Facet Ordering and Crease Assignment in Uniaxial Bases

Robert J. Lang and Erik D. Demaine

1 Introduction

A renaissance of origami began in the mid-twentieth century as the exposure of the works of Japanese master Akira Yoshizawa inspired a wave of creation of new design that continues unabated today. Beginning in the 1960s with the development of box pleating and through the ensuing decades, the state of complexity and sophistication of origami designs grew steadily, leading to ever-more challenging subjects as origami artists including Elias, Hulme, Engel, and Maekawa developed techniques to design origami shapes with specified features.

By the 1990s, these techniques began to assume mathematical form. At roughly the same time, Toshiyuki Meguro in Japan and one of the authors (Robert Lang) in America devised a set of techniques based on disk packing that allowed an origami artist to design a basic form, called a *base*, with an arbitrary configuration of flaps [8, 9]. These techniques and their subsequent diffusion through the origami communities on both sides of the Pacific led to a wave of new origami creation and an “arms race” (or perhaps a “legs race” is more accurate) of arthropodal invention known informally as the “Bug Wars.”

While origami artists were not overly concerned with the mathematical niceties of circle packing so long as it worked in practice, in the mid- to late 1990s, origami began to receive attention from computational geometers, including the other author (Erik Demaine), who began investigating origami design issues from a computational perspective, examining questions of computational complexity, existence, and formal algorithms for the solution of various folding problems.

The first computational geometric description of the *circle packing* design algorithm was provided by Robert Lang in 1996 [9]. We showed that a broad class of origami structure—called *uniaxial bases*—could be designed by solving a nonlinear constrained optimization (NLCO) problem that, under certain conditions, amounted to a disk packing. The construction of the base occurred in two steps. After describing the desired base by a weighted tree graph, one constructed an NLCO from the properties of the tree graph and then solved it for a set of points that ultimately became key vertices of the desired crease pattern. In the second step, the pattern of vertices was filled in with a set of creases utilizing patterns known as *molecules* (a name and concept coined by Meguro). The resulting crease pattern was foldable into a base whose flaps possessed the lengths and connections specified by the original tree graph. We called this algorithm *tree theory*, and incorporated it into a freely available software tool for origami design, TreeMaker [11].

A complete description of a flat folded origami shape requires three things: the locations of the creases; their assignment (mountain or valley); and the stacking order of the folded layers. in our original analysis, we noted that the creases could be classified into four families and that the crease assignments for some of the families were known, and we commented that the remaining crease assignments could *usually* be determined by a bit of experimentation.

However, lack of a complete description of crease assignment (and the related information of stacking order) has remained a hole in tree theory for some ten years. It is by no means assured that the solution to either problem is trivial; in general, finding crease assignment and/or stacking order for a given crease pattern is NP-complete [3]. On the other hand, polynomial-time algorithms for crease assignment and stacking order for closely related problems [2] have been described, giving grounds for hope for the existence of a general algorithm.

In this work, we describe for the first time a relatively simple algorithm for crease assignment in a uniaxial base. The method hinges on the construction of an ordering of the facets, expressed as a digraph, that allows a mountain-valley assignment that satisfies the Justin conditions on layer ordering [5]. Although we defer the proof of the algorithm to a future work due to space limitations, we present it here; this algorithm, plus the NLCO of tree theory, provides a complete computation algorithm for the crease pattern, crease assignment, and stacking order for an arbitrary uniaxial origami base.

2 Tree Theory

2.1 Optimization

We begin with a brief recapitulation of tree theory and relevant terms and concepts.

A *uniaxial base* is a folded shape that can be partitioned into distinct regions, called *flaps*, and for which a particular line can be defined, called the *axis*. Each flap must be incident to the axis and the perpendicular projection of the flap onto the axis must be fully contained within each flap. The connections between flaps are called *hinges*, and the hinges are all perpendicular to the axis. Each flap has a defined *length*, which is simply the length of its projection upon the axis.

The lengths and connections between flaps in a uniaxial base can be described by an edge-weighted graph in which edges represent flaps, edge weights are the flap lengths, and the nodes of the graph represent connections between flaps. Since the paper is simply connected, any shape folded from the paper must be simply connected, and therefore its graph must be as well. We call such a graph the *tree graph* of the base. Given a uniaxial base, constructing its tree graph is simple and straightforward. (See Figure 1.)

Many of the classic bases of origami, and many modern bases of great complexity, are uniaxial bases (although of course many are not). The property of uniaxiality permits a solution of the inverse problem: given a tree graph and a sheet of paper, construct a uniaxial base with the given tree graph (or one that differs by only a proportionality constant), using an algorithm which we call the *TreeMaker algorithm*.

We will present examples in which the sheet of paper is a square, but the algorithm is applicable to any convex polygon P. We first classify nodes and edges within the tree graph: a node of degree 1 is a *leaf node*; all others are *branch nodes*. Similarly, any edge incident to a leaf node is a *leaf edge*; all others are *branch edges*. With each leaf node n_i, we associate a vertex of the crease pattern v_i, which we call a *leaf node vertex*. (Branch nodes do not have unique associated vertices.) The first step of the TreeMaker algorithm is to solve for the positions of the leaf node vertices within the paper P.

Since the tree graph is simply connected, there is a unique shortest path between any two nodes n_i and n_j, called a *tree path*, consisting of an ordered list of nodes and edges. With each such tree path, we associate a *tree length* l_{ij}, which is the sum of the lengths of the edges of the tree path. In our prior work [9], we showed that for any uniaxial base, a necessary condition on the crease pattern was that

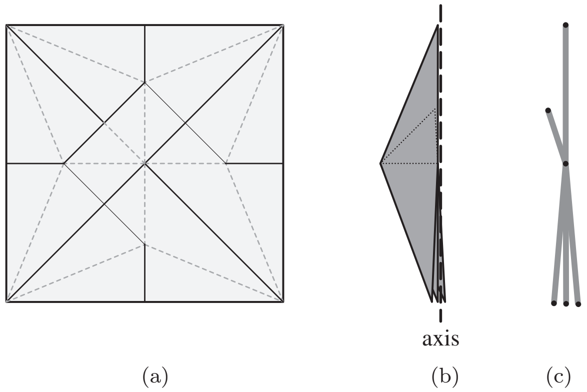

Figure 1. (a) Crease pattern of a uniaxial base. (b) Folded form of the same base, with axis highlighted. (c) Its tree graph. Note that one flap is hidden inside the folded form.

$$|v_i - v_j| \geq l_{ij} \tag{1}$$

for all possible pairs of leaf node vertices.

For an arbitrary tree graph, there is no guarantee that a solution to Equation (1) exists. We therefore introduce a scaling factor m, which is the ratio between the length of an edge of the tree graph and the length of the corresponding flap in the folded form. With this introduction, Equation (1) becomes

$$|v_i - v_j| \geq ml_{ij} \tag{2}$$

for all possible pairs of leaf node vertices.

The scale is a measure of the efficiency of the folded base; a base with a large scale will have a folded form that is relatively large compared to the starting paper. The largest potential base is therefore given by the extremum of the following nonlinear constrained optimization problem:

$$\text{maximize } m \text{ over } \{v_i\} \text{ subject to } |v_i - v_j| \geq ml_{ij}, v_i \in P. \tag{3}$$

A straight line between any two vertices v_i, v_j in the crease pattern is also called a *path* (not to be confused with tree paths, which are defined on the tree graph rather than on the crease pattern). For every tree path between leaf nodes, there is a corresponding path between leaf node vertices on the crease pattern, which we call a *leaf path*. The length of any path in the crease pattern is the Euclidean distance between its endpoints. If the length of a leaf path is equal to its scaled length on the tree graph, corresponding to equality in Equation (2), the path is said to be an *active path*, because its associated constraint in the NLCO is in the active set of constraints. A path that is not active is *inactive*.

A leaf path is a *polygon path* if it is either (a) an active path, or (b) on the convex hull of the leaf node vertices, in which case it is called a *hull path*. If a chain of polygon paths closes on itself, forms a convex polygon, and contains no leaf node vertices in its interior, the enclosed region and boundary is said to be an *active polygon*.

A set of leaf node vertices in a polygon P with convex hull P_H is said to be *well-formed* with respect to a tree graph if it satisfies the following properties:

1. Every point within the convex hull lies within some active polygon.

2. Every active polygon contains at most one inactive hull path.

There is no guarantee that a solution to Equation (3) satisfies these two conditions; it is not uncommon to find a solution with active paths all around the boundary and one or more unconstrained leaf vertices "rattling around" in the interior of the polygon. However, it is usually possible to either add additional edges to the tree graph or to selectively lengthen certain edges of the tree graph to attain the well-formed state—and importantly, this is accomplished without reducing the size of the original graph; the solution to Equation (3) remains an optimum.

2.2 Molecules

Given a well-formed vertex set, we can now construct the crease pattern itself. We first note that any paper outside of the convex hull of the leaf vertices is effectively unused. In practice, it can be folded underneath and the resulting polygon treated as a single sheet of paper; in the interest of brevity, we will ignore it going forward and will assume that the paper P and the convex hull of the leaf node vertices P_H are one and the same.

We now introduce a coordinate system in the folded form: for any point p in the crease pattern, its perpendicular distance from the axis is called the *elevation* e, and its distance along the axis from some fixed reference point (to be defined presently) is called the *depth* d.

The leaf node vertices, by definition, lie on the axis of the base in the folded form, and so have elevation zero. It can be shown that any active path must have constant elevation along its length; thus, every point along an active path between two leaf node vertices has elevation zero and lies on the axis in the folded form. We call such a path an *axial path*. Since all points in the immediate neighborhood of an axial path lie at higher elevation than the axial path, there must be a gradient discontinuity along an axial path in the mapping from the crease pattern to the folded form—in other words, every axial path is folded. We can therefore construct creases along all axial paths in a well-formed vertex set; said creases are *axial creases*. (See Figure 2.)

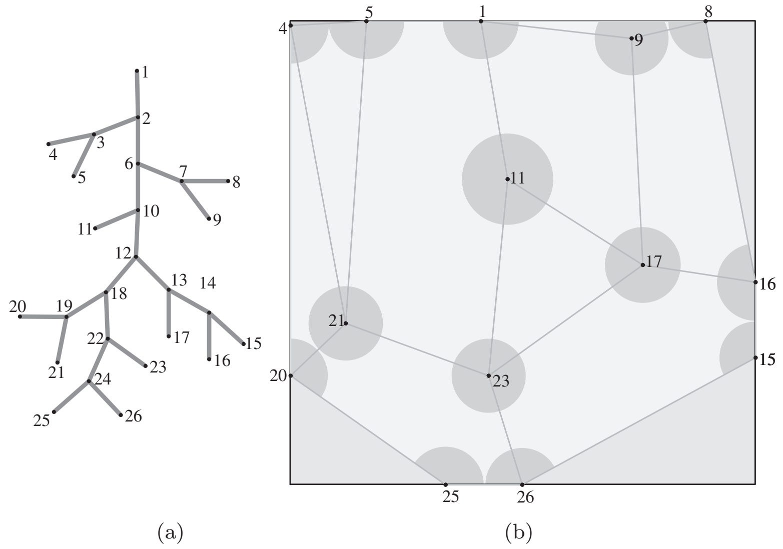

Figure 2. (a) A tree graph. (b) A well-formed vertex set for this tree graph with active and inactive polygon paths highlighted.

We now construct the crease pattern inside of each active polygon. They can be treated independently at this point; we call the crease pattern for any axial-boundary polygon a *molecule*. Inactive polygon paths (which exist only on the boundary of the convex hull) are not forced to be axial paths but we will choose them to be so. Thus, every active polygon has elevation zero on its boundary and, as it turns out, elevation greater than zero everywhere in its interior. The boundary paths of each active polygon are all lines of constant elevation and run parallel to the axis in the folded form. If we inset the boundary by some constant distance h, the resulting smaller polygon must also have a boundary that has constant elevation, but that elevation is now given by h. (See Figure 3.)

The corners of the inset polygon are gradient discontinuities in the mapping from the crease pattern to the folded form, and therefore must be folded points. In other words, there must be creases radiating inward from the corners of the active polygons. We call such creases *ridge creases*. If we denote the vertices of the active polygon by $\{v_i\}$ and the vertices of the inset polygon by $\{v_i'\}$, the inset vertices must obey a set of conditions analogous to the tree conditions, which are

$$|v_i' - v_j'| \geq ml_{ij} - h(\cot\alpha_i + \cot\alpha_j) \tag{4}$$

for all possible pairs of inset vertices, where α_i and α_j are, respectively,

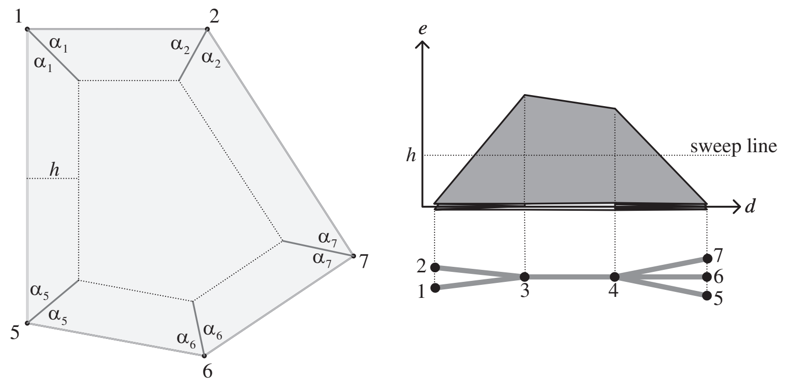

Figure 3. Insetting the boundary of an active polygon produces a polygon with a constant-elevation boundary at higher elevation.

half of the vertex angles at v_i and v_j. The path from v'_i to v'_j is called a *reduced active path*.

As the inset distance h is increased, one of two things happens. Either two inset vertices merge—in which case the polygon degree is reduced—or one (or more) of the reduced active paths becomes active, i.e., the inequality in Equation (4) reaches equality. When this happens, the situation is analogous to an axial path; the reduced path must be a line of constant elevation, the paper on either side of the path lies at higher elevation, and therefore, the path must be a (folded) crease, called a *gusset crease*. The gusset crease(s) divide the reduced polygon into two (or more) separate reduced polygons, each of which has degree lower than the original polygon, and the insetting process continues. Eventually, the ridge creases and vertices merge at a point, and the polygon is filled by a network of ridge and gusset creases, all of which are folded. (See Figure 4.)

These are not the only creases in the crease pattern, however. For a given uniaxial base, the flaps can be positioned in multiple arrangements. As noted earlier, the boundaries between flaps are called hinges; in the crease pattern, hinges are represented by *hinge creases*. A hinge crease can be folded or unfolded, depending on the relative positions of the flaps to either side.

While the tree graph is a discrete structure, we can expand it into a metric space in which a point can be defined anywhere along an edge and characterized by its distance along the edge. We call this space the *metric tree*. If we do this, then along any axial path in the crease pattern, there is a one-to-one mapping between any point on the path and a unique point on the metric tree. In particular, we can identify points

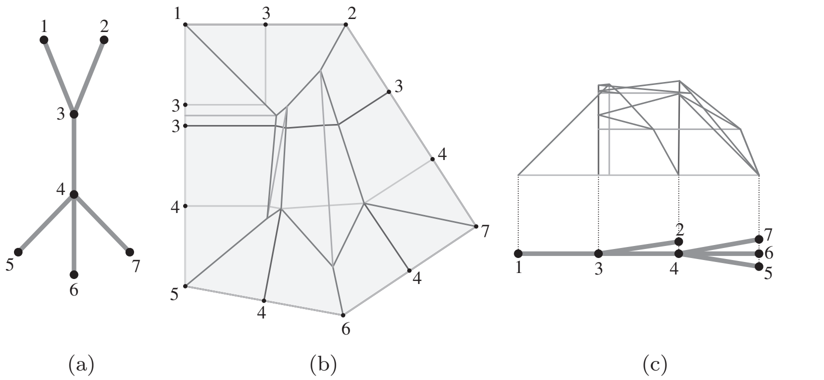

Figure 4. (a) The tree graph. (b) The universal molecule, showing ridge, gusset, folded and unfolded hinge, and pseudohinge creases. (c) The folded form, showing the folded positions of each crease.

on the crease pattern that map to branch nodes on the metric tree. Such points must be incident to hinge creases. And since hinge creases are lines of constant depth (just as axial and gusset creases were lines of constant elevation), we can identify vertices along the axial creases that correspond to the branch nodes along the tree, and then by propagating hinge creases out from them (and reflecting the hinge creases when they hit ridge creases), we can construct all of the hinge creases in the crease pattern (though not, as yet, determine whether they are folded or unfolded).

There is one more type of crease to construct. From every hull path in the crease pattern, there is a chain of ridge creases—called a *ridgeline*—connecting its endpoints. The intermediate vertices along this chain are incident to hinge creases that connect the ridgeline to the hull path, for the most part. However, if the hull path is inactive, there will be at least one vertex formed in the insetting process that is *not* incident to a hinge crease connecting it to the hull. We drop a new crease from each such vertex to the hull path, and call such creases *pseudohinge creases*. Like hinge creases, pseudohinge creases are creases of constant depth. Unlike hinge creases, pseudohinge creases are *always* folded and do not map to branch nodes on the metric tree.

Thus, the crease pattern is composed of creases and vertices that partition each active polygon into regions called *facets*. Each facet is part of a flap that corresponds to an edge of the tree graph; that edge is called the *projection* of the facet. If we take the projection of all of the facets in a molecule, we get a set of tree graph edges that is a subset of the edges of the entire tree graph. This set of edges and their incident nodes form a subgraph of the tree graph, which is called the *subtree* of the tree graph. A facet that is incident to the boundary of a molecule (and which therefore is incident to an axial crease) is an *axial facet*. A facet that is incident to a pseudohinge crease is a *pseudohinge facet*. Since all pseudohinge creases are incident to axial creases, all pseudohinge facets are also axial facets. A *corridor* is a connected set of facets that belong to the same flap in the folded form. A corridor can (and typically does) extend across multiple molecules.

This completes the construction of the crease pattern. Most of this algorithm has been previously described in [9, 10], although we had not previously made the distinction between hinge and pseudohinge creases. The crease pattern within each polygon is called the *universal molecule*, and the algorithm to construct it, the *universal molecule algorithm*. This algorithm provides all of the folds necessary to create the folded form. However, it does not give the crease assignments, nor even fully specify which of the hinge creases are folded, and it says nothing about the stacking order of the layers. It is well known that given a stacking order, the crease assignment is trivially deduced, while even given a full crease assignment, determining a valid stacking order can be NP-complete. Fortunately, determination of the stacking order and crease assignment in uniaxial bases is *not* NP-complete, as we will show in the next section.

3 Facet Ordering

3.1 Ordering Conditions

For an origami crease pattern to be flat foldable, it must satisfy three sets of conditions. The most famous of these are Maekawa's Theorem on crease directions at a vertex ($|M - V| = 2$) [4] and Kawasaki's Theorem on angles between creases around a vertex ($\sum_{i \text{ odd}} \phi_i = \sum_{i \text{ even}} \phi_i = 180°$) [6,7]. Less known are the layer ordering conditions formulated by Jacques Justin [5], which govern the stacking order of overlapping facets in the folded form. In fact, it is these conditions that lead to the computational complexity of many folding problems.

Our crease pattern satisfies Kawasaki's Theorem by design, and Maekawa's Theorem follows automatically if Justin's conditions are satisfied; thus, our focus on determining crease assignment is in fact an attempt to satisfy Justin's conditions. In his original formulation, Justin considered that all creases would be folded. In our crease pattern, we allow for unfolded creases, which necessitates a slight modification of the descriptions of the Justin conditions, of which there are four. (See Figure 5.)

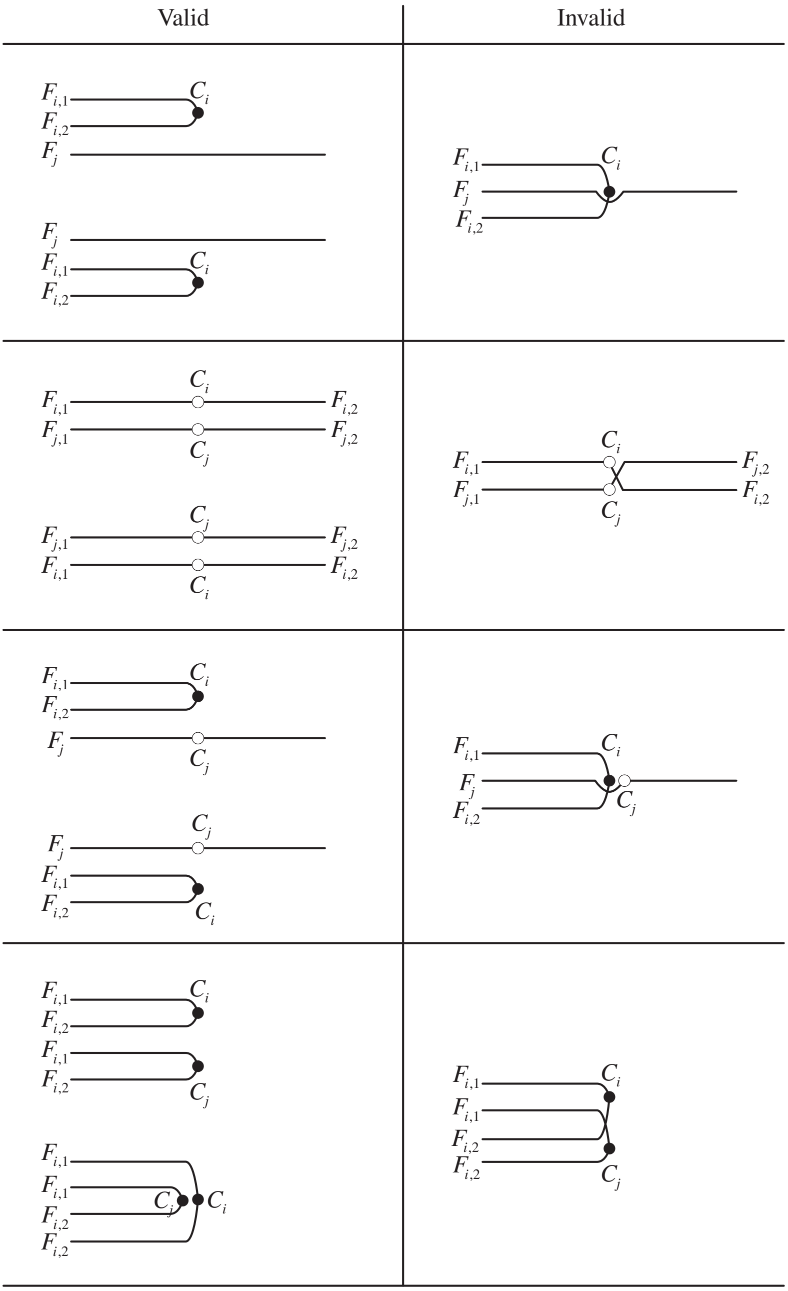

Figure 5. The four Justin conditions, illustrated schematically (edge views of the creases). Both valid and invalid configurations are illustrated.

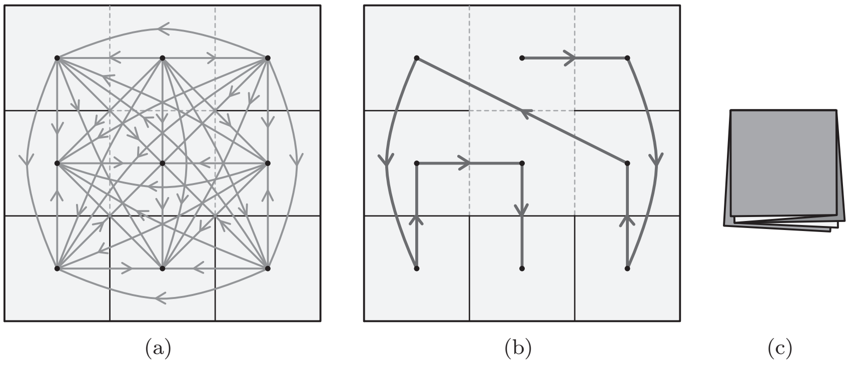

Figure 6. (a) A complete ordering graph. (b) An equivalent ROG. (c) The folded form.

The Justin conditions affect layer ordering among overlapping facets; we therefore must define an ordering relation for any two facets that overlap in the folded form. Such a relationship can readily be described by a directed graph, called an *ordering graph*, or OG, where the nodes of the graph represent the facets of the crease pattern and the directed edges represent order. An edge (F_i, F_j) is in the ordering graph if and only if facet F_i overlaps facet F_j. For convenience, one can draw an embedding of the ordering graph by positioning the nodes at the centroids of their corresponding facets, as shown in Figure 6.

An ordering graph can be fairly complex; for N facets, it can have as many as $N(N-1)/2$ edges (see, e.g., the stamp-folding problem). The Justin conditions apply to edges of the ordering graph that relate facets on either side of creases that overlap in the folded form. The ordering graph is directed, but it need not be simply connected, acyclic, or possessed of any other particular property. However, if an ordering graph *is* acyclic, it can be described by a simpler structure, which we call a *reduced ordering graph*, or ROG. Specifically, one can derived the OG from an ROG; the edge (F_i, F_j) is in an OG if and only if F_i overlaps F_j and there exists a directed path $(F_i, ..., F_j)$ in the ROG.

An ROG can be much simpler—having many fewer edges—than an OG, since in an ROG both edges and paths imply ordering relationships between facets. However, an ROG must necessarily be directed acyclic, leading to a directed acyclic ordering graph, and so not all OGs can be described by an ROG. On the other hand, since ROGs are DAGs (directed acyclic graphs), they are sortable; it is possible to assign an index to every facet such that the ordering relationship between the facets can be inferred simply by comparing the values of the two facet indices.

3.2 Rooted Embedding

As already noted, for a given crease pattern, there are usually several different folded forms with the same tree graph, depending on the arrangement of flaps. In particular, any given flap can be "flipped" about its hinge to point in either the positive or negative depth direction. Not all arrangements are possible, however. For some bases, there are flap directions for which no valid facet ordering exists (or put differently, for which the flaps must intersect one another).

To avoid this problem, we choose a particular flap arrangement that avoids such problems, by assigning depth in a particular way. We pick one node of the tree graph, which we call the *root node*, and assign it a *discrete depth* of zero. We then move out from the root node and assign each node a discrete depth, incrementing the discrete depth counter as we cross each edge. Thus, at the end, every node has a discrete depth that is simply its distance (in hops) from the root node.

We can now assign true depth to every vertex of the base, by setting the difference in depth between any two hinges to be the length of the edge between them, and choosing the sign of the difference in depth from the sign of the difference in their discrete depths. In a physical analogy, this algorithm is equivalent to "picking up the base" by its root node and letting all of the flaps dangle under the force of gravity.

Once the true depth of all hinges and hinge vertices has been assigned, there is sufficient information to assign both depth and elevation to every point in the folded form. This information then allows one to determine which hinge creases are folded; if the depth mapping is smooth across the hinge crease, it is unfolded; if it is gradient discontinuous, the hinge crease is folded.

Within the full tree graph, there is exactly one root node, which has the lowest possible discrete depth, i.e., zero. Each molecule has a subtree associated with it; within each subtree, there is one (or more) nodes with the lowest discrete depth, which may be greater than zero. These nodes are called the *local root nodes* of the subtree associated with the molecule. (To further distinguish the local root node of a subtree from the root node of the tree, we will sometimes call the latter the *global root node*.) The hinge creases associated with the local root nodes are the *local root hinges* of the molecule. The vertices incident to local root hinges are *local root vertices* of the molecule.

3.3 Reduced Ordering Graphs

We are now in a position to construct the ROG for the crease pattern. We begin by constructing directed graphs on each molecule. Each graph is composed of a set of directed paths, called *chains*, of which there are two types.

A *corridor chain* connects facets in the same corridor and is constructed as follows. It begins from an axial facet. We repeatedly add directed edges from the current facet to the next facet; the next facet in the chain is the facet that lies on the other side of either (a) the highest-elevation ridge crease, (b) the gusset crease, or (c) the pseudohinge crease of the current facet (other than the crease just crossed). Repeatedly following this rule gives a connected chain of facets, all within the same corridor, that eventually terminates on another axial facet. Each of the directed edges created in this fashion is a *corridor link*.

The *axial chain* connects facets in distinct corridors and is constructed as follows. We begin with an axial facet positioned immediately CCW from a local root hinge. if this facet is not an in-link of an existing corridor chain, we launch a new corridor chain from this facet and propagate it until it terminates (on some other axial facet). We then look for the next CCW axial facet that is *not* a pseudohinge facet. If the two facets are in different corridors, we add a directed edge from the current facet to this new facet (skipping over any pseudohinge facets between them) and repeat the process until we have reconnected with the first facet with which we started. Each of the directed edges created in this fashion is an *axial link*.

When this process is completed, the resulting graph, called a *molecular ordering graph* (MOG), will be connected. It may not (yet) be an ROG, because it may contain a cycle (if the local root node was a branch node of the subtree). In this latter case, if you delete any single axial link that crosses a local root hinge crease, the graph becomes an ROG and (while it is beyond the scope of this paper to prove) it is a valid facet ordering for the molecule. (See Figure 7.)

However, we must find a valid ROG for the entire crease pattern. We construct the MOG for each molecule—each of which may contain a cycle. We then merge the MOGs into a single directed graph. Two MOGS can merge at any common vertex that is a local root hinge vertex for one of the molecules. To merge two MOGs at a vertex, we delete the axial links on either side of the vertex and add two links connecting the "cut ends" to each other. In this fashion, we merge all molecules into a single directed graph.

At this point, if the global root node is a leaf node of the tree, the resulting graph is acyclic, constitutes an ROG, and produces a valid facet ordering (again, the proof is beyond the scope of this paper). If the global root node is not a leaf node, then this graph contains a cycle; breaking the cycle by deleting an axial link crossing a hinge crease incident to a global root vertex. The resulting graph is then acyclic, is an ROG, and produces a valid facet ordering.

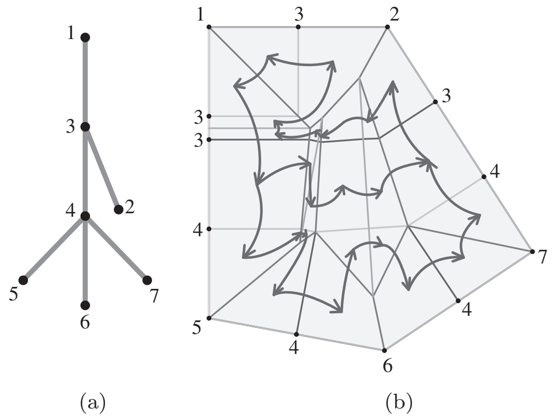

Figure 7. (a) The molecular ordering graph for a single molecule. (b) Deleting the edge crossing the ridge crease incident to node 1 transforms it into a valid ROG.

3.4 Crease Assignment

Once the ROG has been constructed, crease assignment is straightforward. Since the ROG is sortable, each facet can be assigned an index, beginning with the (sole) source facet, such that the relative ordering of the facets implied by the ROG can be deduced by comparing the indices of the two facets. We then two-color the facets as "white-up" (W) and "color-up" (C) so that the facets on either side of every folded crease are of opposite color. (See Figure 8.) Mountain/valley (M/V) assignment of every folded crease can be computed from the two-coloring and the relative ordering of the facets on each side of the crease:

1. $(W \to C) \implies M$.
2. $(C \to M) \implies V$.

Of course, the opposite assignment is equally valid, being equivalent to the interchange of M and V creases. This completes the crease assignment algorithm. (See Figure 9.)

3.5 TreeMaker 5

We have implemented this algorithm in a revised version of our TreeMaker program, in which the optimization, crease construction, and crease assignment algorithms encompass roughly 27,500 lines of code. We have tested the algorithm on a wide range of tree graphs, all producing facet orderings and crease assignments that yield valid folded forms (both mathematically and physically). TreeMaker 5 is cross-platform (Mac, GNU/Linux, and

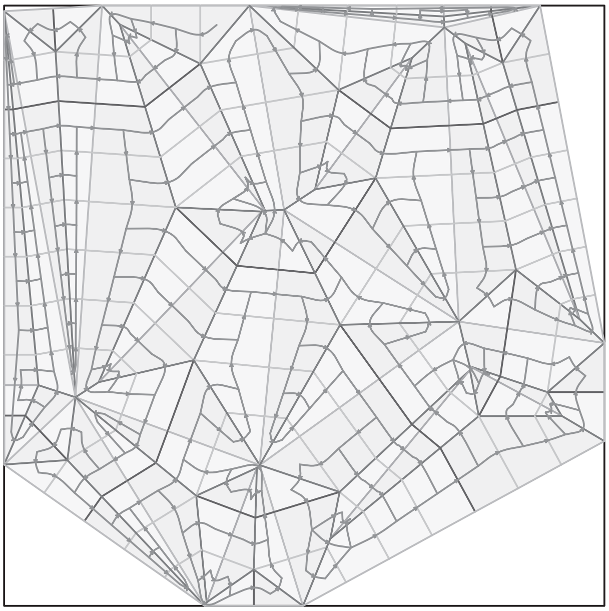
Figure 8. The completed ordering graph and two-coloring.

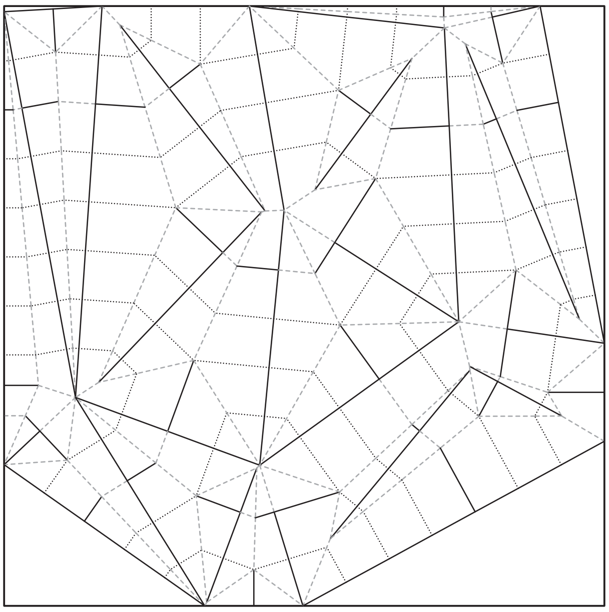
Figure 9. The completed M/V assignment.

Windows), open-source (GPL), and can be downloaded from http://www.langorigami.com.

4 Discussion

This algorithm completes the plan laid out in [9]. Given an arbitrary tree graph, it is now possible to construct the crease pattern, including crease assignment, for a valid base whose projection is the given tree graph. We note that the technique of incrementally constructing ordering graphs and merging them into a single graph is conceptually similar to the technique used in [1]—as is the concept of rooting the folded form. The explicit construction of an ordering graph as we have done here leads to a relatively straightforward computer implementation of the ordering algorithm. It also emphasizes the primacy of the ordering relationship, rather than the crease assignment, as the fundamental mathematical description.

Of course, we have not proven that the ROG satisfies the Justin conditions. To do so requires first, a transformation of the Justin conditions on facets into required properties of the graph, and second, a proof that the constructed graph has those properties. A complete derivation and proof is the subject of ongoing work. We do note, however, that we have tested the algorithm on many individual cases, including intentionally pathological test structures, with success, and so feel that the algorithm may be usefully employed even now. We believe that this approach could be adapted to provide crease assignment and facet ordering for other related folding problems, such as polyhedron flattening (the airbag problem), among others.

Acknowledgment. Both authors would like to acknowledge fruitful discussions with Martin L. Demaine in the course of working on this problem. Robert Lang would like to thank the Massachusetts Institute of Technology for support as a visiting researcher.

Bibliography

[1] Marshall Bern, Erik Demaine, David Eppstein, and Barry Hayes. "A Disk-Packing Algorithm for an Origami Magic Trick." In *Proceedings of the International Conference on Fun with Algorithms, Isola d'Elba, Italy, June 18–20, 1998*, pp. 32–42. Ottawa: Carleton University, 1998.

[2] Marshall Bern, Erik Demaine, David Eppstein, and Barry Hayes. "A Disk-Packing Algorithm for an Origami Magic Trick." In *Origami^3: Proceedings of the Third International Meeting of Origami Science, Mathematics, and*

Education, edited by Thomas Hull, pp. 17–28. Natick, MA: A K Peters, 2002.

[3] Marshall Bern and Barry Hayes. "The Complexity of Flat Origami." In *Proceedings of the Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 175–183. Philadelphia: SIAM, 1996.

[4] Thomas Hull. "The Combinatorics of Flat Folds: A Survey." In *Origami^3: Proceedings of the Third International Meeting of Origami Science, Mathematics, and Education*, edited by Thomas Hull, pp. 29–37. Natick, MA: A K Peters, 2002.

[5] Jacques Justin. "Aspects mathematiques du pliage de papier." In *Proceedings of the First International Meeting of Origami Science and Technology*, edited by H. Huzita, pp. 263–277. Padova, Italy: Dipartimento di Fisica dell'Università di Padova, 1991.

[6] Toshikazu Kawasaki. "On High-Dimensional Flat Origamis." In *Proceedings of the First International Meeting of Origami Science and Technology*, edited by H. Huzita, pp. 131–141. Padova, Italy: Dipartimento di Fisica dell'Università di Padova, 1991.

[7] Toshikazu Kawasaki. "On the Relation between Mountain-Creases and Valley-Creases of a Flat Origami." In *Proceedings of the First International Meeting of Origami Science and Technology*, edited by H. Huzita, pp. 229–237. Padova, Italy: Dipartimento di Fisica dell'Università di Padova, 1991.

[8] Robert J. Lang. "Mathematical Algorithms for Origami Design." *Symmetry: Culture and Science* 5:2 (1994), 115–152.

[9] Robert J. Lang. "A Computational Algorithm for Origami Design." In *Proceedings of the Twelfth Annual Symposium on Computational Geometry*, pp. 98–105. New York: ACM Press, 1996.

[10] Robert J. Lang. *Origami Design Secrets: Mathematical Methods for an Ancient Art.* A K Peters, 2003.

[11] Robert J. Lang. "TreeMaker." Available at http://www.langorigami.com/treemaker.htm, 2003.