When Can a Net Fold to a Polyhedron?[*]

Therese Biedl[†] Anna Lubiw[†] Julie Sun[†]

1 Introduction

In this paper, we study the problem of whether a polyhedron can be obtained from a *net*, i.e., a polygon and a set of creases, by folding along the creases. We consider two cases, depending on whether we are given the dihedral angle at each crease. If these dihedral angles are given the problem can be solved in polynomial time by the simple expedient of performing the folding. If the dihedral angles are not given the problem is NP-complete, at least for orthogonal polyhedra. We then turn to the actual folding process, and show an example of a net with rigid faces that can, in the sense above, be folded to form an orthogonal polyhedron, but only by allowing faces to intersect each other during the folding process.

In the existing literature, a few related problems have been studied. Shephard investigated when a convex polyhedron has a convex net [She75]. Lubiw and O'Rourke showed how to test in $O(n^2)$ time whether an n-vertex polygon (with unknown creases) can be folded into a convex polyhedron [LO96]. Another problem is the reverse of ours: Given a polyhedron, can one obtain a net? This can be done for all convex polyhedra [AO92], as well as for some classes of orthogonal polyhedra [BDD$^+$98]. A fundamental open problem in this area is whether for any convex polyhedron there exists a net such that the edges of the net are also edges of the convex polyhedron, a so-called unfolding with edge cuts only.

Related to our problem of folding a rigid net is the problem of straightening rigid linkages in 3D, which has been studied in [BDD$^+$99]. In fact, our proof that some rigid net cannot be folded without intersecting faces is based on the fact that there exists a linkage in 3D that cannot be straightened without having links intersect [CJ98, BDD$^+$99].

2 Definitions

A *polygonal chain* is a sequence of line segments $[a_i, b_i]$, $i = 0, \ldots, n-1$ that are mutually disjoint, except that $b_i = a_{i+1}$ for $i = 0, \ldots, n-1$ (addition modulo n). The segments are called *edges* and the endpoints of the edges are called *vertices.* A finite region in the plane bounded by a polygonal chain is called a *polygon.* A *chord* of a polygon is a line segment inside the polygon where both endpoints are vertices of the polygon. A *crease-set* of a polygon is a set of chords of the polygon that do not intersect each other except possibly at endpoints. A *net* is a polygon together with a crease-set. An *orthogonal net* is a net where all edges of the polygon and all creases are parallel to a coordinate axis.

[*]These results are part of the third author's Master's Thesis at the University of Waterloo. Research supported by NSERC.

[†]Department of Computer Science, University of Waterloo, Waterloo, ON N2L 3G1, {biedl, alubiw, j4sun}@uwaterloo.ca

A net can also be viewed as a graph; in fact, it is an *outer-planar graph* since no two edges cross and all vertices are on the unbounded face (the *outer face*). It is known that an n-vertex outer-planar graph has at most $2n-3$ edges. The faces that are not the outer face are called *interior faces*. For an outer-planar graph, the incidences between interior faces form a tree.

In a net, for each crease we may or may not specify the *dihedral angle*, i.e., the angle that the two faces incident to the crease will form inside the finished polyhedron. We impose the condition that every crease must indeed be folded, so the dihedral angle cannot be π. Likewise, the dihedral angle cannot be 0 or 2π, because faces are not allowed to overlap. For an orthogonal net, we stipulate that all dihedral angles must be $\pi/2$ or $-\pi/2$.

To be able to test whether a net folds into a polyhedron, we must establish a clear definition of a polyhedron. This is a non-trivial task (see [Cro97] for a history of attempts). We use the following definition, based on Coxeter [Cox63]: A *polyhedron* is a finite connected set of plane polygons, called *faces*, such that (1) if two faces intersect, it is only at a common vertex or a common edge, (2) every edge of every face is an edge of exactly one other face, and (3) the faces surrounding each vertex form a single circuit (to exclude anomalies such as two pyramids with a common apex).

An *orthogonal polyhedron* is a polyhedron each of whose faces is perpendicular to a coordinate axis. For such a polyhedron, we classify each face as an *xy-face*, a *yz-face* or an *xz-face*, depending on which plane the face is parallel to.

3 Known Dihedral Angles

In this section, we show that if dihedral angles are given, we can determine in polynomial time whether folding the net yields a polyhedron. Our computation model here is the real RAM; for the case of orthogonal nets, which is our main interest, basic arithmetic suffices.

First, we show how to find the coordinates of the vertices in 3D after the creases have been folded. Compute the tree of adjacencies between the interior faces of the net. Traverse this tree T in depth-first-search order, starting at an arbitrary leaf in an arbitrary position. For each face, the positions of vertices of this face can then be computed using the positions of the vertices of the parent of this face in T, and the dihedral angle that connects the two faces. This takes $O(n)$ time, where n is the number of vertices of the net, because the number of edges and faces of the net is proportional to the number of vertices.

Now we must verify the three properties of polyhedra. We do so in four steps, not for efficiency, but for clarity of presentation: (1) We reject the input if we can find two faces that intersect in a point that is interior to one or both of the faces; (2) We add additional vertices along the edges of the faces, so that any vertex of the polyhedron is a vertex of all its incident faces; (3) For each edge of each face we find all identical edges of other faces, simultaneously building up the the *incidence graph*, a data structure to store polyhedra [Ede88]; (4) we use the incidence graph to test that the faces surrounding each vertex form a circuit.

For step 1 we consider each pair of faces of the net. The running time of our algorithm is then already $\Omega(n^2)$, so we will not worry about making other steps of the algorithm faster than this. Let $F_1, \ldots, F_f$ be the interior faces of the net, and let m_i be the number of edge of F_i. For any $i < j$, if F_i and F_j lie in the same plane, we can test in $O(m_i m_j)$ time how they intersect by testing every pair of edges, and doing a final inclusion test. If F_i and F_j lie in different planes, we then compute the line of intersection of these planes and compute the intersections of this line with F_i and F_j, forming two sets of disjoint intervals. We can certainly test in $O(m_i m_j)$ time how these intervals intersect.

The total running time of this step is proportional to

$$\sum_{1\leq i<j\leq f} m_i m_j \leq \frac{1}{2}(\sum_{i=1}^{f} m_i)(\sum_{i=1}^{f} m_i) \leq \frac{1}{2}(2m)^2,$$

where m is the number of edges of the net. Since the net is an outer-planar graph, $m \leq 2n-3$, so this step takes $O(n^2)$ time.

The second step is necessitated by the polyhedron property that every edge of every face must be an edge (not just part of an edge) of another face. Without the addition of extra vertices in the net, this condition may be violated if at a vertex v of the polyhedron one incident face F has face-angle π and v is not a vertex of F in the net. We therefore must add v as a vertex of F. We will refer to this situation as "face F grazes vertex v".

To determine all places where a face grazes a vertex, we compare each vertex v of a face to all edges e of faces. If v sits in the interior of e, then we introduce a new vertex to the face containing e, thus splitting edge e into two edges. This takes $O(n^2)$ time.

To show the correctness of this approach, let v be a vertex of the polyhedron, i.e., a point where either one face of the net has a vertex or where the sum of the face-angles of the incident faces is not equal to 2π. Observe that no two consecutive incident faces of v can graze v, because otherwise both faces have face-angle π, which implies that v can have only these two incident faces. But then the sum of face-angles at v is 2π, and both incident faces have no vertex at v, contradicting the definition of a vertex of a polyhedron.

Therefore, if a face F grazes a vertex v, then another face F' incident to v had v as a vertex; therefore this grazing will be detected in the above procedure. Also, the total number of added vertices is at most n, since at most half of the incident faces of a vertex v graze v, while the other half already had v as a vertex in the original net. Therefore we end up with $O(n)$ vertices and edges in the net.

In the third step, we test every pair of edges to see whether they are identical. If we ever find an edge that does not have exactly one identical mate, we reject this input. As we match up edges we build the incidence graph that records the incidences of vertices, edges, and faces of the polyhedron. This step can be performed in $O(n^2)$ time.

Finally, in the fourth step we use the incidence graph to walk around the faces incident with each vertex and verify that they form a circuit. This can be done in time proportional to the degree of the vertex, and therefore in $O(n)$ time overall.

Our total time therefore is $O(n^2)$. We conjecture that this time bound can be improved to $O(n \log n)$ by using 3D sweep techniques, rather than the brute-force approach.

4 Unknown Dihedral Angles

In this section, we show that if dihedral angles are not specified, then the folding problem becomes NP-complete—at least for the case of orthogonal polyhedra, where each dihedral angle must be $\pi/2$ or $-\pi/2$.

For our reductions, we use the NP-complete problem Partition [GJ79]: Can a set S of positive integers be partitioned into $S = S_1 \cup S_2$, such that the sum of the elements in S_1 equals the sum of the elements in S_2?

We start by proving that a 2D version of the folding problem is NP-complete. Rather curiously, this will not immediately imply NP-completeness of the 3D version, and more work must be done.

4.1 2D Orthogonal Foldings

In 2D, the folding problem is the following: Given a sequence of straight line segments with joints between the segments, determine if we can bend all joints such that we obtain a simple polygon. This problem is solvable in polynomial time, because the answer is positive if and only if no link is longer than all other links together [LW95]. In this paper, we study the *orthogonal folding problem*, which is the same problem, except that all joints have to be bent at right angles. Surprisingly, this makes the problem NP-complete.

Given an orthogonal polygon directed counter-clockwise, the horizontal edges fall into two classes: those directed to the right and those directed to the left; furthermore, the sum of the lengths in each class is the same. This observation is the heart of our reduction of Partition to the 2D orthogonal folding problem.

Theorem 1 *The 2D orthogonal folding problem is NP-complete.*

Proof: It is easy to verify that a given assignment of dihedral angles of $\pi/2$ or $-\pi/2$ forms a simple polygon, so the problem is in NP. To show that it is NP-hard, let $S = \{x_1, \ldots, x_n\}$ be an instance of Partition. Set $L = \frac{1}{2}\sum_{i=1}^{n} x_i + 1$, $v = n + 1$, and let $S' = (1, x_1, 1, x_2, 1, \ldots, 1, x_n, 1, L, v, L)$ be an instance of the 2D folding problem where the numbers denote the lengths of the links in order along the chain. The sequence consisting of the first $2n+1$ segments will be called the *jagged sequence*; these segments encode the partition problem, while the other segments serve to complete the polygon.

Assume the instance S of Partition has a solution $S = S_1 \cup S_2$. We construct a solution to the folding problem S' as follows: Working counterclockwise, the link of length x_i points left if $x_i \in S_1$ and right otherwise, all links of length 1 point up, and the remaining links form a "C". See also Figure 1. The resulting polygon is simple, because from the right ends of the links of length L, we can reach at most $\sum_{x_i \in S_1} x_i = \frac{1}{2}\sum_{i=1}^{n} x_i$ units to the left, so $L > \frac{1}{2}\sum_{i=1}^{n} x_i$ is big enough to prevent any of the links of the jagged sequence from reaching the link of length v.

Conversely, assume that S' can be folded into an orthogonal polygon, and after possible rotation, assume that the link of length v is vertical. Let H_l be the lengths of the edges pointing left and H_r be the lengths of the edges pointing right. Since all creases have dihedral angle $\pi/2$ or $-\pi/2$, we know that $H_l \cup H_r = \{x_1, \ldots, x_n, L, L\}$. Since S' folds to a closed polygon, the lengths in H_l sum to the same as the lengths in H_r. Since $L + L > \sum_{i=1}^{n} x_i$, the two L's cannot be in the same subset, so each of H_l and H_r must contain exactly one L. Removing both L's, we obtain a partition of $\{x_1, x_2, \ldots, x_n\}$ with the two parts having equal sums. $\square$

4.2 3D Orthogonal Foldings

Theorem 2 *The 3D orthogonal folding problem is NP-complete.*

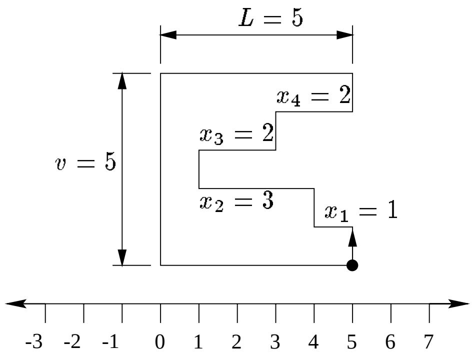

Figure 1: Constructing S' from the instance {1,3,2,2} of PARTITION.

Proof: If we are given the assignment of dihedral angles of $\pi/2$ or $-\pi/2$, then we can verify in polynomial time whether this net folds into a polyhedron (Section 3), so the problem is in NP. To show that it is NP-hard, let $S = \{x_1, \ldots, x_n\}$ be an instance of PARTITION.

We describe the construction of an instance of the 3D orthogonal folding problem in successively more correct refinements. We illustrate these using the PARTITION instance $S = \{1, 2, 1\}$.

The first approach is to extrude the polygon formed in Section 4.1 in the z-direction by 2 units, see Figure 2. The sequence of links becomes in the net a sequence of rectangles all of height 2. However, the problem then arises of how to cover the front face. Note that the right boundary of the front face abuts the (extruded) jagged sequence, and thus the shape of this face depends upon the partition of S into S_1 and S_2. Even cutting the front face into strips attached to the vertical unit length edges of the jagged sequence as shown in Figure 2 does not resolve the problem, because the lengths of the strips still depend on the partition.

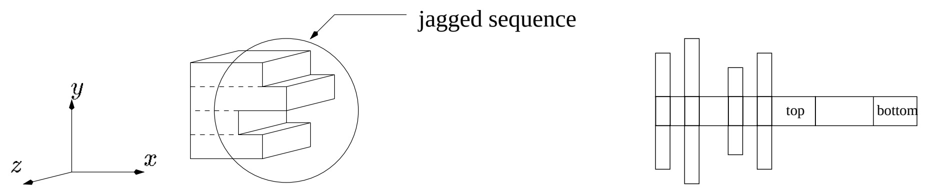

Figure 2: The construction in 2D extruded, and its net.

Our next idea is to make the strips that form the front face equally long, which can be achieved by replicating the jagged sequence. The lengths in the x-direction of the top and bottom faces are set to $\sum_{i=1}^{n} x_i + 1$. The resulting polyhedron looks like a staggered pile of bricks. Its net is independent of any particular partition of S. Unfortunately, this construction is too general: for any input the resulting net folds into a polyhedron. The proof from the two-dimensional case does not transfer because we cannot force the top and the bottom face to be *aligned*, i.e., to have the same xz-projection. See Figure 3.

To force an alignment of the top and the bottom face, we add an extruded rectangle with x-dimension $\frac{1}{2}$ and z-dimension 1 down the middle of the front face. We call this extruded rectangle the *middle notch*. See Figure 4.

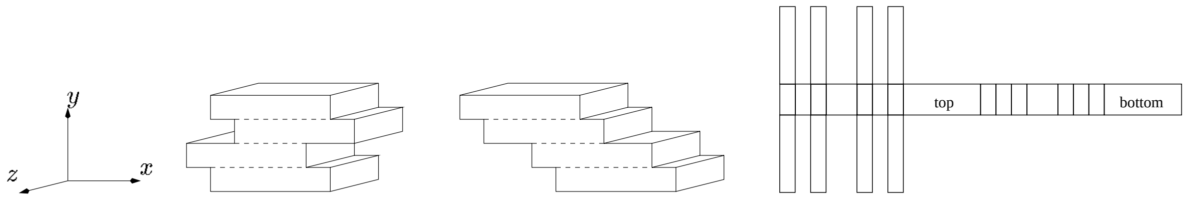

Figure 3: Replicate the jagged sequence to form a pile of bricks. Now the top and bottom face need not align.

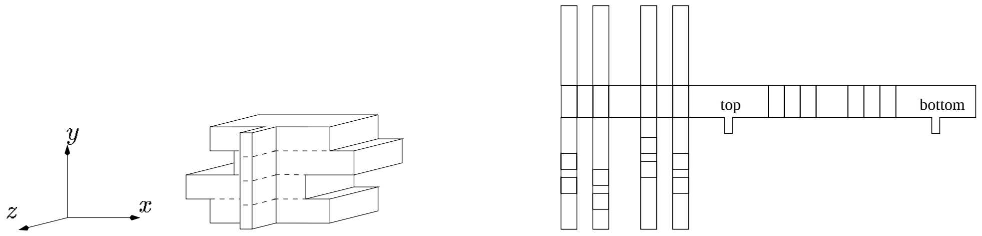

Figure 4: Introducing a middle notch.

Unfortunately, now we have a problem similar to the one we had originally: Each strip now consists of five pieces, the middle three forming part of the middle notch. The lengths of the first and last pieces depend upon the partition of S.

We resolve this problem by scooping out rectangular notches along the front side of the polyhedron (see Figure 5). More precisely, starting $\frac{1}{4}$ units to the right of the left end of the top face, we scoop out notches of length $\frac{1}{2}$ in the x-dimension and length 1 in the z-dimension at $\frac{1}{2}$ unit intervals in the x-direction, until we reach the middle notch. We proceed in a similar way from the right end of the top face. Call the resulting net S''.

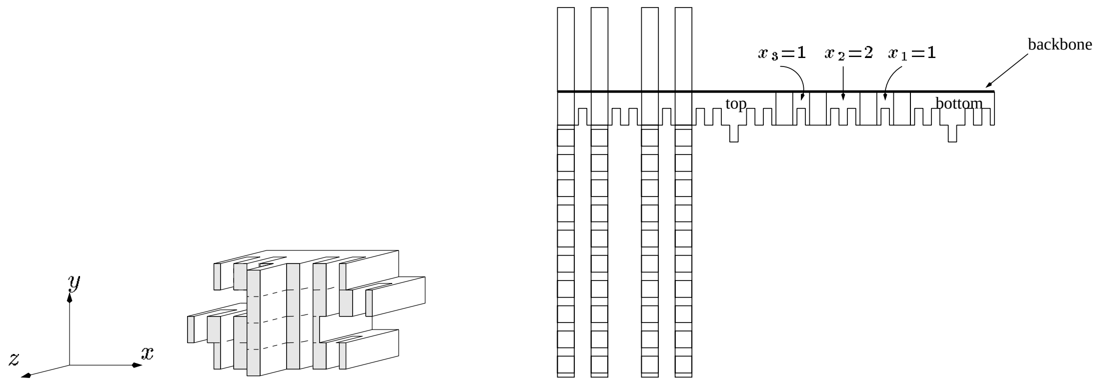

Figure 5: Cutting notches along the front face.

The effect of these notches is that now the strips used to cover the front face of the polyhedron have many creases any of which can form part of the middle notch. Therefore, we need not know the partition to construct the net. On the other hand, we can show that the middle notch continues

to force alignment of the top and the bottom faces, and thus that this construction works:
Claim: S has a solution $S = S_1 \cup S_2$ if and only if S'' folds in an orthogonal polyhedron.
Proof: Following the steps of our construction, one can verify that if S has a solution, then S'' folds into an orthogonal polyhedron. For the other direction, assume S'' folds into an orthogonal polyhedron. We will not show that the polyhedron is like the one shown in Figure 5 (even though this is possible), but only use properties of the polyhedron to extract a partition of S. Rotate the polyhedron so that the face marked "top" in the net is an xz-face; we refer to this face as T. Also rotate such that the edge of length $\sum_{i=1}^{n} x_i + 1$ of T is the *back edge* of T, i.e., it is parallel to the x-axis and has the smallest z-coordinate of all points of T. Let the *backbone* be the intersection of the net with the line through the back edge of T in the net, see Figure 5. By construction of our net, the edges of the backbone lie in one xz-plane P in the folded polyhedron.

As a first step, we show that face T must be aligned with the face marked "bottom" in the net; we refer to this face as B. Since all folds must be folded at right angles, we can read from the net that the set of xz-faces consists exactly of the faces T and B and the faces corresponding to $x_1, \ldots, x_n$. Notice that by construction of our net these faces have their back edge on the backbone and thus in plane P. Note however that T and B stick farther forward in the z-direction than any other part of any other xz-face, because of the middle notch. This implies that T and B must be aligned, for otherwise there would be a line parallel to the y-axis intersecting the top of the middle notch and no other xz-face, a contradiction to the even parity required for the number of intersections between a polyhedron and a line.

In the polyhedron, the backbone forms an orthogonal chain in plane P. Let C be the subchain joining the back edges of T and B; thus C consists of the back edges of one extruded jagged sequence. Let H_l be the lengths of the edges of C pointing left, and H_r be the lengths of the edges of C pointing right. Since all creases have dihedral angle $\pi/2$ or $-\pi/2$, we know that $H_l \cup H_r = \{x_1, \ldots, x_n\}$. Because T and B are aligned, chain C begins and ends with the same x-coordinate. Therefore the lengths in H_l sum to the same as the lengths in H_r, and thus H_l and H_r form the required partition. $\square$

This ends the proof of the claim, and therefore the proof of the theorem. $\square$

5 Rigid Nets

In this section, we show that if the net is made from stiff material, i.e., its faces are rigid, then we cannot always execute the folding process—from net to polyhedron—while keeping faces disjoint.

Theorem 3 *There exists an orthogonal net and an orthogonal polyhedron it can be folded to with the property that the folding cannot be performed while keeping faces rigid and disjoint.*

Proof: Consider the net shown in Figure 6; its folded-up version is shown in Figure 7. The ends of this "extruded chain" are supposed to be very long.

Imagine placing an orthogonal chain on the faces that are shaded in Figure 6, with vertices exactly on the creases that the chain crosses. Now, if we could fold this net of rigid faces without self-intersections, then we could also fold the chain without self-intersections into the position that

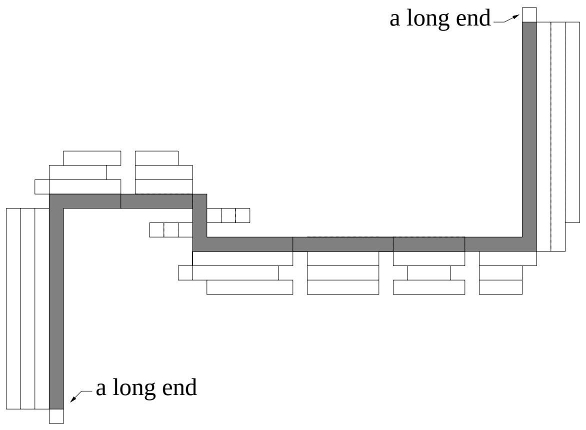

Figure 6: This rigid net cannot be folded without intersections.

it takes on the polyhedron. However, using a proof very similar to the one in [BDD+99], we can show that if the end-links of the chain are sufficiently long, then this chain cannot be *straightened*, i.e., transformed into a straight chain without self-intersections, even allowing arbitrary rotations of links, rather than just the one degree of freedom rotations possible for the chain on the net. Since the chain can be straightened when lying on the net (because it has a planar projection [BDD+99]), the net of rigid faces cannot be folded into the polyhedron without self-intersections. □

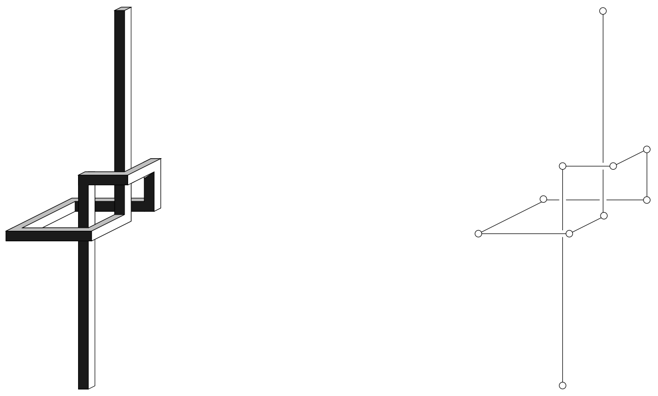

Figure 7: The polyhedron for the rigid net, and the chain embedded on it.

6 Conclusions

In this paper, we studied the problem of determining whether a net folds into a polyhedron. We showed that if the dihedral angle at each crease is given, the question can be answered in polynomial time. However, for orthogonal polyhedra, if the dihedral angle is not given, the problem becomes NP-complete. Finally, we examined the difficulty of folding a net (with dihedral angles given) made of stiff material in which faces cannot intersect. Here we showed a net of an orthogonal polyhedron that cannot fold into the polyhedron without any intersection of faces.

Our NP-completeness result only holds for orthogonal polyhedra. In particular, it is conceivable that the nets we construct (see Figure 5) can fold to non-orthogonal polyhedra that no longer yield the partition. This leads to a question interesting in its own right: can an orthogonal net ever fold to a non-orthogonal polyhedron?

We mention one other problem concerning rigidity: can any feasible folding process be accomplished with rigid non-intersecting faces if extra creases (of dihedral angle π) may be introduced?

References

[AO92] B. Aronov and J. O'Rourke. Nonoverlap of the star unfolding. *Discrete Comput. Geom.*, 8:219–250, 1992.

[BDD$^+$98] T. Biedl, E. Demaine, M. Demaine, A. Lubiw, J. O'Rourke, M. Overmars, S. Robbins, and S. Whitesides. Unfolding some classes of orthogonal polyhedra. In *10th Canadian Conference on Computational Geometry*, pages 70–71, 1998.

[BDD$^+$99] T. Biedl, E. Demaine, M. Demaine, S. Lazard, A. Lubiw, J. O'Rourke, M. Overmars, S. Robbins, I. Streinu, G. Toussaint, and S. Whitesides. Locked and unlocked polygonal chains in 3D. In *SIAM-ACM Conference on Discrete Algorithms*, pages 866–867, 1999.

[CJ98] J. Cantarella and H. Johnston. Non-trivial embeddings of polygonal intervals and unknots in 3-space. *Journal of Knot Theory and its Ramifications*, 7(8):1027–1039, 1998.

[Cox63] H.S.M. Coxeter. *Regular Polytopes.* The Macmillan Company, 1963.

[Cro97] Peter R. Cromwell. *Polyhedra.* Cambridge University Press, 1997.

[Ede88] H. Edelsbrunner. *Algorithms in Combinatorial Geometry.* Springer-Verlag, 1988.

[GJ79] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* Freeman, 1979.

[LO96] A. Lubiw and J. O'Rourke. When can a polygon fold to a polytope? Technical Report 048, Smith College, 1996.

[LW95] W.J. Lenhart and S.H. Whitesides. Reconfiguring closed polygonal chains in euclidean d-space. *Discrete and Computational Geometry*, 13:123–140, 1995.

[She75] G. C. Shephard. Convex polytopes with convex nets. *Math. Proc. Camb. Phil. Soc.*, 78:389–403, 1975.