

Folding a Strip of Paper into Shapes with Specified Thickness

MIT Folding Group*, Lily Chung, Erik D. Demaine, Martin L. Demaine, Jenny Diomidova, Jayson Lynch, Klara Mundilova, Hanyu Alice Zhang

Abstract: *Computational origami design typically focuses on achieving a desired shape of folding, treating multiple layers of paper like a single layer. In this paper, we study when we can achieve a desired shape with a desired constant number of layers throughout the shape, or a specified pattern of layer thicknesses. Specifically, we study the case of a rectangular strip of paper, which is the setting of the first universal computational origami design algorithm [SoCG'99]. Depending on the generality of the target surface and on the number of layers modulo 4, we give a variety of universal design algorithms, polynomial-time decision algorithms characterizing what is possible to fold, and NP-hardness results.*

1 Introduction

Since SoCG'99 [Demaine et al. 00], we have known that *origami is universal* in the sense that any shape of paper can fold into (a scaling of) any desired polyhedral surface. Deciding whether a desired surface can be folded a specified piece of paper remains one of the standard goals in computational origami design. More efficient universal algorithms include the Origamizer algorithm from SoCG 2017 [Demaine and Tachi 17]. Such results rely on a key perspective on what it means to “fold a surface”: multiple overlapping layers of paper are treated the same as a single layer. In other words, the goal is to *cover* the surface, not fold it exactly. (The latter is the notion of polyhedron folding, which is not universal [Demaine and O'Rourke 07].)

The number of layers of coverage can be important, however. Thick material such as sheet metal cannot easily support many overlapping layers (see, e.g., [Ku and Demaine 16]), so we may want to upper bound the number of layers at any point. On the other hand, we may want to lower bound the number of layers at every point as a proxy for structural stiffness. In particular, to optimally balance these two constraints, we may aim for a uniform number of layers at every point. Alternatively, we may want to exactly control the number of layers, varying at

* Artificial first author to highlight that the other authors (in alphabetical order) worked as an equal group. Please include all authors (including this one) in your bibliography, and refer to the authors as “MIT Folding Group” (without “et al.”).

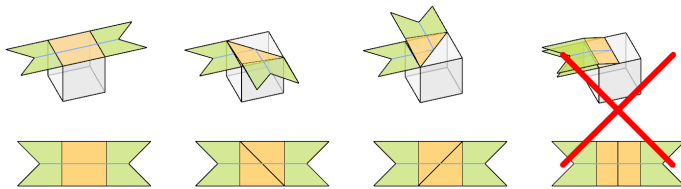


Figure 2: Illustration of allowed folds in the grid + diagonals model. A square can be either increased (left) or folded along the square’s diagonals (middle), but cannot be folded along half-grid lines (right). We draw the mid-line of the strip (in blue) to help visualize the tiling model introduced in Section 2.

different points, to obtain a desired shadow pattern when the folding is held up to the light; Figure 1 shows a simple example.

Our results. In this paper, we analyze what shapes with a specified number of layers, possibly varying throughout the shape, can be folded from a long rectangular strip of paper. Folding a paper strip is the approach taken by the original universality result [Demaine et al. 00] as well as more recent work [Benbernou et al. 20], so it is a natural starting point for an investigation of layer counts. We follow the *grid + diagonals model* of folding [Aichholzer et al. 21, Czajkowski et al. 20, Benbernou et al. 20]; refer to Figure 2. The strip is a grid-aligned $1 \times L$ rectangle and each crease is between two integer points at distance 1 (perpendicular to the strip) or $\sqrt{2}$ (diagonal to the strip). Each perpendicular crease can be folded by $\pm 90^\circ$ or $\pm 180^\circ$, and diagonal folds can be folded by $\pm 180^\circ$ (to keep the folding grid-aligned).

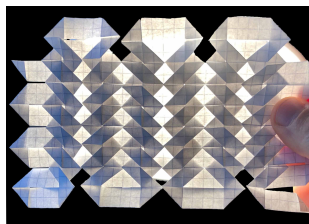


Figure 1: The shadow pattern resulting from folding a strip of paper in a woven pattern.

In this model, the shapes we can build are (connected) polyhedral complexes made up of grid-aligned unit squares and/or triangular half-squares (right isosceles triangles with side lengths $1, 1, \sqrt{2}$). We categorize these into the following special classes; refer to Figure 3. (In the abstract cases, we ignore the constraint that the creases get folded by $\pm 90^\circ$ or $\pm 180^\circ$.)

- A *polyomino* is a 2D shape made up of grid-aligned unit squares.
- A *polyabolo* is a 2D shape made up of grid-aligned unit squares and/or triangular half-squares.
- A *polycube* is a manifold without boundary embedded in 3D that is made up of grid-aligned unit squares; in other words, it is the boundary of the union of finitely many grid-aligned unit cubes.

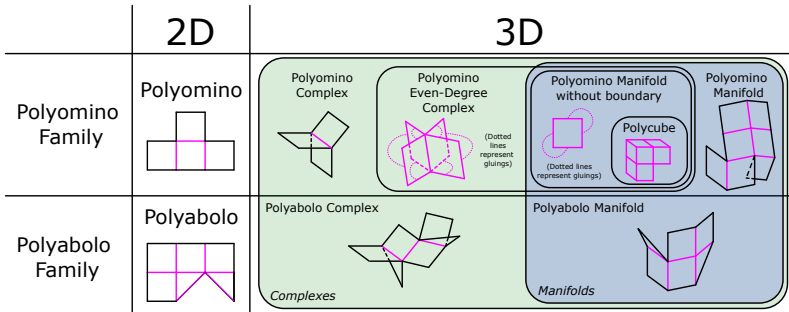


Figure 3: Different classes of target shapes analyzed in this paper. Magenta lines denote edges where faces are glued together, while black lines are boundary. Dotted lines indicate long-distance gluings.

- A **polyabolo manifold** is an abstract manifold, possibly with boundary, made up of unit squares and/or triangular half-squares joined along disjoint pairs of unit edges. (Manifolds allow at most two (half-)squares to be attached at a common edge.)
- A **polyomino manifold** is a polyabolo manifold made up of just unit squares (no triangular half-squares).
- A **polyomino manifold without boundary** is the special case of a polyomino manifold where exactly two squares are attached at each edge. (This notion generalizes polycubes to an abstract setting. Because we allow joins along only unit edges, a manifold without boundary cannot have triangular half-square faces: any diagonal edge would be a boundary edge.)
- A **polyabolo complex** is an abstract complex made up of unit squares and/or triangular half-squares joined arbitrarily along unit edges. (Complexes allow any number of (half-)squares to be attached at a common edge. This case is the most general.)
- A **polyomino complex** is a polyabolo complex made up of just unit squares (no triangular half-squares).
- An **even-degree polyomino complex** is the special case of a polyomino complex where an even number of squares are attached at each edge. (This notion generalizes polyomino manifold without boundary. For the same reason, it does not make sense with triangular half-squares: any diagonal edge would have degree 1.)

In addition, we are given a constraint on the number of layers throughout the surface. In general, we allow each unit square and triangular half-square f to have its own target number k_f of layers in a valid folding, where $k : f \mapsto k_f$ is a function. In particular, we consider the case where k is a constant function such as $k = 2$ or $k = 4$, i.e., we want a uniform-layer folding.

Tables 1 and 2 summarize our results for polyomino and polyabolo shapes, respectively (forbidding and allowing triangular half-squares, respectively). Each

k	Polyomino	Polycube	Even-degree complex	Manifold	Complex
$k = 1$	Only strips (Thm 6.2)	Nothing (Thm 6.3)	NP-hard (Thm 7.2)	Linear time (Thm 5.1)	NP-hard (Thm 7.1)
$k = 2$	Only strips (Thm 6.4)	Everything (Thm 3.1)		Linear time (Thm 4.1)	NP-hard (Thm 7.3)
$k_f \equiv 1 \pmod 2, k_f \geq 3$	Only strips (Thm 6.2)	Nothing (Thm 6.3)	Polynomial time (Thm 5.2)		
$k_f \equiv 0 \pmod 2, k_f \geq 4$	Everything (Thm 3.1)				
$k_f \geq 3$	Polynomial time (Thm 5.2)				
$k_f \in \{2, 4\}$	NP-hard (Thm 7.4)	Everything (Thm 3.1)		NP-hard (Thm 7.4)	

Table 1: Summary of our results for polyomino shapes (made from unit squares) with a specified number k of layers (possibly varying as k_f for each unit-square face f).

k	Polyabolo	Manifold	Complex
$k = 1$	Only strips (Thm 6.1, 6.2)	Linear time (Thm 5.1)	NP-hard (Thm 7.1)
$k = 2$	Linear time (Thm 4.1)		NP-hard (Thm 7.3)
$k_f \equiv 1 \pmod 2, k_f \geq 3$	Only strips (Thm 6.1, 6.2)	Polynomial time (Thm 5.2)	
$k_f \equiv 0 \pmod 4, k_f \geq 4$	Everything (Thm 3.2)		
$k_f \equiv 2 \pmod 4, k_f \geq 6$	Polynomial time (Thm 5.2)		
$k_f \geq 3$	Polynomial time (Thm 5.2)		
$k_f \in \{2, 4\}$	NP-hard (Thm 7.4)		

Table 2: Summary of our results for polyabolo shapes (made from unit squares and right-isosceles triangular half-squares) with a specified number k of layers (possibly varying as k_f for each face f). Throughout, we assume that k_f is even for all triangular half-squares; otherwise, Theorem 6.1 shows that the shape is impossible to fold.

column corresponds to a class of target shapes as categorized above, and each row corresponds to a restriction on the number of layers. In each case, we prove one of the following types of results:

- **Universality / everything:** Every such shape can be folded with the desired number of layers, and there is a linear-time algorithm to find such a folding. In Section 3, we prove the following such results:
 - Universality when the number k_f of layers for each (half-)square f is at least 4 and either even (for polyomino complexes) or 0 modulo 4 (for polyabolo complexes).
 - Universality when every $k_f \in \{2, 4\}$ for even-degree polyomino complexes (e.g., polycubes or polyomino manifolds without boundary).

In particular, these results establish that the grid + diagonals model can

universally fold all polyabolo complexes with exactly $k = 4$ layers everywhere.

- **Polynomial-time characterization:** A polynomial-time algorithm decides which shapes can be folded with the desired number of layers. Such results can be further categorized as follows:
 - **Nothing** can be folded, i.e., the algorithm’s answer is always “no”. In Section 6, we prove such a result for polycubes where all k_f are odd, and if there is any triangular half-square whose k_f is odd.
 - **Only strips** (unit-width rectangles) can be folded, so the algorithm can simply check for such a shape. In Section 6, we prove such results for polyominoes where all k_f are odd and for uniform coverage by $k = 2$ layers. (These results show that none of our polynomial-time characterizations could be further extended to universality.)
 - **Linear time** is possible. In Sections 4 and 5, we prove such results for polyabolo manifolds with $k = 2$ and $k = 1$ respectively. The $k = 2$ algorithm is quite complicated and one of our main results.
 - Larger **polynomial time** results. In Section 5, we prove such results when all k_f are at least 3, and for certain classes of k_f s modulo 4.
- **NP-hardness:** It is NP-hard to decide which shapes can be folded with the desired number of layers. In Section 7, we prove several such results:
 - For uniform coverage with $k = 1$ or $k = 2$ layers, we prove NP-hardness for polyomino complexes. The $k = 1$ result holds even when restricted to even-degree polyomino complexes, in surprising contrast to $k = 2$ where we obtain universality. Both of these results contrast our linear-time algorithms when further restricted to manifolds.
 - For varying coverage with $k_f \in \{2, 4\}$ layers, we prove NP-hardness for polyominoes. This result contrasts our positive results for $k = 2$ and $k = 4$, and contrasts our universality result when we change the shapes to even-degree polyomino complexes (e.g., polycubes).

Related work. Origami design with a uniform number of layers throughout the folding was considered by [Davis et al. 14]. They show that, for any polygon that tiles the plane using only finitely many orientations, it is possible to fold multiple pieces of paper in that shape into a “locked weaving” with a fixed constant number of layers at every point. The constant they achieve depends on the paper shape (in particular, they consider folding rectangles of varying aspect ratios) and on whether the goal is to make a finite-area weaving (with no particular shape) or an infinite plane, but the main problem they address is *which shapes of paper admit a uniform-layer weaving*.

By contrast, our work assumes a *single* piece of paper in the *simple shape* of a unit-width rectangle, and the folding must be in a *specific target shape* (not just a specified number of layers). In some sense, [Davis et al. 14] consider the dual problem, where the piece of paper is in a given shape, while the target has unspecified shape (only a specified number of layers). (We also do not consider the

“locked” constraint from [Davis et al. 14], which attempts to model that the folded model does not “fall apart”).

2 Tiling Model

Given a polyabolo complex with grid-aligned unit square and triangular half-square faces F , and a *layer assignment* $k : F \rightarrow \mathbb{Z}^+$, we characterize when a strip of unit width can be folded in the grid + diagonals model such that every face $f \in F$ is covered with exactly $k(f) = k_f \in \mathbb{Z}^+$ layers. In particular, we assume that the target number of layers is positive, and that the faces are joined along only unit-length edges. Consequently, the only way to cover a half-square face is by making a diagonal fold, resulting in an even number of layers on half-squares.

We fix a local intrinsic orientation of each face of the complex. In particular, we assume all squares are oriented like \blacksquare (axis-aligned) and all half-squares are oriented like \blacktriangle . These orientations let us talk about, e.g., horizontal and vertical edges and/or neighbors of a particular tile but, except in the 2D cases of polyominoes or polyaboloes, these notions may differ from tile to tile.

2.1 Tiles

A *square tile* is a unit square with a multigraph on five labeled vertices: one central vertex for the face, and one additional vertex for each edge. A *half-square tile* is a half-square with a multigraph on three labeled vertices: one central vertex for the face, and one additional vertex for each unit edge (but nothing for the diagonal edge). We call the vertices *face-vertices* and *edge-vertices* accordingly. In both cases, we require the multigraph to be bipartite, allowing edges only between the face-vertex and the edge-vertices. See Figure 4 for some simple examples. Square tiles that differ by rotations are considered different.

A *tiling* of a complex C is a function τ mapping each face of C to a tile of the same shape (square or half-square). By gluing together the multigraphs of each tile (according to its local orientation) at the edge-vertices, a tiling τ induces a bipartite multigraph G_τ on the entire complex, with a face-vertex for each face of the complex and an edge-vertex for each unit edge.

In the rest of this section, we will show that a complex C with layer assignment k has a strip folding if and only if it has a tiling that satisfies certain conditions.

2.2 Complete Tiling

Call a tile *valid* if it is an edge-disjoint union of one or more *subtiles* shown in Figure 4 (of the same shape), which we denote in the text by \blacksquare , \square , and \blacktriangle , respectively. For example, the valid tile \boxplus results from the disjoint union of two copies of \blacksquare and one copy of \square . Equivalently, a valid square tile has equal degree at opposite edge-vertices, and a valid half-square tile has equal degree at the two edge-vertices.

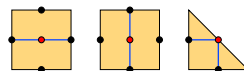


Figure 4: The three types of subtiles that form valid tiles. Face-vertices are colored red.

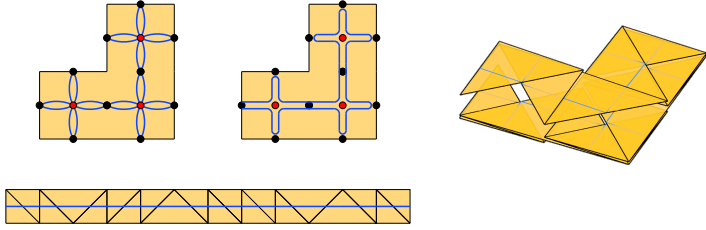


Figure 5: Folding a strip to place its midline along an Euler path of the multigraph of a tiling.

The *size* of a valid tile is the number of subtiles in the union, or equivalently, half the number of edges.

Definition 2.1. A tiling τ of a complex C with layer assignment k is **complete** if the following conditions hold:

1. For each square face f of C , the tile $\tau(f)$ is a valid square tile of size $t_f = k_f$.
2. For each half-square face f of C , the tile $\tau(f)$ is a valid half-square tile of size $t_f = k_f/2$. In particular, if k_f is odd for some half-square face, then a complete tiling cannot exist.
3. The multigraph G_τ has an Euler path. Equivalently,
 - all but zero or two edge-vertices have even degree; and
 - the graph is connected, except possibly for some isolated edge-vertices.

Intuitively, we show that a complex has a complete tiling if and only if it can be folded from a single strip by tracing the Euler path with the strip’s midline; see Figures 2 and 5. To only have grid and diagonal folds, the strip needs to start and end at edge-vertices; this can always be arranged because face-vertices have even degree. Going straight through a face-vertex corresponds to traversing a face without a fold. Making an L-turn (right angle) at a face-vertex corresponds to a diagonal fold. A problem arises if the Euler tour **U-turns** (doubles back) at a face-vertex, as it corresponds to the invalid half-grid fold from Figure 2. On the other hand, U-turns at edge-vertices are allowed: they correspond to folding by 180° along a perpendicular crease. Other perpendicular creases may be folded by 0 or 90° (or just abstractly) depending on the geometry of the complex.

To fix the problem of U-turns at face-vertices, we prove the following:

Lemma 2.2. For any complete tiling τ of a complex, G_τ has an Euler tour without U-turns at face-vertices, which can be found in linear time.

Proof sketch. We construct a modified multigraph $\tilde{G}_\tau = (\tilde{V}, \tilde{E})$ that is still Eulerian, such that any Eulerian path in \tilde{G}_τ can be converted into an Eulerian path in G_τ . Furthermore, we ensure that Eulerian paths in \tilde{G}_τ either go straight or make an L-turn at each face-vertex, forbidding U-turns. Refer to Figure 6. \square

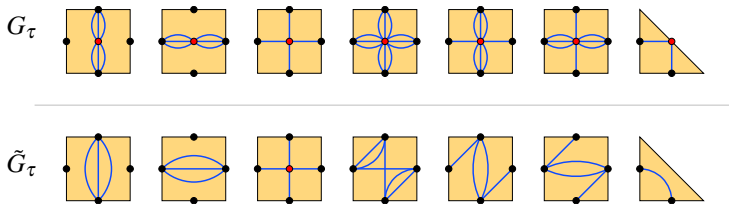


Figure 6: Conversion in the proof of Lemma 2.2 from G_τ (top) to \tilde{G}_τ (bottom).

Finally, we prove that we can trace an Euler path with no U-turns at face-vertices with the strip’s midline to obtain a strip folding of the complex with k_f layers on face f :

Lemma 2.3. *Given a complete tiling τ of a complex C with layer assignment k , and given an Euler path in G_τ without U-turns at face-vertices, we can construct in linear time a strip folding (whose midline traces the Euler path) that achieves the desired layer assignment k .*

Combining Lemmas 2.2 and 2.3, we obtain one direction of the desired result:

Theorem 2.4. *A complex C with layer assignment k can be folded from a strip if and only if (C, k) has a complete tiling. Either conversion can be done in linear time.*

A consequence of the argument in Lemma 2.3 in particular is the following characterization of the strip ends:

Corollary 2.5. *If the multigraph G_τ of a complete tiling τ has two odd-degree edge-vertices, then these two edge-vertices will be the locations of the two ends of the strip. If the multigraph G_τ has no odd-degree edge-vertices, then the strip starts and ends at the same location, which can be any non-isolated edge-vertex.*

2.3 Reduced Tiling

Next we identify a *finite* set of “reduced” tiles and show that the existence of a “reduced tiling” is equivalent to the existence of a complete tiling, and thus equivalent to the existence of a strip folding. Notably, all reduced tiles have constant size. This approach simplifies many of the arguments and is used throughout the paper.

Consider a complete tiling τ of a complex C with layer assignment k , and let f be a face of C . If tile $\tau(f)$ contains three copies of the same subtile, we can *remove* two of them: this change does not affect vertex degree parities, and it does not affect connectivity, so the graph G_τ remains Eulerian. Conversely, we can always *add* two copies of the same subtile: this change does not affect vertex degree parities, and it can only increase connectivity. Such changes will break the property that the size of the tile matches k_f , however; instead, it will match modulo 2.

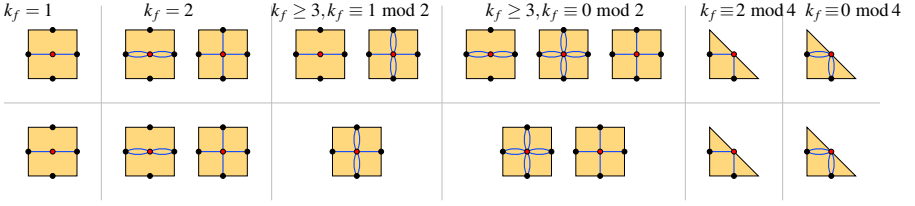


Figure 7: The reduced tiles T up to rotation. Top row: partial reduction from removing pairs of identical subtiles. Bottom row: full reduction from also adding pairs of identical subtiles.

By repeatedly removing pairs of identical subtiles, except the last pair, we can reduce each tile to one of the tiles shown in the top row of Figure 7 (ignoring rotations for the moment). We can further reduce the number of candidate tiles for each face by adding pairs of subtiles, resulting in the tiles shown in the bottom row of Figure 7 (again ignoring rotations). We call this set T of tiles the **reduced tiles**:

$$T = \{\text{[square with horizontal fold]}, \text{[square with vertical fold]}, \text{[square with diagonal fold]}, \text{[square with vertical fold and horizontal fold]}, \text{[square with vertical fold and diagonal fold]}, \text{[square with horizontal fold and diagonal fold]}, \text{[square with vertical fold and horizontal fold and diagonal fold]}, \text{[triangle]}, \text{[triangle]}\}.$$

Definition 2.6. A tiling τ of a complex C with layer assignment k is **reduced** if

1. Every tile $\tau(f)$ is reduced and its size matches k_f modulo 2. Equivalently:

$$\tau(f) \in \begin{cases} \{\text{[square with horizontal fold]}, \text{[square with vertical fold]}\} & \text{if } k_f = 1, \\ \{\text{[square with horizontal fold]}, \text{[square with vertical fold]}, \text{[square with diagonal fold]}\} & \text{if } k_f = 2, \\ \{\text{[square with horizontal fold]}, \text{[square with vertical fold]}\} & \text{if } k_f \geq 3 \text{ and } k_f \equiv 1 \pmod{2}, \\ \{\text{[square with horizontal fold]}, \text{[square with vertical fold]}\} & \text{if } k_f \geq 3 \text{ and } k_f \equiv 0 \pmod{2}, \end{cases} \quad \text{for a square face } f \text{ of } C;$$

$$\tau(f) = \begin{cases} \text{[triangle]} & \text{if } k_f \equiv 2 \pmod{4}, \\ \text{[triangle]} & \text{if } k_f \equiv 0 \pmod{4}, \end{cases} \quad \text{for a half-square face } f \text{ of } C.$$

2. The multigraph G_τ has an Euler path. Equivalently,

- all but zero or two edge-vertices have even degree; and
- the graph is connected, except possibly for some isolated edge-vertices.

For polyabolo manifolds, the even-degree condition can be restated as a modulo-2 form of tile edge matching:

- For all but zero or two boundary edges of the manifold, the corresponding tile edge has even parity; and
- Any two tile edges meeting at a nonboundary edge of the manifold have equal parity.

Corollary 2.7. A complex C with layer assignment k can be folded from a strip if and only if (C, k) has a reduced tiling. The conversion can be done in linear time.

3 Universality Results

In this section, we present some corollaries of the tiling model for specific shapes, which are based on prescribing a tiling of the underlying shape. The presented tilings are trivially connected and all degrees are even. With Corollary 2.7, the corresponding strip folding can be computed in linear time. The following two theorems imply the universality results in Table 1 and Table 2.

Theorem 3.1. *Every even-degree polyomino complex with layer assignment k where every $k_f \equiv 0 \pmod{2}$ has a strip covering.*

Theorem 3.1 implies every polycube can be covered with an even number of layers.

Theorem 3.2. *Every polyabolo complex with layer assignment k where every $k_f \geq 4$ and $k_f \equiv 0 \pmod{2}$ on squares and $k_f \equiv 0 \pmod{4}$ on half-squares has a strip covering.*

Theorem 3.2 implies that every polyomino, polyomino manifold, polyabolo, and polyabolo manifold can be covered with an appropriate layer assignment.

4 2-Layer Algorithm

In this section, we develop a linear-time “ray-shooting” algorithm for covering polyabolo manifolds with exactly $k = 2$ layers.

Theorem 4.1. *There is a linear-time algorithm for determining whether a polyabolo manifold has a two-layer strip folding, and if so, finding such a folding.*

By Corollary 2.5, this problem is equivalent to finding a complete tiling of the manifold using only tiles $\{\square, \blacksquare, \blacktriangle, \blacktriangleright\}$. The challenge is to assign tiles to the squares such that (1) the resulting graph G_τ is connected, and (2) there are at most two parity violations in G_τ .

Given a valid tiling τ of a polyabolo complex C , define the multigraph H_τ as follows: the vertices of H_τ are the edge-vertices of G_τ , and there is an edge between two vertices for every subtile of τ that connects those edge-vertices. A **parity violation** is an edge-vertex of odd degree. This corresponds either to a unit-length boundary edge of the manifold with an incident \blacksquare or \blacktriangle (odd) tile, or a unit-length nonboundary edge with two mismatched incident tiles: \square or \blacksquare (even) on one side, and \blacksquare or \blacktriangle (odd) on the other. Graphs H_τ and G_τ have the same set of parity violations, because all face-vertices in G_τ have even degree.

Consider the unique tiling μ which uses only the tiles \blacksquare and \blacktriangle . A **ray** is a connected component of H_μ . Each ray is either a path or a cycle. Intuitively, a ray is like a bidirectional beam of light that passes through square faces and reflects off the diagonal edges of half-square faces. It continues in both directions until they either meet each other (if it is a cycle) or meet boundaries (if it is a path). The **multiplicity belt** of a ray r in τ is the sequence of multiplicities assigned by τ to each edge of r . Here we use “belt” to mean a sequence which may be either path-shaped or cycle-shaped.





Definition 4.2. A belt consisting of integers in $\{0, 1, 2\}$ is **well-formed** if all of the nonzero elements are contiguous, and all instances of “1” are contiguous.

4.1 Layer Constraint Problem

A **layer constraint belt** is a belt consisting of numbers in $\{1, 2\}$ and occurrences of two possible types of variables. A **group variable** is denoted x_i and can occur any number of times in the belt. **Link variables** come in pairs (y_i, y_j) , where each occurs exactly once in the belt.

An assignment of values $\{0, 1, 2\}$ to the variables of a layer constraint belt ϕ is a **satisfying assignment** if the following are true:

- $x_i \neq 0$ for each group variable x_i ;
- $y_i + y_j = 2$ for each pair (y_i, y_j) of link variables; and
- substituting the assigned values for the variables in ϕ makes a well-formed belt.

The layer constraint problem is to determine, given a layer constraint belt, whether a satisfying assignment exists. In the full paper, we show that complete $k = 2$ tiling reduces to this problem: there is a certain ray such that 0 corresponds to a  tile perpendicular to the ray, 1 corresponds to a  or  tile, and 2 corresponds to a  tile parallel to the ray.

4.2 Solving the Layer Constraint Problem

We now give a linear-time algorithm for solving the layer constraint problem, starting with the case of a path-shaped layer constraint belt of length n .

A well-formed string s is determined by the natural numbers $b_{02}, b_{21}, b_{12}, b_{20}$ such that s consists of b_{02} 0s, followed by $b_{21} - b_{02}$ 2s, then $b_{12} - b_{21}$ 1s, then $b_{20} - b_{12}$ 2s, and finally $n - b_{20}$ 0s. These numbers, which are not always unique, can be thought of as the locations of the boundaries between contiguous sections of s . For clarity, we use integers for the locations of boundaries between characters in the string, and *half-integers* for the locations of the characters themselves.

Our algorithm will work by finding intervals $[l_{02}, r_{02}], [l_{21}, r_{21}], [l_{12}, r_{12}], [l_{20}, r_{20}]$, maintaining the invariant that $b_i \in [l_i, r_i]$ for every possible satisfying string. These intervals can only become smaller; if any of them become empty, then the algorithm reports that no satisfying assignment exists.

A basic operation used by the algorithm is assigning a value $v \in \{0, 1, 2\}$ to a location ℓ . The outcome of this operation depends on the previous contents of the layer constraint string at ℓ . If the contents were a value in $\{0, 1, 2\}$, then no more action is necessary unless the value is different from v , in which case no satisfying assignment exists. If there was a group variable x_i at ℓ , then the algorithm replaces all occurrences of x_i with v , unless $v = 0$ in which case no satisfying assignment exists. If there was a link variable y_i at ℓ , then the algorithm replaces y_i with v and y_j with $2 - v$ where y_j is the other link variable in y_i 's pair. These replacements can be performed in amortized linear time by precomputing pointers to allow fast traversals of group and link variables.

After placing a value at a location ℓ , the algorithm updates intervals as follows.

- If ℓ is assigned 1, then we update r_{21} with the information that $b_{21} < \ell$. That is, we update r_{21} to $\min\{\ell - \frac{1}{2}, r_{21}\}$. Symmetrically, we also learn that $b_{21} > \ell$, i.e., we update l_{21} to $\max\{\ell + \frac{1}{2}, l_{21}\}$.
- If ℓ is assigned 2, then we learn that $b_{02} < \ell$ and $b_{20} > \ell$. Additionally, if we know that $b_{12} > \ell$ (i.e., $l_{12} > \ell$), then we learn that $b_{21} > \ell$; and symmetrically, if we know that $b_{21} < \ell$, then we learn that $b_{12} < \ell$.
- If ℓ is assigned 0, then if we know that $b_{20} > \ell$, we learn that $b_{02} > \ell$; and symmetrically, if we know that $b_{02} < \ell$, then we learn that $b_{20} < \ell$.

We also update intervals to be consistent with each other, using the fact that $b_{02} \leq b_{21} \leq b_{12} \leq b_{20}$. These updates all take constant time.

The algorithm works by repeatedly selecting one of several possible rules to apply. These rules are as follows.

- (i) Assign 0 to a location in $[0, l_{02}]$ or $[r_{20}, n]$.
- (ii) Assign 1 to a location in $[r_{21}, l_{12}]$.
- (iii) Assign 2 to a location in $[r_{02}, l_{21}]$ or $[r_{12}, l_{20}]$.
- (iv) If ℓ contains an occurrence of a group variable, then we learn that $b_{02} < \ell < b_{20}$.
- (v) If ℓ_i, ℓ_j contain a pair of link variables where $\ell_i < \ell_j$, then we learn that $b_{20} > \ell_i$ and $b_{02} < \ell_j$.
- (vi) If ℓ_i, ℓ_j contain a pair of link variables, and we know that $\ell_i, \ell_j \in [b_{02}, b_{20}]$, then assign 1 to both ℓ_i and ℓ_j .
- (vii) If $\ell_i < \ell_j$ contain a pair of link variables, and we know both $\ell_i < b_{21}$ and $b_{02} < \ell_j < b_{20}$, then assign 0 to ℓ_i and assign 2 to ℓ_j .

By tracking which intervals of locations have been scanned for possible rule applications, we can select and apply a rule in amortized constant time. There are at most a linear number of rule applications overall, so the algorithm takes linear time to make as many rule applications as possible.

We can now describe the algorithm in full. For path-shaped layer constraint belts, the algorithm is as follows.

- Initialize the four intervals to the entire range $[0, n]$.
- Scan the constraint belt for a location ℓ containing a 2; if no such location exists then return the trivial all-1s satisfying assignment.
- Nondeterministically guess whether $\ell < b_{21}$ or $\ell < b_{12}$, and update the intervals according to the guess.
- Make as many rule applications as possible.
- Construct a well-formed string s according to the following two cases:
 - If $r_{21} < l_{12}$, then s consists of 0 in the range $[0, l_{02}]$; 2 in the range $[l_{02}, r_{21}]$; 1 in the range $[r_{21}, l_{12}]$; 2 in the range $[l_{12}, l_{20}]$; and 0 in the range $[l_{20}, n]$.
 - Otherwise, s consists of 0 in the range $[0, l_{02}]$; 2 in the range $[l_{02}, l_{20}]$; and 0 in the range $[l_{20}, n]$.
- Return the satisfying assignment corresponding to s .

For cycle-shaped layer constraint belts, we reduce to the path case. The algorithm scans the belt for a 2; if none exists, then it returns the trivial all-1s satisfying assignment. Let a, b be the locations of the two link variables closest to our 2 on either side; if there are no link variables, then the problem is easy. The algorithm guesses the assignments of values to both link variables; if either of these guessed values is not 1, then it locates a 0 and reduces to the path-shaped case. Otherwise, because the 1s must be contiguous, we know that every location separated from the 2 by a and b must be a 1. This eliminates all link variables, and the resulting problem is easy.

5 One- and Many-Layer Algorithms

Single-layer covering is easy because turning creates at least two layers.

Theorem 5.1. *We can decide in linear time whether a given polyabolo manifold has a 1-layer covering.*

For three or more layers, it is easy to ensure connectivity and thus the problem essentially reduces to checking parity constraints across possible tilings. This is in contrast to the NP-hardness results in Section 7.

Theorem 5.2. *If all square faces f of a polyomino complex have layer requirements $k_f \geq 3$, and all triangle faces have even k_f , then there is an $O(n^{4.2131})$ -time algorithm to decide whether the polyomino complex has a strip covering. For the special case of polyomino manifolds, there is an $O(n^3)$ -time algorithm.*

6 Impossibility Results

Theorem 6.1. *If any triangle on the target shape has odd k_f , no strip covering exists.*

Theorem 6.2. *For layer assignments with all odd k_f , only polyominoes of type $1 \times m$ rectangles have a strip k -covering.*

Theorem 6.3. *No polycubes can be folded with an all-odd layer assignment k_f .*

Theorem 6.4. *For polyominoes, only strips have a 2-layer covering.*

7 Hardness Results

In this section, we show several NP-hardness results. All of these problems are in NP, because the crease locations and fold angles all come from a polynomial-size discrete set.

7.1 1-Layer Polyomino Complexes are NP-Hard

Theorem 7.1. *Deciding whether a polyomino complex can be exactly covered by a strip of paper is NP-hard.*

We reduce from positive E1-in-3-SAT [Schaefer 78]. Figure 8 shows the variable gadget and Figure 9 shows the clause gadget. Colored lines indicate glued edges; when multiple lines of the same color intersect, this represents multiple edges being joined at the same place in the complex. These self-joined faces mean the complex is not embeddable. Figure 10 shows how to attach extra cells to an edge in the prior reduction, strengthening the result to even-degree complexes.

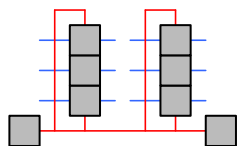


Figure 8: Start square followed by two variable gadgets.

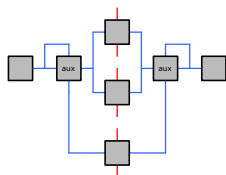


Figure 9: Clause gadget.

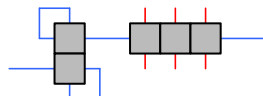


Figure 10: Extra edge gadget.

Theorem 7.2. *Deciding whether an even-degree polyomino complex can be exactly covered by a strip of paper is NP-hard.*

7.2 2-Layer Polyomino Complexes are NP-Hard

Theorem 7.3. *Deciding whether a polyomino complex can be double covered by a strip of paper is NP-hard.*

We reduce from Constraint Logic Satisfaction [Hearn and Demaine 09]. Vertex gadgets are shown in Figures 11 and 12. Edge gadgets are each a single cell shared by two vertex gadgets. Again this theorem only applies to abstract complexes.

7.3 $k_f \in \{2, 4\}$ Polyominoes are NP-Hard

When $k = 2$, we have a polynomial-time algorithm. When $k = 4$, we can make everything. What if some cells have to be covered 2 times and others have to be covered 4 times? We show this to be NP-complete by a reduction from Planar Monotone Rectilinear 3SAT-3 [Darmann et al. 18].

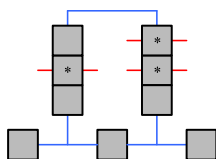


Figure 11: AND gadget. Cells with * are edge gadgets and are shared by two vertex gadgets.

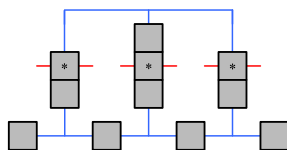







Figure 12: OR gadget. Cells with * are edge gadgets and are shared by two vertex gadgets.

Figure 13 shows a variable gadget. Blue lines show the orientation of forced double lines in a tile, specifically , , or . Squares marked with a 4 require 4 layers; all others require 2 layers. Assigning the * squares to be either all  or all  corresponds to setting the variable TRUE or FALSE. Variables are connected along the middle horizontal lines. Clauses attach vertically and are paths of tiles connecting the variables. A clause must be connected through at least one variable gadget, which occurs if the * tiles are aligned to connect the corresponding top or bottom section of the variable gadget.

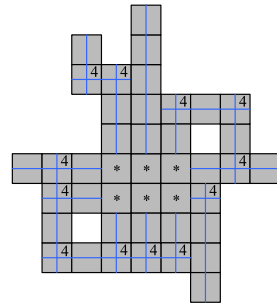


Figure 13: Variable gadget with two positive and one negated literal.

Theorem 7.4. *Deciding whether a polyomino can be exactly covered when each cell is specified to need to be covered exactly 2 or exactly 4 times is NP-hard.*

8 Open Problems

- Is single/double covering polyomino complexes still hard when those complexes are required to be embeddable in \mathbb{R}^3 , ideally grid-aligned?
- What is the complexity when the required number of layers satisfies $k_f \in \{1, 2\}$?
- Can we improve the running time of our polynomial-time algorithms?
- What if each face f has a desired *range* $[a_f, b_f]$ for the number of layers?

Acknowledgments

This work grew out of an open problem session from the MIT class on Geometric Folding Algorithms: Linkages, Origami, Polyhedra (6.849) during Fall 2020.

References

- [Aichholzer et al. 21] Oswin Aichholzer, Hugo A. Akitaya, Kenneth C. Cheung, Erik D. Demaine, Martin L. Demaine, Sándor P. Fekete, Linda Kleist, Irina Kostitsyna, Maarten Löffler, Zuzana Masárová, Klara Mundilova, and Christiane Schmidt. “Folding Polyominoes with Holes into a Cube.” *Computational Geometry: Theory and Applications* 93 (2021), Article 101700.
- [Benbernou et al. 20] Nadia M. Benbernou, Erik D. Demaine, Martin L. Demaine, and Anna Lubiw. “Universal Hinge Patterns for Folding Strips Efficiently into Any Grid Polyhedron.” *Computational Geometry: Theory and Applications* 89 (2020), Article 101633.
- [Czajkowski et al. 20] Kingston Yao Czajkowski, Erik D. Demaine, Martin L. Demaine, Kim Epling, Robby Kraft, Klara Mundilova, and Levi Smith. “Folding Small Polyominoes into a Unit Cube.” In *Proceedings of the 32nd Canadian Conference in Computational Geometry*. Saskatchewan, Canada, 2020.

- [Darmann et al. 18] Andreas Darmann, Janosch Döcker, and Britta Dorn. “The Monotone Satisfiability Problem with Bounded Variable Appearances.” *International Journal of Foundations of Computer Science* 29:6 (2018), 979–993.
- [Davis et al. 14] Eli Davis, Erik D. Demaine, Martin L. Demaine, and Jennifer Ramseyer. “Weaving a Uniformly Thick Sheet from Rectangles.” In *Origami⁶: Proceedings of the 6th International Meeting on Origami in Science, Mathematics and Education (OSME 2014)*, pp. 177–188. Tokyo, Japan: American Mathematical Society, 2014.
- [Demaine and O’Rourke 07] Erik D. Demaine and Joseph O’Rourke. *Geometric Folding Algorithms: Linkages, Origami, Polyhedra*. Cambridge University Press, 2007.
- [Demaine and Tachi 17] Erik D. Demaine and Tomohiro Tachi. “Origamizer: A Practical Algorithm for Folding Any Polyhedron.” In *Proceedings of the 33rd International Symposium on Computational Geometry*, pp. 34:1–34:15. Brisbane, Australia, 2017.
- [Demaine et al. 00] Erik D. Demaine, Martin L. Demaine, and Joseph S. B. Mitchell. “Folding Flat Silhouettes and Wrapping Polyhedral Packages: New Results in Computational Origami.” *Computational Geometry: Theory and Applications* 16:1 (2000), 3–21. Originally in SoCG 1999.
- [Hearn and Demaine 09] Robert A. Hearn and Erik D. Demaine. *Games, Puzzles, and Computation*. CRC Press, 2009.
- [Ku and Demaine 16] Jason S. Ku and Erik D. Demaine. “Folding Flat Crease Patterns With Thick Materials.” *Journal of Mechanisms and Robotics* 8:3 (2016), 031003–1–6.
- [Schaefer 78] Thomas J Schaefer. “The complexity of satisfiability problems.” In *Proceedings of the 10th Annual ACM Symposium on Theory of Computing*, pp. 216–226, 1978.

Lily Chung

Massachusetts Institute of Technology, USA, e-mail: lkdc@mit.edu

Erik D. Demaine

Massachusetts Institute of Technology, USA, e-mail: edemaine@mit.edu

Martin L. Demaine

Massachusetts Institute of Technology, USA, e-mail: mdemaine@mit.edu

Jenny Diomidova

Massachusetts Institute of Technology, USA, e-mail: diomidova@mit.edu

Jayson Lynch

Massachusetts Institute of Technology, USA, e-mail: jaysonl@mit.edu

Klara Mundilova

Massachusetts Institute of Technology, USA, e-mail: kmundil@mit.edu

Hanyu Alice Zhang

Cornell University, USA, e-mail: [hz496@cornell.edu](mailto:haz496@cornell.edu)