$ denote the set of pieces. A placement option $k = (p, \tau, t)$ puts piece $p \in \mathcal{P}$ in orientation τ at translation t such that the set of cells $\text{cells}(k)$ it occupies lies in W . Since the cells of W must be covered exactly, $|W|$ must be the sum of the piece sizes. Here, however, $|W| = 144$ while the pentacubes have 145 cubes, so one cube has to lie outside $N(S)$. We therefore select one pentacube and the cube of it that protrudes, and use in the model the tetracube formed by its remaining four cubes. Then $|W| = 28 \cdot 5 + 4 = 144$.

We construct a binary matrix M whose rows are the placement options. Its columns are partitioned into two groups:

1. Cell columns ($c \in W$): primary constraints requiring that each cell of the target wall is covered by exactly one piece.

along two opposite parallel edges of length j have been removed. Accordingly, as confirmed by Raphaël Cerf (personal communication, October 2026), the cases $n = j^2(j+3) = j(j+1)^2 + (j-1)j$ and $n = j(j+1)^2 - 2j = j^2(j+1) + (j-1)j$ must be excluded from the list in Theorem 3.5. The gap in the proof is that, for these values, the projections p_1, p_2, p_3 can map a non-principal polyomino to a principal one. This was first noticed by the AI agent Claude from Anthropic and then discussed with Raphaël Cerf.

2. Piece columns ($p \in \mathcal{P}$): constraints ensuring that each piece is used at most once (and hence exactly once, since the piece sizes add up to $|W|$).

With binary decision variables $x_k \in \{0, 1\}$ indicating whether placement option k is selected, the exact cover formulation is

$$\sum_{k: c \in \text{cells}(k)} x_k = 1 \quad \forall c \in W, \quad (15)$$

$$\sum_{k: \text{piece}(k)=p} x_k \leq 1 \quad \forall p \in \mathcal{P}. \quad (16)$$

The structure of M is illustrated in Table 5.

Placement Option (k)	Target Wall Cells ($c \in W$)				Pieces ($p \in \mathcal{P}$)			
	c_1	c_2	$\dots$	$c_{ W }$	p_1	p_2	$\dots$	$p_{ \mathcal{P} }$
Option 1: (p_1, τ_1, t_1)	1	0	$\dots$	0	1	0	$\dots$	0
Option 2: (p_1, τ_2, t_2)	0	1	$\dots$	1	1	0	$\dots$	0
Option 3: (p_2, τ_1, t_3)	1	1	$\dots$	0	0	1	$\dots$	0
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$

Table 5: Schematic representation of the exact cover matrix M . Rows represent valid placements of pieces; columns enforce exact coverage of the target wall cells W and the piece usage constraints for $\mathcal{P}$.

Using Knuth’s `ssxccc-binary` solver [12], which is based on his Dancing Links algorithm [10, 11], we find a valid cover of the target wall (Figure 30). Together with the upper bound of Section 7.2, this proves that the maximum volume for the pentacubes is 52.

We additionally tried to enumerate all possible constructions, but stopped the backtracking search after several weeks, at which point we had already found more than 35 million solutions.

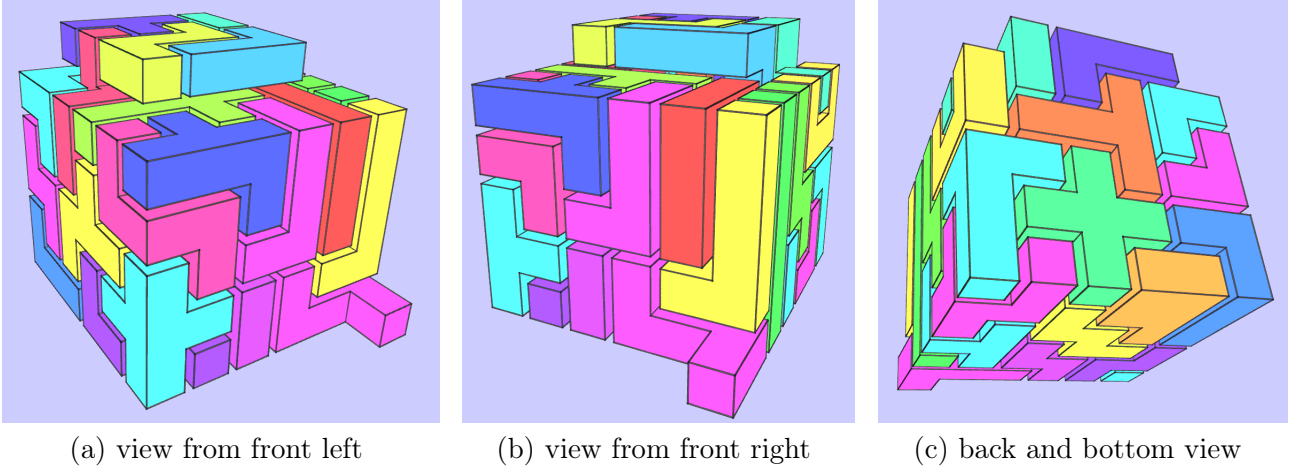


Figure 30: One optimal wall built from 29 pentacubes. It encloses a volume of 52. Each piece is shrunk to make it easier to distinguish. These pictures are taken from the viewer <https://erikaroldanroa.github.io/fence-challenge/archive/polycube-viewer/Polycubefences.html>; the cube lab <https://erikaroldanroa.github.io/fence-challenge/cube-lab/?wall=52> offers more interaction: the wall can be cut open and spread apart, and the pieces can be manipulated.

7.4 Hexacubes and heptacubes

For larger polycubes, exact solvers are computationally infeasible. Instead, we find lower bounds by applying a greedy placement heuristic to the target wall of a prescribed box-shaped target region. With this, we achieved the following results:

- **Hexacubes:** We can greedily cover the target wall of the $11 \times 11 \times 11$ cube, thus achieving an enclosed volume of $V = 1\,331$. The optimal isoperimetric solution yields an upper bound of 1 633.
- **Heptacubes:** Using the $28 \times 33 \times 28$ box as target region (whose closed neighborhood is the $30 \times 35 \times 30$ box), the greedy heuristic achieved an enclosed volume of $V = 25\,872$. The optimal isoperimetric solution yields an upper bound of 37 446.

Our implementation of the greedy placement algorithm currently only considers box-shaped target regions. It computes a score for each placement option and greedily selects the option with the highest score. The score of a placement option k is $s(k) = 2H(k) + A(k) - 2I(k)$, where $H(k)$, $A(k)$, and $I(k)$ denote the numbers of unit cubes of the placed polycube that lie in the target wall, that are adjacent to previously placed cubes, and that lie in the target region, respectively.

In the current solutions, many cubes lie outside the target wall, creating bumps rather than contributing to the enclosed volume. Local rearrangements could flatten these bumps and thereby improve the volume; we leave this computational refinement for future work.

Interactive lab. In the *cube lab* (<https://erikaroldanroa.github.io/fence-challenge/cube-lab/>) one can try building a polycube wall and explore our solutions. Additional visualizations of the solutions are available at <https://zopfbit.github.io/polycube-solutions/>.

8 Independent confirmation of hexomino optimality with integer programming

We now want to demonstrate that the ILP encoding, which we used in our previous paper [16], is also able to determine that there is no hexomino fence with an area of more than 1597. To this end, we reduce each of the 35 hexominoes to a straight or L-shaped piece with the same effect on the fence. A replacement piece does not capture the re-entrant corners of the original hexomino, so the reduction requires a correction (penalty) for the cells in these corners. Figure 31 shows three examples. In total, there are eight shapes to which the 35 pieces can be reduced. Some hexominoes can be feasibly used in two different ways, and in this case both reductions have to be considered. Since different hexominoes and different choices of reduction can lead to the same combination of L-shaped pieces, only 24 distinct combinations remain.

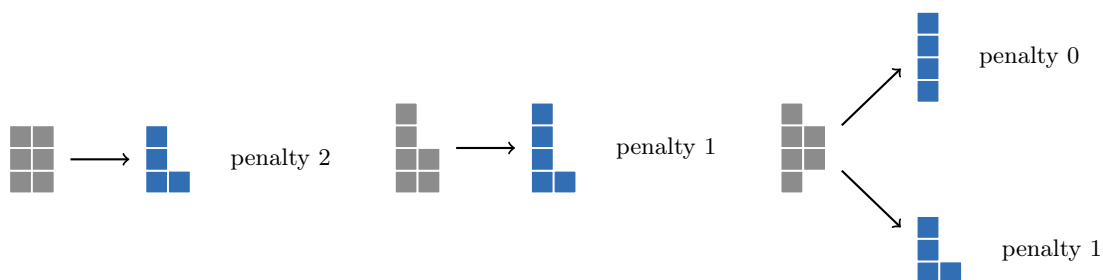


Figure 31: Three examples of the reduction. Left and middle: a hexomino with a single replacement piece. Right: a hexomino that can be replaced in two different ways. The penalty is the correction for re-entrant cells. It is 0 for the upper possibility of the right piece, because the hexomino can simply be used reflected in the y -axis.

For each combination we keep the smallest correction, which gives the largest bound and is therefore safe. These 24 simplified problems can all be solved exactly with the commercial ILP solver Gurobi, each within a few minutes. The results are listed in Table 6. The maximum, 1597, is attained in the first row. Hence no hexomino fence can enclose more than 1597 cells, and the configuration shown in Figure 14 is optimal.

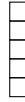
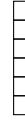
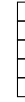
								$A_0 - \text{correction} = A$
3	2	1	3	11	6	4	5	$1634 - 37 = 1597$
3	2	1	3	10	6	4	6	$1627 - 35 = 1592$
4	2	1	3	11	6	4	4	$1628 - 37 = 1591$
4	2	1	3	10	6	4	5	$1622 - 35 = 1587$
3	2	1	3	9	6	4	7	$1619 - 35 = 1584$
4	2	1	3	9	6	4	6	$1616 - 35 = 1581$
2	2	1	3	11	6	5	5	$1617 - 36 = 1581$
3	2	1	3	11	6	5	4	$1614 - 36 = 1578$
2	2	1	3	10	6	5	6	$1611 - 34 = 1577$
3	2	1	3	10	6	5	5	$1608 - 34 = 1574$
2	2	1	3	9	6	5	7	$1603 - 34 = 1569$
3	2	1	3	9	6	5	6	$1602 - 34 = 1568$
1	2	1	3	11	6	6	5	$1600 - 35 = 1565$
2	2	1	3	11	6	6	4	$1598 - 35 = 1563$
1	2	1	3	10	6	6	6	$1594 - 33 = 1561$
2	2	1	3	10	6	6	5	$1593 - 33 = 1560$
2	2	1	3	9	6	6	6	$1586 - 33 = 1553$
1	2	1	3	9	6	6	7	$1586 - 33 = 1553$
0	2	1	3	11	6	7	5	$1583 - 36 = 1547$
1	2	1	3	11	6	7	4	$1582 - 36 = 1546$
0	2	1	3	10	6	7	6	$1577 - 34 = 1543$
1	2	1	3	10	6	7	5	$1576 - 34 = 1542$
1	2	1	3	9	6	7	6	$1570 - 34 = 1536$
0	2	1	3	9	6	7	7	$1569 - 34 = 1535$

Table 6: Sensible combinations of reducing the 35 hexominoes to different straight and L-shaped pieces, with corrections for re-entrant cells. The first eight columns give how many pieces of each type occur in the replacement problem. A_0 is the optimum of the replacement problem, found with Gurobi, and $A = A_0 - \text{correction}$ is the resulting upper bound.

It is important to mention that this approach assumes that all 35 hexominoes are only used in a “sensible” configuration, so that their contribution is actually captured by the one or two assigned L-shaped pieces. Otherwise, the number of combinations of possible L-shaped pieces would be infeasibly large. In contrast, the complete search described in Section 4 allows each hexomino to be used with any pair of incoming and outgoing attachment edges and thus provides a completely rigorous bound.

9 Conclusion

We hope that this paper conveys how much mathematical richness is hidden in this recreational geometric problem from the 1960s. Using the new dynamic programming approach we have proven optimality for the pentomino and hexomino fence, and the abstraction of the reference polygon allowed us to characterize all optimal solutions. Furthermore, we now have good bounds even for very large n -omino fences and many results on the generalization to three dimensions.

Still, many questions remain open, and we would be delighted to see further research that improves these bounds or generalizes the dynamic programming approach to other grids.

AI Usage Statement. We have developed all fence algorithms, so the dynamic programming approach including all its optimizations and the various approaches for the 3D walls, entirely without

A1266
A1267
A1268
A1269
A1270
A1271
A1272
A1273
A1274

A1275
A1276

A1277
A1278
A1279
A1280
A1281

A1282

A1283
A1284
A1285
A1286
A1287

A1288
A1289

A1290
A1291

A1292
A1293

A1294
A1295

A1296
A1297

A1298
A1299

A1300
A1301
A1302

A1303
A1304

A1305
A1306
A1307

A1308
A1309
A1310

AI assistance. This includes the programs that create good fences or walls and that establish rigorous upper bounds.

In many other ways, the creation of this paper was supported by the AI assistant Claude of Anthropic. In particular, the AI assistant translated parts of Section 4 from an earlier German draft, created and improved figures, and was instrumental in keeping a consistent notation. It provided optimized programs for running and evaluating the computational experiments in Section 6.3, and it identified literature on isoperimetric problems. It helped to formulate the precise problem specification in Sections 2 and 7, and it provided drafts for the proofs of Lemmas 3, 4, 5 and 8. The authors take full responsibility for this work.

Funding. ALR acknowledges funding from a postdoctoral fellowship of the Fonds de Recherche du Québec – Nature et Technologies (ref. 326641) during the course of this project.

Acknowledgments. The initial inspiration for this project came from Dr. Erich Fuchs, who sparked Mia Müßig’s enthusiasm for this problem many years ago when she was still a school student. We thank Corbin Masur, Frederik Safner, and Lennard Scharein, who helped with the maximum wall search in Section 7. We would also like to thank Jonas Handwerker for proofreading the initial draft of Section 3. Finally, we thank the participants of the Fence Challenge workshops.

References

[1] Laurent Alonso and Raphaël Cerf. “The three dimensional polyominoes of minimal area”. *The Electronic Journal of Combinatorics* 3.1 (Sept. 1996). DOI: 10.37236/1251 (cit. on pp. 38, 39).

[2] Gill Barequet, Günter Rote, and Mira Shalah. “ $\lambda > 4$: an improved lower bound on the growth constant of polyominoes”. *Commun. ACM* 59.7 (2016), pp. 88–95. URL: <https://doi.org/10.1145/2851485> (cit. on p. 31).

[3] Béla Bollobás and Imre Leader. “Edge-isoperimetric inequalities in the grid”. *Combinatorica* 11 (1991), pp. 299–314. DOI: 10.1007/BF01275667 (cit. on p. 39).

[4] Vuong Bui. *A convolutional approach to bounding the number of polyominoes*. 2026. arXiv: 2511.00461 [math.CO] (cit. on p. 31).

[5] Victor G. Feser. “Pentomino farms”. *Journal of Recreational Mathematics* 1 (1968), pp. 675–682 (cit. on pp. 3, 5).

[6] Martin Gardner. “Mathematical games”. *Scientific American* 228.5 (1973), pp. 102–107. URL: <http://www.jstor.org/stable/24923053> (cit. on p. 3).

[7] Sarel Har-Peled, Elfarouk Harb, and Qizheng He. *How to catch k grid points*. 2026. arXiv: 2607.10824 [cs.CG] (cit. on p. 17).

[8] W. Keith Hastings. “Monte Carlo sampling methods using Markov chains and their applications”. *Biometrika* 57.1 (1970), pp. 97–109. DOI: 10.1093/biomet/57.1.97 (cit. on p. 35).

[9] Iwan Jensen and Anthony J. Guttmann. “Statistics of lattice animals (polyominoes) and polygons”. *Journal of Physics A: Mathematical and General* 33.29 (July 2000), pp. L257–L263. DOI: 10.1088/0305-4470/33/29/102 (cit. on pp. 31, 51).

[10] Donald E. Knuth. *Combinatorial Algorithms, Part 2*. Vol. 4B. The Art of Computer Programming. Addison-Wesley, 2023 (cit. on pp. 40, 41).

[11] Donald E. Knuth. “Dancing links”. *Millennial Perspectives in Computer Science*. Ed. by Jim Davies, Bill Roscoe, and Jim Woodcock. Palgrave, 2000, pp. 187–214. arXiv: cs/0011047 (cit. on pp. 40, 41).

[12] Donald E. Knuth. *ssxcc-binary: A CWEB program for exact cover with color controls*. CWEB implementation of Dancing Links with color controls. 2020. URL: <https://cs.stanford.edu/~knuth/programs/ssxcc-binary.w> (cit. on p. 41).

A1311
A1312
A1313

A1314
A1315
A1316
A1317

A1318
A1319
A1320
A1321

A1322
A1323
A1324
A1325

A1326
A1327
A1328

A1329
A1330

A1331
A1332

A1333
A1334

A1335
A1336

A1337
A1338

A1339
A1340
A1341

[13] T. Yung Kong and Azriel Rosenfeld. “Digital topology: introduction and survey”. *Computer Vision, Graphics, and Image Processing* 48.3 (1989), pp. 357–393. DOI: 10.1016/0734-189X(89)90147-3 (cit. on pp. 4, 37).

[14] Alexis Langlois-Rémillard, Mia N. Müßig, Helmut Podhaisky, Erika Roldan, Günter Rote, Maximilian Brömme, Matthes Norden, Maximilian Prietzel, Daniel Yu, and Mykhailo Lyader. *Fence Challenge: interactive labs for polyomino fences and polycube walls*. 2026. URL: <https://erikaroldanroa.github.io/fence-challenge/hub/> (cit. on p. 3).

[15] Alexis Langlois-Rémillard, Mia N. Müßig, Helmut Podhaisky, Erika Roldan, Günter Rote, Maximilian Brömme, Matthes Norden, Maximilian Prietzel, Daniel Yu, and Mykhailo Lyader. *Polyomino Fences*. URL: <https://github.com/PhoenixSmaug/Pentomino-Farm> (cit. on pp. 3, 17, 34).

[16] Alexis Langlois-Rémillard, Mia N. Müßig, and Érika Roldán. “Maximale zäune mit polyformen”. *Mitteilungen der Deutschen Mathematiker-Vereinigung* 33.3 (Aug. 2025). English version: *Extremal fences with polyforms*, arXiv:2607.22379 (2026). DOI: 10.1515/dmvm-2025-0056. arXiv: 2607.22379 [math.HO] (cit. on pp. 3, 17, 42).

[17] Albrecht Ludwig Friedrich Meister. “Generalia de genesi figurarum planarum et inde pendentibus earum affectionibus”. (Latin). *Novi Commentarii Societatis Regiae Scientiarum Gottingensis* 1 (1769), p. 144 (cit. on p. 7).

[18] Ivan Niven. *Maxima and Minima Without Calculus*. Mathematical Association of America, 1981, pp. 234–236 (cit. on pp. 22, 33).

[19] OEIS Foundation Inc. *The On-Line Encyclopedia of Integer Sequences*. A000105: Number of free polyominoes (or square animals) with n cells. URL: <https://oeis.org/A000105> (cit. on p. 31).

[20] Azriel Rosenfeld. “Digital topology”. *The American Mathematical Monthly* 86.8 (1979), pp. 621–630. DOI: 10.1080/00029890.1979.11994873 (cit. on p. 4).

[21] G. C. Shephard. “Combinatorial properties of associated zonotopes”. *Canadian Journal of Mathematics* 26.2 (1974), pp. 302–321. DOI: 10.4153/CJM-1974-032-5 (cit. on p. 25).

[22] Takakazu Shimauchi. “Pentomino farm”. (Japanese). *Sugaku Seminar* (Mar. 1978), pp. 11–16 (cit. on p. 3).

[23] Ellen Veomett and A. J. Radcliffe. “Vertex isoperimetric inequalities for a family of graphs on $\mathbb{Z}^k$ ”. *The Electronic Journal of Combinatorics* 19.2 (2012), P45. DOI: 10.37236/2426 (cit. on p. 39).

A1342

A Optimizing the total enclosed area

A1343
A1344
A1345
A1346
A1347
A1348
A1349
A1350
A1351
A1352
A1353
A1354
A1355
A1356
A1357

For fences whose enclosed cells form several regions, Definition 1 counts only the *largest* region. This was convenient, because it gave us a cyclic order of the pieces around the boundary of this region (Section 2.2), and thereby a grip on optimizing the fence along this order by dynamic programming. An alternative, equally natural definition counts all enclosed cells: the *total enclosed area* $A_{\text{tot}}(\Phi)$ of a fence Φ is the number of its enclosed cells. For the arrangement of Figure 1, this is the hatched plus the cross-hatched area, $51 + 21 + 21 + 5 + 2 + 1 = 101$ cells. It seems unlikely that a fence whose enclosed cells fall apart into several regions could be optimal, at least when all pieces have the same size. For pentominoes, we will confirm this: the largest total enclosed area is again 128, and it is attained by exactly the same fences (Corollary 1).

Holes inside pieces, like those of C and H in Figure 1, count towards $A_{\text{tot}}(\Phi)$. Free cells that are 8-adjacent only to cells of a single piece, like the holes of C and H , are enclosed in every placement. They add a constant to the total area, and they do not affect the optimization. This is not true for every hole, however: For example, the heptomino with a hole leaves a diagonal gap next to its hole; this hole can either be connected to the outside region or be part of the enclosed area, depending on how the piece is built into the fence. Pentominoes and hexominoes have no holes.

The walk around the outside. We construct a walk along the boundary of the fence, as in Section 2.2, except that we walk along the outer boundary.

Like a region, the outside of a fence is 8-connected. Its complement, which consists of the fence cells and the enclosed cells, is finite and 4-connected: the fence Φ is 4-connected, and from every enclosed cell, a horizontal row of enclosed cells leads to a fence cell. Hence, as in Section 2.2, the boundary of the outside forms a single closed walk. We orient it clockwise around the fence, so that the outside lies on the left side and the fence on the right side. At a vertex where two cells of the outside touch diagonally, like the vertex where G and R touch in Figure 1, the walk turns right and goes through the gap. We record the fence cells along this walk; at a left turn, we also record the diagonally opposite cell that meets the walk only at a vertex. The recorded cells are the fence cells that are 8-adjacent to the outside, and consecutive recorded cells are 4-adjacent. Replacing every recorded cell by the piece that contains it and merging consecutive repetitions, we obtain the cyclic sequence of the *pieces around the outside*.

In Figure 1, the pieces around the outside, in clockwise order starting at the top left, are $ABCDE-FQRQFGHIEJKLMNOCP$. The pieces C , E , F , and Q occur twice, and S and T do not occur at all.

Definition 5. A fence Φ is simple if it consists of a single piece, or if its pieces can be arranged in a cyclic sequence $P_1, P_2, \dots, P_k$ with $k \geq 2$ (indices taken modulo k) such that there is a Jordan curve Z with the following properties:

- Z runs in the interior of the fence, that is, in the interior of the union Φ of the pieces (regarded as closed subsets of the plane);
- Z visits every piece exactly once, in the given order: for each i , the intersection $Z \cap P_i$ is an arc that lies in the interior of P_i , except for its endpoints. One endpoint is an interior point of an edge that P_i shares with P_{i-1} , and the other one is an interior point of an edge that P_i shares with P_{i+1} ;
- all enclosed cells of Φ lie inside Z .

The edges through which Z passes from one piece to the next are the attachment edges from Section 3.

Since Z runs in the interior of the fence, a free cell inside Z cannot be 8-connected to a free cell outside Z . Hence the enclosed cells of a simple fence are precisely the free cells inside Z . However, they can form several regions, as in Figure 32.

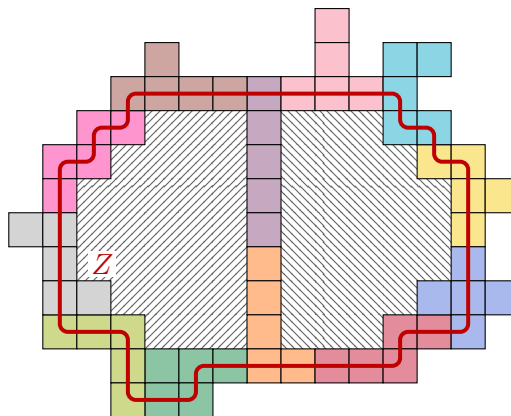


Figure 32: A simple fence of the 12 pentominoes. The I-pentomino and the L-pentomino form a wall that separates the enclosed area into two regions of 29 cells each (hatched in different directions). The total enclosed area is 58. The Jordan curve Z visits the pieces in the cyclic order in which they occur around the outside, and it passes from one piece to the next through an attachment edge.

Lemma 6. Let Φ be a fence.

(a) If every piece of Φ occurs exactly once in the cyclic sequence of pieces around the outside, then Φ is simple.

(b) If some pieces of Φ do not occur in this sequence, then the pieces that do occur form a fence Φ' with $A_{\text{tot}}(\Phi') > A_{\text{tot}}(\Phi)$.

(c) If some piece P occurs more than once in this sequence, then Φ can be decomposed into two fences Φ_1 and Φ_2 , each consisting of at least two pieces, that share the piece P and no other piece, such that every enclosed cell of Φ is an enclosed cell of Φ_1 or of Φ_2 . In particular, $A_{\text{tot}}(\Phi) \leq A_{\text{tot}}(\Phi_1) + A_{\text{tot}}(\Phi_2)$.

Consequently, if Φ is not simple, then (b) or (c) applies.

Proof. Let ω be the walk around the outside.

(a) Let Z be the closed curve that runs parallel to ω at offset $\frac{1}{4}$ on its right side. Along an edge of ω , Z runs through the fence cell on the right side of this edge. At a right turn of ω , it stays in this cell, and at a left turn, it goes around the vertex through the diagonally opposite cell. Thus, Z passes through the recorded cells in the order in which they are recorded, and it passes from one recorded cell to the next through an interior point of their common edge. In particular, Z runs in the interior of the fence. The walk ω uses every edge only once, and at a vertex that it passes twice, it turns right both times. Therefore, Z is a Jordan curve. If every piece occurs exactly once around the outside, then Z visits every piece exactly once, as required in Definition 5. Finally, Z surrounds all points of the complement of the outside that have distance more than $\frac{1}{4}$ from ω , because ω is the boundary of this complement. An enclosed cell is not 8-adjacent to the outside, and so it has distance at least 1 from ω . Hence, all enclosed cells lie inside Z , and Φ is simple.

(b) Since consecutive recorded cells are 4-adjacent, the pieces around the outside form a 4-connected set, and hence a fence Φ' . Removal of the pieces that are not used in Φ' increases the enclosed area, and therefore $A_{\text{tot}}(\Phi') > A_{\text{tot}}(\Phi)$.

(c) We cut the fence into two parts along a Jordan curve γ through P and the outside region, see Figure 33. More precisely, we pick two occurrences of P in the cyclic sequence. Then the sequence has the form P, σ_1, P, σ_2 , where σ_1 and σ_2 are nonempty. Let a be a cell of P that is recorded at the first occurrence, and let o_1 be a cell of the outside that is 8-adjacent to a (the cell on the left side of the corresponding edge of ω , or the corner cell of the outside at a left turn). Similarly, choose c and o_2 for the second occurrence. We form the Jordan curve γ from four parts: a straight segment from the center of a to the center of o_1 , a path from o_1 to o_2 through the centers of cells of the outside (which is 8-connected), a straight segment from o_2 to c , and a path from c back to a through the centers of cells of P (which is 4-connected). The fences Φ_1 and Φ_2 will then consist of P together with the pieces inside γ and outside γ , respectively. The remainder of the proof will establish formally that this construction achieves the claimed goals.

First, γ is indeed a Jordan curve, if we choose shortest paths in the above construction: In a shortest path, two cells that are not consecutive are not 8-adjacent, and so the path cannot cross itself.

The curve γ crosses ω exactly twice, between a and o_1 and between o_2 and c . Elsewhere, it can touch ω only at vertices where it passes diagonally between two cells of the outside. So, the part of ω that runs along σ_1 lies on one side of γ , and the part that runs along σ_2 lies on the other side. Every piece other than P meets γ at most in isolated vertices, and hence it lies entirely inside or entirely outside γ . In particular, the pieces of σ_1 other than P lie on one side of γ , say inside, and the pieces of σ_2 other than P lie outside; thus σ_1 and σ_2 have no piece other than P in common.

Let Φ_1 consist of P and the pieces inside γ , and let Φ_2 consist of P and the pieces outside γ . Each of them contains at least two pieces. A 4-connected path of fence cells from a piece inside γ to P , drawn through the centers of its cells, can leave the inside of γ only through P , since γ runs only through P and through the outside. Hence Φ_1 is 4-connected, and similarly, Φ_2 is 4-connected; thus, both are fences.

Finally, let e be an enclosed cell of Φ . It does not meet γ , and we assume without loss of generality that it lies inside γ . Suppose that e is not enclosed in Φ_1 . Then there is an 8-connected path of free cells of Φ_1 from e to a cell far away, outside the bounding box of Φ . Let g be the first cell on this

path that is not an enclosed cell of Φ ; it exists because the last cell of the path is not enclosed in Φ . It is 8-adjacent to an enclosed cell of Φ , so it does not belong to the outside of Φ . Since g is free in Φ_1 , it belongs to a piece of Φ_2 other than P , and thus it lies outside γ . But the path from e to g , drawn through the centers of its cells, does not meet γ : it never enters P , and up to g , it runs through enclosed cells of Φ , which are not even 8-adjacent to the outside of Φ . This contradiction shows that e is enclosed in Φ_1 . $\square$

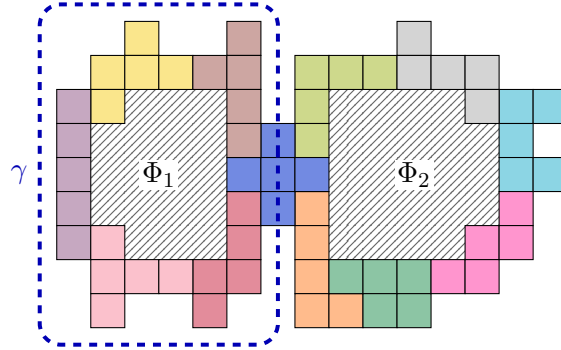


Figure 33: A fence of the 12 pentominoes in which the X-pentomino (dark) occurs twice around the outside: once from above, and once from below. The dashed closed curve γ runs through the outside and through X. It cuts the fence into two simple fences that share the X-pentomino: Φ_1 , with six pieces, encloses 18 cells, and Φ_2 , with seven pieces, encloses 23 cells.

In the example of Figure 33, the total enclosed area is the sum of the total areas of the two subfences. This is not true in general, because a hole inside the shared piece P is counted in both subfences.

For a more elaborate example, we look again at Figure 1. Although this arrangement is not a fence, the walk around its outside is defined in the same way, and we have seen that the pieces S and T do not occur. The other 18 pieces form a fence of total area 110, in which the cells of S and T become free and join the regions that surround them. Cutting this fence through the pieces C , E , F , and Q , which occur twice, in any order, we obtain five simple fences, consisting of the pieces $ABCP$, $CDEJKLMNO$, $EFGHI$, FQ , and QR , respectively. The last two, in the upper right corner, enclose nothing. The other three have total areas 22, 60, and 29. Their sum, 111, exceeds 110 by one, because the hole of C is counted twice.

Lemma 6 reduces every fence to simple fences. Since the area that can be enclosed by i pieces grows roughly quadratically with i , splitting a fence into two parts that share a piece is expensive. The following lemma makes this precise. A fence built from pieces of $\mathcal{S}$ uses each piece of $\mathcal{S}$ at most once.

Lemma 7. *Let $\mathcal{S}$ be a family of N polyomino pieces, and let $u_1, u_2, \dots, u_N$ be numbers with the following properties.*

(i) *For $1 \leq i \leq N$, u_i is an upper bound on the total enclosed area of every simple fence that is built from i pieces of $\mathcal{S}$.*

(ii) *$u_i + u_j \leq u_{i+j-1}$ for all $i, j \geq 2$ with $i + j \leq N + 1$.*

(iii) *$u_1 \leq u_2 \leq \dots \leq u_N$.*

Then:

(a) *Every fence Φ that is built from i pieces of $\mathcal{S}$, $1 \leq i \leq N$, satisfies $A_{\text{tot}}(\Phi) \leq u_i$. In particular, u_N is an upper bound on the total area enclosed by any fence built from pieces of $\mathcal{S}$.*

(b) *If, in addition, the inequalities in (ii) are strict, then every fence that is built from all N pieces of $\mathcal{S}$ and is not simple satisfies $A_{\text{tot}}(\Phi) < u_N$.*

The inequalities $u_2 \leq u_3 \leq \dots \leq u_N$ in (iii) follow from (ii) when $u_2 \geq 0$, by taking $j = 2$; only $u_1 \leq u_2$ must be checked separately. For larger polyominoes, which may enclose holes inside the pieces, it would likely be advantageous to take those holes out of the game and adjust the lemma. Then, u_1 becomes zero, and double-counting of those holes is avoided.

Proof. (a) We use induction on i . If Φ is simple, in particular if $i = 1$, then $A_{\text{tot}}(\Phi) \leq u_i$ by (i). Otherwise, by Lemma 6, case (b) or (c) of this lemma applies. In case (b), the fence Φ' consists of $i' < i$ pieces, and by induction and (iii), $A_{\text{tot}}(\Phi) < A_{\text{tot}}(\Phi') \leq u_{i'} \leq u_i$. In case (c), the fences Φ_1 and Φ_2 consist of $i_1 \geq 2$ and $i_2 \geq 2$ pieces with $i_1 + i_2 = i + 1$, and hence $i_1, i_2 < i$. By induction and (ii),

$$A_{\text{tot}}(\Phi) \leq A_{\text{tot}}(\Phi_1) + A_{\text{tot}}(\Phi_2) \leq u_{i_1} + u_{i_2} \leq u_{i_1+i_2-1} = u_i. \quad (17)$$

The second statement holds because $u_i \leq u_N$ by (iii).

(b) Let Φ be a fence that is built from all N pieces and is not simple. In case (b) of Lemma 6, we get $A_{\text{tot}}(\Phi) < A_{\text{tot}}(\Phi') \leq u_{i'} \leq u_N$ as before, by part (a). In case (c), the last inequality in (17) is strict, and $A_{\text{tot}}(\Phi) < u_N$. $\square$

To apply Lemma 7, we need upper bounds for simple fences. For pieces without holes, these are provided by the dynamic program of Section 3, since the total area of a simple fence is the area of a closed walk of the kind that the dynamic program considers:

Lemma 8. *Let Φ be a simple fence with $k \geq 2$ pieces whose boundaries are simple closed curves, and let Z be a curve as in Definition 5. We orient Z counterclockwise and number the pieces $P_1, \dots, P_k$ in the order in which Z visits them.*

(a) *If $k = 2$ and Z crosses the same edge twice, then $A_{\text{tot}}(\Phi) = 0$.*

(b) *Otherwise, for each i , let β_i be the part of the boundary of P_i that lies inside Z between the attachment edge through which Z enters P_i and the attachment edge through which Z leaves P_i , excluding these two edges. Then $\beta_1, \beta_2, \dots, \beta_k$ form a closed walk β , and the area enclosed by β , as computed by the Shoelace formula (1), is $A_{\text{tot}}(\Phi)$.*

Proof. Every piece P_i is a topological disk, and the arc $Z \cap P_i$ cuts it into two parts. Let P_i^+ be the part inside Z .

(a) In this case, the arc $Z \cap P_i$, for $i = 1, 2$, and the segment between the two crossing points on the common edge bound a part of P_i . These two parts lie on different sides of the common edge, and together they form a disk bounded by Z . Hence they are the parts P_1^+ and P_2^+ , they cover the inside of Z , there is no free cell inside Z , and $A_{\text{tot}}(\Phi) = 0$.

(b) The curve Z meets each attachment edge only at one point, where it crosses the edge. Hence exactly one endpoint of the edge lies inside Z , and we call the segment from the crossing point to this endpoint the *inner part* of the edge. The boundary of P_i^+ consists of the arc $Z \cap P_i$, the inner parts of the two attachment edges of P_i , and β_i . Thus, β_i leads from the endpoint inside Z of the attachment edge where Z enters P_i to the endpoint inside Z of the attachment edge where Z leaves P_i . The latter edge is the one where Z enters P_{i+1} , so consecutive sections fit together to a closed walk β .

When we replace the arc $Z \cap P_i$ by the path along the boundary of P_i^+ that consists of the inner part of an attachment edge, β_i , and the inner part of the other attachment edge, the winding number around a point decreases by 1 if the point lies in the interior of P_i^+ , and otherwise it does not change. After doing this for all i , the inner parts of the attachment edges cancel, and we obtain β . Hence β winds once around every point inside Z that is not covered by a piece, and zero times around every other point that is not on β . Since the Shoelace formula integrates the winding number, it yields the number of free cells inside Z , which are precisely the enclosed cells of Φ . $\square$

The walk β runs counterclockwise around the enclosed cells, with the pieces on its right side, and β_i is a section of the boundary of P_i between an incoming and an outgoing attachment edge, as described in Section 3.1; it may degenerate to a single point, like the section of K in Figure 1. Consecutive pieces share the attachment edge between them. So, β is one of the closed boundaries whose Shoelace area is maximized by the dynamic program of Section 3, and for pentominoes, $A_{\text{tot}}(\Phi)$ is bounded by the entry $A_{\text{max}}(k)$ of Table 2.

Corollary 1. *The largest total enclosed area of a fence built from pentominoes is 128. Every such fence with total enclosed area 128 uses all 12 pentominoes, is simple, and has a single region of 128 cells. Thus, the optimal fences with respect to the total enclosed area are the same as the optimal fences with respect to the area of Definition 1, namely the 5760 solutions found in Section 3.3.*

Proof. We apply Lemma 7 to the family $\mathcal{S}$ of the $N = 12$ pentominoes, with $u_i = A_{\max}(i)$ from Table 2. By the construction in Section 3.3, $A_{\max}(i)$ is the largest area, computed by the Shoelace formula, of a closed boundary that the dynamic program can form from i distinct pentominoes, maximized over all choices of the pentominoes, of their cyclic order, and of their incoming and outgoing attachment edges, regardless of whether the pieces overlap. By Lemma 8 and the remark following it, $A_{\max}(i)$ is an upper bound on the total enclosed area of every simple fence of i pentominoes, for $i \geq 2$ (in the case of Lemma 8(a), trivially). A single pentomino encloses nothing, and $A_{\max}(1) = 0$. Thus, condition (i) holds. Condition (iii) is obvious from the table. Condition (ii) holds even with strict inequality, as can be checked in the table for all pairs $2 \leq i \leq j$ with $i + j - 1 \leq 12$. For example, $A_{\max}(4) + A_{\max}(6) = 13 + 34 = 47 < A_{\max}(9) = 78$. The smallest gap occurs for $i = j = 2$, where $1 + 1 < 6 = A_{\max}(3)$.

By Lemma 7(a), every fence Φ built from pentominoes satisfies $A_{\text{tot}}(\Phi) \leq A_{\max}(12) = 128$, and the optimal fences of Section 3.3 attain this bound. Now let Φ be a fence with $A_{\text{tot}}(\Phi) = 128$. Lemma 7(a) excludes the case that Φ does not use all 12 pentominoes, because $A_{\max}(11) = 111$. By Lemma 7(b), Φ is simple. By Lemma 8, the walk β of Φ is a closed boundary of the dynamic program with area 128, the optimum. Therefore, Φ is one of the optimal placements that were enumerated in Section 3.3, and all of them consist of a single region of 128 cells. $\square$

The proof gives more: a fence of pentominoes that is not simple encloses at most 112 cells in total. Indeed, in case (b) of Lemma 6, $A_{\text{tot}}(\Phi) < A_{\text{tot}}(\Phi') \leq A_{\max}(i') \leq A_{\max}(11) = 111$, and in case (c), $A_{\text{tot}}(\Phi) \leq A_{\max}(i_1) + A_{\max}(i_2)$ with $i_1 + i_2 = i + 1 \leq 13$, which is at most $A_{\max}(2) + A_{\max}(11) = 112$.

For hexominoes, whose boundaries are also simple closed curves, Lemma 8 holds as well. The bounds of Section 4 should also apply to the walk β of a simple fence that uses all 35 hexominoes. However, β may touch itself, and so the rotation condition $D_1 + \dots + D_{35} = 360^\circ$, which was derived for the boundary of a region, would have to be justified with the help of the Jordan curve Z (for example, by applying the theorem on turning tangents to the boundary of each part P_i^+). To show that no hexomino fence encloses a total area of 1598 or more, it would suffice to derive upper bounds u_i for simple fences of $i < 35$ hexominoes that satisfy the conditions of Lemma 7 together with $u_{35} = 1597$, for example with the simpler techniques of Section 5.2. Since nonsimple fences pay a substantial penalty, rather weak bounds should suffice. However, we have not carried out this proof.

B Hexomino area distribution of random orders

We also ran OPTIMALORDEREDFENCE on various orders for the 35 hexominoes. Since here the total number of orders is infeasibly large ($\frac{34!}{2} \approx 1.5 \cdot 10^{38}$), we sampled 10^5 orders uniformly at random and show the resulting distribution in Figure 34.

The shape closely mirrors the pentomino distribution in Figure 9: a roughly bell-shaped curve with a sharp left tail and a gradual right tail. The smallest maximum hexomino fence area was 1458, which we observed three times among the samples.

We also experimented with a simple genetic algorithm that tries to minimize the maximum hexomino fence area by swapping hexominoes in the order. Figure 35 shows the best order found after 6 hours of running time, with a maximum area of 1448.

C Expected length of reference vectors

As an extension of Section 6.3, we now determine the expected length of the longest reference vector $v^{\max}$ of a random n -omino, which we denote by $\mathbb{E}[|v^{\max}|]$. We reuse the Metropolis algorithm to sample 100 000 random n -ominoes for each $n \in \{10, 20, \dots, 490, 500\}$ and show their average reference vector length in Figure 36.

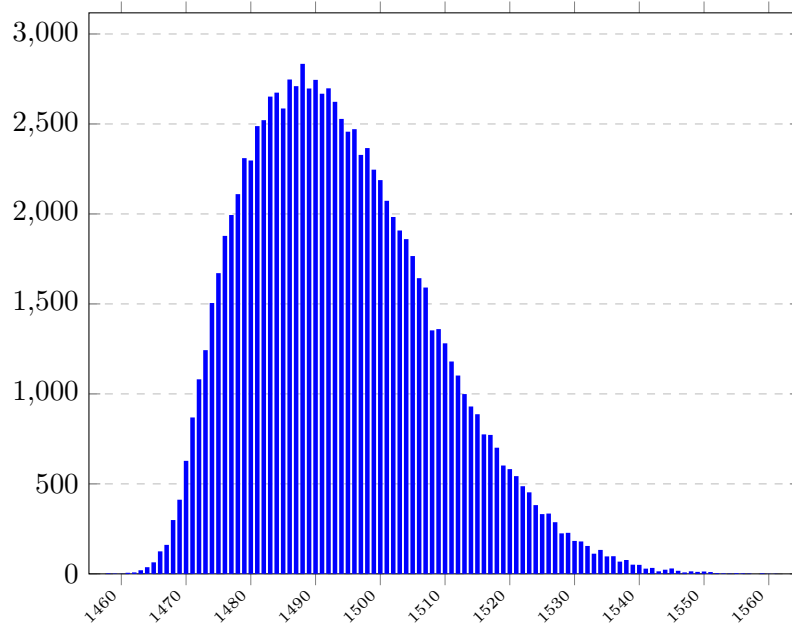


Figure 34: Distribution of the maximum achievable fence area across 10^5 random hexomino orders.



Figure 35: A hexomino order whose best fence has area 1448, the smallest maximum among all orders that we found.

Fitting a curve of the form $a \cdot n^b - c$ to these data, we obtain $1.464 \cdot n^{0.643} - 0.479$. We want to contrast the exponent with a related quantity: Jensen and Guttmann [9] concluded from numerical experiments that the radius of gyration (the root-mean-square distance of the cells from the centroid) of an n -omino scales with the exponent 0.64115. It is remarkable that the exponents are quite close. This means that the square of the diameter, i.e., of the *maximum* distance, does not grow much faster than the *average* squared distance. Indeed, as can be seen in Figure 25, typical polyominoes do not fill the area densely, but they grow mainly along linear structures.

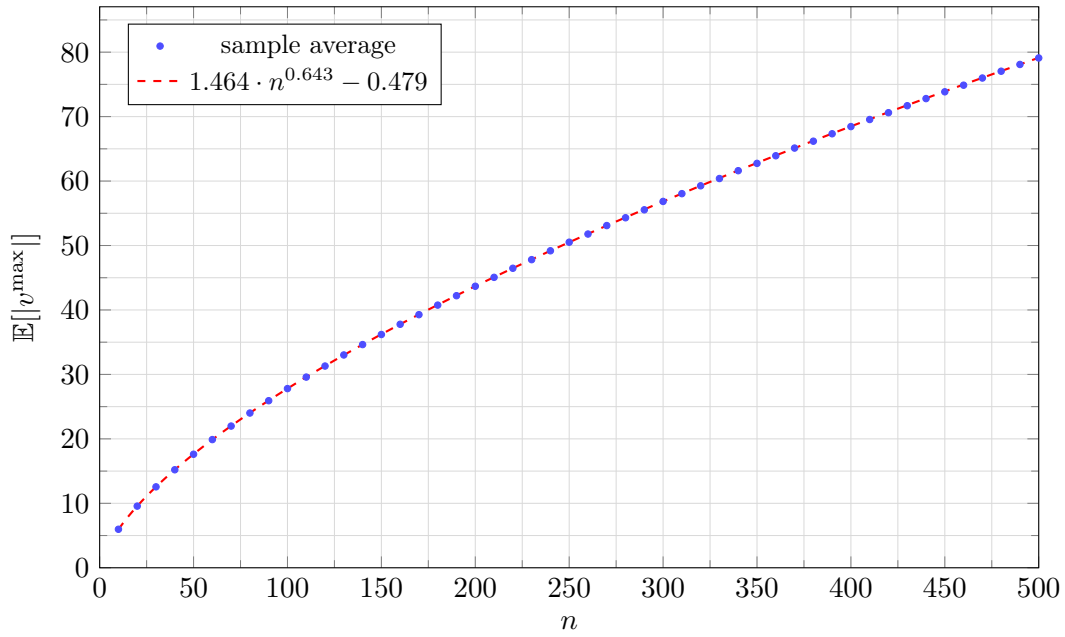


Figure 36: Monte Carlo estimate of $\mathbb{E}[v^{\max}]$ using 100 000 random n -ominoes each for $n = 10, 20, \dots, 490, 500$.

A1577 (A. Langlois-Rémillard) AALTO UNIVERSITY, OTAKAARI 1 F 02150, ESPOO, FINLAND
A1578 *Email address:* alexis.langlois-remillard@aalto.fi

A1579 (M. N. Müßig) LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN, GERMANY
A1580 *Email address:* nienna@miamuessig.de

A1581 (H. Podhaisky) HALLE, GERMANY

A1582 (G. Rote) FREIE UNIVERSITÄT BERLIN, GERMANY
A1583 *Email address:* rote@inf.fu-berlin.de

A1584 (E. Roldan) UNIVERSITÉ DE GENÈVE, SWISS
A1585 *Email address:* roldan@mis.mpg.de

A1586 (M. Norden) FREIE UNIVERSITÄT BERLIN, GERMANY

A1587 (M. Brömme) FREIE UNIVERSITÄT BERLIN, GERMANY
A1588 *Email address:* maximilian.broemme@aviomatik.de

A1589 (M. Prietzel) FREIE UNIVERSITÄT BERLIN, GERMANY
A1590 *Email address:* maximilian.prietzel@proton.me

A1591 (D. Yu) FREIE UNIVERSITÄT BERLIN, GERMANY
A1592 *Email address:* barzindyne@gmail.com

A1593 (M. Lyader) GEORG-CANTOR-GYMNASIUM, GERMANY
A1594 *Email address:* mykhailolyader@gmail.com