Flat Folding an Unassigned Single-Vertex Complex (Combinatorially Embedded Planar Graph with Specified Edge Lengths) Without Flat Angles

Lily Chung ✉ 🄳
Massachusetts Institute of Technology, Cambridge, MA, USA

Erik D. Demaine ✉ 🄳
Massachusetts Institute of Technology, Cambridge, MA, USA

Dylan Hendrickson ✉ 🄳
Massachusetts Institute of Technology, Cambridge, MA, USA

Victor Luo ✉
Massachusetts Institute of Technology, Cambridge, MA, USA

──── **Abstract** ────

A foundational result in origami mathematics is Kawasaki and Justin's simple, efficient characterization of flat foldability for unassigned single-vertex crease patterns (where each crease can fold mountain or valley) on flat material. This result was later generalized to cones of material, where the angles glued at the single vertex may not sum to $360°$. Here we generalize these results to when the material forms a *complex* (instead of a manifold), and thus the angles are glued at the single vertex in the structure of an arbitrary planar graph (instead of a cycle). Like the earlier characterizations, we require all creases to fold mountain or valley, not remain unfolded flat; otherwise, the problem is known to be NP-complete (weakly for flat material and strongly for complexes). Equivalently, we efficiently characterize which combinatorially embedded planar graphs with prescribed edge lengths can fold flat, when all angles must be mountain or valley (not unfolded flat). Our algorithm runs in $O(n \log^3 n)$ time, improving on the previous best algorithm of $O(n^2 \log n)$.

1 Introduction

The graph flat folding problem asks whether a given combinatorially embedded planar graph with prescribed edge lengths can be "folded flat" onto a line. More precisely, a ***flat folding*** is an assignment of x coordinates to vertices that respects the edge lengths, together with a partial order on the edges (which defines the stacking order among edges with overlapping x extents) that respects the combinatorial planar embedding and avoids crossings (edges penetrating connections between higher and lower edge endpoints, and improperly nested edge endpoint connections) [2, 9, 11].[1] Equivalently, a flat folding is a sequence or continuum

---

[1] In [2], flat foldings are called "linear folded states". Here we use "flat foldings" so that they match up with the corresponding notions in computational origami.
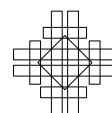
of planar embeddings that respect the combinatorial planar embedding, avoid crossings, and converge to the correct edge lengths and to lying on a line [1, 3]. Figure 1 shows an example of a flat folding, as well as some non-examples.
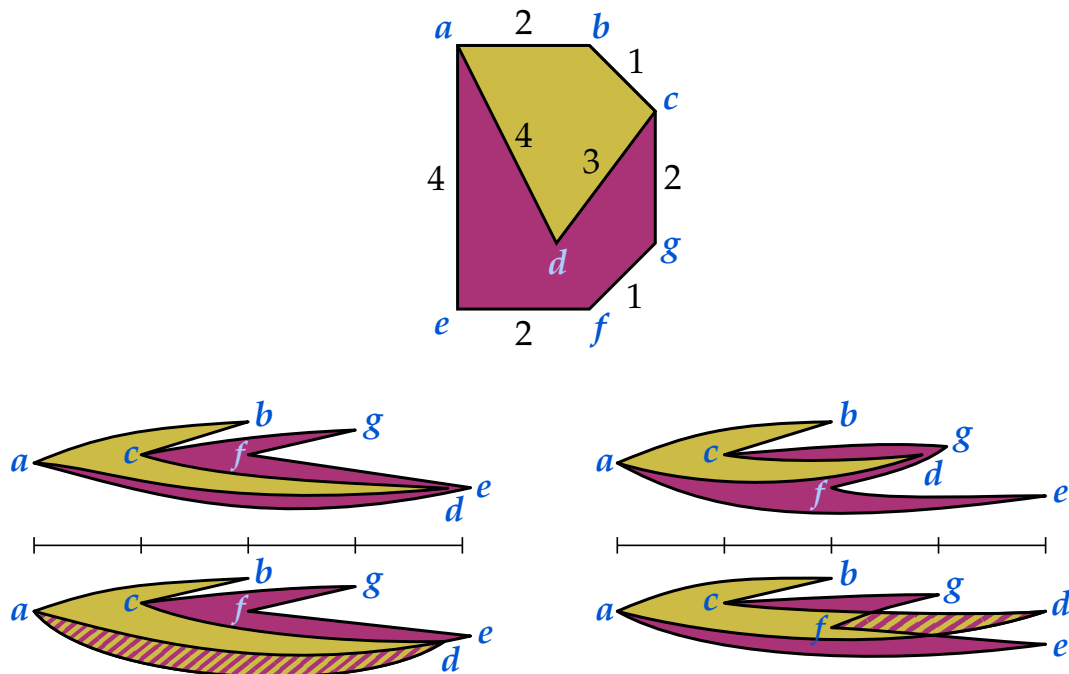
Figure 1 Four attempts to fold a graph with assigned edge lengths. Top left: A valid folding of the graph. Top right: Invalid because the lengths of edges *ad* and *cd* do not correspond to the original edge lengths. Bottom left: Invalid for two reasons: the cyclic ordering of edges at vertex *a* is not respected, and the folding exhibits incorrect layering at vertices *d* and *e*. Bottom right: Invalid since edges cross over each other.

It is known that the graph flat folding problem is strongly NP-complete in general, and solvable in linear time if all edge lengths are equal [2]. But there are two natural variations on the problem, posed in the same paper [2]. In any flat folding, we can identify the angles between consecutive edges around a vertex (as determined by the combinatorial planar embedding) as either **valley** (0°), **mountain** (360°), or **unfolded/flat** (180°). (At each vertex, these angles must sum to 360°, so there is either one mountain or two flats, and the rest are valleys.) Now we can vary two aspects of the problem:

1. What if we are also given the angle (valley/mountain/flat) between every consecutive pair of edges around each vertex?
2. What if we forbid flat angles, and instead require just valleys and mountains?

These parameters define four versions of the problem, as summarized in Table 1. The original paper [2] proved NP-completeness of the version with no angles given and allowing flat angles. Recent work shows that, if the angles are given, the problem becomes solvable in linear time (independent of whether flat angles are allowed) [3]. The remaining problem, studied here, is the version where the angles are not given, but flat angles are forbidden.

Connection to weak embeddings of graphs.   Although not stated explicitly, this no-flat-angles graph flat folding problem can be solved in polynomial time by a reduction to "weak embeddings of graphs". A key feature of this version of graph folding (in particular

■ **Table 1** Complexity of different models of graph flat folding (based on [3, Table 1], which in turn is based on open problems from [2]). Our new result is in the bottom-left.

	Flat angles forbidden	Flat angles allowed
Angles given	Linear time [3]	
Angles unspecified	$O(n^2 \log n)$ [4] $\rightarrow$ $\boldsymbol{O(n \log^3 n)}$ [new]	NP-complete [2]

distinguishing it from the NP-complete version with unknown angles that can be flat) is that the relative coordinates of the vertices are determined by the input: fixing one edge to go right from the origin, any path in the graph alternates between going right and left by the specified edge lengths, so a depth-first search fixes the vertex coordinates (and checks geometric closure constraints on cycles in the graph). The graph flat folding problem is then equivalent to asking whether this mapping from vertices to coordinates is a ***weak embedding*** of the graph, meaning that the vertices can be perturbed in the plane within ε-radius disks (for any $\varepsilon > 0$), and the edges can be similarly perturbed to Jordan curves within distance ε of the corresponding line segments, so that we obtain a strict embedding (no intersections except as intended at shared vertices). Recognizing weak embeddings was recently solved in $O(n^2 \log n)$ time [4],[2] so the same result applies to no-flat-angles graph flat folding.

Our results. In this paper, we give a faster algorithm for the no-flat-angles graph flat folding problem. Specifically, we show how to determine whether a graph can be folded flat without flat angles in $O(n \log^3 n)$ time, which is tight up to logarithmic factors.

We extend this result to the case where some angles are specified as flat, and the problem asks to determine mountain or valley for each of the remaining angles. (The same extension also follows from the reduction to weak embedding.) Thus what makes graph flat folding hard is not the existence of flat angles, but deciding which angles are flat.

Application to single-vertex origami. The version of the graph flat folding problem we study is particularly natural when viewed from the lens of computational origami.

Define a ***single-vertex complex*** to consist of m polygons in 3D where the polygons all share a common vertex v, and all the shared edges between these polygons are incident to v, as in Figure 2 (left). If we intersect such a single-vertex complex with a small sphere centered at v, we obtain a planar graph embedded on the sphere, whose m edge lengths are proportional to the m polygon angles at v. In the example of Figure 2 (left), we obtain the planar graph in Figure 1 (top). A flat folding of the single-vertex complex into the plane (according to standard origami definitions [11]) corresponds to a flat folding of the combinatorially embedded planar graph with prescribed edge lengths [2, 3]. Figure 2 (right) shows such a flat folding, corresponding to the graph flat folding in Figure 1 (top left). We can similarly consider the case of a single-vertex abstract complex – that is, an abstract metric space (not embedded in 3D) formed by gluing planar polygons along edges, which all share a common vertex – together with the cyclic ordering of polygons around each shared edge. Intersecting a single-vertex abstract complex with a small intrinsic sphere centered at the shared vertex produces a graph flat folding problem, and we can construct an arbitrary combinatorially embedded planar graph with prescribed edge lengths by a suitable

---

[2] The same paper [4] develops an $O(n \log n)$ algorithm for weak embedding of graphs, but only when the given map is "simplicial", meaning that edges do not pass through other vertices. This property does not hold in general in the graph flat folding problem.
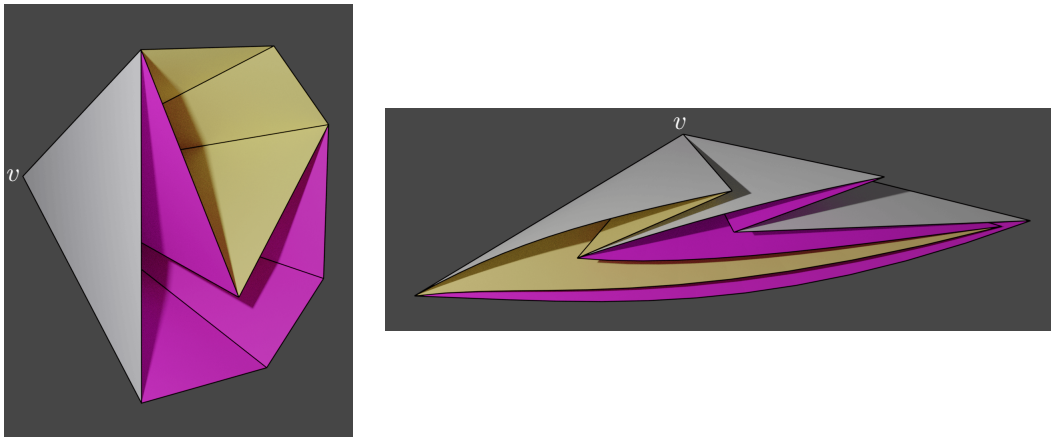
Figure 2 Unfolded and folded states of the single-vertex complex corresponding to the planar graph in Figure 1.

single-vertex abstract complex. Indeed, we can construct a multigraph in this way, so we generally allow graphs with multiple edges between the same two vertices. Therefore graph flat folding is equivalent to origami flat foldability of single-vertex (abstract) complexes.

When the planar graph is a cycle corresponding to 360° of total angle of polygons glued at a single vertex, we obtain what is known as a ***single-vertex crease pattern*** [11, Section 12.2]: creases emanating from a single vertex on a piece of paper. At the first OSME (Origami Science/Mathematics/Education) conference in 1989, Justin [16] and Kawasaki [17] presented characterizations of which single-vertex crease patterns fold flat: exactly those whose alternating sum of angles is zero. (A complete proof of this characterization was not published until Hull's 1994 paper [13]; see [15, Section 5.9].) Crucially, this linear-time characterization assumes that all creases must be folded either mountain or valley (none can be left unfolded flat at an angle of 180°); otherwise, single-vertex flat foldability becomes weakly NP-complete [10].

We see a similar behavior in Table 1 (bottom row), where allowing mountain, valley, and flat angles makes the problem NP-complete (even strongly), while our result shows that allowing just mountain or valley makes the problem solvable in near-linear time. Thus our result can be seen as a generalization of the Justin–Kawasaki Theorem from flat paper to complexes with similar running time. Previously, the theorem was generalized to cones of paper, where the angles sum to a value other than 360° [11, Section 12.2.1], but ours is the first generalization from manifolds to complexes with near-linear running time.

The top row of Table 1 corresponds to single-vertex ***mountain-valley patterns***, where each crease is marked as mountain or valley. (Some creases could be marked unfolded/flat, but this is equivalent to removing the crease.) The previous work on given-angle complexes [3] can similarly be seen as a generalization of the previously known linear-time characterization of single-vertex mountain-valley patterns [5], [11, Section 12.2.2].

Organization.   The rest of this paper is organized as follows. First we restate two needed previous results in Section 2. We then give a high-level overview of our algorithm in Section 3, and detail the various components of the algorithm in Sections 4, 5, 6, and 7.

2 Background

Our results rely on two previous results, which we restate here for completeness.

First, based on results of Hull [14], Demaine and O'Rourke [11] characterized the flat-foldable mountain/valley assignments of a cycle, which we will apply to each face in a connected combinatorially embedded planar graph:

▶ **Lemma 1** ([11, Corollary 12.2.12]). *Let f be a simple cycle with edge lengths $\theta_1, \ldots, \theta_n$. If the edge lengths are all equal, then a crease assignment on f is flat foldable in precisely the following cases:*

- Case A: *The cycle f is an interior face with equal-length edges, and there are exactly 2 more valley folds than mountain folds.*
- Case B: *The cycle f is an exterior face with equal-length edges, and there are exactly 2 more mountain folds than valley folds.*

Otherwise, take any maximal sequence $e_m, \ldots, e_{m+k-1}$ of k contiguous equal-length edges surrounded by strictly longer edges, so that[3]

$$\theta_{m-1} > \theta_m = \cdots = \theta_{m+k-1} < \theta_{m+k}$$

Then a crease assignment is flat foldable in precisely the following cases:

- Case C: *k is odd, and there are an equal number of mountain and valley folds incident to edges $e_m, \ldots, e_{m+k-1}$. Additionally, replacing all of the edges $e_{m-1}, \ldots, e_{m+k}$ with a single edge of length $\theta_{m-1} - \theta_m + \theta_{m+k}$ yields a flat-foldable face with the same crease assignment.*
- Case D: *k is even, and the numbers of mountain and valley folds incident to edges $e_m, \ldots, e_{m+k-1}$ differ by ± 1. Additionally, replacing all of the edges $e_m, \ldots, e_{m+k-1}$ with a single new vertex yields a flat-foldable face, where the crease assignment is the same except that it assigns the new vertex to be the same type as the majority of the folds incident to $e_m, \ldots, e_{m+k-1}$. (That is, the new vertex is a mountain fold in this assignment if the number of mountain folds was 1 greater than the number of valley folds.)*

Second, Abel et al. [3] proved that a flat folding of a graph is equivalent to a compatible folding of each face:[4]

▶ **Theorem 2** ([3, Theorem 2]). *Let G be a connected multigraph with an assignment of measures to every angle in G. That is, for each angle a we are given its measure $m_a \in \{0°, 180°, 360°\}$. Suppose that, for every face f, the restriction of this assignment to f yields a flat-foldable mountain-valley assignment when f is treated as a simple cycle. Suppose also that the assignment is **compatible** in that the sum of angles around each vertex is equal to 360°. Then there exists a flat folding of G whose angles have the assigned measures.*

Figure 3 shows an example of combining compatible flat foldings of individual faces to obtain a flat folding of the entire graph.

---

[3] The indices should be understood as being modulo n.

[4] A similar style of result ("faces being valid implies global validity") was obtained in the context of upward drawings of graphs [6, Theorem 3]. It also does not allow flat angles. That result, however, does not deal with prescribed edge lengths, which significantly complicates whether faces are flat foldable.
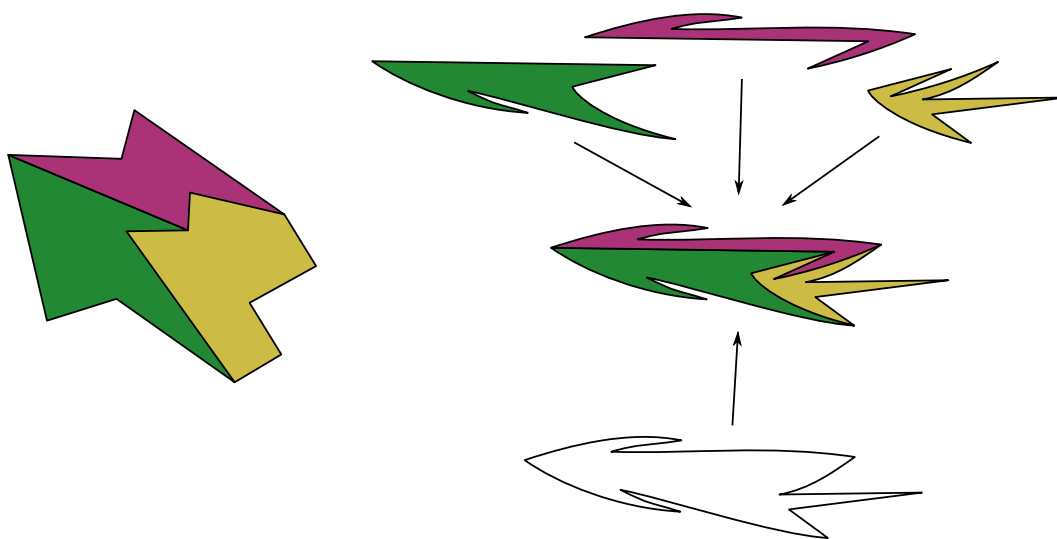
Figure 3 Left: A graph with three interior faces. Right: Given compatible flat foldings of all four faces, a flat folding of the graph can be generated.

3 Algorithm Overview

In this section, we provide a high-level outline of our algorithm for determining whether a connected combinatorially embedded planar (multi)graph with prescribed edge lengths can be folded flat. This algorithm takes as input a combinatorial embedding of the graph G (which we allow to have multiple edges between the same two vertices) and an assignment of lengths to the edges of G. We assume for now that the graph is connected and that we are given a single face of G designated as the *exterior*; in Section 7, we will remove both constraints.

By Theorem 2, determining whether such a graph has a flat folding is equivalent to determining whether there are compatible flat-foldable crease assignments for each face. To accomplish this, we reduce the graph flat folding problem to a boolean constraint satisfaction problem, with constraints deriving both from the requirement that each face needs to be flat foldable and from the compatibility requirement between faces. The variables will correspond to angles in each face of the graph (including some angles only present in virtual intermediate states), and indicate whether that angle is a valley fold or a mountain fold. The resulting constraint satisfaction problem has the following structure:

- Each clause specifies an exact number of true variables in some set.
- Each variable appears in exactly two clauses.
- The graph whose vertices are clauses and whose edges are variables, connecting the two clauses in which each variable appears, is bipartite. In particular, each vertex appears in exactly one clause on each side of the bipartition.
- The same graph is planar.

We call such a problem *planar bipartite positive ∗-in-∗SAT-E2*. This terminology generalizes the standard notion of "positive i-in-kSAT" [12, 18] where every clause requires satisfying exactly i out of (up to) k variables, which are never negated (hence "positive"), to the situation where number of variables and required true variables may vary from clause to clause. The standard suffix "-E2" represents the requirement that every variable appears in exactly two clauses [12, 8]. The "planar" prefix is also standard [12, 18], while the "bipartite" prefix is new (and makes sense only with the "-E2" requirement).

In Section 4, we describe the constraints which express that each face must be folded flat. In Section 5, we describe the constraints capturing compatibility between faces, and prove that the resulting constraint problem is equivalent to flat folding the graph. In Section 6, we show that planar bipartite positive $*$-IN-$*$SAT-E2 can be solved in $O(n \log^3 n)$ time through a reduction to a flow problem. In Section 7, we put the pieces together to obtain our main result, and describe three extensions: to graphs with some prescribed flat angles, to disconnected graphs, and to graphs with unknown exterior face.

4   Single Face Constraints

In this section, we describe the constraints obtained from the requirement that each face of the graph is folded flat. Although faces of the graph may not be bound by simple cycles (in the case of cut vertices), there exists a simple cycle corresponding to each face. This cycle can be constructed by enumerating the face's incident edges and angles in order, duplicating any repeated vertices or edges. Although there may exist flat foldings of this simple cycle which do not correspond to flat foldings of the original face, Theorem 2 tells us that compatible flat foldings of the corresponding simple cycles are sufficient for flat foldability of the full graph. From here on, when we discuss flat foldability of an individual face, we will actually be referring to flat foldability of the corresponding simple cycle.

Now consider flat folding a single face f. We check (and henceforth assume) that the edge lengths satisfy the basic closure property (mentioned in Section 1) that the number of edges is even and the alternating sum of edge lengths is zero; otherwise, flat folding is impossible. It remains to determine flat-foldable mountain/valley assignments.

We introduce a boolean variable x_a for each angle a in f. These variables represent an assignment of creases: $x_a = 0$ if a is a valley fold (0°) and $x_a = 1$ if a is a mountain fold (360°). We present an algorithm which, given the edge lengths and interior/exterior assignment of f and a variable x_a assigned to each angle a in f, generates a set of constraints C_f on the variables x_a, possibly introducing additional variables, such that solutions to this constraint problem correspond to flat-foldable crease assignments of f. The constraints are of the form "exactly c variables from a set S are true," which we write

$$\sum_{x \in S} x = c.$$

Additionally, each constraint generated will be colored either red or blue; this coloring will be used later to show that the constraint satisfaction problem is bipartite. The algorithm essentially follows Lemma 1:

- If all edges of f have equal length, then let V be the set of all angles in f, and let b be -1 if f is an interior face, or $+1$ if f is an exterior face. Generate just the red constraint

$$\sum_{a \in V} x_a = \frac{|V|}{2} + b. \tag{1}$$

- Otherwise, not all of the edges of f have equal length. Find a sequence $e_m, \ldots, e_{m+k-1}$ of k consecutive equal-length edges, such that $\theta_{m-1} > \theta_m = \cdots = \theta_{m+k-1} < \theta_{m+k}$; this is guaranteed to exist by considering a maximal sequence of consecutive edges with minimum length. Let S be the set of angles in f incident to $e_m, \ldots, e_{m+k-1}$.
- If k is odd (i.e., $|S|$ is even), generate the red constraint

$$\sum_{a \in S} x_a = \frac{|S|}{2}. \tag{2}$$

Having done so, replace all the edges $e_{m-1}, \ldots, e_{m+k}$ with a single edge of length $\theta_{m-1} - \theta_m + \theta_{m+k}$ to construct a smaller face f', and recursively output the constraints in $C_{f'}$.

- If instead k is even (i.e., $|S|$ is odd), introduce two fresh boolean variables y and z, and generate the following red and blue (respectively) constraints:

$$y + \sum_{a \in S} x_a = \frac{|S| + 1}{2}; \tag{3}$$

$$y + z = 1. \tag{4}$$

Then replace all the edges $e_m, \ldots, e_{m+k-1}$ with a single new angle whose associated variable is z to construct a smaller face f', and recursively output the constraints in $C_{f'}$.
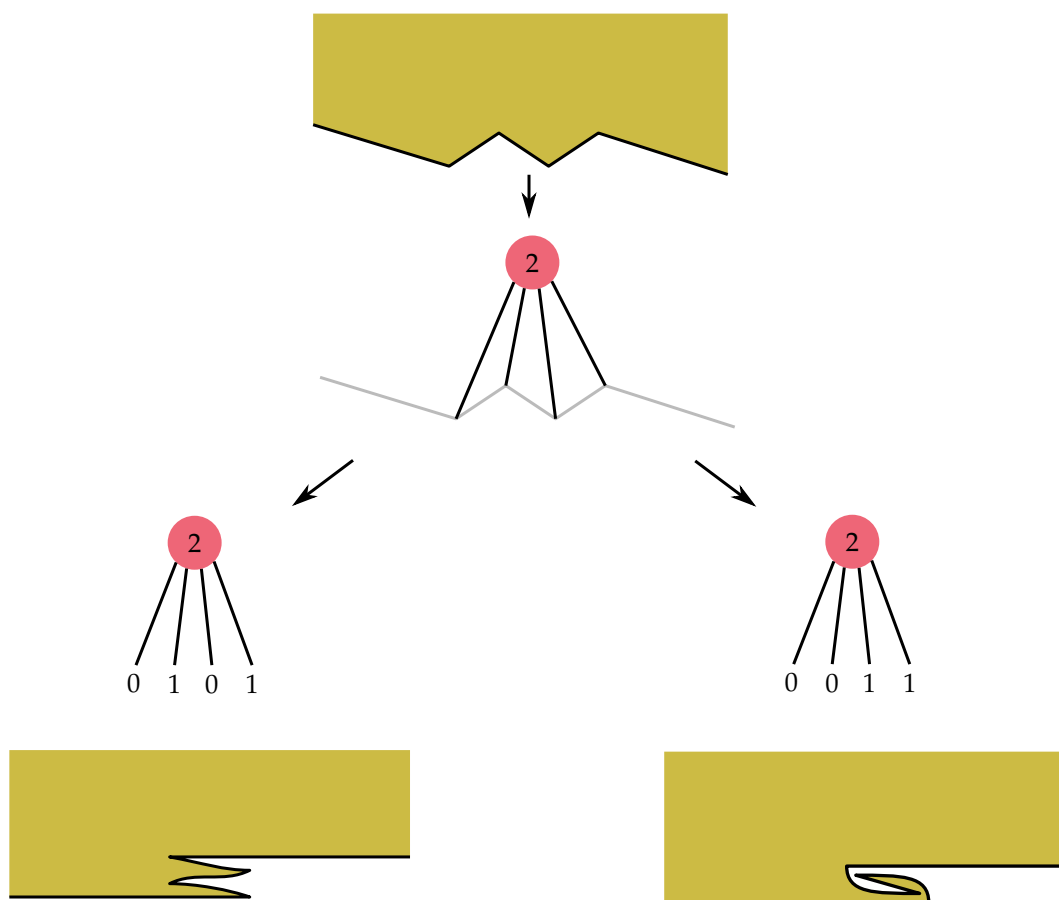

Figure 4 The case where k is odd. In this diagram, circular vertices represent constraints and edges represent boolean variables. Top: The red constraint expresses that exactly half of the creases must be mountain folds and the others must be valley folds. Bottom: Two possible satisfying variable assignments and the associated local foldings.

We now show that solutions to the constraints generated by this algorithm correspond to flat-foldable crease assignments of f.

▶ **Theorem 3.** *A mountain/valley assignment for f is flat foldable if and only if it can be extended to a satisfying assignment of C_f.*
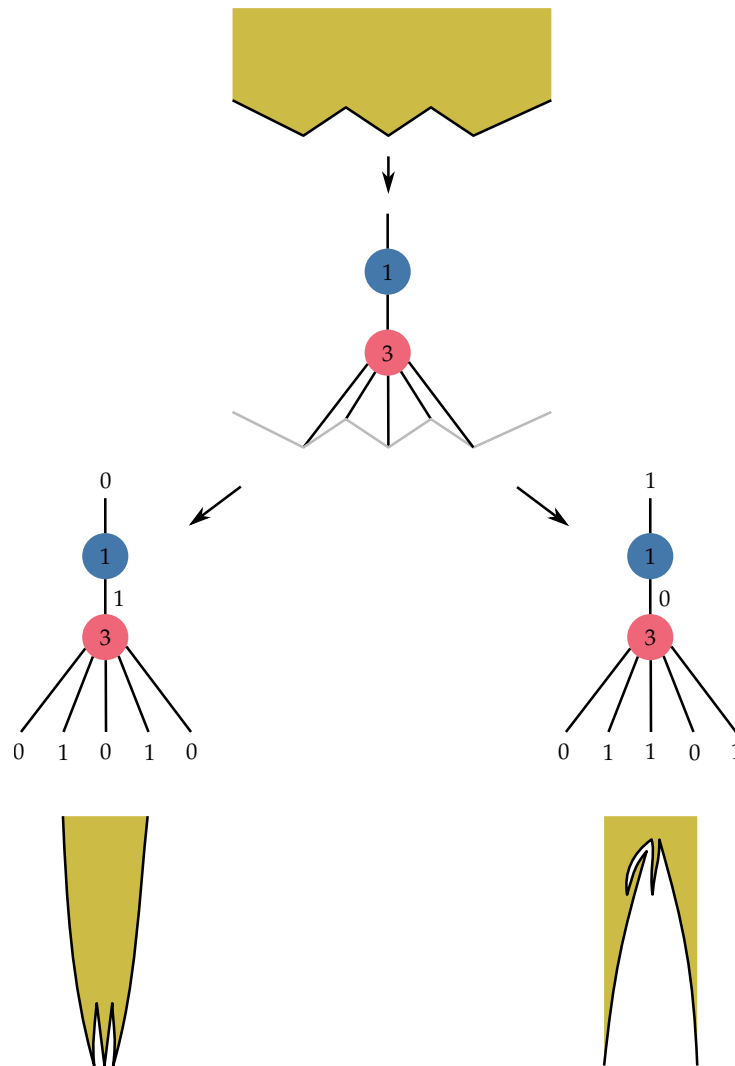
Figure 5 The case where k is even. The pair of constraints expresses that the number of mountain folds and valley folds must differ by 1, and the majority value is equal to the newly generated variable.

We may need to extend the assignment to account for variables introduced in the case where k is even. The values for these variables are forced by the constraints added when the variables are introduced, and can be determined by considering variables in the order they were added.

Proof. The proof is by induction on the size of f. If all the edges of f have equal length, then it is immediate by cases A and B of Lemma 1 that an assignment is flat foldable if and only if equation (1) holds.

When the edges are not all equal in length, the algorithm finds some maximal sequence $e_m, \ldots, e_{m+k-1}$ of k equal-length edges surrounded by strictly longer edges, whose incident angles we call S.

If k is odd, then by case C of Lemma 1, the assignment is flat foldable for f if and only if it both assigns an equal number of mountain and valley folds to the angles in S, and is also a flat-foldable crease assignment for f', where f' is the face resulting from replacing the

edges $e_{m-1}, \ldots, e_{m+k}$ with a single edge of length $\theta_{m-1} - \theta_m + \theta_{m+k}$. The first condition is just equation (2), and the second is equivalent by the inductive hypothesis to the set of constraints $C_{f'}$ obtained by recursion on f'. An example of this case is shown in Figure 4.

If k is even, then by case D of Lemma 1, the assignment is flat foldable if and only if the mountain and valley folds assigned to the angles in S differ by 1, and it is also a flat-foldable crease assignment for f' when suitably extended. Here f' is the face resulting from replacing the edges $e_m, \ldots, e_{m+k-1}$ with a single angle, and the assignment is extended to assign the new angle to be the same type as the majority of the folds it assigned to the angles in S. Equation (3) constrains the number of folds to differ by 1, where y is the minority fold type, and equation (4) constrains z to be the opposite of y, so z is the majority fold type. By the inductive hypothesis, the constraints $C_{f'}$ obtained by recursion on f' are equivalent to the statement that f' is flat foldable under the assignment extended to assign z to the new angle. An example of this case is shown in Figure 5.

In all cases, we find that the assignment is flat foldable if and only if it satisfies the constraints. ◀

We also show that these constraints can be computed efficiently and satisfy certain properties which will be useful for solving them.

▶ **Theorem 4.** *The algorithm for computing C_f takes time linear in the number of angles in f. The variables and clauses of C_f form a graph in which graph vertices correspond to clauses and graph edges correspond to variables, when an additional* blue *graph vertex is added for each angle of f. Then this graph is a bipartite (i.e. 2-colored) forest with linearly many vertices, and there is a planar embedding of this graph within f such that each vertex corresponding to an angle of f is located at the vertex of f incident to that angle.*

Proof. Let n be the number of angles in f. We prove by induction that C_f forms a graph as described with at most $2n$ vertices.

In the case where the edges all have equal length, C_f is a star graph whose central vertex is a red clause and whose outer vertices are the blue angles of f, so it is a bipartite forest with $n + 1 \leq 2n$ vertices. The planar embedding can be achieved by placing the central vertex within f and drawing edges to all the vertices of f.

When the edges of f are not all equal in length, the algorithm finds some sequence of k edges whose $k + 1$ incident angles we call S. Let T be the star graph whose central vertex is the red clause added in this step and whose outer vertices are the blue angles of S; this is a bipartite forest with $k + 2$ vertices.

When k is odd, the graph C_f is simply the disjoint union of $C_{f'}$ and T, where f' is a face with $n - k - 1$ angles. By the inductive hypothesis $C_{f'}$ is a bipartite forest with at most $2(n-k-1)$ vertices, so C_f is a bipartite forest with at most $k+2+2(n-k-1) = 2n-k \leq 2n$ vertices. The planar embedding of C_f is obtained from the planar embedding of $C_{f'}$ by simply placing T alongside it; none of the edges need to cross because the angles in S are contiguous in f.

When k is even, the graph C_f is formed from the disjoint union of $C_{f'}$ and T by adding an edge from the red central vertex of T to the blue vertex corresponding to some angle a' of f', where f' is a face with $n - k$ angles. By the inductive hypothesis $C_{f'}$ is a bipartite forest with at most $2(n - k)$ vertices, so C_f is a bipartite forest with at most $k + 2 + 2(n - k) = 2n - k + 2 \leq 2n$ vertices. The planar embedding of C_f is obtained from the planar embedding of $C_{f'}$ by first placing T alongside it as before; again none of the edges cross because S is contiguous in f. Then the edge from the central vertex of T to the vertex corresponding to a' can be added without crossing because a' occurs in the same place in f''s cyclic order of angles as S does in f.

Thus C_f is a linear-sized bipartite forest with the desired planar embedding. We need to show that it can be computed in linear time. It is straightforward to charge the work performed by the algorithm at each step to the newly created vertices, except for finding the sequence $e_m, \ldots, e_{m+k-1}$ of equal-length edges surrounded by strictly longer edges. We cannot accomplish this by simply scanning through the edges of the face at each iteration, since this would take linear time and there might be linearly many iterations. We instead solve this by maintaining a cyclic doubly-linked list C, each of whose entries corresponds to a maximal contiguous sequence of equal-length edges. Additionally we keep a list M of such entries of C which are surrounded by longer entries. These can be computed once at the beginning of the algorithm in linear time, and then maintained at each iteration. At each iteration a sequence $e_m, \ldots, e_{m+k-1}$ is obtained by taking the first entry from M and removing it from both M and C. When the new face f' is computed, we add any new edges to C and check whether any of the newly adjacent pairs of entries have equal length; if so we consolidate them into a single entry of C. We also check whether any of the newly adjacent entries have become surrounded by strictly longer entries; if so we add them to M. These checks take constant time in each iteration since at most two new pairs of adjacent entries can be created. So computing C_f takes linear time overall.   ◀

5   Compatibility Constraints

Next, we describe the constraints needed to ensure that the crease assignments are compatible between faces. The angles around each vertex must sum to 360°; this means exactly one of these angles is a mountain fold, as shown in Figure 6. So for each vertex v of the graph, we generate a blue constraint C_v:

$$\sum_{a \in A_v} x_a = 1, \tag{5}$$

where A_v is the set of angles incident to v.

▶ **Theorem 5.** *A connected combinatorially embedded planar multigraph with prescribed edge lengths has a flat folding with no flat angles if and only if the constraint satisfaction problem consisting of*

- *for each face f, the constraints C_f described in Section 4, and*
- *for each vertex v, the constraint C_v described above*

is satisfiable. Moreover, these constraints can be computed in time linear in the number of angles in the graph.

Proof. Suppose the graph has such a flat folding, and assign variables representing angles in the graph based on whether the corresponding angle is a mountain or a valley fold in the flat-folded state; this is only a partial assignment since some variables do not correspond to angles of the original graph. Each face (and thus its corresponding simple cycle) is folded flat, and the variables which are not yet assigned are disjoint between faces, so by Theorem 3 we can extend the assignment to an assignment of all variables which satisfies C_f for every face f. The assignment also satisfies C_v since exactly one angle incident to v has measure 360° in the folded state.

Conversely, suppose there is a satisfying assignment. Then assign each angle to be mountain or valley based on the value of the corresponding variable. By Theorem 3, this gives a flat-foldable crease assignment for each face. These crease assignments are compatible because the variable assignments satisfy each C_v, so by Theorem 2 there is a flat folding with these angle assignments.
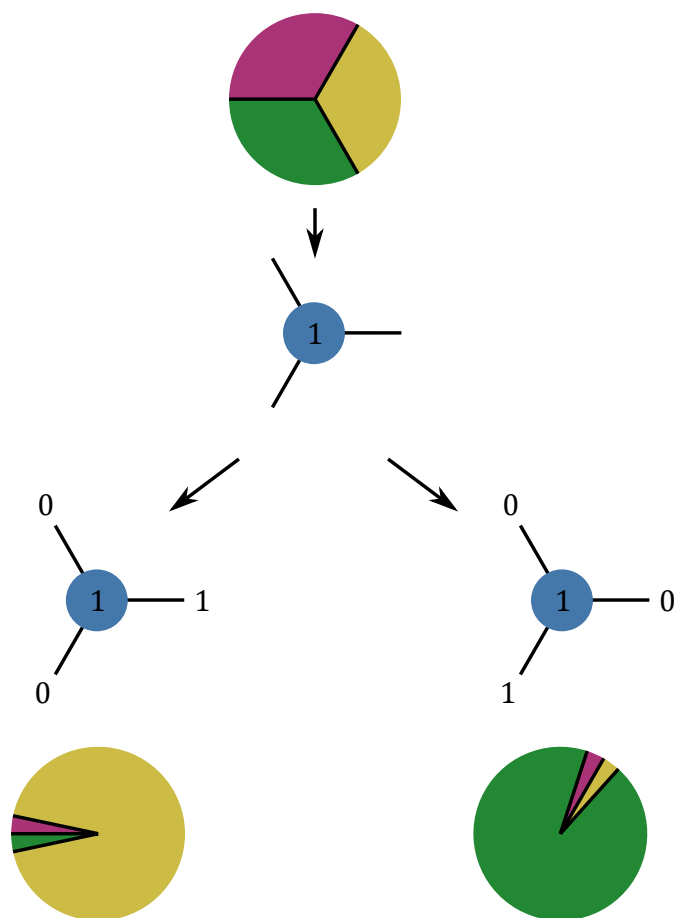
Figure 6 A vertex folds flat under a given crease assignment if and only if exactly one if the incident angles is a mountain fold.

Finally, we show that the set of constraints can be computed in linear time. By Theorem 4 each set of face constraints C_f can be computed in time linear in the number of angles incident to f. Since the sets of angles incident to different faces are disjoint, it takes linear time overall to compute the face constraints. Similarly, computing each vertex constraint C_v takes time linear in the number of angles incident to v, and these are all disjoint from each other as well. So the set of constraints can be computed in time linear in the number of angles of the graph.   ◀

6   Solving the Constraint Satisfaction Problem

What remains is solving the constraint satisfaction problem consisting of C_f and C_v for each face and vertex of the graph. Inspecting the constraints reveals that they are an instance of planar bipartite positive ∗-IN-∗SAT-E2:

- Each constraint has the form $\sum_{x \in S} x = c$ for some set S of variables and constant c; this is a clause saying exactly c variables in S are true.
- The red and blue clauses provide the bipartition. Each variable is in exactly one red clause and exactly one blue clause. For each angle a incident to a face f and a vertex v, the variable x_a appears in one red clause belonging to C_f and one blue clause C_v. All other variables satisfy this condition because the subgraph corresponding to each C_f is bipartite according to Theorem 4.
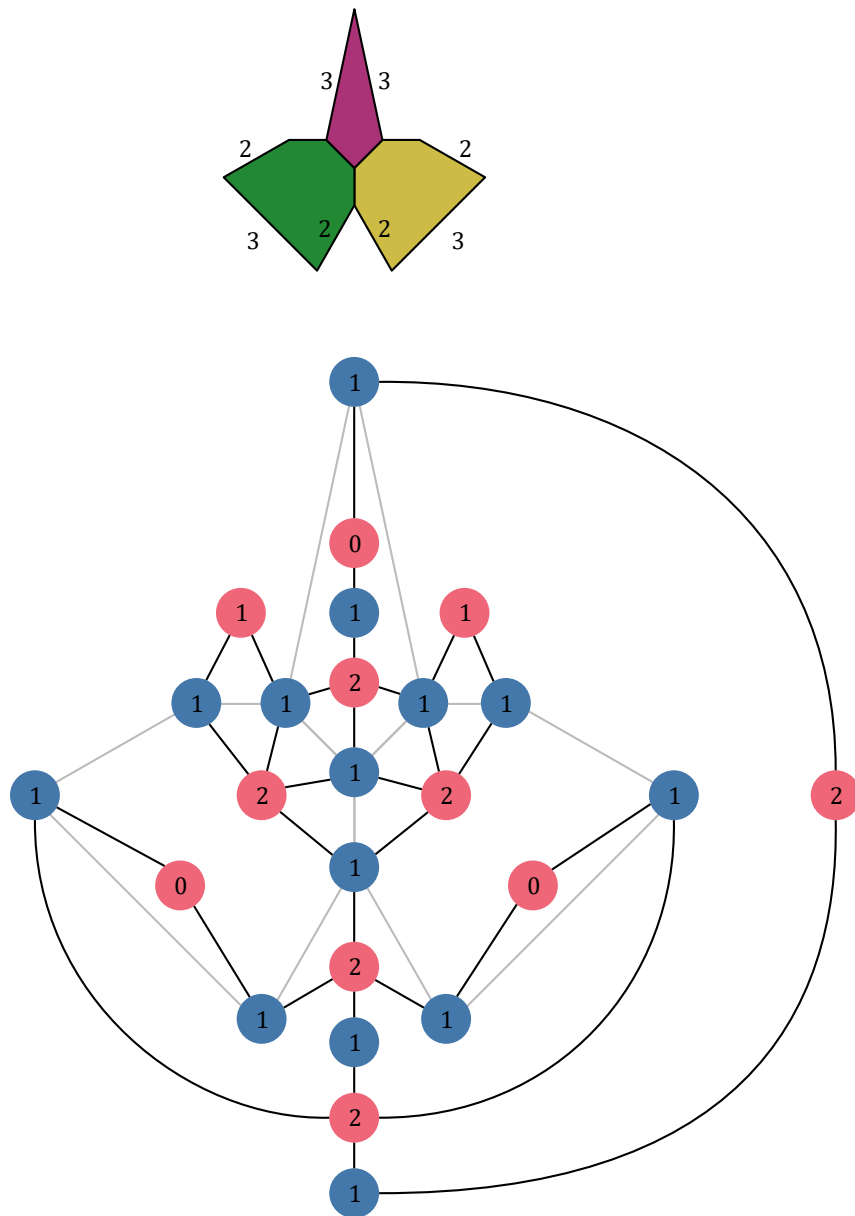
Figure 7 Top: An example graph with assigned edge lengths. Unlabeled edges have length 1. Bottom: The resulting instance of planar bipartite positive ∗-in-∗SAT-E2 (overlaid on the original graph in gray). Since this instance is unsatisfiable, the original graph cannot be folded flat.

- The graph corresponding to the constraint satisfaction problem is planar. We can place each clause C_v at the corresponding vertex v. Then for each face f we can place the graph corresponding to C_f inside f; by Theorem 4 this can be done without violating planarity. An example of the planar embedding constructed for the entire constraint satisfaction problem is shown in Figure 7.

All that remains to be shown is that planar bipartite positive ∗-in-∗SAT-E2 can be solved efficiently. We now describe a fairly standard reduction to a max-flow problem, which can be solved in near-linear time.

▶ **Theorem 6.** *Planar bipartite positive* $*$-IN-$*$SAT-E2 *can be solved in* $O(n \log^3 n)$ *time, where n is the number of clauses.*

Proof. We use the graph with clauses as vertices and variables as edges, as described earlier and shown in Figure 7. For each red clause r which expects ℓ_r true variables, we add a new source vertex and an edge from the source vertex to r with capacity ℓ_r. Similarly, for every blue clause b expecting ℓ_b true variables, we add a new sink vertex and an edge from b to the sink vertex with capacity ℓ_b. Finally, we assign a capacity of 1 to each edge corresponding to a variable, which goes from a red clause to a blue clause. This gives us an instance of multi-source multi-sink planar max-flow, for which the maximum possible flow can be determined in time $O(k \log^3 k)$ [7] where k is the number of vertices in the flow graph. Since the flow graph has exactly twice as many vertices as there were clauses, the maximum flow can be determined in time $O(n \log^3 n)$.

We will assume that

$$T := \sum_{\text{red } r} \ell_r = \sum_{\text{blue } b} \ell_b,$$

since this is clearly required for the constraint problem to be satisfiable.

To solve the constraint satisfaction problem, we ask if the maximum flow has value T; this is clearly an upper bound an the maximum flow.

An integer flow uses some set of edges corresponding to variables, which specifies an assignment. The flow constraint on the edges to the appropriate source or sink forces the flow to use at most ℓ_c variables in clause c, and in order to reach the target flow T we must use exactly this many variables in each clause. Thus the desired flow exists if and only if the instance of planar bipartite positive $*$-IN-$*$SAT-E2 is solvable.   ◀

7 Putting Things Together

Combining Theorem 5 and Theorem 6 immediately gives our main result:

▶ **Corollary 7.** *We can determine whether a connected combinatorially embedded planar multigraph with prescribed edge lengths and exterior face has a flat folding with no flat angles in $O(n \log^3 n)$ time, where n is the number of angles in the graph.*

Proof. The constraint problem instance can be computed in linear time, and so it has linearly many clauses, which can thus be solved in time $O(n \log^3 n)$.   ◀

This result can be extended in three ways, described next.

7.1 Extension to Specified Flat Angles

First, we can allow flat (180°) angles in the folded graph, provided the input specifies *which* angles are flat, leaving the remaining angles free to be mountain or valley (but not flat).

To accomplish this, first observe that for there to be a flat folding, each vertex must have exactly zero or two flat angles. We do not create variables for flat angles, since their angle is already known. Within a face that contains a flat angle, we treat the two edges around the flat angle as a single longer edge. At a vertex v which has two flat angles, we need all other angles to be valley, so the constraint C_v is now

$$\sum_{a \in A_v} x_a = 0, \tag{6}$$

where A_v includes only non-flat angles at v. The rest of the algorithm is as before.

7.2 Extension to Disconnected Graphs

Second, we can account for the case where the graph is disconnected. Here we assume that the connected components are arranged in a rooted forest (i.e., a collection of rooted trees), where each non-root component specifies which interior face of its parent it is to reside in. This condition can arise from folding an arbitrary single-vertex complex, where some faces share the central vertex but no edges; then the structure of the complex requires a certain arrangement of components within faces. We first check that each connected component is foldable. If this is the case, then the only obstacle to foldability is being able to fit the folded state of each child graph G_i inside the designated face p_i of its parent.

We define the ***folded diameter*** of a graph or face to be the maximum distance between any pair of vertices in its folded state. Since the locations of all flat angles are specified, the relative vertex coordinates are determined and can be computed in linear time (as described in Section 1), so this value can be computed easily without knowledge of the folding. It turns out that we can fit G_i inside p_i in a folding if and only if the folded diameter of G_i is at most the folded diameter of p_i. To show this, we can imagine applying cases C and D of Lemma 1 to p_i repeatedly until all the edge lengths are equal. Because the face transformations in those cases preserve folded diameter, it follows that the remaining edges all have length equal to the folded diameter of p_i. Thus, if the folded diameter of G_i is less than or equal to the length of one of these edges, we can place a folding of G_i along it in the folded state. On the other hand, if the folded diameter of G_i is greater than the folded diameter of p_i, then we can clearly never fit a folding of G_i inside a folding of p_i.

7.3 Finding an Exterior Face

Third, instead of assuming that the exterior face is given, we can determine in linear time a face that is a suitable exterior face if any face is. Observe that the exterior face must be ***full-diameter*** in the sense that its folded diameter (defined in Section 7.2 above) equals the folded diameter of the entire graph, because some vertex of the minimum (and maximum) coordinate must be on the exterior face in any flat folding. We claim that *every* full-diameter face is an equally suitable exterior face: if there is a flat folding with any one full-diameter face as exterior face, then there is a flat folding with any desired full-diameter face as exterior face. Thus, to determine whether the graph is flat foldable, we can simply find any full-diameter face and specify it as the exterior face.

To prove the claim, consider a flat folding of the graph, say with exterior face e. Take any non-exterior full-diameter face f, with diameter realized by vertices v and w. Face f consists of two folded paths connecting v and w. By the argument in Section 7.2 above, in any folding of f resulting from Lemma 1, we can select v and w such that the two folded paths are separable: we can draw a straight line segment s from v to w that is layered in between the two folded paths. Because s is full diameter, it partitions the edges of the graph into two halves H_1 and H_2, where H_1 is entirely before H_2 in the layer order. Some vertices (including v and w) have some incident edges in H_1 and other incident edges in H_2. We can imagine splitting each such vertex x into two vertices x_1 and x_2, where x_i is incident to the edges that lie within H_i, so that H_1 and H_2 become disconnected from each other. We then swap the layer order of the two halves, placing H_2 before H_1, and for each split vertex x, reconnect the two halves x_2 and x_1, which corresponds to a cyclic shift of the edges incident to x. This process is illustrated in Figure 8. Intuitively, we can view the folding as lying on an American football (prolate spheroid), where the two poles represent the minimum and maximum vertex coordinates; then this transformation corresponds to spinning the line along which we cut this football open to define the extremes in the other dimension (layer order). Thus we still obtain a flat folding of the graph, but now f is the exterior face.
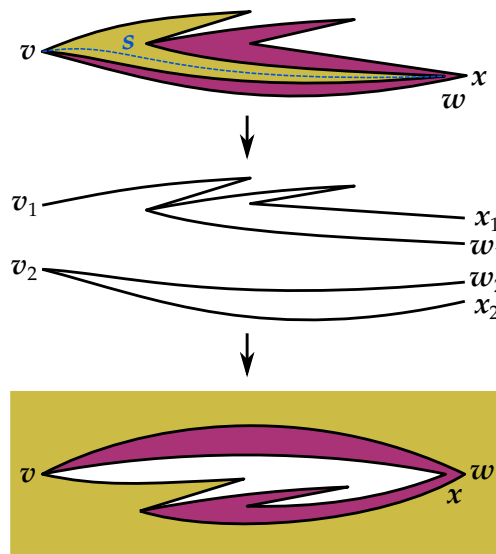
■ **Figure 8** Cutting a flat folding apart and reassembling it with a different exterior face.

7.4   Finale

Putting these extensions together, we have the following more general result:

▶ **Corollary 8.** *Given a combinatorially embedded planar multigraph with prescribed edge lengths and some angles specified as flat, we can determine in $O(n \log^3 n)$ time whether there is a flat folding that has precisely the specified angles flat.*

On the other hand, if the set of flat angles is not specified, it is NP-complete to determine whether there is a flat folding [2], so this implies that the hard part is deciding which angles should be flat.

___ **References** ___

1   Timothy G. Abbott, Erik D. Demaine, and Blaise Gassend. A generalized carpenter's rule theorem for self-touching linkages. arXiv:0901.1322, 2009. `arXiv:0901.1322`.

2   Zachary Abel, Erik D. Demaine, Martin L. Demaine, Sarah Eisenstat, Jayson Lynch, Tao B. Schardl, and Isaac Shapiro-Ellowitz. Folding equilateral plane graphs. *International Journal of Computational Geometry and Applications*, 23(2):75–92, April 2013.

3   Zachary Abel, Erik D. Demaine, Martin L. Demaine, David Eppstein, Anna Lubiw, and Ryuhei Uehara. Flat foldings of plane graphs with prescribed angles and edge lengths. *Journal of Computational Geometry*, 9(1):74–93, 2018.

4   Hugo A. Akitaya, Radoslav Fulek, and Csaba D. Tóth. Recognizing weak embeddings of graphs. *ACM Transactions on Algorithms*, 15(4), October 2019. `doi:10.1145/3344549`.

5   Marshall Bern and Barry Hayes. The complexity of flat origami. In *Proceedings of the 7th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 175–183, Atlanta, January 1996.

6   Paola Bertolazzi, Giuseppe Di Battista, Giuseppe Liotta, and Carlo Mannino. Upward drawings of triconnected digraphs. *Algorithmica*, 12(6):476–497, 1994.

7   Glencora Borradaile, Philip N. Klein, Shay Mozes, Yahav Nussbaum, and Christian Wulff-Nilsen. Multiple-source multiple-sink maximum flow in directed planar graphs in near-linear time. *SIAM Journal on Computing*, 46(4):1280–1303, 2017. `doi:10.1137/15M1042929`.

8   M. Chlebík and J. Chlebíková. Approximation hardness of dominating set problems in
        bounded degree graphs. *Information and Computation*, 206(11):1264–1275, 2008. `doi:`
        `10.1016/j.ic.2008.07.003`.

9   Robert Connelly, Erik D. Demaine, and Günter Rote. Infinitesimally locked self-touching
        linkages with applications to locked trees. In J. Calvo, K. Millett, and E. Rawdon, editors,
        Physical Knots: Knotting, Linking, and Folding of Geometric Objects in 3-space, pages 287–311.
        American Mathematical Society, 2002.

10   Erik D. Demaine. 6.849: Geometric folding algorithms: Linkages, origami, polyhedra: Lecture
        5. MIT class, fall 2010. URL: `https://courses.csail.mit.edu/6.849/fall10/lectures/`
        `L05.html?notes=4`.

11   Erik D. Demaine and Joseph O'Rourke. *Geometric Folding Algorithms: Linkages, Origami,
        Polyhedra.* Cambridge University Press, 2007.

12   Ivan Tadeu Ferreira Antunes Filho. Characterizing boolean satisfiability variants. M.eng. thesis,
        Massachusetts Institute of Technology, 2019. URL: `https://erikdemaine.org/theses/`
        `ifilho.pdf`.

13   Thomas Hull. On the mathematics of flat origamis. *Congressus Numerantium*, 100:215–224,
        1994.

14   Thomas Hull. The combinatorics of flat folds: a survey. In *Origami³: Proceedings of the
        3rd International Meeting of Origami Science, Math, and Education*, pages 29–38, Monterey,
        California, March 2001.

15   Thomas C. Hull. *Origametry: Mathematical Methods in Paper Folding.* Cambridge University
        Press, December 2020.

16   Jacques Justin. Aspects mathematiques du pliage de papier (Mathematical aspects of paper
        folding). In H. Huzita, editor, *Proceedings of the 1st International Meeting of Origami Science
        and Technology*, pages 263–277, Ferrara, Italy, December 1989. Originally appeared in *L'Ouvert*,
        number 47, 1987, pages 1–14. URL: `https://publimath.univ-irem.fr/biblio/IST87008.`
        `htm`.

17   Toshikazu Kawasaki. On the relation between mountain-creases and valley-creases of a flat
        origami. In H. Huzita, editor, *Proceedings of the 1st International Meeting of Origami Science
        and Technology*, pages 229–237, Ferrara, Italy, December 1989. An unabridged Japanese
        version appeared in *Sasebo College of Technology Report*, 27:153–157, 1990.

18   Wolfgang Mulzer and Günter Rote. Minimum-weight triangulation is NP-hard. *Journal of the
        ACM*, 55(2), May 2008. `doi:10.1145/1346330.1346336`.