Threadable Curves*

Joseph O'Rourke Emmely Rogers†

Abstract

We define a plane curve to be *threadable* if it can rigidly pass through a point-hole in a line L without otherwise touching L. Threadable curves are in a sense generalizations of monotone curves. We have two main results. The first is a linear-time algorithm for deciding whether a polygonal curve is threadable—$O(n)$ for a curve of n vertices—and if threadable, finding a sequence of rigid motions to thread it through a hole. We also sketch an argument that shows that the threadability of algebraic curves can be decided in time polynomial in the degree of the curve. The second main result is an $O(n \operatorname{polylog} n)$-time algorithm for deciding whether a 3D polygonal curve can thread through a hole in a plane in $\mathbb{R}^3$, and if so, providing a description of the rigid motions that achieve the threading.

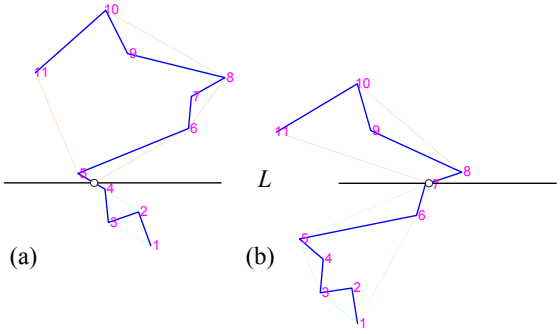

Figure 1: Two snapshots of a 10-segment polygonal chain passing through a point-hole in the x-axis.

1 Introduction

We define a simple (non-self-intersecting) open planar curve C to be *threadable* if there exists a continuous sequence of rigid motions that allows C to pass through a point-hole o in an infinite line L without any other point of C ever touching L. For fixed L, we will take L to be the x-axis and o to be the origin; equivalently we can view C as fixed and L moving (Lemma 1). C could be a polygonal chain or a smooth curve. C is open in the sense that it is not closed to a cycle. An example is shown in Fig. 1; animations are available at `http://cs.smith.edu/~jorourke/Threadable/`.

Note that our definition requires "strict threadability" in the sense that no other point of C touches L. So, for example, the curve illustrated in Fig. 2 is not threadable.

This notion has appeared in the literature in another guise. In particular, a threadable curve C corresponds to a "generalized self-approaching curve" with width π in both directions, as defined in [AAI+01]. However, those authors do not explore that concept, and in any case, our explorations focus on different properties of C. Nevertheless, our algorithms are quite similar to those for recognizing self-approaching curves and increasing-chord paths in [ACG+12].[1]

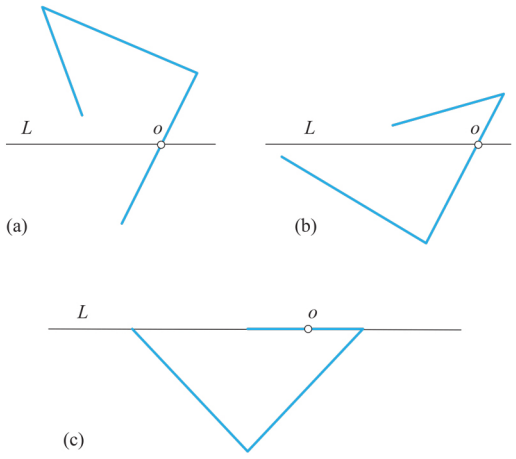

Figure 2: A curve that is not threadable. To pass completely through o, an edge would have to lie on L.

*Full version: `https://arxiv.org/abs/1801.08003`, v3.

†Department of Computer Science, Smith College, Northampton, MA, USA. `{jorourke,erogers}@smith.edu`.

[1]We thank Anna Lubiw for these references.

One could view our topic as a specialized motion-planning problem, but it seems not directly addressed in the literature. Work of Yap [Yap87], discussed in Section 7, can be viewed as a higher dimensional version. Research examining the fabrication of hydraulic tubes [AFM03], as well as work on "producible protein chains" [DLO06], lead to workspace-clearance concerns, to which we return in Section 8. We will see that classical computational geometry tools suffice to address our problems, but some interesting questions are raised.

1.1 Definition Consequences

We now explore a few consequences of the definition.

Lemma 1 *If a curve C is threadable, then through every point $p \in C$ there is a line L that meets C in exactly p: $L \cap C = \{p\}$, and L properly crosses C at p.*

Note that L tangent to C is insufficient for threadability, for then C would locally lie on one side of L. This is why the lemma insists on proper crossings.

What is perhaps not immediate is the implication in the other direction to Lemma 1:

Lemma 2 *If a curve C has the property that through every point $p \in C$ there is a line L that meets C in exactly p, and L properly crosses C at p, then C is threadable.*

The reason this is not immediate, is that it is conceivable that the orientation of the line changes discontinuously at some point $p \in C$, requiring an instantaneous rigid "jump" motion of C to pass through L, rather than a continuous rigid motion. A proof is deferred until we can rule out this discontinuity (Section 3).

1.2 Monotone Curves

A *monotone curve* C is defined as one that meets all lines parallel to some line L in a single point (if *strictly monotone*), or which intersects every such line in either a point or a segment (if *non-strictly monotone*). Every strictly monotone curve is threadable, and one can view threadability as a generalization of monotonicity, allowing the orientation of L to vary.

2 Butterflies

Define the *butterfly* $\mathrm{bf}(p)$ for $p \in C$ to be the set of all lines L satisfying the threadability condition at p: those lines that meet C in exactly p and properly cross C at p. Let L be one line in $\mathrm{bf}(p)$, and view C as passing through L at p. Then the convex hull H^+ of the chain from p upward is above L and meets L exactly at p, and the hull H^- of the chain from p downward is below L and again meets L exactly at p. (Here "upward" and "downward" are not meant literally, but just convenient shorthand for the two portions of the curve delimited by a roughly horizontal L.) If either hull met L in more than just p, then strict threadability would be violated at L. Now rotate L counterclockwise about p until it hits C at some point other than p, and similarly clockwise. The stopping points determine the butterfly *wing-lines*. See Fig. 3.

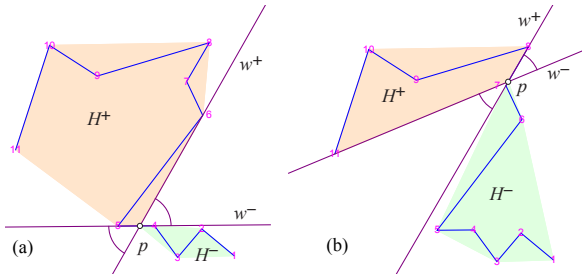

Figure 3: Here C is fixed, and two $\mathrm{bf}(p)$'s are shown. Note the hulls H^+ and H^- meet at exactly p. (a) The stopping point ccw is vertex 6 and cw it is vertices $4, 5$.

Thus $\mathrm{bf}(p)$ is an open double wedge. Its two boundary wing-lines w^+ and w^- (which are not part of $\mathrm{bf}(p)$) must both be externally supported by points of C distinct from p. Each wing must touch C on at least one of its two halves with respect to p. Note by our definition, $\mathrm{bf}(p)$ can never be a line; rather it becomes empty when the wing-lines merge to one line.

3 Upper and Lower Hulls

It is not difficult to see that the upper convex hull H^+ changes continuously (say, under the Hausdorff distance measure) as p moves along C, and similarly for H^-. This has long been known in the work on computing "kinetic" convex hulls of continuously moving points (although we have not found an explicit statement). Roughly, because each point in the convex hull of a finite set of points is a convex combination of those points, moving one point p a small amount ε changes the hull by at most a small amount δ. For more detail, see [Nie17].

Because the hulls change continuously, the butterflies change continuously as well. So we have finally established Lemma 2: If there is a line through every $p \in C$ meeting the threadability criteria, then indeed C is threadable: there are continuous rigid motions that move C through a point-hole in a line.

And now this is an immediate consequence of Lemma 2 and our definition of $\mathrm{bf}(p)$:

Lemma 3 *A curve C is threadable if and only if* $\mathrm{bf}(p)$ *is never empty for any $p \in C$.*

We can also now see the following characterization, which is the basis of the algorithm in the next section:

Lemma 4 *A curve C is threadable iff, for every $p \in C$, the upper and lower hulls intersect in exactly p, i.e., $H^+ \cap H^- = \{p\}$.*

Proof. ($\Rightarrow$:) Suppose C is threadable, but $H^+ \cap H^- \neq \{p\}$. We then show C could not be threadable.

- Case 1: $H^+ \cap H^-$ is a 2D region (Fig. 4(a)). Then p is strictly interior to one of H^+ or H^-. So, the butterfly $= \varnothing$. Therefore C is not threadable by Lemma 3.
- Case 2: $H^+ \cap H^-$ is a segment (Fig. 4(b)). Note the intersection could not consist of ≥ 2 segments, for that would violate the convexity of convex hulls. So, the butterfly wings reduce to a line; so the butterfly is empty. And again, C is not threadable by Lemma 3.

($\Leftarrow$:) Assume $H^+ \cap H^- = \{p\}$ for every p. Then, by the definition of $\mathrm{bf}(p)$, for every p the butterfly is non-empty, because one could rotate a line through p until it hits $H^{\pm}$. So Lemma 3 implies that C is threadable. □

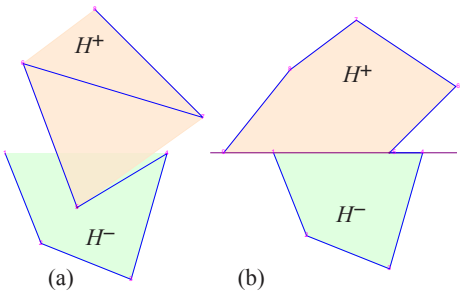

Figure 4: (a) An example of Case 1: $H^+ \cap H^-$ is a 2D region. (b) An example of Case 2: $H^+ \cap H^-$ is a segment.

4 Algorithm for Threadability

In light of Lemma 4, we can detect whether a polygonal chain is threadable by computing H^+ and H^- for all p along C, and verifying that p never falls inside either hull, i.e., ceases to be a nonflat vertex of either hull. Let p be a point on $C = (v_1, v_2, \ldots, v_n)$, which we view as moving "vertically downward" from v_1 (top) to v_n (bottom). Let the edges of C be $e_i = (v_{i-1}v_i)$. We concentrate on constructing $H = H^+$ as p moves downward along C. Clearly the same process can be repeated to construct H^-.

As p moves down along C, $H = \mathrm{hull}\{v_1, \ldots v_{i-1}, p\}$ grows in the sense that the hulls form a nested sequence. Thus once a vertex of C leaves ∂H, it never returns to ∂H (where ∂H is the boundary of H.) At any one time, p is a vertex of H. Let a_1, a_2 be the vertices of H right-adjacent to p, and b_1, b_2 the vertices left-adjacent, so that (b_2, b_1, p, a_1, a_2) are consecutive vertices of H. Finally, let A and B be the lines through a_1a_2 and b_1b_2 respectively. See Fig. 5.

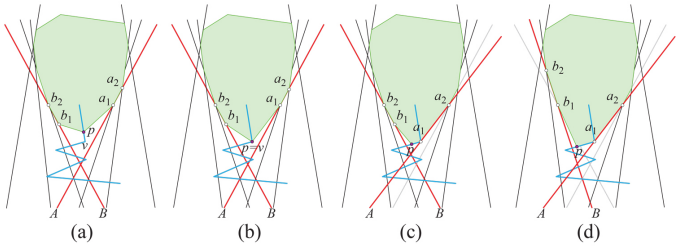

Figure 5: Algorithm snapshots. (a) H grows without combinatorial change until p reaches v. (b) $p = v$ event. (c) a_1, a_2 updated. e_i crosses B. (d) b_1, b_2 updated.

We now walk through the algorithm, whose pseudocode is displayed in the full version. Let p be on the interior of an edge $e_i = (v_{i-1}v_i)$. The portion of e_i already passed by p must lie inside H, and the remaining portion outside H. As long as p remains within the wedge region delimited by A, B, and ∂H, the combinatorial structure of H remains fixed (Fig. 5a). If p crosses A or B—say A—then a_1 leaves H and a_1, a_2 become the next two vertices counterclockwise around ∂H. If p reaches the endpoint v_i of e_i, then if e_{i+1} angles outside H, v_i becomes a new a_1 or b_1 depending on the direction of e_{i+1}. If instead, e_{i+1} turns inside H, advancing p would enter H and we have detected that C is not threadable by Lemma 4.

All the updates just discussed are constant-time updates: detecting if e_i crosses A or B, updating a_1, a_2 and b_1, b_2, and detecting if e_{i+1} turns inside H, entering $\triangle b_1 v_i a_1$.

At the end of the algorithm, H is the hull of C. It may seem surprising that we can compute the hull of C in linear time (rather than $O(n \log n)$), but Melkman showed long ago that the hull of any simple polygonal chain can be computed in linear time [Mel87]. The chain C acts almost as a pre-sorting of the points, leading to an $O(n)$ algorithm for threadability.

4.1 Rigid Motions

At any stage where the butterfly $\mathrm{bf}(p)$ is non-empty, we could choose the line L to bisect $\mathrm{bf}(p)$. This choice was used to produce the online animations cited in Section 1. To prepare for an analogous 3D-computation in Section 6, we explain the bisection choice in terms of vectors normal to L. Fig. 6(a) shows the possible L choices through p dictated by the two incident edges of H^+ and the two incident edges of H^-, illustrated by rightward rays from p along L. Rotating these 90° in (b) of the figure yields the possible vectors normal

to L. The intersection of the H^+ and H^- constraints yields an interval corresponding to bf(p), which is then bisected to select a particular N and therefore L. (The intersection always yields an interval [rather than two intervals] because each of the H^+ and H^- constraints is $\leq$ a semicircle.)

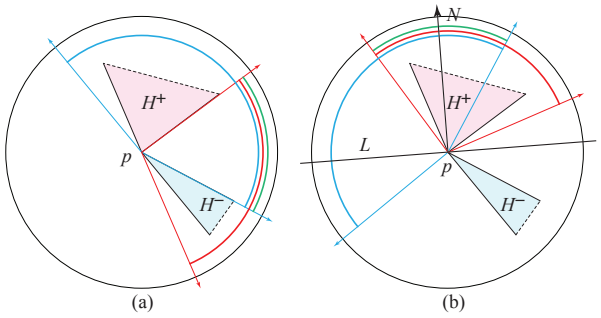

Figure 6: (a) Rightward rays along possible L's. (b) Normal vector N to L (black).

Let H_j^+ and H_j^-, $j = 1, \ldots, m$ be the sequence of hulls at the points at which there is a combinatorial change in either. Let $r_j \subseteq e$ be the range of p along edge e of C between $\{H_j^+, H_j^-\}$ and $\{H_{j+1}^+, H_{j+1}^-\}$. Then as p moves along r_j, the wings of the butterfly bf(p) have the same set of tangency points on the hulls. With L chosen as the bisector of bf(p), translation of p along r_j leads to translation and rotation of L. It is not difficult to see that the rotation implied by p moving along r_j reverses at most once, from clockwise to counterclockwise or vice versa. This is evident in Fig. 7, where the butterfly angle θ bisected to yield L has at most one local maximum. Thus each slide of p along r_j leads to at most two monotonic rotations. We call a slide and a simultaneous monotonic rotation an *elementary rigid motion*. But note that, although "elementary," these motions are not pure rotations and pure translations, but rather the particular mix determined by the slide and the butterfly bisection. We leave these elementary motions as the output rigid motions, not further analyzed into explicit analytical expressions.

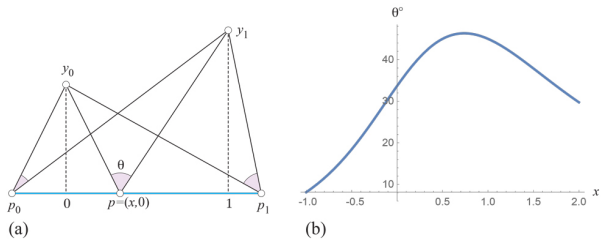

Figure 7: (a) p slides along edge e from p_0 to p_1, $r_j = (p_0, p_1) \subseteq e$. (b) The butterfly angle θ has at most one local maximum throughout the range.

Thus the sequence of $O(n)$ hulls provides a set of $O(n)$ elementary rigid motions to thread C, which we used to produce the online animations.

4.2 Difficult-to-Thread Curves

One easy consequence of our analysis is that a threadable curve need never "back-up" while threading through a hole, because p never enters $H^{\pm}$ as it progresses along the chain. However, one could define the "difficulty" of threading by, say, integrating the absolute value of the back-and-forth rotations necessary to thread. Then variations on the curve shown in Fig. 8 are difficult to thread in this sense. For each pair of adjacent spikes require a rotation by θ, and with many short spikes, there is no bound on $\sum |\theta|$ even for a fixed-length chain.[2]

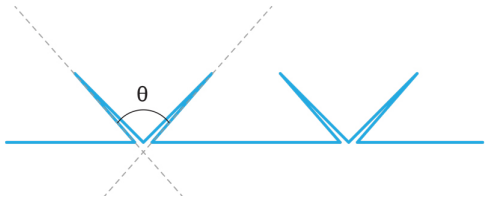

Figure 8: A threadable curve that requires repeated rotations. Animation: `http://cs.smith.edu/~jorourke/Threadable/`, Example 2.

5 Algebraic Curves

In the full version, we sketch an argument that shows detection of threadability for algebraic curves is achievable in time $O(d^4)$, where d is the degree of the curve.

6 Threadable Curves in 3D

The results in Section 4 can be extended to $\mathbb{R}^3$, asking whether a 3D polygonal chain C can pass through a point-hole in a plane. First we roughly sketch an algorithm. We claim without proof that the natural generalization of the 2D lemmas hold in 3D as well.

Again Lemma 4 is the key: we need that $H^+ \cap H^- = \{p\}$ holds for all p on C. Again computing H^+ and H^- will suffice to answer all questions; see Fig. 9. But now what was the simple wedge region between hull supporting lines A, B, and ∂H, becomes a more complex region R bounded by $O(n)$ hull-supporting planes, and the portion of ∂H formed by the faces incident to p, i.e., what is called star(p) in simplicial-complex theory (which has size $O(n)$). Setting aside complexity issues temporarily, the next edge e_{i+1} on which p will travel

[2] Thanks to Anna Lubiw for this observation.

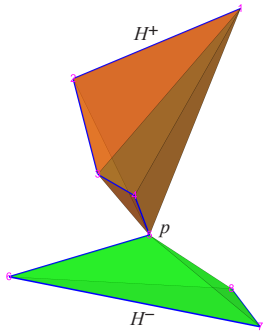

Figure 9: Upper and lower hulls for a 3D polygonal chain. Animation: `http://cs.smith.edu/~jorourke/Threadable/`, Example 6.

must be intersected with the planes bounding this region R, to determine whether R changes combinatorially, and if so, which supporting plane is first pierced by e_{i+1}.

The planes bounding R that are not determined by faces in star(p) are the planes incident to an edge of link(p), i.e., the edges of star(p) not incident to p, which form a topological circle. See Fig. 10. When e_{i+1} pierces a plane A supporting face $\triangle abc$ of H, with ab an edge of link(p), then ab is deleted from the link, and ac and cb added, and the planes incident to these new link edges are added to those defining R.

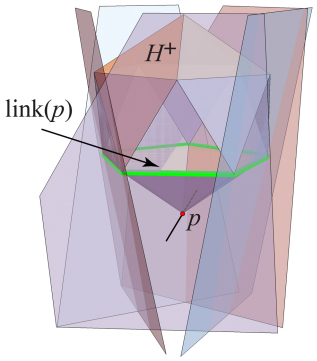

Figure 10: Faces sharing an edge with link(p) are extended to form the lower part of R. H^+ does not change combinatorially until p crosses one of those planes.

This allows H to be maintained throughout the movement of p along C. As in 2D, C is threadable if and only if p never enters either hull.

If C is threadable, selecting planes in the more complex bf(p) regions and determining rigid motions that achieve the threading are more complicated tasks than in 2D.

6.1 Updating the hull H quickly

Timothy Chan's powerful dynamic data structure for updating 3D convex hulls [Cha10] provides the tools needed to update the hull H quickly. Here "quickly" means in amortized expected $O(\text{polylog}\, n)$ time. His "nonvertical ray shooting" queries permit determining if the next edge e_{i+1} intersects a supporting plane of the region R described above, and if so, which one. Then that plane can be deleted, and new planes inserted according to the new link(p), as identified above. Thus the computation of the hulls H^+ and H^-—and therefore threadability detection—can be achieved in $O(n\, \text{polylog}\, n)$ time.

6.2 Butterfly "bisecting" planes

The equivalent of the butterfly bf(p) in 3D is a more complicated region than in 2D, and choosing a plane P through p separating H^+ and H^- (the analog of L) is correspondingly more complicated. As in 2D, we identify P by its normal vector N, say, pointing toward H^+. The outward normals to the faces of H^- incident to p form a convex geodesic polygon on the Gaussian sphere, with each node a face normal, and each geodesic arc corresponding to the dihedral angle along the edge shared by two adjacent faces. See, e.g., [BLS07]. Any point within this geodesic polygon corresponds to a normal vector whose plane supports H^- at p. Repeating this for H^+ yields another geodesic polygon corresponding to the faces of H^+ incident to p. Using outward face normals leads to normals pointing toward H^-; reflecting this geodesic polygon through the origin then orients the normals for H^+ and H^- consistently. See Fig. 11. Then the butterfly region bf(p) is determined by the intersection I of these two geodesic polygons.

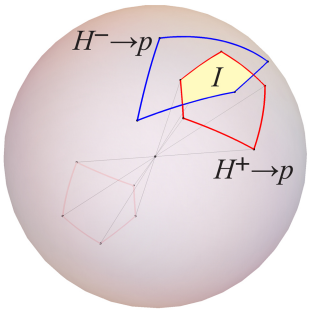

Figure 11: Gaussian sphere. The blue polygon represents the faces of H^- incident to p, and the red polygon the faces of H^+ incident to p. Any point in the (yellow) intersection I is the normal vector N of a plane P in bf(p).

The equivalent of bisecting bf(p) in 2D would be choosing the centroid of the intersection region I on

the Gaussian sphere; of course any point in the interior of I would suffice. In 2D we argued that, as p slides along an edge e between combinatorial changes in either H^+ or H^-, the rigid rotation reverses direction at most once, which led to a linear-size description of the rigid motions. In 3D, even with p on one edge e between combinatorial changes, it seems that the intersection region I on the Gaussian sphere might change $\Omega(n)$ times, requiring recalculation of $N \in I$. This complicates describing the rigid motions in a concise manner. We leave finding a clean notion of what should constitute a "elementary rigid motion" in 3D to future work, but we note that the rigid motions for threading are analytically determined and could be detailed to any precision desired.

7 Higher Dimensional Generalizations

There are two natural generalizations to higher dimensions, but neither seems a fruitful line of future inquiry. The first retains the curve as a 1-dimensional object which must pass through a hole in a hyperplane in $\mathbb{R}^d$.

The second generalization replaces the curve with a polygon P, which must pass through a slit in L. This topic has been explored previously, in two versions. We cite [Yap87] and [BVK05] and leave further discussion to the full version.

8 Open Problems

1. In $\mathbb{R}^3$, can finding a plane P separating H^+ and H^-, as sketched in Section 6, be achieved in $O(n \operatorname{polylog} n)$ time? In other words, can the intersection I of the two geodesic polygons be maintained in amortized expected $O(\operatorname{polylog} n)$ time?
2. Is there a natural definition of what constitutes an "elementary rigid motion" in $\mathbb{R}^3$, and how many such motions are needed to thread a polygonal curve of n segments?
3. If C were a hydraulic tube, it would be necessary to ensure clearance regions above and/or below L are empty of other objects to avoid collisions [AFM03]. If C represents a polygonal protein chain, clearance within a cone is important in some models [DLO06]. Finding minimum clearance regions requires more careful selection of L in $\mathrm{bf}(p)$, rather than just using the bisector as we suggest in Section 4.1. The question is most relevant in $\mathbb{R}^3$.
4. Suppose instead of C passing through a line, C were to pass through a point hole in a polygonal k-chain. What is the complexity of finding a threading motion as a function of n and k?
5. If C is not threadable, what is the shortest slit in L through which C could pass? Or, in $\mathbb{R}^3$, the smallest radius hole in a plane? Likely Yap's door width algorithm [Yap87] could apply to the 2D problem, but it would be attractive to find a hull-based approach in 2D and 3D.

Acknowledgements. We thank Mikkel Abrahamsen, Anna Lubiw, Joseph Mitchell, and the referees for helpful suggestions.

References

[AAI+01] O. Aichholzer, F. Aurenhammer, C. Icking, R. Klein, E. Langetepe, and G. Rote. Generalized self-approaching curves. *Discrete Appl. Math.*, 109:3–24, 2001.

[ACG+12] S. Alamdari, T.M. Chan, E. Grant, A. Lubiw, and V. Pathak. Self-approaching graphs. In *Internat. Symp. Graph Drawing*, pages 260–271. Springer, 2012.

[AFM03] E.M. Arkin, S.P. Fekete, and J.S.B. Mitchell. An algorithmic study of manufacturing paperclips and other folded structures. *Comput. Geom. Theory Appl.*, 25:117–138, 2003.

[BLS07] T. Biedl, A. Lubiw, and M. Spriggs. Cauchy's theorem and edge lengths of convex polyhedra. *Algorithms Data Structs.*, pages 398–409, 2007.

[BVK05] P. Bose and M. Van Kreveld. Generalizing monotonicity: On recognizing special classes of polygons and polyhedra. *Internat. J. Comput. Geom. & Appl.*, 15(06):591–608, 2005.

[Cha10] T.M. Chan. A dynamic data structure for 3-D convex hulls and 2-D nearest neighbor queries. *J. ACM*, 57(3):16, 2010.

[DLO06] E.D. Demaine, S. Langerman, and J. O'Rourke. Geometric restrictions on producible polygonal protein chains. *Algorithmica*, 44(2):167–181, 2006.

[Mel87] A.A. Melkman. On-line construction of the convex hull of a simple polyline. *Info. Proc. Letters*, 25(1):11–12, 1987.

[Nie17] M. Nientker. Convex hulls change continuously as one point moves continuously, October 2017. `https://math.stackexchange.com/q/2529897`.

[Yap87] C.-K. Yap. How to move a chair through a door. *IEEE J. Robotics Automation*, 3(3):172–181, 1987.