Threadable Curves

Joseph O'Rourke and Emmely Rogers*

January 25, 2018

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. Our main result is a linear-time algorithm for deciding whether a polygonal curve is threadable, and if so, finding a sequence of rigid motions to thread it through a hole. In addition, we sketch arguments that show that the threadability of algebraic curves can be decided in time polynomial in the degree of the curve, and that threading a 3D polygonal curve through a point-hole in a plane can be decided in quadratic time. Finally, we connect threadable curves to the problem known as "moving a chair through a doorway."

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.[1] 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 the concept, but instead say: "One might also consider a symmetric situation, where curves are ϕ-self-approaching in both directions. Generalizations to 3D are also completely open." So it appears our explorations (which also touch on 3D) are new, and in any case, focus on different properties of C.

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

[1] We thank Anna Lubiw for this reference.

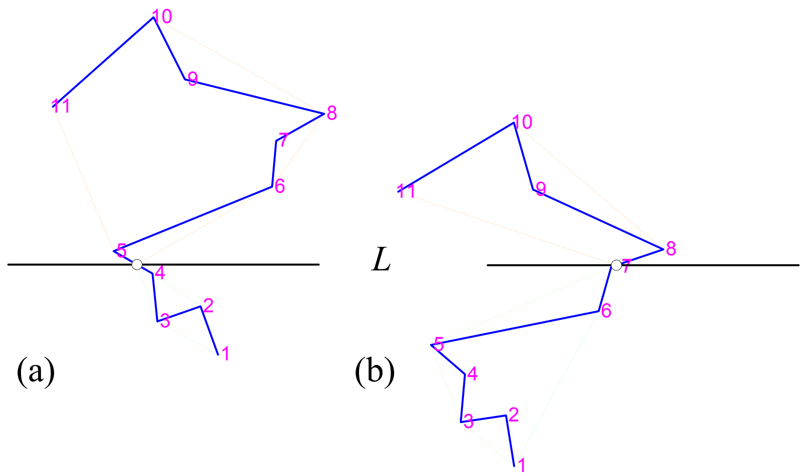

Figure 1: Two snapshots of a 10-segment polygonal chain passing through a point-hole in the x-axis.

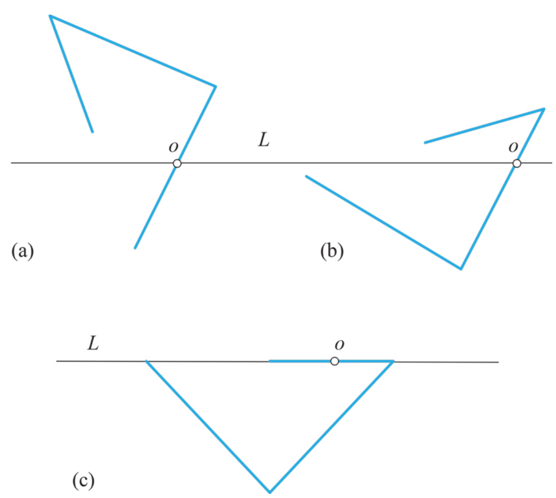

Figure 2: (a,b) A curve that is not threadable: two snapshots partially through o. (c) To pass completely through o, an edge would have to lie on L.

1.1 Definition Conseqences

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.*

Proof: This is a nearly immediate consequence of the definition, because at any one time the x-axis serves as L, meeting C at $p = o$. So one can imagine C fixed and L undergoing rigid motions. $\square$

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 "jump" rigid 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 line parallel to L 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. Monotone curves and especially monotone polygons have played a significant role in computational geometry. It remains to be seen if threadability inherits any of the advantages of monotonicity.

2 Butterflies

Define the *butterfly* $bf(p)$ 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 $bf(p)$, 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 L.) If either hull met L in more than just p, then strict threadability would be violated at L. Now rotate L counterclockwise until it hits C at some point other

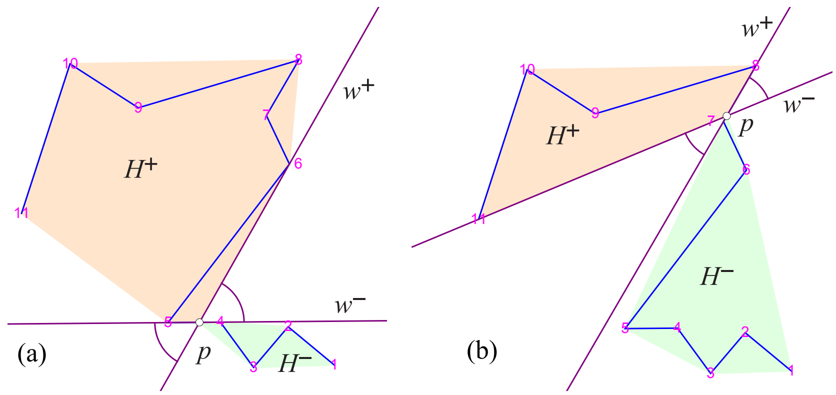

Figure 3: Here C is fixed, and two $bf(p)$'s are shown. Note the hulls H^+ and H^- meet at exactly p. (a) The stopping point ccw is vertex 6 and cw it is vertices $4, 5$.

than p, and similarly clockwise. The stopping points determine the butterfly *wing-lines*. See Fig. 3.

Thus $bf(p)$ is an open double wedge. Its two boundary wing-lines w^+ and w^- (which are not part of $bf(p)$) 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, $bf(p)$ can never be a line; rather it becomes empty when the wings-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 $bf(p)$:

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

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

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

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). 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. 5). 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 $bf(p)$, for every p the butterfly is non-empty, because one could rotate a line through p until it hit $H^{\pm}$. So Lemma 3 implies that C is threadable.

$\square$

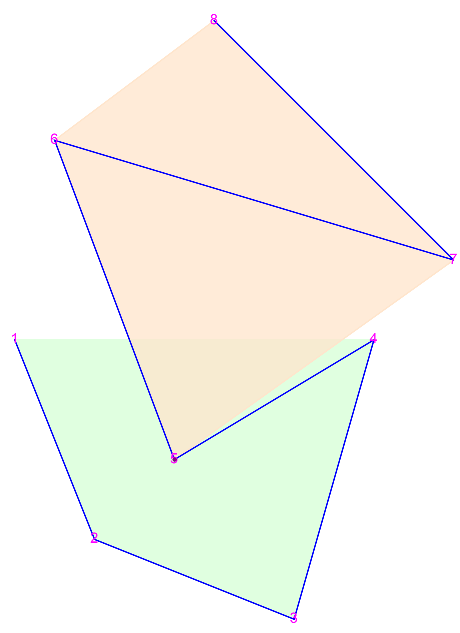

Figure 4: An example of Case 1: $H^+ \cap H^-$ is a 2D region.

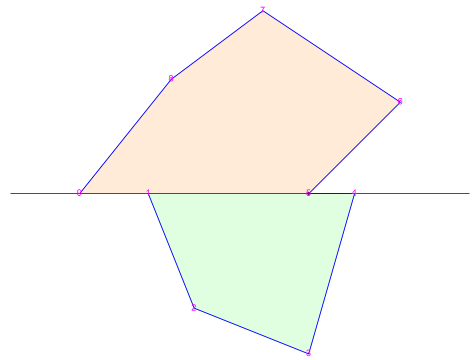

Figure 5: An example of Case 2: $H^+ \cap H^-$ is a segment.

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. Let p be a point on $C = (v_1, v_2, \dots, v_n)$, 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, $H = \text{hull}\{v_1, \dots v_{i-1}, p\}$ 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 $a_1 a_2$ and $b_1 b_2$ respectively. See Fig. 6.

We now walk through the algorithm, whose pseudocode is displayed as Algoirthm 1. 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. 6a). 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

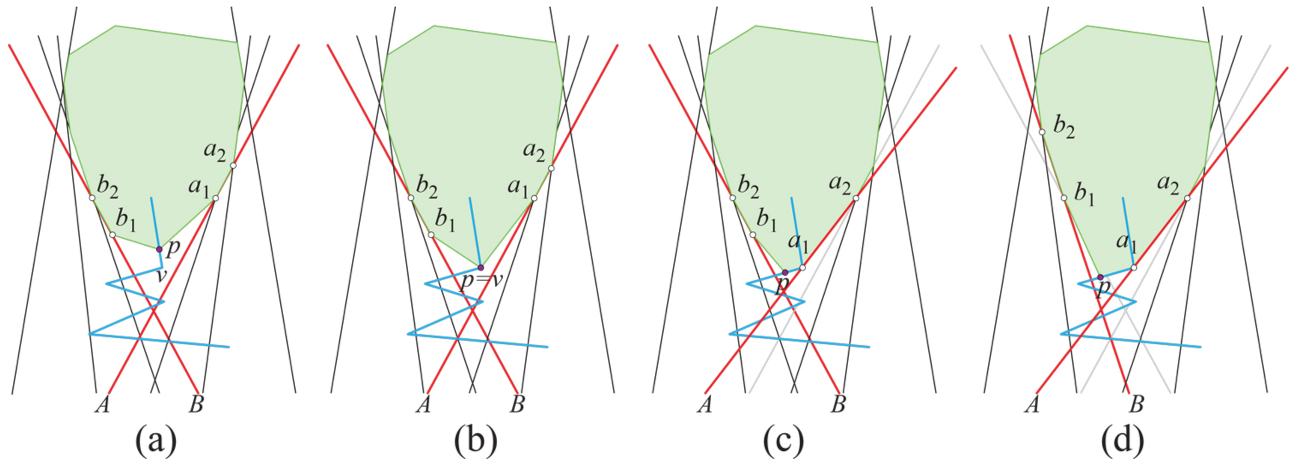

Figure 6: Algorithm snapshots. (a) H grows without combinatorial change until p reaches v. (b) $p = v$ event. (c) a_1, a_2 updated. e_i crosses B. (d) b_1, b_2 updated.

Algorithm 1: Threadable Curve Algorithm: Upper Hull

Input : Polygonal chain $C = \{v_1, \ldots, v_n\}$
Output: Upper convex hull H

```
// p:  Moving point on edge e_i = (v_{i-1} v_i).  Fig. 6(a).
// H:  Upper convex hull of {v_1, ..., v_{i-1}, p}.
// a_1, a_2:  Vertices of H right-adjacent to p.
// b_1, b_2:  Vertices of H left-adjacent to p.
// A:  line through a_1 a_2.
// B:  line through b_1 b_2.
```

while *p has not reached last vertex v_n* **do**
  Compute next event on e_i: Intersect e_i with A and B.
  if *Next event is vertex $v = v_i$.* **then** `// Fig. 6(b)`
    if *Turn at $p = v$ enters $\triangle a_1, v, b_1$ and so enters H* **then**
      return NotThreadable
    end
    if *Turn at $p = v$ angles outside H, so next edge e_{i+1} is on H* **then**
      Update A or B. `// Fig. 6(c).`
    end
  end
  if *Next event is intersection with A or B* **then**
    Update A or B, whichever intersected.
    `// Fig. 6(d).`
  end
end

computed in linear time [Mel87].[2] The chain C acts almost as a pre-sorting of the points.

4.1 Rigid Motions

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. Choosing, say, the line L that bisects $bf(p)$, the range of p along e leads to a translation of p along r_j and a rotation of L. Thus the sequence of hulls provides a set of rigid motions to thread C, which we used to produce the online animations cited in Section 1.

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. 7 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.[3]

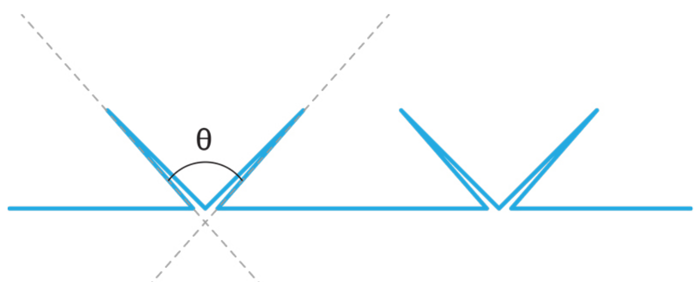

Figure 7: A threadable curve that requires repeated rotations. Animation: `http://cs.smith.edu/~jorourke/Threadable/`.

5 Algebraic Curves

Here we sketch an argument that shows detection of threadability for algebraic curves is achievable in time polynomial in the degree of the curve. We use this lemma:

[2] See Dan Sunday's description: `http://geomalgorithms.com/a12-_hull-3.html`.

[3] Thanks to Anna Lubiw for this observation.

Lemma 5 *Let C have a non-empty butterfly at $p_1 \in C$, and an empty butterfly at $p_2 \in C$. Then for some $p^* \in C$ between p_1 and p_2, $bf(p^*)$ is empty and the wing-lines coincide in a line L that is tangent to C at two (or more) points.*

Proof: The existence of p^* follows from the continuity of the butterflies: As a point p moves from p_1 to p_2, the non-empty butterfly at p_1 must disappear before p_2 is reached. Let p be close to the disappearing point p^*, with $bf(p)$ non-empty with wings w^+ and w^-. Each of w^+ and w^- must be tangent to C at a point, and the two tangency points must be distinct. As p approaches p^*, at some stage these tangency points will no longer discontinuously change. Then at p^*, $A = B = L$ passes through those limit tangency points, t_1 and t_2 in Fig. 8. □

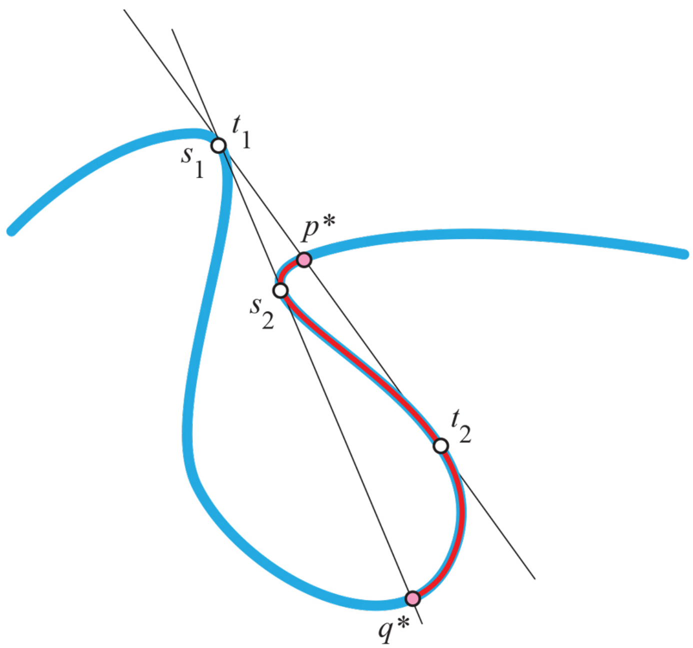

Figure 8: A non-threadable smooth curve. Red section has no butterflies. Both $bf(p^*) = \varnothing$ and $bf(q^*) = \varnothing$. t_1, t_2 and s_1, s_2 are the wing-line tangency points, for p^*, q^* respectively.

This lemma allows us to detect threadability by checking all the double tangencies (*bi-tangents*) of C, as follows. Let L be a bi-tangent of C, tangent at t_1 and t_2. If L does not cross C at some other point p, then it is irrelevant to threadability. Suppose L does cross C uniquely at p. Then check whether or not this implies an empty $bf(p)$. This depends on whether p is between t_1 and t_2 (p^* in Fig. 8) or outside those tangencies along L (q^* in the figure), and whether C is locally left or right at the tangency points.

If, for every bi-tangent L, and every corresponding crossing p, $bf(p)$ is non-empty, then C is threadable. Otherwise, it is not threadable. The time-complexity of this algorithm is dependent on the number of bi-tangents. The other computations (intersecting L with C, whether C is left or right at a tangency) are achievable within the degree of C.

It is known that the number of bi-tangents to a curve of algebraic degree d is $O(d^4)$ [Økl17], and they can be listed in that time. So, without delving into details, we can see that the threadability of an algebraic curve can be decided in time polynomial in 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. Here we just roughly sketch an algorithm.

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 A, B, and ∂H,

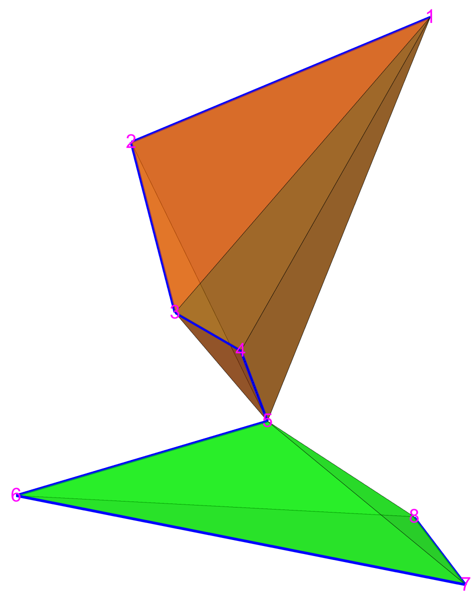

Figure 9: Upper and lower hulls for a 3D polygonal chain.

becomes a more complex region R bounded by $O(n)$ planes, and a portion of ∂H (which has size $O(n)$). Although it seems quite likely the intersection of the next edge e_{i+1} on which p will travel with the planes bounding this region R could be computed in $O(\log n)$ time with appropriate data structures, we have not pursued this in detail, and so we only claim linear-time per step, which leads to quadratic time overall. Again, if p never enters either hull, then C is threadable. And if C is threadable, selecting planes in the more complex $bf(p)$ regions leads to rigid motions that achieve the threading.

7 Moving a Chair through a Doorway

Finally we make a connection between threadable curves and a classic problem, moving a polygon through a segment-slit in a line, or, as it was known, "moving a chair through a doorway." Chee Yap solved the problem with an innovative quadratic algorithm [Yap87], which both computed the "door-width" of the polygon—the narrowest door through which it could pass—and provided the rigid motions necessary to execute the passage. Here we cannot improve his results, but instead reinterpret a version of the problem in terms of threadability.

Given a simple polygon $\mathcal{P}$, partition it at vertices a and b into two chains, which we'll call P and Q. See Fig. 10. For p on P and q on Q, say the

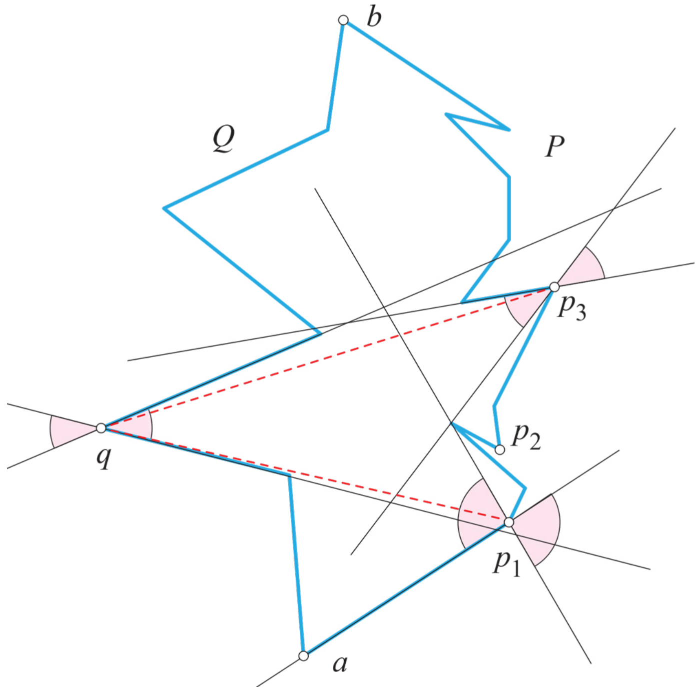

Figure 10: $bf(q)$ is compatible with $bf(p_1)$ and with $bf(p_3)$, but not with $bf(p_2)$.

butterflies $bf(p)$ and $bf(q)$ are *compatible* if the segment pq falls inside both butterflies. Here the butterflies are defined for each chain P or Q, ignoring the other chain. Then it would be possible to place the polygon $\mathcal{P}$ in a slit in the line L containing pq. The doorway width needed there is $|pq|$. Note however, that the set of points $p \in P$ that are compatible with $bf(q)$ need not form a subchain of P, as illustrated in Fig. 10.

Imagine now parametrizing p and q along the chains P and Q over $[0, 1]$, and creating the following surface S over the 1×1 square, whose lowerleft corner is $p(0) = q(0) = a$ and upper-right corner is $p(1) = q(1) = b$:

$$\begin{aligned} d(p,q) &= |pq| \text{ if } bf(p) \text{ and } bf(q) \text{ are compatible} \\ d(p,q) &= \infty \text{ if incompatible} \end{aligned}$$

Finding a continuous path ρ on S between the a and b corners, avoiding all $d(p, q) = \infty$ points of S, corresponds to finding a sequence of motions to move $\mathcal{P}$ through a segment-slit in a line L, entering with a and exiting with b, such that the intersection of $\mathcal{P}$ with L is always a segment (pq). And the lowest path ρ—the ρ whose maximum on S is minimized—corresponds to the door-width of such a motion.

Note that the condition that the intersection of $\mathcal{P}$ with L is always a segment is not needed to move $\mathcal{P}$ through a slit in L. In fact, it may be that it is better not to attempt to maintain this condition. Fig. 11 illustrates this. Here $\mathcal{P}$ fits

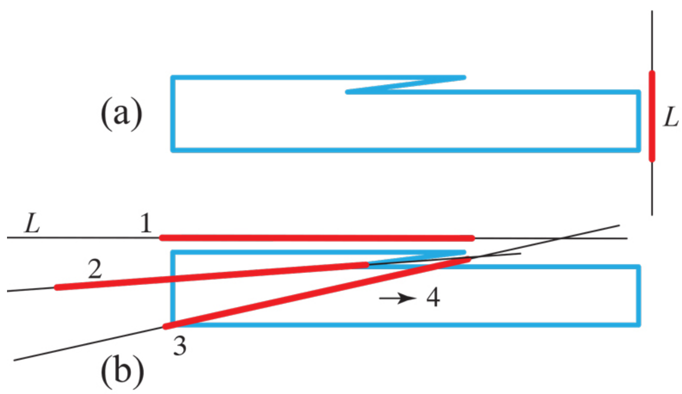

Figure 11: (a) Moving a polygon $\mathcal{P}$ through a slit (red) in L. (b) Moving such that the intersection with L is always a segment.

through a rather narrow doorway (a), but at some stages the intersection $L \cap \mathcal{P}$ is two segments. Insisting on $L \cap \mathcal{P}$ remaining one segment requires a wider doorway (b). This leads us to a generalization of monotone polygons.

7.1 Threadable Polygons

We noted in Section 1.2 that threadable polygonal chains can be viewed as generalizations of monotone chains. Similarly we can define a generalization of a monotone polygon: a *threadable polygon* $\mathcal{P}$ can be partitioned by vertices a and b into two chains P and Q such that, for every $p \in P$, there is a line L that meets $\partial\mathcal{P}$ at exactly two points $p \in P$ and $q \in Q$, and crosses each there, and similarly when reversing the roles of p and q. Note that if $\mathcal{P}$ is threadable, then the chains P and Q are threadable. However, it could be both chains are threadable, but the polygon is not (at least as partitioned by a, b). Fig. 11a provides an example when partitioned into upper and lower chains. A threadable polygon $\mathcal{P}$ can move through a slit in L so that $L \cap \mathcal{P}$ is always a single segment, as described above. Properties of threadable polygons remain to be explored.

7.2 Backing Up

Finally we note a difference between threading a curve through a point in a line L and moving a polygon through a slit in L: the polygon might have to "back-up" or reverse, in that some points of $\partial\mathcal{P}$ might first be on one side of L, later on the opposite side, and later still back on the original side. An example is shown in Fig. 12,[4] based on an idea in [JO90]. Fig. 12a shows the polygon, with two slots a and b aligned with two spikes a' and b'. The door-width is just a bit greater than $|aa'| = |bb'|$, which is greater than the height of the rectangle. Points along the edge x of $\partial\mathcal{P}$ exhibit the back-up phenomenon. Fig. 12b shows snapshots of L reaching the b' spike at position 2. Then L rotates $(3, 4)$ to reach the b slot. Because the a slot is narrow, it cannot be used to circumvent the b' spike, so L must move over to the b slot. Then it can move up the slot and over the tip of the b' spike. Fig. 12c shows snapshots as L performs similar manuevering to clear the a' spike. Notice that the segment x is traversed three times: forward (left-to-right), backward, and forward again. In terms of the surface S mentioned previously, this polygon requires a path ρ shaped like the letter `S` narrowly skirting $d(p, q) = \infty$ regions.

Adding more spikes and slots to the example can lead to points of $\partial\mathcal{P}$ passing back-and-forth through L a linear number of times.

8 Open Problems

1. Define the *minimum above/below clearance* for a threadable curve C as the minimum width region above and below L through which points of C pass as it threads through the hole o, with width the dimension parallel to L. (See again Fig. 1, and the animations at `http://cs.smith.edu/~jorourke/Threadable/`.) If C were a rigid pipe (e.g., a hydraulic tube), it would be necessary to ensure the clearance regions are empty of other objects to avoid collisions. Finding the minimum requires more careful selection of L in $bf(p)$, rather than just using the bisector as we suggest in Section 4.1.
2. Detail an algorithm for threadability of a polygonal curve in $\mathbb{R}^3$, as sketched in Section 6. Can $O(n \log n)$ be achieved?

[4] Incidentally, this is a threadable polygon when partitioned into upper and lower chains.

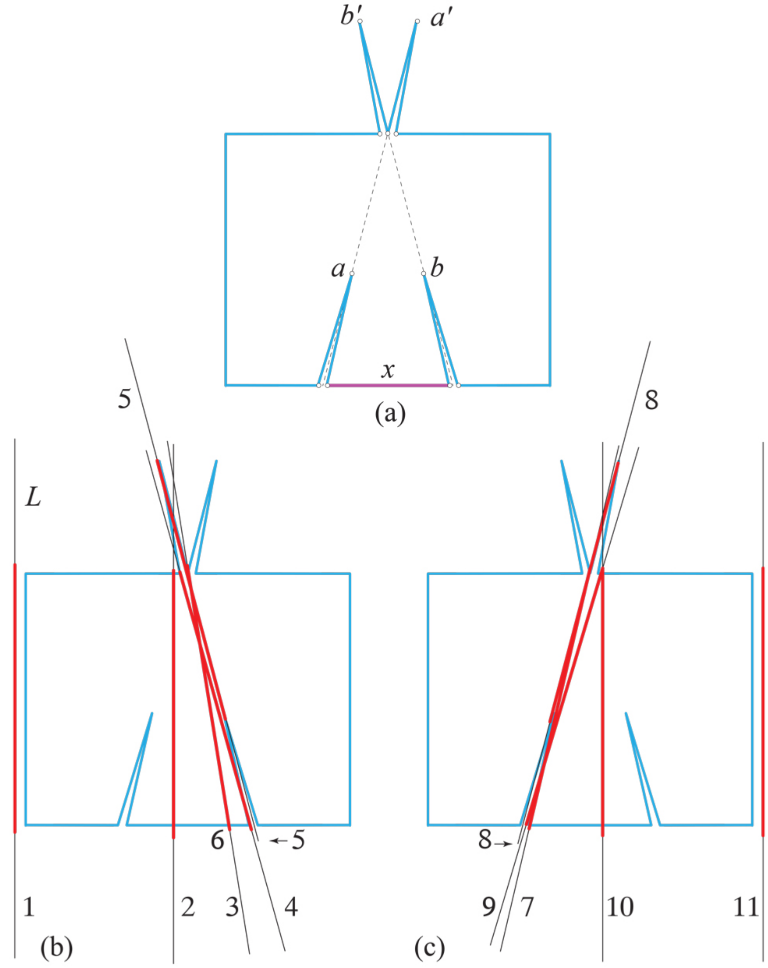

Figure 12: (a) A polygon that requires "back-up." (b,c) Snapshots of the position of L. The red subsegment of L is the doorway.

3. Can a simple, connected algebraic curve of degree d have $\Omega(d^4)$ bi-tangents? The d^4 bound mentioned in Section 5 is achieved for quartics by disconnected, closed zero-sets [O'R17].

4. Do threadable polygons (Section 7.1) have interesting/useful properties?

References

[AAI$^+$01] Oswin Aichholzer, Franz Aurenhammer, Christian Icking, Rolf Klein, Elmar Langetepe, and Günter Rote. Generalized self-approaching curves. *Discrete Appl. Math.*, 109:3–24, 2001.

[JO90] Susan Jones and Joseph O'Rourke. A note on moving a chair through a doorway. *Algorithms Review*, 1(3):139–149, 1990. S. Jones is now S. Dorward.

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

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

[Økl17] Jan-Magnus Økland. Number of double tangents to an algebraic curve of degree d. Mathematics Stack Exchange, October 2017. `https://math.stackexchange.com/q/2494999/237`.

[O'R17] Joseph O'Rourke. Number of bitangents to connected algebraic curve. MathOverflow, October 2017. `https://mathoverflow.net/q/284929`.

[Yap87] Chee-Keng Yap. How to move a chair through a door. *IEEE Journal on Robotics and Automation*, 3(3):172–181, 1987.