Computing Signed Permutations of Polygons

Greg Aloupis * Prosenjit Bose† Erik D. Demaine‡ Stefan Langerman* Henk Meijer§
Mark Overmars¶ Godfried T. Toussaint*

Abstract

Given a planar polygon (or chain) with a list of edges $\{e_1, e_2, e_3, \ldots, e_{n-1}, e_n\}$, we examine the effect of several operations that *permute* this edge list, resulting in the formation of a new polygon. The main operations that we consider are: *reversals* which involve inverting the order of a sublist, *transpositions* which involve interchanging subchains (sublists), and *edge-swaps* which are a special case and involve interchanging two consecutive edges. Using these permuting operations, we explore the complexity of performing certain actions, such as convexifying a given polygon or obtaining its mirror image. When each edge of the given polygon has also been assigned a *direction* we say that the polygon is *signed*. In this case any edge involved in a reversal changes direction. The complexity of some problems varies depending on whether a polygon is signed or unsigned. An additional restriction in many cases is that polygons remain simple after every permutation.

1 Introduction

Much focus has been placed recently on the problem of sorting a permutation of n integers by *reversals* [HP98, BP95]. As one might guess, a single reversal is applied to a consecutive set of these integers and the result is that their order is inverted. The key problem that arises is determining the minimum number of reversals necessary to sort a given permutation. This number is called the *reversal distance* of the permutation. A variation of this problem involves *signed* permutations [BMY01]. In this case any integer affected by a given reversal also changes *parity*.

Each of these interesting combinatorial problems has its roots in bioinformatics and molecular biology [HP98, HP96, BP95, CJM+00, BMY01]. Specifically, genomes have been modeled as linear sequences, where each element in a sequence is a *block* of smaller elements that are never separated. A popular model for mutation involves inverting parts of these sequences. In order to determine the number of such mutations needed to transform one genome to another, one may compute the reversal distance of the associated permutations. An extension of this model is to consider the direction of each *block*. This leads to the study of signed permutations.

The problem of computing *transposition distance* also stems from bioinformatics. In this case, a transposition involves exchanging two disjoint sets of consecutive integers in a permutation. Computing reversal distance has been shown to be NP-hard [Cap99] for unsigned permutations, but for the signed version a linear time algorithm exists [BMY01]. Computing transposition distance is of unknown complexity [CJM+00]. The reader may also be interested in [BHK01, BP98, EEK+01].

In this paper we extend the ideas mentioned above from one dimension to two. Instead of considering permutations of integers, we consider permutations of *edges* which form polygons or chains. We define operations such as *edge-swaps*, *reversals* and *transpositions*, in analogy to $\mathbb{R}^1$. We introduce the notions of *signed permutations of polygons and chains*. These concepts give rise to a wide range of problems to be solved.

2 Definitions

First we introduce the notion of a *signed polygon* or *signed permutation* of a polygon. Any polygon P may be described by a list of edges $\{e_1, e_2, e_3, \ldots, e_{n-1}, e_n\}$. A signed polygon is no different, except that each edge is also assigned a direction. The same holds for chains. This is a generalization of the notion of parity that is used in $\mathbb{R}^1$. If the directions of all edges are consistent as we traverse a polygon or chain, then we say that this polygon or chain is *oriented*.

Without loss of generality, suppose that we are dealing with an oriented polygon. A *transposition* of two edges A and B involves interchanging their positions so that the resulting polygon remains oriented. This is illustrated in Figure 1. If A and B are consecutive, this operation is defined as an *edge-swap* or plainly *swap*. It

*School of Computer Science, McGill University. {athens,sl,godfried}@uni.cs.mcgill.ca
†Carleton University. jit@scs.carleton.ca
‡MIT Laboratory for Computer Science, 200 Technology Square, Cambridge, MA 02139, USA, edemaine@mit.edu
§Queens University. henk@cs.queensu.ca
¶University of Utrecht. markov@cs.uu.nl

is not difficult to see that entire subchains may also be transposed. A *single-edge* transposition involves transposing an edge with an empty subchain. A *reversal* of a

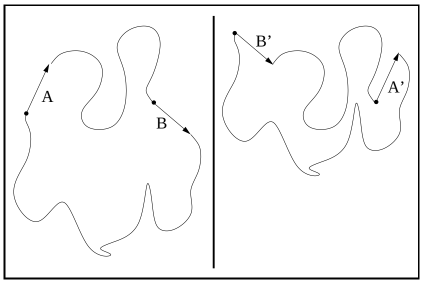

Figure 1: Transposing two edges A and B.

subchain belonging to a polygon involves inverting the order of the edges in the subchain. Geometrically this rotates the subchain rigidly by an angle of π so that its endpoints are placed exactly at each other's original location. For unsigned polygons this operation appears identical to the *flipturn* operation introduced by Joss and Shannon [GZ01]. However, here we allow reversals to take place on any subchain, not only on pockets. For signed polygons the direction of each edge involved in the reversal is switched, as is done for parity in $\mathbb{R}^1$. In Figure 2 we illustrate a reversal of subchain $\{e_i, \ldots, e_j\}$ for a signed (initially oriented) polygon. One can see

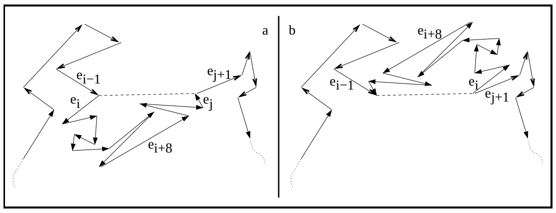

Figure 2: Reversing a subchain of a signed polygon.

that for unsigned polygons, an edge-swap is merely a transposition *or* a reversal of two consecutive edges. For signed polygons there is a difference in the resulting direction of each edge.

Each of the operations above results in the same shape when used on a polygon, regardless of the direction of its edges. However, for chains one can imagine alternate definitions. This will be discussed in more detail in the full paper. At the moment, we will mention only an interpretation which is used in Theorem 3.6 in the next section: Any operation performed on an oriented chain must preserve orientation. For example, the reversal of an edge will not only change its direction; The edge will have to be shifted to recreate an oriented chain.

3 Permuting Polygons

In this section we mention some results concerning permutations of simple polygons or chains. Proofs have been condensed and more details will be available in the full paper.

In Figure 3a we show a polygon which does not admit any edge-swaps. Examples such as this one may be extended easily to create any n-gon which will not admit edge-swaps. Figure 3b shows a polygon which does

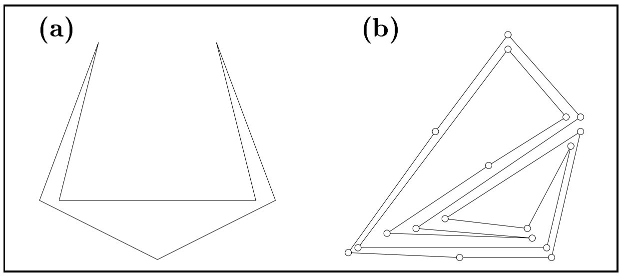

Figure 3: Polygon (a) does not admit edge-swaps. Polygon (b) does not admit single-edge transpositions.

not admit single-edge transpositions, with the exception of a few edge-swaps for some edges that are almost collinear. These transpositions cannot change the basic shape. Thus we see that sometimes local permutations will not be sufficient to achieve desired reconfigurations.

Theorem 3.1 *A simple polygon may be convexified with $O(n^2)$ reversals while maintaining simplicity after each reversal.*

Proof: This result holds for the more restricted reversal operation of flipturns [ABC$^+$00]. □

Theorem 3.2 *A star-shaped polygon can be convexified with $O(n^2)$ edge-swaps while maintaining star-shapedness after each edge-swap, and this bound is tight in the worst case.*

Proof (sketch): Let k be a point in the kernel and without loss of generality suppose that the polygon is oriented clockwise. If the polygon is not convex, there must exist two successive edges $\overrightarrow{ab}$ and $\overrightarrow{bc}$ which form a left hand turn (see Figure 4a). Since the polygon is star-shaped, b is the only vertex in the cone formed by the half-lines ka and kc. If we edge-swap $\overrightarrow{ab}$ and $\overrightarrow{bc}$, we obtain the configuration shown in Figure 4b. The new position of b (shown as b') must be somewhere in the triangle (a, c, k'). Thus the swapped edges are still visible from k, and they do not interfere with the other edges of the polygon, so the polygon remains star-shaped. The polygon will become convex only when there are no swaps to be made on left hand turns. Any pair of edges may be swapped at most once, since

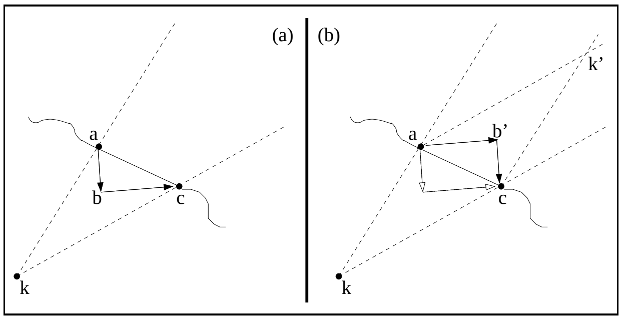

Figure 4: Using edge-swaps to convexify a star-shaped polygon.

they will never form a left hand turn again after being swapped. Therefore $O(n^2)$ swaps suffice to convexify a star-shaped polygon. In the full paper we will show that this bound is tight. □

Theorem 3.3 *Determining whether a signed polygon may be permuted (using transpositions or swaps) so that its shape is rotated by an angle of π takes $\Theta(n \log n)$ time in the algebraic decision tree model.*

Proof (sketch): Each edge of the polygon must have its *opposite* also present in the polygon. This property also suffices if we do not impose the restriction of maintaining simplicity at all times. By "opposite" we mean an edge with the same angle, but opposite direction. Each such pair must be interchanged. We can determine whether every edge is matched by performing a radial sort.

If we translate every vector to the origin we obtain a set of n points. The shape of the polygon can be rotated if and only if every such point has a reflection through the origin. Thus we obtain the matching lower bound by a reduction from *Set Equality* to this problem. For details on this technique, see [LS00, ACG+]. □

Theorem 3.4 *Determining whether we can obtain the mirror image of a signed polygon by permutations requires $\Theta(n \log n)$ time in the algebraic decision tree model.*

Proof (sketch): In order to be able to obtain a mirror image, there must exist an axis through which every edge has its reflection present (allowing translation of one edge). This is illustrated in Figure 5. Which lines in the plane may serve as an axis of reflection? Suppose we place each edge (vector) at the origin. The axis must be *on* one of the edges or it must be a *bisection* of two consecutive edges in angular order. The symmetries of this set of vectors may be found in $O(n \log n)$ time using the Knuth-Morris-Pratt string matching algorithm [KMP77]. This is pointed out by Eades [Ead88], who also mentions that several independent authors have produced matching lower bounds. □

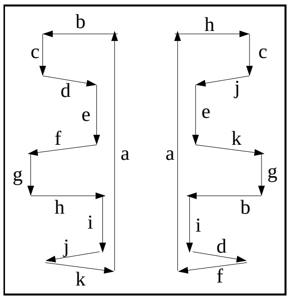

Figure 5: Two polygons that are mirror images and have different permutations of the same edge list.

Theorem 3.5 *Given an oriented polygon P and a rectangle R, deciding whether P can be permuted by transpositions into an oriented polygon P' that can be drawn inside R is (weakly) NP-complete.*

Proof: Consider a partition problem with $S = \{a_0, a_1, \ldots, a_{n-1}\}$ and $a_i > 0$ for all i. Let $A = \sum_{a \in S} a/2$. Deciding whether there is a subset S' of S with $\sum_{a \in S'} a = A$ is NP-hard. Consider the following polygon P. Denote the edges of P in counter-clockwise order by $\{e_0, e_1, \ldots, e_{2n+3}\}$. The edges with even indices are parallel to the x-axis; the edges with odd indices are parallel to the y-axis. Let ϵ be a positive number less than a_i for all i. The edge e_0 has length $a_0 + \epsilon$. Edges e_i for $i = 2, 4, 6, \ldots, 2n-2$ have length $a_{i/2}$. Edge e_{2n} has length A. Edge e_{2n+2} has length $A + \epsilon$. Edges e_i for $i = 1, 3, 5, \ldots, 2n+1$ have length 1. Edge e_{2n+3} has length $n + 1$.

We also assign directions to the edges, so that the edges form a counter-clockwise traversal of P. All edges of length 1 go up. The edges e_i for $i = 0, 2, 4, \ldots, 2n-2$ go from left to right. The few remaining edges go down and right to left, as illustrated in Figure 6(a) with n=7.

Let R be a rectangle of size $A+\epsilon$ by $n+1$. W.l.o.g assume that R has $(-\epsilon, 0)$ and $(A, n+1)$ as its left-bottom and right-top corner. Suppose P can be permuted into a polygon P' that can be drawn in the rectangle R. Again w.l.o.g assume that e_{2n+2} of P' lies along the top side of R and e_{2n+3} along the left side of R. This implies that the left and right endpoints of e_{2n} are $(0, y)$ and (A, y) for some value of y with $1 \le y \le n$. Moreover the edge e_0 lies below e_{2n}. The edges form a counter-clockwise traversal of P'. Since e_{2n} has a direction that goes from right to left, the horizontal edges above e_{2n} connect the left endpoint of e_{2n} with the right endpoint of e_{2n+2}, so their lengths must add up to A. Therefore the partition has a solution if and only if P can be permuted into a polygon P' that fits in R. Figure 6(b) shows a permutation of the polygon in Figure 6(a) that fits in rectangle R. □

This technique can be extended for the problems of permuting polygons so that they fit within strips or

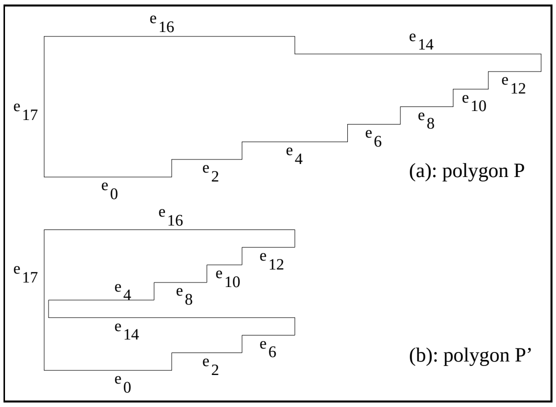

Figure 6: Polygons P and P' with 18 vertices.

circles of a given size.

Theorem 3.6 *The maximum endpoint distance for all permutations of an oriented chain may be computed in $O(n \log n)$ time.*

Proof (sketch): Fix one endpoint at the origin. Endpoint distance depends only on the direction of each edge. If we knew the direction in which to position the second endpoint, it would be a simple matter to select the direction of each edge in order to maximize the distance. Thus we can position two vectors at the origin for each edge, representing its possible directions. Now we sort the vectors radially and compute the sum of all vectors in one halfplane determined by a line ℓ through the origin. This represents the maximum distance in a direction perpendicular to ℓ. By rotating ℓ and updating the sum whenever a vector enters or exits the rotating halfplane, we can obtain the endpoint distance over all directions. This procedure takes $O(n \log n)$ time using $O(n)$ space. $\square$

Acknowledgements

Much of this work was completed at the *Workshop on Geometric Aspects of Molecular Reconfiguration* organized by Godfried Toussaint at the Bellairs Research Institute of McGill University in Barbados, February 2002. We appreciate the helpful discussions with the other participants: David Bremner, Vida Dujmovic, Herbert Edelsbrunner, Jeff Erickson, Ferran Hurtado, Patrick Morin, Joseph O'Rourke, Suneeta Ramaswami, Ileana Streinu, Yusu Wang. We also appreciate the comments of an anonymous referee.

References

[ABC+00] Hee-Kap Ahn, Prosenjit Bose, Jurek Czyzowicz, Nicolas Hanusse, Evangelos Kranakis, and Pat Morin. Flipping your lid. *Geombinatorics*, X(2):57–63, 2000.

[ACG+] Greg Aloupis, Carmen Cortes, Francisco Gomez, Michael Soss, and Godfried T. Toussaint. Lower bounds for computing statistical depth. *To appear in Computational Statistics and Data Analysis.*

[BHK01] Piotr Berman, Sridhar Hannenhalli, and Marek Karpinski. 1.375-approximation algorithm for sorting by reversals. Technical Report DIMACS TR:2001-41, 2001.

[BMY01] David A. Bader, Bernard M.E. Moret, and Mi Yan. A linear time algorithm for computing inversion distance between signed permutations with an experimental study. *Journal of Computational Biology*, 8(5):483–491, 2001.

[BP95] Vineet Bafna and Pavel A. Pevzner. Sorting by reversals: Genome rearrangements in plant organelles and evolutionary history of X chromosome. *Mol. Biol. Evol.*, 12(2):239–246, 1995.

[BP98] Vineet Bafna and Pavel A. Pevzner. Sorting by transpositions. *SIAM J. Discrete Math.*, 11(2):224–240, May 1998.

[Cap99] Alberto Caprara. Sorting permutations by reversals and Eulerian cycle decompositions. *SIAM J. Discrete Math.*, 12(1):91–110, 1999.

[CJM+00] Mary E. Cosner, Robert K. Jansen, Bernard M.E. Moret, Linda A. Raubeson, Li-San Wang, Tandy Warnow, and Stacia Wyman. A new fast heuristic for computing the breakpoint phylogeny and experimental phylogenetic analyses of real and synthetic data. In *Proc. 8th Int'l Conf. on Intelligent Systems for Molecular Biology ISMB-2000, San Diego*, pages 104–115, 2000.

[Ead88] Peter Eades. Symmetry finding algorithms. In *Computational Morphology: A Computational Geometric Approach to the Analysis of Form, Godfried T. Toussaint (ed.), North-Holland.* 1988.

[EEK+01] Henrik Eriksson, Kimmo Eriksson, Johan Karlander, Lars Svensson, and Johan Wästlund. Sorting a bridge hand. *Discrete Math.*, 241(1–3):289–300, October 2001.

[GZ01] Branko Grunbaum and Joseph Zaks. Convexification of polygons by flips and by flipturns. *Discrete Mathematics*, 241:333–342, 2001.

[HP96] Sridhar Hannenhalli and Pavel A. Pevzner. To cut or not to cut (applications of comparative physical maps in molecular evolution). In *Seventh Anuual ACM-SIAM Symposium on Discrete Algorithms*, pages 304–313, 1996.

[HP98] Sridhar Hannenhalli and Pavel A. Pevzner. Transforming cabbage into turnip (polynomial algorithm for sorting signed permutations by reversals). In *Proc. 27th Annual ACM Symposium on the Theory of Computing*, pages 178–189, 1998.

[KMP77] Donald E. Knuth, James H. Morris, and Vaughan R. Pratt. Fast pattern matching in strings. *SIAM Journal of Computing*, 6:322–350, 1977.

[LS00] Stefan Langerman and William Steiger. An optimal algorithm for hyperplane depth in the plane. In *Symposium on Discrete Algorithms*, pages 54–59, 2000.