Convexifying Monotone Polygons

Therese C. Biedl*† Erik D. Demaine* Sylvain Lazard‡†
Steven M. Robbins§ Michael A. Soss§

Abstract

This paper considers reconfigurations of polygons, where each polygon edge is a rigid link, no two of which can cross during the motion. We prove that one can reconfigure any monotone polygon into a convex polygon; a polygon is *monotone* if any vertical line intersects the interior at a (possibly empty) interval. Our algorithm computes in $O(n^2)$ time a sequence of $O(n^2)$ moves, each of which rotates just four joints at once.

1 Introduction

An interesting area in computational geometry is the reconfiguration of (planar) *linkages*: collections of line segments in the plane (called *links*) joined at their ends to form a particular graph. A *reconfiguration* is a continuous motion of the linkage, or equivalently a continuous motion of the joints, that preserves the length of each link. We further enforce that links do not cross, that is, do not intersect during the motion. For a survey of work on linkages where crossing is allowed, see the paper by Whitesides [12].

The case of noncrossing links has had a recent surge of interest. The most fundamental question [10] is still open: Can every chain be reconfigured into any other chain with the same sequence of link lengths? Here a *chain* is a linkage whose underlying graph is a path. Because reconfigurations are reversible, an equivalent formulation of the question is this: Can every chain be *straightened*, that is, reconfigured so that the angle between any two successive links is π? This question has been posed independently by several researchers, including Joseph Mitchell, and William Lenhart and Sue Whitesides [8]. It has several applications, including hydraulic tube and wire bending, and sheet metal folding [10].

At first glance, it seems intuitive that any chain can be "unraveled" into a straight line, but experimentation reveals that this is a nontrivial problem. Indeed, no such "general unraveling" motions have been formally specified. Because the problem is so elusive, it is

*Department of Computer Science, University of Waterloo, Waterloo, Ontario N2L 3G1, Canada, {biedl, eddemaine}@uwaterloo.ca

†Research performed during a post-doctoral position at McGill University.

‡INRIA Lorraine – LORIA, Projet ISA, 615 rue du jardin botanique B.P. 101, 54602 Villers les Nancy Cedex, France, lazard@loria.fr

§School of Computer Science, McGill University, 3480 University Street, Montréal, Québec H3A 2A7, Canada, {stever, soss}@cgm.cs.mcgill.ca

natural to look at special classes of linkages and prove that at least they can be straightened. For example, consider the class of *monotone* chains, where every vertical line intersects the chain at a point or not at all. Such a chain can easily be straightened by repeatedly (1) rotating the first link until it lines up with the second link, and (2) fusing these links together into a single “first link.” This motion induces no crossings because it preserves monotonicity throughout.

This paper addresses the analogous question of straightenability for *polygons* (a linkage whose underlying graph is a cycle): Can every polygon be reconfigured into a convex polygon? In other words, can every polygon be *convexified*? This question was also raised by the researchers mentioned above. Note that a convex polygon can be reconfigured into any other convex polygon with the same clockwise sequence of link lengths, and hence the question is equivalent to the fundamental question for polygons [10]: Can every polygon be reconfigured into any other polygon with the same clockwise sequence of link lengths?

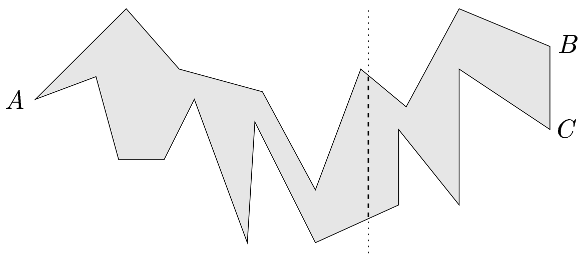

Figure 1: A monotone polygon: Every vertical line intersects the interior at an interval. The extreme left and right vertices split the polygon boundary into an upper chain from A to B and a lower chain from A to C. The vertical edge (B, C) belongs to neither chain.

In this paper we focus on the case of monotone polygons. Similar to the case of chains, a polygon is *monotone* if the intersection of every vertical line with the interior of the polygon is an interval, that is, either a single vertical line segment, a point, or the empty set. See Figure 1 for an example. A monotone polygon consists of two chains, the upper chain and the lower chain. Each chain is *weakly monotone* in the sense that the intersection with a vertical line is either empty, a single point, or a vertical edge.[1] The left [right] ends of the upper and lower chains may be identical (like point A in Figure 1), or they may be connected by a vertical edge (like edge (B, C) in Figure 1). The vertical edge (B, C) belongs to neither chain.

In contrast to monotone chains, it is nontrivial to convexify monotone polygons. In this paper, we show that this is possible by a fairly simple motion consisting of a sequence of $O(n^2)$ moves. We use just a single type of move, changing the angles of only four joints, which we show is the fewest possible. While the proof of correctness is nontrivial, our algorithm for computing the motion is simple and efficient, taking $O(n^2)$ time.

1.1 Related Work

Let us briefly survey the work on reconfiguring linkages whose links are not allowed to cross.

[1]All straight (angle-π) vertices are removed, so there is no possibility of two adjacent vertical edges.

The most related result, by Bose, Lenhart, and Liotta [3], is that all monotone-separable polygons can be convexified. A *monotone-separable* polygon is a monotone polygon whose upper and lower chains are separated by a line segment (connecting the common ends of the chains). Their motion involves translating almost all joints in the upper chain at once, and appears not to extend to general monotone polygons.

The only other result about convexifying classes of polygons is that every star-shaped polygon can be convexified [6]. A polygon is *star-shaped* if its boundary is entirely visible from a single point. This motion rotates all joints simultaneously, and it seems difficult to find a motion involving few joints [13].

The remaining related results are for types of linkages other than polygons. For tree linkages, it is known that the answer to the fundamental problem is "no" [2]: There are some trees which cannot be reconfigured into other trees with the same link lengths and planar embedding. Indeed, there can be an exponential number of trees with the same link lengths and planar embedding that are pairwise unreachable. The complexity of determining whether a tree can be reconfigured into another remains open.

Another way to change the problem is to allow linkages in higher dimensions. If we start with a polygon in the plane, and allow motions in three dimensions, then every polygon can be convexified [11]. Indeed, a 1935 problem by Erdős [5] asks whether a particular sequence of moves through 3D, each rotating only two joints (called a "flip"), converges in finite time. While the answer is positive [9], the number of moves is unfortunately unbounded in n, as first proved by Joss and Shannon [7]. Recently, it was shown that $O(n)$ moves of a different kind suffice [1]; each rotates at most four joints.

If the polygon lies in three dimensions and we want to convexify it by motion through three dimensions, then it is surely not convexifiable if it is knotted. But there are unknotted polygons that cannot be convexified [1]. The complexity of determining whether a polygon in 3D can be convexified also remains open. Amazingly, Cocan and O'Rourke [4] have shown that every polygon in d dimensions can be convexified through d dimensions for any $d \geq 4$.

1.2 Outline

The rest of this paper is organized as follows. Section 2 begins with a more formal description of the problem. Section 3 describes our algorithm for computing the motion. Sections 4 and 5 prove its correctness and bound its performance, respectively. We conclude in Section 7.

2 Definitions

This section gives more formal definitions of the concepts considered in this paper: linkages, configurations, and motions.

Consider a graph, each edge labeled with a positive number. Such a graph may be thought of as a collection of distance constraints between pairs of points in a Euclidean space. A *realization* of such a graph maps each vertex to a point, also called a *joint*, and maps each edge to the closed line segment, called a *link*, connecting its incident joints. The link length must equal the label of the underlying graph edge. If a graph has one or more such realizations, we call it a *linkage*.

An embedding of a linkage in space is called a *configuration* of the linkage. In a *simple configuration*, any pair of links intersect only at a common endpoint, and in this case the links must be incident at this joint in the linkage. We consider only simple configurations in this paper. A *motion* of a linkage is a continuous movement of its joints respecting the link lengths such that the configuration of the linkage remains simple at all times.

In this paper, we consider linkages embedded in the plane whose graph is a single cycle. The configurations are simple polygons, and so divide the plane into the *exterior* and *interior* regions (distinguished by the fact that the interior region is bounded). Joints of an n-link linkage are labeled $j_0, j_1, \ldots, j_{n-1}$ in a counterclockwise manner: traversing the boundary in sequence $j_0, j_1, \ldots, j_{n-1}$ keeps the interior region on the left. The *joint angle* θ_i is the interior angle at joint j_i: $\theta_i = \angle j_{i+1} j_i j_{i-1} \in (0, 2\pi)$. We call a joint *straight* if the joint angle is π, *convex* if the joint angle is strictly less than π, and *reflex* otherwise. A configuration is *convex* if none of its joints are reflex.

The question considered is whether every monotone configuration of a cyclic linkage (or *polygon*) can be *convexified*, that is, moved to a convex configuration. We show this is true by giving an algorithm to compute such a motion.

3 Algorithm

As input, the algorithm requires a description (that is, the joint coordinates) of a polygon with n links. In each main step, the algorithm computes a sequence of $O(n)$ moves that ultimately straighten a joint. This joint angle is then held fixed so that it remains straight forever after, effectively reducing the number of joints. As the algorithm continues, the configuration has fewer and fewer nonstraight joints. We stop when no reflex joint is left, and so the polygon is convex as desired.

First we need some notation for basic geometric concepts. Let p and q be two points in the plane. Define $\|pq\|$ to be the Euclidean distance between p and q. If p and q are distinct, define $\mathrm{ray}(p, q)$ to be the ray originating at p and passing through point q. We use "left" and "right" in two different senses, one for points and one for rays. A point r is *left* or *right* of $\mathrm{ray}(p, q)$ if it is strictly left or right (respectively) of the oriented line supporting $\mathrm{ray}(p, q)$. A point p is *left* or *right* of point q if p has a strictly smaller or larger x coordinate than q, respectively. In both cases, we use *nonstrictly* left/right to denote left/right or equality, i.e., neither left nor right.

The algorithm works as follows:

Algorithm Convexify

- Until the polygon is convex:
  - Find a rightmost reflex joint (a joint with maximum x coordinate, breaking ties arbitrarily).
  - Relabel joints counterclockwise along the polygon so that this rightmost reflex joint is j_1, and all straight joints are ignored.
  - If j_1 belongs to the lower chain:

1. Compute the largest index p such that $j_2, \ldots, j_{p-1}$ are right of ray(j_0, j_1).
2. Until a joint has straightened:
   (a) Perform the following move (see Figure 2) until either the joints j_0, j_1, and j_{p-1} become collinear; or one of the joints $\{j_0, j_1, j_{p-1}, j_p\}$ straightens:

      i. Fix the positions of joints j_p, j_{p+1}, $\ldots$, j_{n-1}, and j_0.
      ii. Fix all joint angles except those at j_0, j_1, j_{p-1}, and j_p.
      iii. Rotate j_1 clockwise about j_0.
      iv. Move joint j_{p-1} as uniquely defined by maintaining the distances $\|j_1 j_{p-1}\|$ and $\|j_{p-1} j_p\|$.
      v. Move joint j_i, $2 \le i \le p-2$, as uniquely defined by maintaining the distances $\|j_1 j_i\|$ and $\|j_i j_{p-1}\|$.

   (b) Update the coordinates of j_1 and j_{p-1}.
   (c) If j_0, j_1, and j_{p-1} are collinear, then decrement p, because with the new positions, j_{p-1} is on ray(j_0, j_1). Also update the coordinates of the new j_{p-1}.
3. Update the coordinates of any remaining joints that have moved.

– If j_1 belongs to the upper chain, the algorithm is similar.

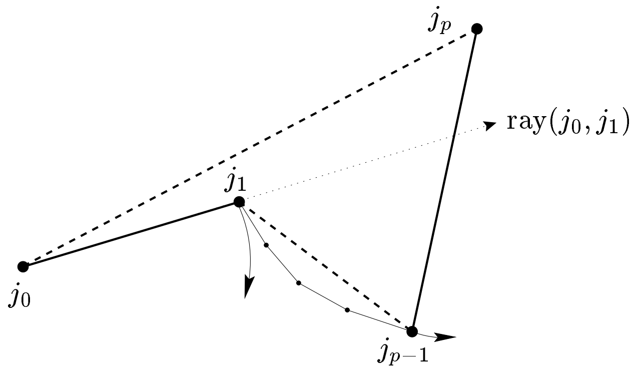

Figure 2: The movement of joints j_1 and j_{p-1}. The thick dashed lines represent "virtual links" between j_1 and j_{p-1}, and between j_0 and j_p, the lengths of which are preserved throughout the move.

First let us justify that the algorithm is well-defined.

Lemma 1 *The definition of p in Step 1 is well-defined and at least* 3.

Proof: Because j_1 is reflex, ray(j_0, j_1) must intersect the polygon elsewhere than the segment (j_0, j_1). This implies that there are joints on both sides of the ray, and hence p is well-defined. Furthermore, because j_1 is reflex and the joints are oriented counterclockwise on the polygon, j_2 is right of ray(j_0, j_1). Hence, 2 is a valid value for $p-1$, so $p \ge 3$. □

Note further that the number of simultaneously rotating joints (four) is the best possible, because any motion of a polygon that rotates just three joints reconfigures a virtual triangle, which is rigid.

The next two sections prove correctness and analyze the performance of this algorithm.

4 Proof of Correctness

In this section, we prove the following theorem.

Theorem 1 *Given any monotone polygon, Algorithm Convexify computes a convexifying motion, during which the polygon remains simple and monotone.*

The difficulty is in showing that the polygon remains monotone and simple during the motion. For the remainder of this section, assume without loss of generality that the link (j_0, j_1) is on the lower chain. First we need some trivial but important observations.

Lemma 2 *During each move, no reflex angle becomes convex and no convex angle becomes reflex.*

Proof: Because all other joint angles are fixed, any such transition means that a joint j_0, j_1, j_{p-1}, or j_p straightens, which stops the move. □

Lemma 3 *Joints $j_2, \ldots, j_{p-1}$ are convex.*

Proof: Consider such a joint j_i. Because j_1 is a rightmost reflex vertex, j_i is convex if it is right of j_1. By monotonicity and the property that j_i is right of ray(j_0, j_1), j_i cannot be left of j_1. The only case that remains is when j_i has the same x coordinate as j_1. Because we ignore straight joints, j_i must in fact be j_2 in this case. Because j_1 is reflex, j_2 must be below j_1. Now j_3 must be strictly right of j_2, and hence the angle at j_2 is convex. □

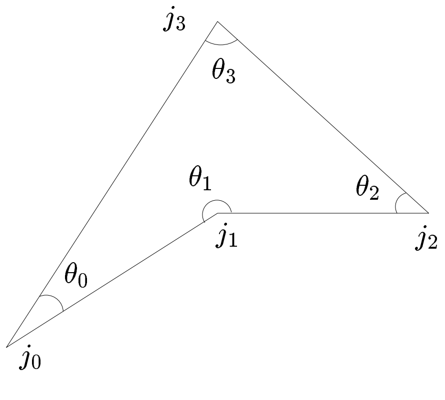

Figure 3: Illustration of Lemma 4.

Next we need a general result about movement of quadrangles.

Lemma 4 [1] *Consider a simple quadrangle j_0, j_1, j_2, j_3 (in counterclockwise order) with j_1 reflex. (See Figure 3.) Let θ_i denote the interior angle at joint j_i. If the linkage moves so that j_1 rotates clockwise about j_0, then θ_1 decreases and θ_i increases for all $i \in \{0, 2, 3\}$, until θ_1 straightens. In other words, all of the angles approach π.*

By the definition of p, j_1 is reflex in the quadrangle $Q = (j_0, j_1, j_{p-1}, j_p)$, so we can apply Lemma 4 to Q and obtain the following result about our motion:

Lemma 5 *During each move in Step 2a, j_{p-1} rotates counterclockwise about j_p, and the joint angles θ_1 and θ_{p-1} both approach π.*

Next we analyze the movement of j_1 relative to j_{p-1}'s reference frame, determined by fixing the position of j_{p-1} and keeping the axes parallel to the world frame's. This can be visualized by imagining that during the motion we translate the entire linkage so that j_{p-1} stays in its original position.

Lemma 6 *j_1 rotates counterclockwise about j_{p-1}.*

Proof: Consider the relative movement of j_1 and j_p about j_{p-1}. Joint j_p is rotating counterclockwise about j_{p-1} and the angle $\angle j_1 j_{p-1} j_p$ is increasing by Lemma 5. Hence, j_1 must also be rotating counterclockwise about j_{p-1}. $\square$

We are now in the position to prove that the polygon remains simple and monotone throughout the motion.

Proof (Theorem 1): The only way that simplicity or monotonicity can be violated is that either a link intersects another link, or a vertical link rotates in the "wrong" direction. The wrong direction for link (j_i, j_{i+1}) on the lower [upper] chain is when j_{i+1} becomes left [right] of j_i. Consider the first time at which a link intersects another, or a vertical link rotates in the wrong direction.

Suppose first that a vertical link (j_i, j_{i+1}) rotates in the wrong direction. Because only joints $j_1, \ldots, j_{p-1}$ move, we must have $0 \leq i \leq p-1$. We distinguish three cases:

Case 1: Link (j_0, j_1) is vertical

Because j_1 is reflex and j_2 is nonstrictly right of j_1, j_1 must be above j_0. But j_1 rotates clockwise about j_0, so monotonicity is preserved.

Case 2: Link (j_1, j_2) is vertical

Because j_1 is reflex, j_1 is above j_2. By Lemma 6, j_1 rotates counterclockwise about j_{p-1}. Hence, the rigid triangle $j_{p-1} j_1 j_2$ rotates counterclockwise about j_{p-1}. Thus because the link (j_1, j_2) is vertical, j_1 is above j_2, and j_{p-1} is (nonstrictly) right of j_1, j_1 moves left of j_2. Thus, monotonicity is preserved.

Case 3: Link (j_i, j_{i+1}) is vertical, $2 \leq i \leq p-1$

We show that joints j_i and j_{i+1} are both convex. First, if $i \leq p-2$, then this follows by Lemma 3. Second, if $i = p-1$, then j_{p-1} and j_p must be right of j_1 in order for them to be on opposite sides of ray(j_0, j_1). Hence, j_{p-1} and j_p must be convex because they are right of j_1 which is a rightmost reflex vertex.

Thus, joints j_{i-1} and j_{i+2} are both left of the link (j_i, j_{i+1}); that is, (j_{i-1}, j_i) belongs to the lower chain and (j_{i+1}, j_{i+2}) belongs to the upper chain. This means that link (j_i, j_{i+1}) joins the top chain to the bottom chain (like link BC in Figure 1). The polygon remains monotone no matter which way the link moves.

Now suppose that two links intersect each other, but the polygon remained simple and monotone before this time. By Lemma 5, the joint angles θ_1 and θ_{p-1} approach π, and hence the chain of moving links $(j_0, \ldots, j_p)$ cannot self-intersect. Hence, the only concern is whether any of these links could intersect the rest of the polygon.

In the following, refer to Figure 4. Let u denote the original position of $\mathrm{ray}(j_0, j_1)$, and v denote the downward vertical ray emanating from j_0. Let W be the wedge right of ray u and left of ray v.

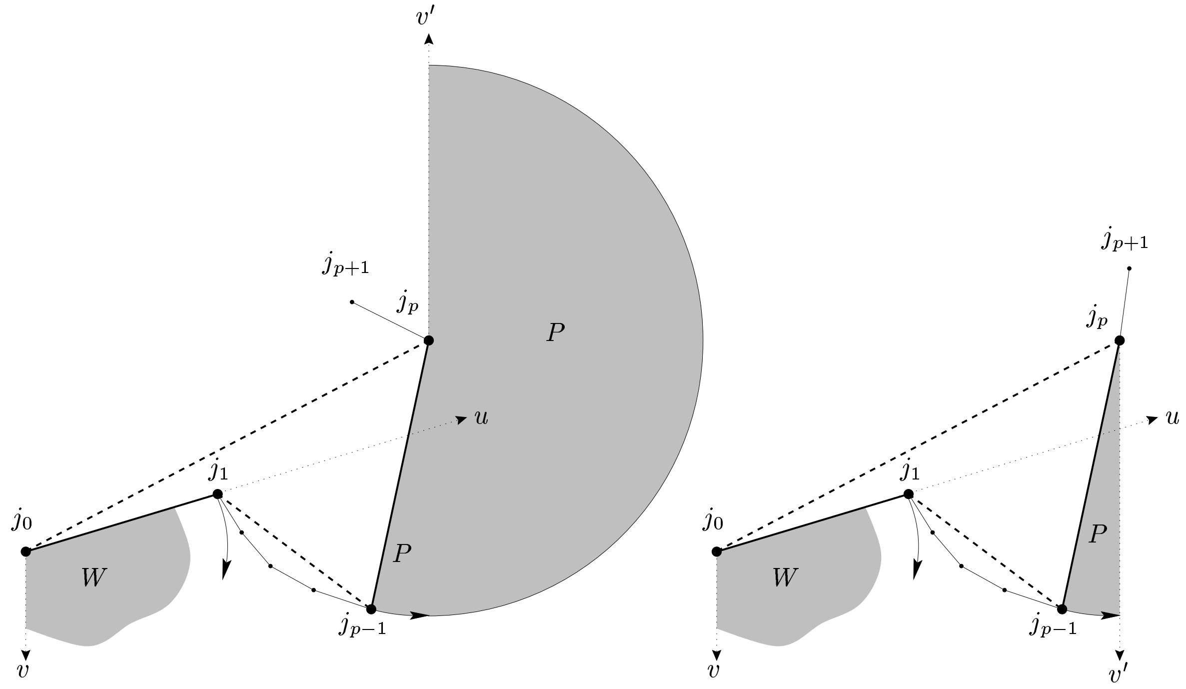

Figure 4: Definition of v, v', W, and P. (Left) When (j_p, j_{p+1}) is not on the lower chain. (Right) When (j_p, j_{p+1}) is on the lower chain.

Joint j_1 starts on the boundary of W; because it rotates clockwise about j_0, it enters region W at the start of the move. By the choice of p, joints $j_2, \ldots, j_{p-1}$ all lie to the right of u. These joints must also lie to the left of v, because otherwise the polygon would not be monotone. Thus, after the move starts, the chain $j_1, \ldots, j_{p-1}$ lies inside W.

We now argue that the only joints that can be inside W are $j_1, \ldots, j_{p-1}$. We know that j_0 and j_p are not interior to W. If some other joint lies inside W, the chain must cross one of the boundaries of W. The chain cannot cross ray v, because that would violate monotonicity. Nor can the chain cross ray u, because that would require a reflex vertex to the right of j_1 or a violation of monotonicity. Because none of $j_p, \ldots, j_{n-1}, j_0$ are moving, this chain remains outside W during the motion.

To establish that the polygon remains simple, we claim further that the joints $j_1, \ldots,$ j_{p-1} never leave W. Suppose to the contrary that one does. Let j_i be the first such joint to leave W. It must reach either ray u or ray v. Consider each possibility in turn.

j_i **crosses** v**:** Because j_1 stays reflex (unless it straightens which stops the move), it cannot be the first to cross v. If j_i crosses v for $1 < i < p$, then we have both j_0 and j_i left of

j_1, so the chain is not monotone, a contradiction.

j_i crosses u: Because j_1 rotates clockwise about j_0, it never crosses u. If j_2 crosses, j_1 cannot be reflex, a contradiction. If j_{p-1} crosses, the move must have already stopped from j_0, j_1, and j_{p-1} becoming collinear because j_1 rotates clockwise about j_0. Finally, if j_i crosses u for $2 < i < p-1$, j_i must be reflex because j_{i-1} and j_{i+1} are both right of u, contradicting Lemma 3.

Hence, the links of the chain $j_0, \ldots, j_{p-1}$ always remain inside W, so they cannot intersect links outside of W.

This leaves just one moving link, (j_{p-1}, j_p). By Lemma 5, j_{p-1} rotates counterclockwise about j_p. Define v' to be the vertical ray emanating from j_p that points away from the interior of the polygon, preferring upward if both directions are possible. Thus, v' points downward [upward] when the link (j_p, j_{p+1}) is [not] on the lower chain. Note that v' does not change throughout the move.

Let P be the pie wedge bounded by the link (j_{p-1}, j_p), the ray v', and the counterclockwise circular arc, centered at j_p, starting at j_{p-1} and ending on v' (at distance $\|j_p j_{p-1}\|$ from j_p). Because monotonicity is preserved up to this point, P is empty of nonmoving links. P also contains the entire sweep of (j_{p-1}, j_p): if v' points upward, j_{p-1} cannot cross v' without first becoming collinear with j_0 and j_1; if v' points downward, it cannot cross without first straightening j_p. □

5 Time and Move Bounds

Finally we establish the time and move bounds on the algorithm. Our model of computation is a real random-access machine supporting comparisons, basic arithmetic, and square roots.

Lemma 7 *Each iteration of Step 2 takes constant time.*

Proof: The computations in this step are computing which of the five candidate events for stopping the motion occurs first (Step 2a), and updating $O(1)$ coordinate positions (Steps 2b and 2c).

In order to compute the halting event, we define the following (refer to Figure 5). Here x^* denotes that x changes during the move.

$$\begin{aligned}
&a = \|j_0 j_1\|, \quad b = \|j_1 j_{p-1}\|, \quad c = \|j_{p-1} j_p\|, \\
&d = \|j_p j_0\|, \quad m^* = \|j_1 j_p\|, \\
&\varphi_0^* = \angle j_p j_0 j_1^*, \quad \varphi_1^* = \angle j_0 j_1 j_{p-1}^*, \\
&\varphi_{p-1}^* = \angle j_1^* j_{p-1}^* j_p, \quad \varphi_p^* = \angle j_{p-1}^* j_p j_0.
\end{aligned}$$

Note that m^* increases during the move, because j_1^* rotates clockwise about j_0, increasing φ_0^* and thus (by law of cosines) m^*. Hence, we parameterize the move by the diagonal distance m^*. More precisely, we determine the halting event by computing the value of $(m^*)^2$ for each of the five candidate events, and choosing the event with smallest $(m^*)^2$.

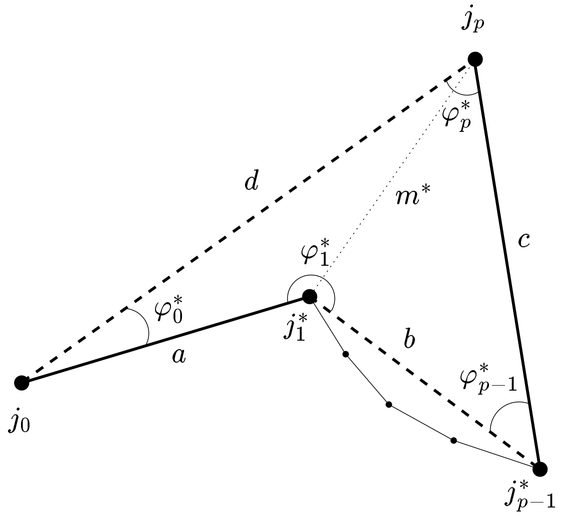

Figure 5: Illustration of the proof of Lemma 7.

Now the event that j_i straightens happens when its angle θ_i^* is π, and the event that j_0, j_1^*, and j_{p-1}^* become collinear happens when $\varphi_1^* = \pi$. Note however that j_1^* can only straighten after j_0, j_1^*, and j_{p-1}^* become collinear (because $\angle j_0, j_1^*, j_{p-1}^*$ is initially reflex), and hence we do not need to consider this event. Using the law of cosines, we obtain the following solutions to these event equations:

j_0 **straightens:** $(m^*)^2 = a^2 + d^2 + 2ad\cos(\angle j_{n-1}j_0j_p)$ and $\angle j_{n-1}j_0j_p < \pi$.

j_{p-1}^* **straightens:** $(m^*)^2 = b^2 + c^2 + 2bc\cos(\angle j_{p-2}j_{p-1}j_1)$.

j_p **straightens:** Provided $\angle j_0j_pj_{p+1} < \pi$ (which is necessary for this event), $(m^*)^2$ is the solution of a quadratic polynomial involving a, b, c, d, and $\cos(\angle j_0j_pj_{p+1})$. This reduces to an arithmetic expression involving square roots.

j_0, j_1^*, j_{p-1}^* **collinear:** $(m^*)^2 = (ac^2 + bd^2 - ab(a+b))/(a+b)$, because $\cos\alpha_1^* = -\cos\beta_1^*$.

The minimum of these event occurrences determines when to stop the move. The new joint coordinates can then be computed as follows: Suppose that we know the new coordinates of joints j_{i-1} and j_i (initially $i = 0$). Let v be the vector from j_i to j_{i-1}, rescaled to have the known length $\|j_ij_{i+1}\|$. Rotate this vector clockwise by the new θ_i, which only involves cosines and sines of this angle, and then add it to the point j_i. The result is the new position of joint j_{i+1}^*. We can similarly compute the new coordinates of j_{p-1}^* from the coordinates of j_p and j_{p+1}. Thus, each update of j_1^*, j_{p-1}^* (Step 2b), and possibly j_{p-2}^* (Step 2c) takes constant time as desired. The full update (Step 3), which is only executed $O(n)$ times, takes linear time. □

Theorem 2 *Algorithm Convexify computes $O(n^2)$ moves in $O(n^2)$ time.*

Proof: By Lemma 7, any one move takes $O(1)$ time to compute. Any execution of Step 2 therefore takes $O(n)$ time, because initially $p \le n-1$, and p decreases with every move, until a joint straightens and Step 2 terminates. All other steps can also be implemented in $O(n)$ time. One iteration of the main loop thus takes $O(n)$ time. Because each iteration straightens a joint, the polygon becomes convex after $O(n)$ iterations. Hence, there are $O(n^2)$ moves, which are computed in $O(n^2)$ time. □

6 Lower Bound

This section proves a simple lower bound on reconfiguring polygons, showing that we cannot hope to do better than rotating four joints at once.

Theorem 3 *Any motion of a polygon requires changing at least four joint angles at once.*

Proof: Suppose that only three joints, A, B, and C, rotate. Because the joints between A and B have fixed angles, the distance between A and B is fixed. Thus, we can ignore the joints between A and B and just consider the "virtual link" (A, B). Similarly, we can consider virtual links (B, C) and (C, A). We are left with a triangle A, B, C of virtual links. Because a triangle is rigid, no motion of only three joints can move the linkage. □

7 Conclusion

We have presented an $O(n^2)$-time algorithm to compute a sequence of $O(n^2)$ moves, each rotating the minimum possible number of four joints at once, that reconfigures a given monotone polygon into a convex polygon with the same link lengths. This is the first algorithm to convexify a nontrivial class of polygons by rotating a constant number of joints in each move. By running the algorithm twice we can find a motion between any two monotone polygons with the same clockwise sequence of link lengths.

Several interesting open problems remain. First, can our algorithm be improved to use $o(n^2)$ moves each rotating $o(n)$ joints? More generally, what is the tradeoff between the number of simultaneously rotated joints and the number of moves?

Our result adds to the class of polygons that are known to be convexifiable; previously, the only nontrivial classes were star-shaped polygons [6] and monotone-separable polygons [3]. A natural area of research is to explore more general classes of polygons. For example, is there a convexifiable class containing both monotone and star-shaped polygons (other than trivial classes like the union)?

Acknowledgments

We thank William Lenhart, Anna Lubiw, Godfried Toussaint, and Sue Whitesides for helpful discussions. This work was partially supported by FCAR and NSERC.

References

[1] 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. Technical Report 060, Smith College, 1999. A preliminary version appeared in the *Proceedings of the 10th Annual ACM-SIAM Symposium on Discrete Algorithms*, Baltimore, Maryland, January 1999, pages 866–867.

[2] Therese Biedl, Erik Demaine, Martin Demaine, Sylvain Lazard, Anna Lubiw, Joseph O'Rourke, Steve Robbins, Ileana Streinu, Godfried Toussaint, and Sue Whitesides. On reconfiguring tree linkages: Trees can lock. In *Proceedings of the 10th Canadian Conference on Computational Geometry*, Montréal, Québec, Canada, August 1998. `http://cgm.cs.mcgill.ca/cccg98/proceedings/cccg98-biedl-reconfiguring.ps.gz`.

[3] Prosenjit Bose, William Lenhart, and Giuseppe Liotta. Personal communication, 1999.

[4] Roxana Cocan and Joseph O'Rourke. Polygonal chains cannot lock in 4D. In *Proceedings of the 11th Canadian Conference on Computational Geometry*, Vancouver, Canada, August 1999. `http://www.cs.ubc.ca/conferences/CCCG/elec_proc/c17.ps.gz`.

[5] Paul Erdős. Problem 3763. *American Mathematical Monthly*, 42:627, 1935.

[6] H. Everett, S. Lazard, S. Robbins, H. Schröder, and S. Whitesides. Convexifying star-shaped polygons. In *Proceedings of the 10th Canadian Conference on Computational Geometry*, Montréal, Québec, Canada, August 1998. `http://cgm.cs.mcgill.ca/cccg98/proceedings/cccg98-everett-convexifying.ps.gz`.

[7] Branko Grünbaum. How to convexify a polygon. *Geombinatorics*, 5:24–30, July 1995.

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

[9] Béla de Sz. Nagy. Solution to problem 3763. *American Mathematical Monthly*, 46:176–177, March 1939.

[10] Joseph O'Rourke. Folding and unfolding in computational geometry. In *Proceedings of the Japan Conference on Discrete and Computational Geometry*, Lecture Notes in Computer Science, Tokyo, Japan, December 1998. To appear.

[11] Godfried Toussaint. The Erdős-Nagy theorem and its ramifications. In *Proceedings of the 11th Canadian Conference on Computational Geometry*, Vancouver, Canada, August 1999. `http://www.cs.ubc.ca/conferences/CCCG/elec_proc/fp19.ps.gz`.

[12] Sue Whitesides. Algorithmic issues in the geometry of planar linkage movement. *Australian Computer Journal*, 24(2):42–50, May 1992.

[13] Sue Whitesides. Personal communication, October 1998.