Computing Flat-Folded States

Hugo A. Akitaya, Erik D. Demaine, Jason S. Ku

Abstract: *In this paper, we introduce a facewise definition for global flat foldability on crease patterns with n convex faces that constrains $O(n^3)$ conditions on the layer orders between pairs of overlapping faces, and prove that it is equivalent to the established pointwise definition. We use this formulation to show that (1) such a facewise layer order can be verified in $O(\min\{n^2p, n^2+mp^2\}) = O(n^3)$ time, where m and p parameterize the complexity of the folding; and (2) all valid folded states of a crease pattern can be implicitly computed in $O(n^3 + \sum_{i=1}^{k} t_i^3 2^{t_i})$ time and $O(\sum_{i=1}^{k} s_i)$ space, where t_i and s_i parameterize a decomposition of the problem into k independent components. Lastly, we prove that unassigned crease patterns on n faces can have at most $2^{O(n^2)}$ folded states, while there exist assigned crease patterns on convex paper that achieve that bound, and assigned crease patterns on square paper that have $2^{\Omega(n \log n)}$ folded states.*

1 Introduction

Given a crease pattern, what flat origami does it fold into? This basic question is one of the oldest in the field of computational origami, first considered by Bern and Hayes [96]. Akitaya et al. [15] proved that deciding whether a crease pattern admits a global flat-folded state is NP-hard. But what if we want to find the flat foldings anyway? How can we compute them in exponential time? Are there cases where we can find the flat foldings efficiently? How do we even represent a flat folding in a way that can be efficiently verified?

In this paper, we aim to solve these problems. A central challenge with rigorously representing, verifying, or computing folded states is that the established definition for a global folded state from [Demaine and O'Rourke 07] (which applies to 3D folding, not just flat) specifies the layer order between an infinite set of pairs of points — all those that overlap and are not creased. We call this the ***pointwise*** definition of global foldability. The naïve representation of this definition is not finite, so it is not suitable for algorithms.

By contrast, many have proposed *facewise* conditions on a flat-folded state, where we specify the layer order between pairs of faces (polygons). The earliest is perhaps by Justin [Justin 94], who introduced the superposition of all crease faces in the flat-folded geometry (as the "s-net"). Later work [Schneider 04, Lang and Demaine 06] elaborated on and used this definition; others have developed representations for the special case of $m \times n$ map folding [Morgan 12, Nishat 13, Jia

et al. 23]. In particular, the FOLD file format for representing crease patterns and folded states [Demaine et al. 16a] allows specifying the layer order between pairs of faces. But software cannot formally check whether such a layer ordering (and the corresponding folded state) is actually valid, because there is no known equivalence between the pointwise definition and the various (essentially equivalent) facewise definitions.

In this paper, we begin with some definitions in Section 2, and then introduce in Section 3 a definition of global flat foldings of crease patterns with convex faces that constrains a finite set of conditions on a facewise layer order defined between pairs of overlapping faces, and prove that it is equivalent to the established pointwise definition. This approach provides a definition that is much easier to work with when specifying, checking, or finding flat foldings.

In Section 4, we use this formulation to show that such a facewise layer order for a crease pattern with n convex faces can be verified in $O(\min\{n^2p, n^2+mp^2\}) = O(n^3)$ time, where m is the complexity of the arrangement of folded creases, and p is the "ply" of the folding, informally the maximum thickness of the folding. Thus we prove that verifying a layer ordering is in P, and finding a layer ordering is in NP. Past work gave a linear-time verification algorithm for the special case of $m \times n$ map folding [Nishat 13]. Here we assume that we have already determined which faces of the crease pattern overlap in the folded-state geometry, which is necessary to even specify a layer order. Given a crease pattern, the geometry of the folded state and overlaps can be found in polynomial time on a real RAM, but it remains open how many bits of precision are necessary to do this correctly on a word RAM. (The geometry involves square roots of crease lengths, similar to [Demaine et al. 20], so we run into the open problem of sum-of-square-roots.) Thus it remains open whether crease pattern flat foldability is in NP.

In Section 5, we provide an algorithm to compute all valid folded states by (1) constructing a pruned constraint graph linking unassigned layer orders to related flat-foldability constraints; (2) decompose this graph into k connected components where component i contains t_i unassigned binary variables defining the layer order between two overlapping convex faces; and (3) solving for the $s_i \leq 2^{t_i}$ valid assignments of each component separately. This algorithm runs in $O(n^3 + \sum_{i=1}^{k} t_i^3 2^{t_i})$ time and $O(\sum_{i=1}^{k} s_i)$ space. We have implemented this algorithm in open-source software [Ku 22]. In many practical crease patterns, the t_is and s_is turn out to be relatively small, making this software reasonably efficient.

Lastly, in Section 6, we prove that crease patterns on n faces can have at most $2^{O(n^2)}$ folded states. We then describe two generalizable crease patterns on n faces: one on convex paper that admits $2^{\Omega(n^2)}$ valid foldings, and one on square paper that admits $2^{\Omega(n \log n)}$ valid foldings. These results formalize and extend early work [Bern and Hayes 96] that sketches a construction of a square crease pattern with $2^{\Omega(n \log n)}$ valid folded states, and posed the open problem of whether crease patterns could admit more solutions. By contrast, for the special case of $1 \times n$ stamp folding, the number of folded states is known to be $2^{\Theta(n)}$ [Uehara 10].

2 Basic Definitions

We define a ***crease pattern*** $\Sigma = (V, E)$ to be a planar straight-line graph embedded in $\mathbb{R}^2$ with vertices (points) V and edges (open line segments) E. We call an edge of a crease pattern a ***crease***, and call any point in $V \cup E$ a ***crease point***. The crease pattern partitions the plane into open subsets; the *bounded* such open subsets are the (polygonal) ***faces*** F of the crease pattern. The ***degree*** of a polygon is the number of vertices bounding the polygon, and the degree of a vertex is the number of creases incident to the vertex. The crease pattern's paper P is the union of the closures of its faces.

A crease pattern is ***face-convex*** if each of its faces is convex. A crease ***bounds*** a face if the crease is contained in the face's closure. An ***interior*** crease bounds two faces and a ***boundary*** crease bounds one face (and thus lies on the boundary of the paper P). A crease pattern is ***well-bounded*** if the number of degree-2 vertices (which we can assume all occur on the boundary) is at most a constant fraction of the total number of vertices $|V|$. In this case, the average face degree is $O(1)$, so $|V|$ and $|E|$ are both $\Theta(|F|)$, and $n = |F|$ is a reasonable measure of the complexity of the crease pattern. A crease pattern that is not well-bounded can easily be made well-bounded by triangulation, or otherwise cleaning up the boundary.

An ***isometric flat folding*** of crease pattern $\Sigma = (V, E)$ with paper P is a function $f : P \to \mathbb{R}^2$ such that (1) the set of points not differentiable under f is a subset of $V \cup E$; and (2) if $\gamma : [0, \ell] \to P$ is a piecewise-geodesic curve on P parameterized with respect to arc-length, then $f \circ \gamma : [0, \ell] \to \mathbb{R}^2$ is also a piecewise-geodesic curve parameterized with respect to arc-length (of the same total arc-length ℓ). It is not hard to show that under these conditions f must be continuous and nonexpansive, and that f restricted to any face of the crease pattern is a rigid transformation (allowing reflection).

Point sets $P_1, P_2, \ldots, P_k$ ***overlap*** under f if the intersection $\bigcap_i f(P_i)$ has dimension equal to the minimum dimension of the P_is. Specifically, we will discuss the overlap between pairs or larger collections of faces (two-dimensional, so the overlap must be two-dimensional); pairs of creases (both one-dimensional, so the overlap must be one-dimensional), and the overlap between a crease and a face (so the overlap must be one-dimensional). Note that for two creases to overlap, their foldings must be collinear. An interior crease is ***folded*** if its bounding faces overlap in the folding and ***unfolded*** otherwise. The ***overlap set*** of a point $p \in f(P)$ is the set of faces that contain a point q such that $f(q) = p$. The ***ply*** of an isometric flat folding is the maximum size of any overlap set for points in $f(P)$. Other recent work [Eppstein 23] has used ply and treewidth of an associated planar graph to provide a fixed-parameter tractable algorithm for testing flat foldability of a crease pattern.

It is known that a crease pattern with a single vertex incident to all folded creases obeys the so-called Kawasaki-Justin Theorem: the alternating sum of angles between consecutive folded creases when cyclically ordered around the vertex equals zero [Demaine and O'Rourke 07]. This condition turns out to be necessary and sufficient: a crease pattern $\Sigma = (V, E)$, together with a labeling of each crease as either folded or unfolded, has an isometric folding if and only if every vertex

(restricted to folded creases) locally obeys the Kawasaki-Justin Theorem.

3 Flat Foldability from Convex Faces

An isometric flat folding f does not completely describe a flat folding; we still need to specify the layer order of overlapping faces. Indeed, for general crease patterns, the Kawasaki-Justin Theorem is not enough to guarantee flat foldability, because the layer orders required by individual vertices might be incompatible with each other. *Geometric Folding Algorithms* [Demaine and O'Rourke 07, Chapter 11] (referred to henceforth as *GFA*) defines a function λ to describe the layer ordering of an arbitrary folding (not necessarily flat) We call λ a "pointwise" layer order function, as it is defined on every pair of points on the paper that are mapped to the same point in the folding.

While this definition is very general, it is hard to work with computationally, as the conditions apply to a possibly infinite set of point pairs. In this section, we provide a new "facewise" definition of layer order that is specific to flat foldings of crease patterns with convex faces. In Section 3.1, we first reproduce the pointwise definition from *GFA* restricted to flat foldings which allows some simplification. Then, in Section 3.2, we identify the finite set of constraints that a facewise layer order Λ must satisfy, where Λ is defined between every pair of overlapping faces. Finally, in Section 3.3, we prove the equivalence of the two definitions (pointwise and facewise) of flat foldability.

3.1 Pointwise Definition of Global Flat Foldability

This section summarizes the *GFA* definition from [Demaine and O'Rourke 07, Chapter 11], with slight simplifications from the restriction to flat foldings. A ***pointwise layer order*** function λ defines $\lambda(p,q)$ for any two noncrease points $p,q \in P$ with $f(p) = f(q)$, where $\lambda(p,q) \in \{+1,-1\}$ according to whether p is above or below q, relative to a global notion of upward.[1] Call λ ***valid*** if it satisfies four conditions:

1. **Antisymmetry condition:** If $\lambda(p,q)$ is defined, then $\lambda(q,p) = -\lambda(p,q)$. Intuitively, if p is above q, then q is below p.
2. **Transitivity condition:** If λ is defined pairwise on p,q,r, and $\lambda(p,q) = -\lambda(r,q)$, then $\lambda(r,p) = \lambda(q,p)$. Intuitively, if q is between p and r, then p has the same order relative to q and r.
3. **Consistency condition:** If $(p,q),(p',q') \in \mathrm{dom}(\lambda)$ are path-connected (endpoints of a path) in $\mathrm{dom}(\lambda)$, then $\lambda(p,q) = \lambda(p',q')$.
4. **Noncrossing conditions:** For any two distinct nonboundary (possibly crease) points $p,q \in P$ for which $f(p) = f(q)$, and for any sufficiently small $\varepsilon > 0$, let $N_p \subseteq P$ be the open disk neighborhood of p that remains within ε distance of $f(p)$ in $\mathbb{R}^2$, and similarly define N_q. By choosing ε sufficiently small, we can

[1] The *GFA* definition for $\lambda(p,q)$ is relative to q's normal, which is necessary when there is no global notion of upward, but we simplify here for flat folding.

assume that N_p and N_q have no vertices except possibly p and q respectively, and that the closures $\overline{N}_p$ and $\overline{N}_q$ are disjoint. We require that (f,λ) restricted to points in $P' = \partial N_p \cup \partial N_q$ (which has two connected components) be a ***valid 1D folded state*** defined as follows.

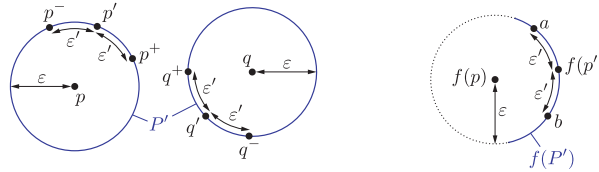

Figure 1: *Depiction of two points $p,q \in P$ for which $f(p) = f(q)$, in the crease pattern (left) and in the folding (right).*

Consider a 1D piece of paper $P' \subseteq P$ that is the disjoint union of one or more topological circles and for which $f(P')$ is a subset of a 2D circle as shown in Figure 1. Looking at the top side of P, we obtain a notion of clockwise for each such topological circle of P'. For any two distinct nonboundary (possibly crease) points $p',q' \in P'$ for which $f(p') = f(q')$, and for any sufficiently small $\varepsilon' > 0$ (in particular $\varepsilon' < \varepsilon$), define $p^+, p^- \in P'$ to be the clockwise- and counterclockwise-next points (respectively) from p' for which $f(p^+), f(p^-)$ are at distance ε' from $f(p')$; and similarly define q^+, q^-. Assume (by ε' being sufficiently small) that the clockwise segment of P' from p^- to p^+ has no crease points except p', and similarly the clockwise segment of P' from q^- to q^+ has no crease points except q'. Because $f(P')$ is a subset of a 2D circle and $\varepsilon' < \varepsilon$, $f(p^+), f(p^-), f(q^+), f(q^-)$ occupy at most two points (the two points of the 2D circle at distance ε' from $f(p') = f(q')$). We decompose into three cases paralleling the naming conventions introduced in [Akitaya et al. 15]; see Figure 2.

(a) **Tortilla-tortilla condition:** Two of $f(p^+), f(p^-), f(q^+), f(q^-)$ occupy the same point a, and two occupy the other point b. If $f(p^+) = f(p^-) = a$ and $f(q^+) = f(q^-) = b$ (corresponding to Case 2 of *GFA*), then there are no requirements (crossing is impossible). Otherwise (corresponding to first two subcases of Case 4 of *GFA*), assume by symmetry that $f(p^+) = f(q^+) = a$ and $f(p^-) = f(q^-) = b$. Then we require that $\lambda(p^+, q^+) = \lambda(p^-, q^-)$.

(b) **Taco-tortilla condition:** Three of $f(p^+), f(p^-), f(q^+), f(q^-)$ occupy the same point a, and one occupies the other point b. (This corresponds to last two subcases of Case 4 of *GFA*.) Assume by symmetry that $f(p^+) = f(q^+) = f(q^-) = a$ and $f(p^-) = b$. Then we require that $\lambda(q^+, p^+) = \lambda(q^-, p^+)$.

(c) **Taco-taco condition:** All four points $f(p^+), f(p^-), f(q^+), f(q^-)$ occupy the same point a. (This corresponds to Case 5 of *GFA*.) Then we require that $\lambda(p^+, q^+) = \lambda(p^-, q^+)$ if and only if $\lambda(p^+, q^-) = \lambda(p^-, q^-)$. We here note that this condition is equivalent to:

$$\lambda(p^+, q^+) + \lambda(p^-, q^+) + \lambda(p^+, q^-) + \lambda(p^-, q^-) = 0 \mod 4.$$

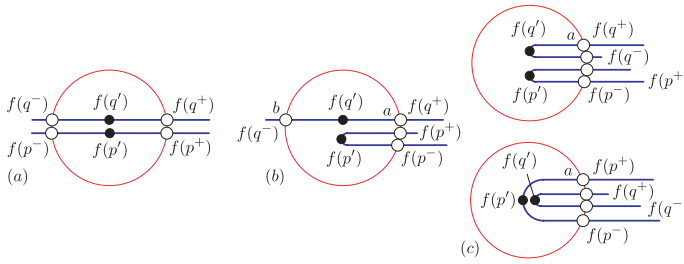

Figure 2: *Depictions of the three noncrossing conditions relevant to flat folding, specifically (a) tortilla-tortilla, (b) taco-tortilla, and (c) taco-taco.*

3.2 Facewise Definition of Global Flat Foldability

In this section, we define "facewise" layer orderings for a face-convex crease pattern. A ***facewise layer order*** function Λ defines Λ_{ij} for any two faces F_i, F_j that overlap under f, where $\Lambda_{ij} \in \{+1, -1\}$ according to whether F_i is above or below F_j, relative to a global notion of upward. Order Λ is ***valid*** if it satisfies the following five types of constraints[2], also paralleling the naming conventions of [Akitaya et al. 15].

1. **Antisymmetry constraints:** For every pair of overlapping faces F_i, F_j, define ***antisymmetry*** constraint (i, j) to be $\Lambda_{ij} = -\Lambda_{ji}$. Informally, if F_i is above F_j, then F_j is below F_i.
2. **Transitivity constraints:** For every triple of overlapping faces F_i, F_j, F_k, define ***transitivity*** constraint (i, j, k) to be that the orders Λ_{ij}, Λ_{jk}, Λ_{ki} are not all equal. Informally, if three faces overlap, there must exist a total order between them.
3. **Tortilla-tortilla constraints:** For every unfolded crease e_{ij} bounding faces F_i, F_j, define the following ***tortilla-tortilla*** constraints:

   - For every face F_k that overlaps e_{ij}, define tortilla-tortilla constraint (i, j, k, k) to be $\Lambda_{ik} = \Lambda_{jk}$.
   - For every unfolded crease e_{kl} that overlaps e_{ij} and bounds F_k, F_l where F_k overlaps F_i, define tortilla-tortilla constraint (i, j, k, l) to be $\Lambda_{ik} = \Lambda_{jl}$.

   Informally, if a face overlaps an unfolded crease, that face cannot be ordered between the two faces bounding the crease.
4. **Taco-tortilla constraints:** For every folded crease e_{ij} bounding F_i, F_j, define the following ***taco-tortilla*** constraints:

   - For every face F_k that overlaps e_{ij}, define taco-tortilla constraint (i, j, k, k) to be $\Lambda_{ik} = \Lambda_{jk}$.
   - For every unfolded crease e_{kl} that overlaps e_{ij} and bounds F_k, F_l where F_k overlaps F_i, define taco-tortilla constraint (i, j, k, l) to be $\Lambda_{ik} = \Lambda_{jk}$.

[2] We use the word 'constraint' to distinguish from 'condition' in the pointwise definition.

Λ_{ik}	Λ_{il}	Λ_{jk}	Λ_{jl}	
1	1	1	1	valid
1	1	1	−1	invalid by taco-taco (i,j,k,l)
1	1	−1	1	invalid by taco-taco (i,j,k,l)
1	1	−1	−1	valid
1	−1	1	1	invalid by taco-taco (i,j,k,l)
1	−1	1	−1	invalid by transitivity (i,k,l)
1	−1	−1	1	invalid by transitivity (i,k,l)
1	−1	−1	−1	invalid by taco-taco (i,j,k,l)
−1	1	1	1	invalid by taco-taco (i,j,k,l)
−1	1	1	−1	invalid by transitivity (i,j,k)
−1	1	−1	1	valid
−1	1	−1	−1	invalid by taco-taco (i,j,k,l)
−1	−1	1	1	invalid by transitivity (i,j,k)
−1	−1	1	−1	invalid by taco-taco (i,j,k,l)
−1	−1	−1	1	invalid by taco-taco (i,j,k,l)
−1	−1	−1	−1	valid

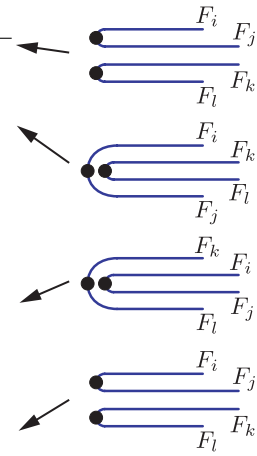

Figure 3: *Case analysis of all possible orders between faces F_i, F_j, F_k, F_l corresponding to a taco-taco constraint (i,j,k,l), assuming without loss of generality that $\Lambda_{ij} = \Lambda_{kl} = 1$.*

Informally, if a face overlaps a folded crease, that face cannot be ordered between the two faces bounding the crease.

5. **Taco-taco constraints:** For every pair of overlapping folded creases e_{ij}, e_{kl}, where e_{ij} bounds faces F_i, F_j and e_{kl} bounds faces F_k, F_l where all faces overlap, define ***taco-taco*** constraint (i,j,k,l) to be $\Lambda_{ik} + \Lambda_{jk} + \Lambda_{il} + \Lambda_{jl} = 0 \mod 4$. Informally, if two folded creases and their adjacent faces overlap, either the faces bounding one crease lie entirely above the faces bounding the other (the sum is ± 4), or the faces bounding one crease nest inside the faces bounding the other (the sum is 0). Note that this condition alone is not enough to forbid all crossings configurations between F_i, F_j, F_k, F_l, but the transitivity constraints are enough to forbid the other crossing configurations, as can be seen in Figure 3.

3.3 Equivalence

Theorem 1. *An isometric flat folding of a face-convex crease pattern has a valid pointwise layer order if and only if it has a valid facewise layer order.*

Proof. We show how to convert any valid pointwise layer order λ into a corresponding valid facewise layer order Λ, and vice versa.

$\lambda \to \Lambda$: Given λ, we construct Λ by defining Λ_{ij} for every pair of overlapping faces F_i, F_j. By convexity of F_i and F_j, the intersection of their foldings is convex and thus connected. Thus, the set $X = \{(p,q) \mid p \in F_i, q \in F_j, f(p) = f(q)\} \subseteq \operatorname{dom} \lambda$ is path-connected. By the Consistency Condition, λ is constant over X, and this

constant value is our definition of Λ_{ij}. It remains to show that Λ satisfies the facewise antisymmetry, transitivity, taco-taco, taco-tortilla, and tortilla-tortilla constraints.

- **Antisymmetry constraint** (i,j): F_i and F_j overlap, so there are points $p_i \in F_i$ and $p_j \in F_j$ with $f(p_i) = f(p_j)$ where $\lambda(p_i,p_j)$ is defined. Then $\Lambda_{ij} = \lambda(p_i,p_j)$ by our construction of Λ, $\lambda(p_i,p_j) = -\lambda(p_j,p_i)$ by the antisymmetry condition, and $-\lambda(p_j,p_i) = -\Lambda_{ji}$ by our construction of Λ. Thus $\Lambda_{ij} = -\Lambda_{ji}$.
- **Transitivity constraint** (i,j,k): F_i, F_j, and F_k all overlap, so there are points $p_i \in F_i$, $p_j \in F_j$, and $p_k \in F_k$ with $f(p_i) = f(p_j) = f(p_k)$ where $\lambda(p_i,p_j)$, $\lambda(p_j,p_k)$, and $\lambda(p_k,p_i)$ are all defined. If $\Lambda_{ij} = \Lambda_{jk}$, then $\lambda(p_i,p_j) = \lambda(p_j,p_k)$ by our construction of Λ, $\lambda(p_i,p_j) = -\lambda(p_k,p_j)$ by the antisymmetry condition, $\lambda(p_k,p_i) = \lambda(p_j,p_i)$ by the transitivity condition, $\lambda(p_k,p_i) = -\lambda(p_i,p_j)$ by the antisymmetry condition, and $\Lambda_{ki} = -\Lambda_{ij}$ by our construction of Λ. Symmetrically, if $\Lambda_{jk} = \Lambda_{ki}$ then $\Lambda_{ij} = -\Lambda_{jk}$, and if $\Lambda_{ki} = \Lambda_{ij}$ then $\Lambda_{jk} = -\Lambda_{ki}$. Thus $\Lambda_{ij}, \Lambda_{jk}, \Lambda_{ki}$ are not all equal.
- **Tortilla-tortilla constraint** (i,j,k,k): F_k and crease e_{ij} overlap, so there are points $p \in e_{ij}$ and $q \in F_k$ with $f(p) = f(q)$. Let $P' \subset P$ and $Q' \subset P$ be the 2D circles centered on p and q respectively, with radii ε chosen sufficiently small such that $P' \subset F_i \cup F_j \cup e_{ij}$ and $Q' \subset F_k$ and $P' \cup Q'$ contains no vertices; and define $p' \in P' \cap e_{ij}$, $q' \in Q'$ such that $f(p') = f(q')$. For ε' sufficiently small, there are points $p^+ \in F_i \cap P'$, $q^+ \in F_k \cap Q'$ with $f(p^+) = f(q^+)$ and points $p^- \in F_j \cap P'$, $q^- \in F_k \cap Q'$ with $f(p^-) = f(q^-)$ such that the distances from p^+, p^- to p' along P', and from q^+, q^- to q' along Q', are all ε'. Then $\lambda(p^+,q^+) = \lambda(p^-,q^-)$ by the tortilla-tortilla condition. Because $p^+ \in F_i$, $p^- \in F_j$, and $q^+, q^- \in F_k$, by our construction of Λ we have $\Lambda_{ik} = \Lambda_{jk}$.
- **Tortilla-tortilla constraint** (i,j,k,l): creases e_{ij}, e_{kl} overlap, so there are points $p \in e_{ij}$ and $q \in e_{kl}$ with $f(p) = f(q)$. Let $P' \subset P$ and $Q' \subset P$ be the 2D circles centered on p and q respectively, with radii ε chosen sufficiently small such that $P' \subset F_i \cup F_j \cup e_{ij}$ and $Q' \subset F_k \cup F_l \cup e_{kl}$ and $P' \cup Q'$ contains no vertices; and define $p' \in P' \cap e_{ij}$, $q' \in Q' \cap e_{kl}$ such that $f(p') = f(q')$. For ε' sufficiently small, there are points $p^+ \in F_i \cap P'$, $q^+ \in F_k \cap Q'$ with $f(p^+) = f(q^+)$ and points $p^- \in F_j \cap P'$, $q^- \in F_l \cap Q'$ with $f(p^-) = f(q^-)$ such that the distances from p^+, p^- to p' along P', and from q^+, q^- to q' along Q', are all ε'. Then $\lambda(p^+,q^+) = \lambda(p^-,q^-)$ by the tortilla-tortilla condition. Because $p^+ \in F_i$, $p^- \in F_j$, $q^+ \in F_k$, and $q^- \in F_l$, by our construction of Λ we have $\Lambda_{ik} = \Lambda_{jl}$.
- **Taco-tortilla constraint** (i,j,k,k): F_k and crease e_{ij} overlap, so there are points $p \in F_k$ and $q \in e_{ij}$ with $f(p) = f(q)$. Let $P' \subset P$ and $Q' \subset P$ be the 2D circles centered on p and q respectively, with radii ε chosen sufficiently small such that $P' \subset F_k$ and $Q' \subset F_i \cup F_j \cup e_{ij}$ and $P' \cup Q'$ contains no vertices; and define $p' \in P'$, $q' \in Q' \cap e_{ij}$ such that $f(p') = f(q')$. For ε' sufficiently small there are points $q^+ \in F_i \cap Q'$, $q^- \in F_j \cap Q'$, $p^+ \in F_k \cap P'$ with $f(q^+) = f(q^-) = f(p^+)$ such that the distances from p^+ to p' along P', and from q^+, q^- to q' along Q', are all ε'. Then $\lambda(q^+,p^+) = \lambda(q^-,p^+)$ by the taco-tortilla condition. Because $q^+ \in F_i$, $q^- \in F_j$, and $p^+ \in F_k$, by our construction of Λ we have $\Lambda_{ik} = \Lambda_{jk}$.

- **Taco-tortilla constraint** (i,j,k,l): creases e_{ij}, e_{kl} overlap, so there are points $q \in e_{ij}$ and $p \in e_{kl}$ with $f(p) = f(q)$. Let $P' \subset P$ and $Q' \subset P$ be the 2D circles centered on p and q respectively, with radii ε chosen sufficiently small such that $Q' \subset F_i \cup F_j \cup e_{ij}$ and $P' \subset F_k \cup F_l \cup e_{kl}$ and $P' \cup Q'$ contains no vertices; and define $p' \in P' \cap e_{kl}$, $q' \in Q' \cap e_{ij}$ such that $f(p') = f(q')$. For ε' sufficiently small, there are points $q^+ \in F_i \cap Q'$, $q^- \in F_j \cap Q'$, $p^+ \in F_k \cap P'$ with $f(q^+) = f(q^-) = f(p^+)$ such that the distances from p^+ to p' along P', and from q^+, q^- to q' along Q', are all ε'. By the taco-tortilla condition, $\lambda(p^+, q^+) = \lambda(p^-, q^-)$. Because $p^+ \in F_i$, $p^- \in F_j$, and $q^+ \in F_k$, by our construction of Λ we have $\Lambda_{ik} = \Lambda_{jl}$.
- **Taco-taco constraint** (i,j,k,l): creases e_{ij}, e_{kl} overlap, so there are points $p \in e_{ij}$ and $q \in e_{kl}$ with $f(p) = f(q)$. Let $P' \subset P$ and $Q' \subset P$ be the 2D circles centered on p and q respectively, with radii ε chosen sufficiently small such that $P' \subset F_i \cup F_j \cup e_{ij}$ and $Q' \subset F_k \cup F_l \cup e_{kl}$ and $P' \cup Q'$ contains no vertices; and define $p' \in P' \cap e_{ij}$, $q' \in Q' \cap e_{kl}$ such that $f(p') = f(q')$. For ε' sufficiently small, there are points $p^+ \in F_i \cap P'$, $p^- \in F_j \cap P'$, $q^+ \in F_k \cap Q'$, $q^- \in F_l \cap Q'$ with $f(p^+) = f(p^-) = f(q^+) = f(q^-)$ such that the distances from p^+, p^- to p' along P', and from q^+, q^- to q' along Q', are all ε'. By the taco-taco condition, $\lambda(p^+, q^+) + \lambda(p^-, q^+) + \lambda(p^+, q^-) + \lambda(p^-, q^-) = 0 \mod 4$. Because $p^+ \in F_i$, $p^- \in F_j$, $q^+ \in F_k$, and $q^- \in F_l$, by our construction of Λ we have $\Lambda_{ik} + \Lambda_{jk} + \Lambda_{il} + \Lambda_{jl} = 0 \mod 4$.

$\Lambda \to \lambda$: For any two noncrease points p, q, let F_i, F_j be the faces containing p, q respectively. Define $\lambda(p,q) = \Lambda_{ij}$ wherever $f(p) = f(q)$.

- **Antisymmetry condition**: Given points p, q with $f(p) = f(q)$, let F_i, F_j be the faces containing p, q respectively. Then $\lambda(p,q) = \Lambda_{ij}$ by our construction of Λ, $\Lambda_{ij} = -\Lambda_{ji}$ by the antisymmetry constraint, and $-\Lambda_{ji} = -\lambda(q,p)$ by our construction of Λ. Thus $\lambda(p,q) = -\lambda(q,p)$.
- **Transitivity condition**: Given points p, q, r with $f(p) = f(q) = f(r)$, let F_i, F_j, F_k be the faces containing p, q, r respectively. If $\lambda(p,q) = -\lambda(r,q)$, then $\Lambda_{ij} = -\Lambda_{kj}$ by our construction of Λ, $\Lambda_{ij} = \Lambda_{jk}$ by the antisymmetry constraint, $\Lambda_{ki} = -\Lambda_{ij}$ by the transitivity constraint, $\Lambda_{ki} = \Lambda_{ji}$ by the antisymmetry constraint, and thus by our construction of Λ, we have $\lambda(r,p) = \lambda(q,p)$.
- **Consistency condition**: Given (p,q), (p',q') that are path-connected in $\mathrm{dom}(\lambda)$, since λ is not defined for crease points, p, p' must exist in the same face F_i, and q, q' must exist in the same face F_j. By our construction of Λ, $\lambda(p,q) = \Lambda_{ij} = \lambda(p',q')$.
- **Tortilla-tortilla condition**: Consider points p', q', p^+, p^-, q^+, q^- relevant to a tortilla-tortilla condition, where $f(p^+) = f(q^+)$ and $f(p^-) = f(q^-)$ and p', q' are either unfolded crease points or not crease points. There are three cases:
  - If neither are crease points, then p^+, p^- are part of the same face F_i, and q^+, q^- are part of the same face F_j, and by our construction of Λ, we have $\lambda(p^+, q^+) = \Lambda_{ij} = \lambda(p^-, q^-)$.

- If exactly one is a crease point, assume it is p' without loss of generality. Let F_i be the face containing p^+, F_j be the face containing p^-, and F_k be the face containing q^+, q^-. Then F_i and F_j are adjacent along an unfolded crease e_{ij} that contains p' which overlaps F_k at q'. Then $\Lambda_{ik} = \Lambda_{jk}$ by the tortilla-tortilla constraint, and by our construction of Λ, we have $\lambda(p^+, q^+) = \lambda(p^-, q^-)$.
- Otherwise, if both are crease points, let F_i, F_j, F_k, F_l be the faces containing p^+, p^-, q^+, q^- respectively. Then F_i, F_j are adjacent along an unfolded crease e_{ij} that contains p' and F_l, F_k are adjacent along an unfolded crease e_{kl} that contains q'. If e_{ij} and e_{kl} overlap, then $\Lambda_{ik} = \Lambda_{jl}$ by the tortilla-tortilla constraint, and $\lambda(p^+, q^+) = \lambda(p^-, q^-)$ by our construction of Λ. Otherwise, e_{ij} overlaps F_k and F_l, and e_{kl} overlaps F_i and F_j, so by the tortilla-tortilla constraint $\Lambda_{ik} = \Lambda_{jk} = \Lambda_{jl}$. By our construction of Λ, we have $\lambda(p^+, q^+) = \lambda(p^-, q^-)$.

- **Taco-tortilla condition**: Consider points p', q', p^+, p^-, q^+, q^- relevant to a taco-tortilla condition, where $f(p^+) = f(q^+) = f(q^-)$, q' is a folded crease point, and p' is either a unfolded crease point or not a crease point. There are two cases:
  - If p' is not a crease point, let F_i, F_j be the faces containing q^+ and q^- respectively, and let F_k be the face containing p^+, p^-. Then F_i, F_j are adjacent along a folded crease e_{ij} that contains q', and e_{ij} overlaps F_k. Then $\Lambda_{ik} = \Lambda_{jk}$ by the taco-tortilla constraint, and by our construction of Λ, we have $\lambda(q^+, p^+) = \lambda(q^-, p^+)$.
  - Otherwise, if p' is an unfolded crease point, let F_i, F_j, F_k, F_l be the faces containing q^+, q^-, p^+, p^- respectively. Then F_i, F_j are adjacent along a folded crease e_{ij} that contains q', and F_k, F_l are adjacent along an unfolded crease e_{kl} that contains p'. If e_{ij} and e_{kl} overlap, then $\Lambda_{ik} = \Lambda_{jk}$ by the taco-tortilla constraint, and $\lambda(p^+, q^+) = \lambda(p^-, q^-)$ by our construction of Λ. Otherwise, e_{ij} overlaps F_k, so $\Lambda_{ik} = \Lambda_{jk}$ by the taco-tortilla constraint. By our construction of Λ, we have $\lambda(q^+, p^+) = \lambda(q^-, p^-)$.

- **Taco-taco condition**: Consider points p', q', p^+, p^-, q^+, q^- relevant to a taco-taco condition, where $f(p^+) = f(p^-) = f(q^+) = f(q^-)$ and p', q' are folded crease points. Let F_i, F_j, F_k, F_l be the faces containing p^+, p^-, q^+, q^- respectively. Then F_i, F_j are adjacent along a folded crease e_{ij} that contains p', and F_k, F_l are adjacent along a folded crease e_{kl} that contains q'. There are two cases:
  - If e_{ij} and e_{kl} do not overlap, then e_{ij} overlaps F_k and F_l, and e_{kl} overlaps F_i and F_j, so by the taco-tortilla constraint, $\Lambda_{ik} = \Lambda_{jk} = \Lambda_{il} = \Lambda_{jl}$. In particular, $\Lambda_{ik} + \Lambda_{jk} + \Lambda_{il} + \Lambda_{jl} = \pm 4 = 0 \mod 4$. By our construction of Λ, we have $\lambda(p^+, q^+) + \lambda(p^-, q^+) + \lambda(p^+, q^-) + \lambda(p^-, q^-) = 0 \mod 4$.
  - Otherwise, if e_{ij} and e_{kl} overlap, then we can apply the taco-taco constraint, $\Lambda_{ik} + \Lambda_{jk} + \Lambda_{il} + \Lambda_{jl} = 0 \mod 4$. By our construction of Λ, we have $\lambda(p^+, q^+) + \lambda(p^-, q^+) + \lambda(p^+, q^-) + \lambda(p^-, q^-) = 0 \mod 4$.

Thus we can convert a valid pointwise layer order λ into a corresponding valid facewise layer order Λ, and visa versa, completing the proof. □

4 Checking Validity of a Facewise Layer Order

Given a facewise layer order Λ of an isometric flat folding f of a well-bounded face-convex crease pattern of size n, we can naïvely check whether it is valid in $O(n^3)$ time by checking each face pair, crease pair, crease-face pair, and face triple for overlap. Because the crease pattern is well-bounded and the intersection of convex faces can be evaluated in time linear in the degree of the faces, each of these overlaps takes $O(1)$ time on average to evaluate. Thus it will take $O(n^2)$ time to compute the overlap between all pairs, and $O(n^3)$ time to compute the overlap between all triples. Each constraint corresponds to one of these overlaps, and can be checked in $O(1)$ time.

- If face pair F_i, F_j overlaps, we can check that $\Lambda_{ij}, \Lambda_{ji}$ are both defined and satisfy antisymmetry in $O(1)$ time.
- If a crease pair or crease-face pair overlaps, we can check whether the overlaps between locally adjacent faces satisfy the preconditions for a tortilla-tortilla, taco-tortilla, or tortilla-tortilla constraint, and then check it in $O(1)$ time.
- If a face triple overlaps, we can check transitivity in $O(1)$ time.

However, we can do better for foldings with low ply by making use of the ***cell adjacency graph***: the planar straight-line graph formed by the union of $f(V)$ and $f(E)$. This structure was previously exploited in both [Mitani 08] and [Eppstein 23]. The cell adjacency graph partitions the plane into open subsets: bounded subsets are called the ***cells*** C of the cell adjacency graph. Note that the overlap set must be identical for all points within the same cell, which we identify as the overlap set of the cell. The ***cell vertices*** are points $f(V)$, together with any points of $f(E)$ that are on the boundary of more than two cells. Removing cell vertices from $f(E)$ results in a set of disjoint open segments that comprise the ***cell edges***. For a well-bounded crease pattern, we let $m = |C| = O(n^2)$ denote the size of the cell adjacency graph.

Theorem 2. *A facewise layer order Λ of an isometric flat folding f of a well-bounded face-convex crease pattern of size n, with ply p and cell adjacency graph of size m, can be checked for validity in $O(\min\{n^2 p, n^2 + mp^2\})$ time.*

Proof. First, check antisymmetry of Λ directly in $O(n^2)$ time. Then, we construct the cell adjacency graph's vertices, edges, and cells by computing the arrangement of the folded images of crease pattern creases. This can be done via a standard plane sweep in $O(n \log n + m)$ time. In addition, during the line sweep, it is possible to build a dictionaries mapping each cell to: its adjacencies to other cells via shared edges in $O(m)$ time, and its overlap set of faces in $O(mp)$ time, since the size of each cell's overlap set is at most p.

Next, for each cell, we compute the unique linear order of the faces in its the overlap set using Λ as a constant-time comparator. To do this, start with any cell C and sort the faces in its overlap set in $O(p^2)$ time, and check that the order between each pair of faces F_i, F_j is consistent with Λ_{ij}. Then traverse the cell adjacency graph (e.g., via DFS) to incrementally sort the other overlap sets of other cells. To transition from cell C to cell C', remove any deleted faces and add any new faces;

to add a face, do a linear scan to see where to put it, and to make sure it's order is consistent with Λ.[3]

This step takes $O(mp^2)$ time, since there are m cells and each face update takes $O(p)$ time. However, the running time is also bounded by $O(n^2p)$: each added face in a transition between cells is due to a crease pattern crease between the cells; and each crease (of which there are $|E| = O(n)$) overlaps at most $O(n)$ cell edges; so the total number of added faces is $O(n^2)$, each of which takes $O(p)$ time. If we are unable to find a linear order for each cell that is consistent with Λ, that means for some three faces F_i, F_j, F_k in some cell's overlap set, there is no linear order consistent with Λ, which can only occur if transitivity constraint (i, j, k) is violated. Conversely, if a consistent linear order for each cell can be found, Λ satisfies all transitivity constraints.

Finally, we check the remaining constraints in $O(mp)$ time. For each edge e between two cells C, C', check each constraint type at the edge in $O(p)$ time:

- **Tortilla-tortilla:** Linear scan the order of the overlap set for both C and C' using a two-finger algorithm to check that orders are consistent for faces that are in both C and C'.
- **Taco-taco:** For each crease c between faces F_i, F_j in the overlap set of C such that c overlaps e and F_i is below F_j, replace F_i in C's order with an open parentheses and replace F_j with a close parentheses. Then we can check that the parentheses in the stack are balanced via a linear scan with a stack.
- **Taco-tortilla:** Augment the above taco-taco algorithm as follows: for each crease between faces $F_i \in C, F_j \in C'$, replace F_i in C's order with an open parentheses and add a close parentheses at infinity for F_j.

So overall, this algorithm runs in $O(\min\{n^2p, n^2 + mp^2\})$ time as desired. □

5 Computing Folded States

Existing software that seeks to compute one or more folded states of a crease pattern include Oripa [Mitani 12], Orihime [Meguro 21], and Oriedita [Oriedita 23]. This software all generally follows the algorithm described by [Mitani 08] to compute folded states — (1) construct the overlap set for each cell of the adjacency graph, and (2) brute-force search for a linear order of faces within each cell that avoids self-intersection of the paper — though it is not clear whether their methods correspond to the same set of facewise constraints that are presented in this paper. This algorithm necessarily requires exponential time to compute an exponential number of folded states.

5.1 Constraint Graph

We present an alternative algorithm which constructs a bipartite ***constraint graph*** between the facewise layer orders Λ_{ij} (the ***variables***) and the set of validity con-

[3]Note that if we already knew the flat folding were valid, we could binary search to perform face updates in $O(\log p)$, e.g., via AVL Trees; but to verify Λ, we must check the order of each added face with respect to every other face in the cell, to ensure Λ does not violate transitivity.

straints defined in Section 3.2, with edges connecting each constraint to its associated variables. Note that each vertex associated with a constraint has constant degree, so the number of edges of the constraint graph is linear in the number of vertices. As observed at the start of Section 4, we can compute the set of variables and the set of constraints (and thus the entire constraint graph) in $O(n^3)$ time. For foldings with low ply p, it is possible to compute the transitivity constraints in $O(n^2p^2)$ time. The construction is similar to the proof of Theorem 2: walk the cell adjacency graph and compute all pairs of faces in a cell that overlap each added face in $O(p^2)$ time.

5.2 Initial Variable Assignment based on Crease Assignment

First, it is common for some or all folded creases in a crease patterns to be ***assigned*** as either mountain or valley relative to one side of the paper. Effectively, a crease assignment for crease e means that, for the pair of faces F_i, F_j that bound e, the assignment of Λ_{ij} is prescribed. Further, partial knowledge of the assignment of some variables can allow us to infer the assignment of other variables. For example, if faces F_i, F_j, F_k are known to share transitivity constraint (i, j, k), and we know that $\Lambda_{ij} = \Lambda_{jk} = +1$, then we can infer that $\Lambda_{ki} = -1$, because they cannot all be equal.

As variables are assigned, for each constraint, the variables involved in the constraint may be in one of three states: unassigned, or assigned either $+1$ or -1. Because each constraint relates at most four variables, we can precompute a constant-sized lookup table storing every partially assigned state of the variables associated with each type of constraint, identifying that either:

1. some unassigned variable's assignment is implied by the partial assignment;
2. no assignment of unassigned variables can be inferred; but some assignment of the unassigned variables can satisfy the constraint; or
3. no assignment of the unassigned variables can satisfy the constraint.

We make these initial assignments via breadth-first search. Initialize a queue with each variable associated with crease assignments. To process a variable from the queue, record the variable's assignment and mark the vertex associated with that variable. Then, for each constraint adjacent to the variable in the constraint graph, check in the lookup table whether any unassigned variables can be inferred and add them to the queue. If ever a variable is assigned conflicting values, or a constraint cannot be satisfied, then the crease pattern does not admit a valid layer order. This initial assignment stage runs in $O(n^3)$ time because each variable gets added to the queue at most once, and each edge of the graph gets traversed $O(1)$ times.

5.3 Component Separation

At this point, we could do a brute-force exponential search for an assignment of the remaining variables that satisfy the constraints. However, for many crease patterns, we can speed up the search by identifying groups of variables that are assignable independently from each other. After initial assignment, remove all assigned variable vertices from the constraint graph to form the ***pruned constraint graph***, which may be disconnected into multiple components. If it is, the assignment of variables in one

component is independent from the assignment of variables in another component. In practice, these components are often small, so an exponential search on each one is much faster than an exponential search on the whole graph.

Suppose each component i contains t_i unassigned variables, where each Λ_{ij} involves two faces F_i, F_j, for a total of $n_i \leq 2t_i$ distinct involved faces. Thus the number of constraints in component i of the pruned constraint graph is at most $O(n_i^3) = O(t_i^3)$.[4] We can solve component i by trying all 2^{t_i} binary assignments to the t_i unassigned variables, and for each such assignment, checking the $O(t_i^3)$ constraints in $O(t_i^3)$ time. Thus the total running time is $O(t_i^3 2^{t_i})$.

After enumerating all assignments for each component, we can list all folded states via the Cartesian product of the assignments from each independent component; or we can count the total number of folded states by multiplying the number of assignments found for each component. Storing only the assignments for each component provides an implicit representation of all folded states, which can represent an exponential number of folded states in polynomial space; see Figure 5 for one such example. This immediately leads to the following theorem.

Theorem 3. *Given a well-bounded face-convex crease pattern with k pruned constraint graph components, where component i has t_i unassigned variables, all of its valid folded states can be implicitly computed in $O\left(n^3 + \sum_{i=1}^{k} t_i^3 2^{t_i}\right)$ time. If s_i is the number of valid assignments of the variables from component i, then we can store an implicit representation of all $\prod_{i=1}^{k} s_i$ states in $O(\sum_{i=1}^{k} s_i)$ space.*

The above algorithm works well in practice, but it is actually possible to infer the assignment of even more variables, beyond the local inferences presented in Section 5.2. A followup paper [Ku et al. 24] treats the set of the nontransitivity constraints as a linear system to infer the initial assignment of a larger number of variables, and uses additional polynomial-time procedures to increase the separation of initially unassigned variables into components. Such techniques can reduce the number of unassigned variables in each component, often leading to exponential time gains in the final brute-force step of the solve.

5.4 Implementation

This algorithm has been implemented in an open-source web application called ***Flat-Folder*** [Ku 22]. A primary motivator for the development of this algorithm and software is to generate folded states in FOLD format [Demaine et al. 16b] as test case input for future origami simulation software. Thus, we have compiled three crease pattern datasets that are publicly available from the Flat-Folder GitHub repository [Ku 22]:

- The **instagram** dataset contains 366 flat-foldable crease patterns from 58 different origami designers from around the world. Most of these crease patterns are fully mountain/valley assigned, with number of faces ranging from $n = 7$ to $n = 11{,}859$.

[4] We conjecture that this bound is not tight. The number c_i of transitivity constraints can be cubic in the number n_i of faces, while the number t_i of variables can be quadratic in the number n_i of faces. But these are both upper bounds, and it is not clear that they maintain a $c_i = O(t_i^{3/2})$ relationship.

- The **grids** dataset contains 375 flat-foldable crease patterns designed by Daniel Brown. These crease patterns fold all possible diagonal-grid color-change patterns that have four-fold rotational symmetry on square grids having side length ≤ 4. All the crease patterns in this data set have vertices that lie on an integer grid. Another followup paper [Brown and Ku 24] discusses the design of the crease patterns in this data set.
- The **unsatisfiable** dataset contains 12 crease patterns that are known to be not flat foldable.

Flat-Folder can successfully find at least one valid folded state for every example in the dataset within seconds or minutes, and can compute all states for most examples in a similar amount of time. Of course, computing all states for examples having large components in their pruned constraint graph will take longer, since our running time scales exponentially with component size. For efficiency, we cap the number of folded states generated per component to 10,000, and report metrics about each solve in a spreadsheet within the repository.

The example with the most found folded states under this filter is Satoshi Kamiya's Ryujin 2.1 (instagram #100). This crease pattern consists of 5,325 vertices, 11,097 edges, and 5,773 faces. The constraint graph consists of 453,998 variables, 119,105 taco-taco constraints, 586,781 taco-tortilla constraints, 0 tortilla-tortilla constraints, and 30,523,575 transitivity constraints. The pruned constraint graph consists of 248 components, 238 of which have two satisfying assignments, three of which have three satisfying assignments, four of which have four satisfying assignments, one of which has five satisfying assignments, one of which has 950 satisfying assignments, and one of which has 633,184 satisfying assignments (when evaluated with no cap on states per component). Together these comprise over $9 \cdot 10^{84}$ folded states, specifically:

9,182,612,107,624,936,419,438,282,251,782,996,996,651,024,099,336,268,033,225,057,196,673,656,643,305,603,072,000.

See [Ku 22] and [Ku et al. 24] for more details on the examples and software performance.

6 Bounding the Maximum Number of Folded States

In this section, we give bounds on the maximum number of possible folded states in terms of the number n of convex faces in the crease pattern. We prove that the maximum number of folded states is $2^{\Theta(n^2)}$ for convex paper, and $2^{\Omega(n \log n)}$ for square paper, independent of whether the creases are assigned.

Theorem 4. *A face-convex isometrically flat-foldable crease pattern with n faces can admit at most $2^{O(n^2)}$ valid folded states, even when the creases are unassigned.*

Proof. Each valid folded state corresponds to a different facewise layer order Λ. There are at most $n(n-1)$ layer orders between pairs of faces, and each layer order has two possible assignments. So at most $2^{n(n-1)} = 2^{O(n^2)}$ valid layer orders assignments, and thus valid folded states, can possibly exist. □

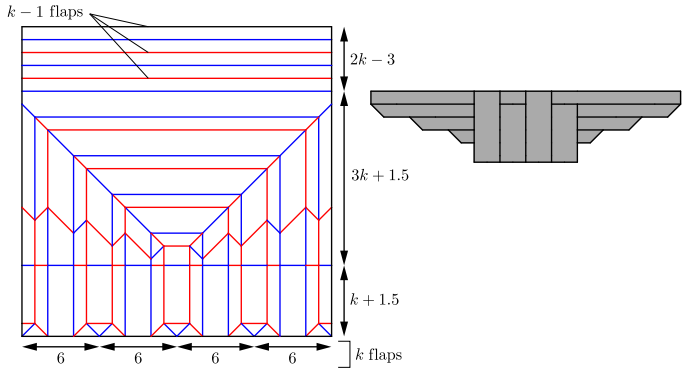

Figure 4: *A generalizable square crease pattern on a $6k \times 6k$ grid with $\Theta(k)$ faces admitting $2^{\Omega(k \log k)}$ valid folded states, here for $k = 4$.*

Section 2 of [Bern and Hayes 96] provides a rough sketch of a construction for a square crease pattern that admits $n^{\Omega(n)} = 2^{\Omega(n \log n)}$ valid folded states, and questions whether there exist crease patterns that can admit more solutions. While the idea behind their construction works, they do not provide specifics of their construction, so it is hard to verify. In particular, care needs to be taken with the initial pleating step when folding the square down to a rectangle, so as not to introduce too many crease pattern faces. Below we provide a precise construction for a square crease pattern that achieves the same bound with the same general approach, as well as a new construction for a crease pattern on convex paper that meets the asymptotic upper bound on valid folded states.

Theorem 5. *A face-convex isometrically flat-foldable crease pattern of a square paper with n faces can have $n^{\Omega(n)} = 2^{\Omega(n \log n)}$ valid folded states, even when the creases are assigned.*

Proof. Consider the family of crease patterns on square paper with assigned creases aligned to a $6k \times 6k$ grid as shown in Figure 4. This crease pattern has $n = 2(k-1) + 17k + 2 = 19k = \Theta(k)$ faces, where k independent flaps on the bottom overlap $k-1$ pleats on the top. Each of the k bottom flaps can be in k different positions relative to the top $k-1$ pleats, leading to $k^k = 2^{k \lg k} = 2^{\Theta(n \log n)}$ valid folded states. □

Theorem 6. *A face-convex isometrically flat-foldable crease pattern of a convex paper with n faces can have $2^{\Omega(n^2)}$ valid folded states, even when the creases are assigned.*

Proof. Consider the family of crease patterns on rectangular paper with height 1 and width $2k(k+2) - 1$ for $k \geq 1$, containing 45°-diagonal mountain creases as shown in Figure 5. This crease pattern has $n = 4k = \Theta(k)$ faces, where k red faces each independently overlap k blue faces when folded. Specifically, for red face F_i

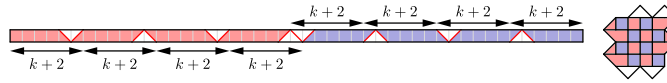

Figure 5: *A generalizable crease pattern on* $1 \times 2k(k+2)-1$ *rectangular paper with* $\Theta(k)$ *faces admitting* $2^{\Omega(k^2)}$ *valid folded states, here for* $k = 4$.

and blue face F_j, layer order Λ_{ij} can be independently assigned to either $\{+1,-1\}$. Thus, this crease pattern admits exactly $2^{k^2} = 2^{\Omega(n^2)}$ valid folded states. □

We conjecture that it may not be possible to achieve $2^{\Omega(n^2)}$ valid folded states from a square. Reaching that bound seems to require a folding with at least $\Omega(n^2)$ independent layer-order pairs, i.e., there must be $\Theta(n)$ flaps that must each independently overlap $\Theta(n)$ other flaps (as for the convex construction of Theorem 6). In order to get a flap to independently overlap many other flaps, some of these flaps should have high aspect ratio. But making flaps with high aspect ratio from a square seems to require the production of many crease pattern faces: making a flap with aspect ratio r seems to require the addition of $\Theta(r)$ faces. But for the crease pattern to retain $O(n)$ faces, there can only be $O(1)$ faces on average associated with each independent flap. The square bound construction of Theorem 5 does not have any high-aspect-ratio flaps; instead it gains exponential blowup from having a linear number of separable components, each having a linear number of valid states. The convex construction by contrast has a quadratic number of separable components, each with a constant number of states, which seems difficult to achieve from square paper.

Acknowledgments

We thank the anonymous referees for their helpful comments.

References

[Akitaya et al. 15] Hugo A. Akitaya, Kenneth C. Cheung, Erik D. Demaine, Takashi Horiyama, Thomas C. Hull, Jason S. Ku, Tomohiro Tachi, and Ryuhei Uehara. "Box pleating is hard." In *Japanese Conference on Discrete and Computational Geometry and Graphs*, pp. 167–179. Springer, 2015.

[Bern and Hayes 96] Marshall Bern and Barry Hayes. "The complexity of flat origami." In *Proceedings of the 7th Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 175–183, 1996.

[Brown and Ku 24] Dan Brown and Jason S. Ku. "Folding all 4×4 Rotationally-Symmetric Diagonal-Grid 2-Color Patterns." In *8OSME*, 2024.

[Demaine and O'Rourke 07] Erik D. Demaine and Joseph O'Rourke. *Geometric Folding Algorithms: Linkages, Origami, Polyhedra*. Cambridge University Press, 2007.

[Demaine et al. 16a] Erik D. Demaine, Jason S. Ku, and Robert J. Lang. "A New File Standard to Represent Folded Structures." In *Abstracts from the 26th Fall Workshop on Computational Geometry*, 2016. See https://github.com/edemaine/fold for the latest FOLD standard.

[Demaine et al. 16b] Erik D. Demaine, Jason S. Ku, and Robert J. Lang. "A new file standard to represent folded structures." In *Abstr. 26th Fall Workshop Computat. Geometry*, pp. 27–28, 2016.

[Demaine et al. 20] Erik D. Demaine, Adam C. Hesterberg, and Jason S. Ku. "Finding Closed Quasigeodesics on Convex Polyhedra." In *Proceedings of the 36th International Symposium on Computational Geometry (SoCG 2020)*, pp. 33:1–33:13, 2020. See full paper at arXiv:2008.00589.

[Eppstein 23] David Eppstein. "A Parameterized Algorithm for Flat Folding." In *Proceedings of the 35th Canadian Conference on Computational Geometry*, pp. 35–42, 2023.

[Jia et al. 23] Yiyang Jia, Jun Mitani, and Ryuhei Uehara. "Clarifying the Difference between Origami Fold Models by a Matrix Representation." *Thai Journal of Mathematics* 21:4 (2023), 1061–1079. Available online (https://thaijmath2.in.cmu.ac.th/index.php/thaijmath/article/view/1565).

[Justin 94] Jacques Justin. "Towards a Mathematical Theory of Origami." In *Proceedings of the 2nd International Meeting of Origami Science and Scientific Origami*, edited by Koryo Miura, pp. 15–29. Otsu, Japan, 1994.

[Ku et al. 24] Jason S. Ku, Akira Terao, and Kenji N. Terao. "An Algebraic Approach to Layer Ordering Constraints for Origami Flat-Foldability." In *8OSME*, 2024.

[Ku 22] Jason S. Ku. "Flat-Folder: A Crease Pattern Solver." https://github.com/origamimagiro/flat-folder, 2022.

[Lang and Demaine 06] Robert J. Lang and Erik D. Demaine. "Facet ordering and crease assignment in uniaxial bases." In *Origami^4: Proceedings of the 4th International Conference on Origami in Science, Mathematics, and Education*, pp. 189–205. AK Peters Pasadena, California, 2006.

[Meguro 21] Toshiyuki Meguro. "Origami Homepage: Orihime." http://mt777.html.xdomain.jp/index.html, 2021.

[Mitani 08] Jun Mitani. "The folded shape restoration and the rendering method of origami from the crease pattern." In *Proc. Int. Conf. on Geometry and Graphics*, pp. 1–7, 2008.

[Mitani 12] Jun Mitani. "ORIPA: Origami Pattern Editor." https://mitani.cs.tsukuba.ac.jp/oripa/, 2012.

[Morgan 12] Tom Morgan. "Map Folding." Master's thesis, Massachusetts Institute of Technology, 2012. Available online (https://erikdemaine.org/theses/tmorgan.pdf).

[Nishat 13] Rahnuma Islam Nishat. "Map Folding." Master's thesis, University of Victoria, 2013. Available online (https://dspace.library.uvic.ca/items/f18edf3b-a3ed-4940-92bd-901d4c35f01a).

[Oriedita 23] Oriedita. "Oriedita Main Page." https://oriedita.github.io, 2023.

[Schneider 04] Jonathan Schneider. "Flat-foldability of origami crease patterns." Manuscript, Swarthmore College, 2004. https://www.sccs.swarthmore.edu/users/05/jschnei3/origami.pdf.

[Uehara 10] Ryuhei Uehara. "Stamp Foldings with a Given Mountain-Valley Assignment." In *Origami^5: Proceedings of the 5th International Conference on Origami in Science, Mathematics and Education*, pp. 585–597. Singapore: A K Peters, 2010.

Hugo A. Akitaya
University of Massachusetts, Lowell, MA, USA, e-mail: hugo_akitaya@uml.edu

Erik D. Demaine
Massachusetts Institute of Technology, Cambridge, MA, USA, e-mail: edemaine@mit.edu

Jason S. Ku
National University of Singapore, Singapore, e-mail: jasonku@nus.edu.sg