Area- and Boundary-Optimal Polygonalization of Planar Point Sets

Sandor Fekete* Stephan Friedrichs† Michael Hemmer* Melanie Papenberg* Arne Schmidt*
Julian Troegel*

Abstract

Given a set of points in the plane, we consider problems of finding polygonalizations that use all these points as vertices and that are minimal or maximal with respect to covered area or length of the boundary. By distinguishing between polygons with and without holes, this results in eight different problems, one of which is the famous Traveling Salesman Problem. Starting from an initial flexible integer programming (IP) formulation, we develop two specific IPs and report preliminary results obtained by our implementation.

1 Introduction

Two of the fundamental structures of Computational Geometry are planar point sets and polygons. Often they come closely related, for example when asking for a *polygonalization*: for a given set V of n points in the plane, find a polygonalization with vertex set V, possibly subject to some objectives and constraints. In the 1990s, the generation of random polygons on a given point set was considered with the motivation of getting polygons as input for geometric algorithms. Auer et al. [1] named five different heuristics for the generation of random polygons. O'Rourke et al. [8] focused on random polygons, while Mitchell et al. [12] worked on random monotone polygons.

Given the geometric character of polygons, natural objectives are *boundary length* and *area*, which may be *minimized* or *maximized*. Furthermore, we may consider *simple polygons* (without holes and self-intersections) or, more generally, *polygons with holes*. Just like that, we have a family of eight basic problems of polygonalizing a set V; see Table 1 and Figure 2.

	general		simple	
	area	boundary	area	boundary
min	MinArea	MinBound	SMinArea	SMinBound
max	MaxArea	MaxBound	SMaxArea	SMaxBound

Table 1: Problem overview

*Department of Computer Science, TU Braunschweig, Germany. s.fekete@tu-bs.de, mhsaar@gmail.com, papenberg.melanie@gmail.com, arne.schmidt@tu-bs.de, j.troegel@tu-bs.de

†Max Planck Institute for Informatics, Saarbrücken, Germany. sfriedri@mpi-inf.mpg.de.

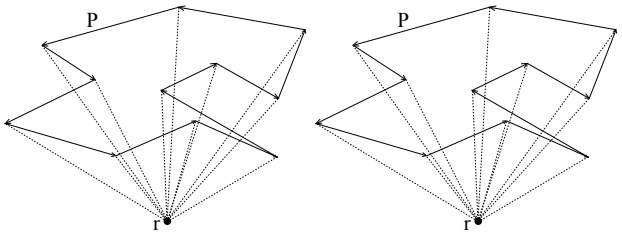

Figure 1: Calculation of the area of P: Using some reference point r, each edge forms an oriented triangle. Counterclockwise triangles contribute positive area (left), others are subtracted (right).

1.1 Related Work on Complexity

Without doubt, the most prominent family member is SMinBound, the Euclidean Traveling Salesman Problem (TSP): Thanks to the triangle inequality, a shortest tour for a given set of vertices is always non-crossing. This classical version of the problem is NP-hard, with well-established benchmark sets [9]. The complexity of MinBound is unknown; note that an optimal 2-factor does not necessarily yield an optimal solution, so the problem may still turn out to be NP-hard. Problems of maximum-boundary length have also been studied, and are related (but not identical) to the Maximum Traveling Salesman Problem [5, 2], whose complexity on a planar point set is Problem #49 of the famous Open Problems Project list [3]. To the best of our knowledge, the complexity of MaxBound as well as SMaxBound is open. Dumitrescu and Tóth [4] gave a $2/\pi$-approximation algorithm for a broad class of instances of SMaxBound. Note that all of these problems have to deal with the additional difficulty of computing a sum of square roots, which is a classical open problem: #33 in [3].

As opposed to the difficulty concerning Euclidean distances, the area of a polygon with rational vertices is rational and can be computed efficiently; see Figure 1. Optimizing area is also a natural problem. While it has received less attention than TSP and its variants, its complexity has been resolved. Fekete [7, 6] showed that SMinArea and SMaxArea are both NP-hard problems; using the same construction, we can conclude that MinArea is also NP-hard. With a separate construction (not given here due to space limitations), we can show that MaxArea is NP-hard as well.

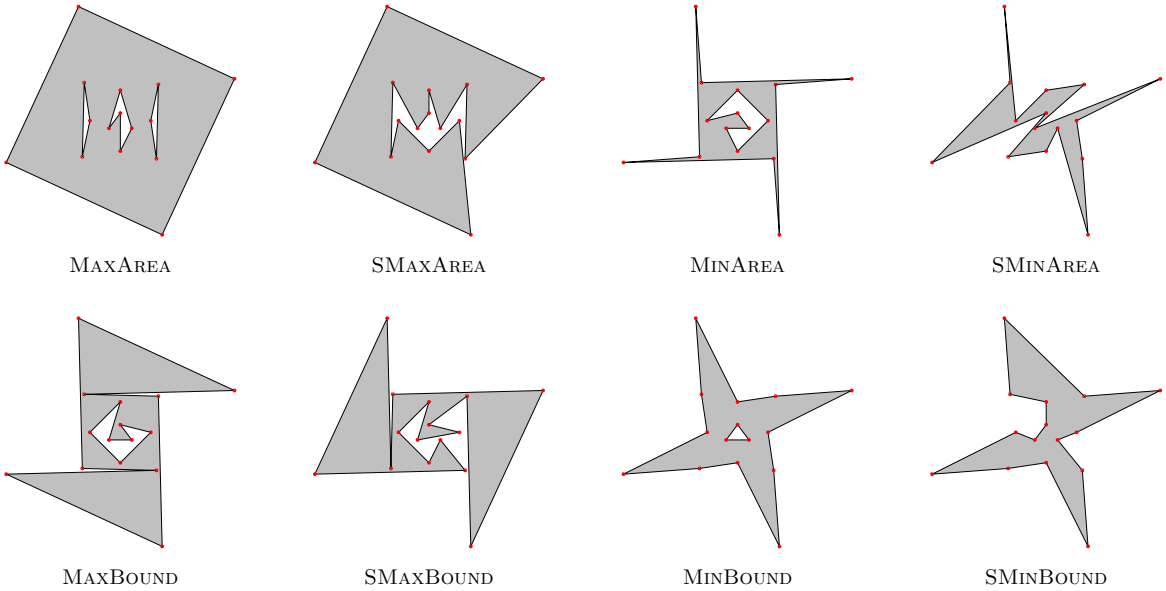

Figure 2: Different optimal solutions on the same input set V for all eight problems.

1.2 Computing Optimal Polygons

As discussed, all problem variants are either known or conjectured to be NP-hard. In the theory community, this is typically used as an incentive for studying approximation algorithms. While several of our problem variants have been studied in this regard (e. g., Euclidean and Max TSP allow polynomial-time approximation schemes, while others allow constant-factor approximations), this is not the goal of this paper. Instead, we want to demonstrate that it is still possible to compute provably optimal solutions for instances of these NP-hard problems, by combining methods of combinatorial optimization (most notably, integer linear programming) with geometric insights. As it turns out, this yields some relatively generic approaches that are suitable for all our problems, and thus possibly for related ones as well. As the ability to check the true optimal values for some instances can provide better comparisons, our approach may also be beneficial for the study of approximation algorithms and heuristics.

2 IP Formulation

Our approach is an IP-based algorithm that solves a relaxation and then, on-demand, adds constraints in a separation phase. (For an introduction to linear and integer programming, see [10].)

2.1 Edge-Based IP for Polygonal Subdivisions

Let $E = L \cup R$ be the set of all oriented edges of the complete graph induced by V, where L and R are the edges directed to the left and right, respectively. By e_{ij} we denote the edge from v_i to v_j and by $z_{ij} \in \{0, 1\}$ its corresponding variable, by $X_R(e_{ij})$ the set of edges in R crossing e_{ij}. Depending on the considered problem, f_{ij} either denotes the Euclidean length of e_{ij} or the signed area of the triangle that is formed by e_{ij} and the origin, which is used as reference point; see Figure 1. Hence, the objective function is given by

$$\min/\max \sum_{i \neq j} f_{ij} z_{ij} + f_{ji} z_{ji} \tag{1}$$

This is subject to the following constraints:

$$\forall i : \sum_{j \neq i} \begin{matrix} z_{ij} = 1 \\ z_{ji} = 1 \end{matrix} \tag{2}$$

$$\forall i \neq j : z_{ij} + z_{ji} \leq 1 \tag{3}$$

$$\forall i < j \text{ and } k_{ij} = |X_R(e_{ij})| : \quad k_{ij} z_{ij} + k_{ij} z_{ji} + \sum_{e_{kl} \in X_R(e_{ij})} (z_{kl} + z_{lk}) \leq k_{ij} \tag{4}$$

$$z_{ij} \in \{0, 1\} \tag{5}$$

Because (2) fixes in- and out-degree of each vertex to one, (3) ensures that for each edge at most one direction is selected, while (4) prevents crossing edges, solving this IP results in an arrangement of non-intersecting oriented (and non-trivial) edge cycles. (1) – (5) generates disjoint, non-trivial, oriented cycles by merely inducing $O(n^2)$ constraints and variables.

However, a polygon is represented by one outer edge cycle, which is oriented counterclockwise and, in the case of general polygons, possibly some inner clockwise cycles that represent the holes. Hence, clockwise cycles incident to the outer face is invalid, as are clockwise cycles that enclose vertices, because they represent holes in holes. Both types are eliminated by the following additional family of constraints.

$$\forall \text{ invalid directed cycles } C : \sum_{\forall e \in C} z_e \leq |C| - 1. \tag{6}$$

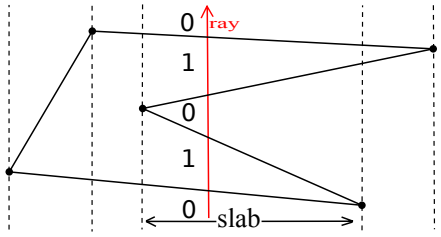

Figure 3: Five points inducing four inner slabs. Winding numbers are indicated along one ray.

However, because there is an exponential number of constraints of type (6), these are added on demand in a separation phase. IP (1) – (6) is named BasicIP.

2.2 A More Efficient Polygonalization

The main issue with BasicIP is that it requires an enormous number of separation steps as it produces many arrangements of boundary cycles that do not represent polygons, let alone a connected or even simple polygon. For example, in the case of MinArea, the IP first prefers boundary cycles with negative area, which then need to be excluded in additional separation steps. It is therefore desirable to enforce a set of boundary cycles that represents a valid polygonal arrangement right away.

Consider a valid polygonal arrangement and a vertical ray from bottom to top. This ray may cross several boundary edges. Let e be a crossed edge. If $e \in R$, the winding number is incremented by one, while it is decremented otherwise; see Figure 3.

Let S be the subdivision of $\mathbb{R}^2$ into $n+1$ vertical slabs induced by V. Shooting a vertical ray along each slab ensures that every face of any arrangement of edge cycles that are induced by V is visited. For slab $s \in S$, we denote by $[e_1^{R_s}, \ldots, e_{K_s}^{R_s}]$ the sequence of edges oriented to the right and crossing s in the order from bottom to top[1], analogously for edges in L. The following pairs of constraints ensure that the winding number alters between 0 and 1 for every slab.

$$\forall s \in S \text{ and } \forall k = 1, \ldots, K_s: \quad \sum_{i=1}^{k} z_{e_i^{R_s}} - z_{e_i^{L_s}} \begin{matrix} \geq 0 \\ \leq 1 \end{matrix} \qquad (7)$$

As k increases to K_s, the sum simulates a walk along the ray from bottom to top, thereby adding or subtracting one to the winding number if a corresponding edge is selected.

This yields an extra $O(n^3)$ constraints as $K_s \in O(n^2)$ and we call the IP (1) – (7) SlabsIP.

[1]It suffices to consider the order of edges with respect to a vertical ray within the slab, because intersecting edges change their order, so they cannot be selected at the same time.

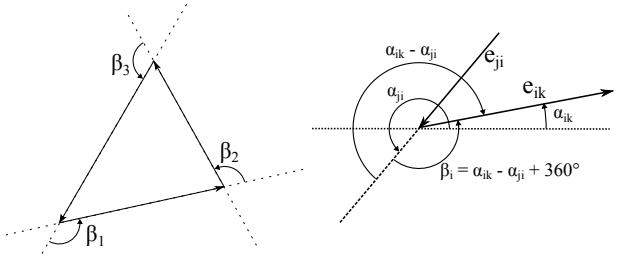

Figure 4: Left: Calculation of Boundary Number. Right: Calculation of enclosing angles.

2.3 Boundary Index

SlabsIP produces valid polygonal arrangements. However, especially when solving SMinArea, many solutions consisting of small distinct polygons have to be eliminated in separation steps. Therefore, it is desirable to add constraints that encode the sum B of positive and negative boundary cycles. In the case of simple polygons, we have $B = 1$, while for general polygons $B \leq 1$ holds.

First observe that for each boundary cycle, we can sum up the angles β_i that are enclosed by two edges at vertex i (see Figure 4). For each counterclockwise cycle, these angles add up to $+360°$, and to $-360°$ for clockwise ones. Hence, summing up at every vertex and dividing by 360° yields B.

Let α_{ij} denote the angle between e_{ij} and the x-axis, in counterclockwise orientation. Now the angle between e_{ij} and e_{jk} at v_j is $\alpha_{jk} - \alpha_{ij}$. This is correct modulo 360°, and we obtain

$$\forall i: \sum_{j \neq i} \alpha_{ij} z_{ij} - \sum_{j \neq i} \alpha_{ji} z_{ji} + 360 y_i \begin{matrix} \leq +180 \\ \geq -180 \end{matrix} \qquad (8)$$

$$y_i \in \{-1, 0, +1\}. \qquad (9)$$

In order to obtain B, we sum up over all angles in (8):

$$B = \frac{1}{360} \sum_{i=1}^{n} \left(\sum_{j \neq i} \alpha_{ij} z_{ij} - \sum_{j \neq i} \alpha_{ji} z_{ji} + 360 y_i \right)$$

However, because we already ensure closed cycles, the α_{ij} cancel out and for general polygonswe only add the following constraint[2]:

$$\sum_{i=1}^{n} y_i \leq 1 \qquad (10)$$

This adds only $O(n)$ constraints and variables. However, note that this does not completely avoid separations steps. For instance, an arrangement of two polygons, one of which has one hole, has boundary index one. Such a solution must still be cut off in a separation step. IP (1) – (10) is named BIndexIP.

[2]For simple polygons, we use $\sum_{i=1}^{n} y_i = 1$

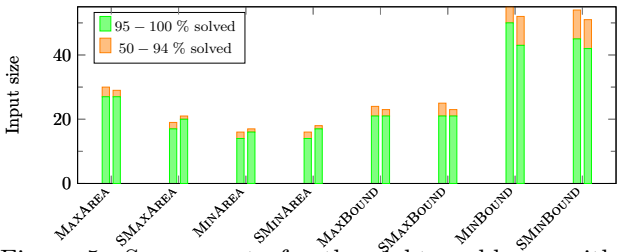

Figure 5: Success rate for the eight problems, with 30 instances for each input size (y-axis). Left bars: SlabsIP. Right bars: BIndexIP.

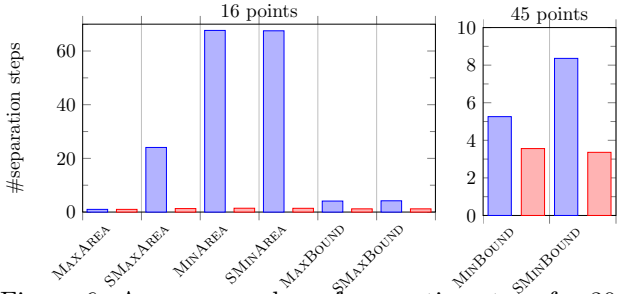

Figure 6: Average number of separation steps for 30 instances of the given input size. All instances where solved within the time limit. Blue: SlabsIP. Red: BIndexIP.

3 Experiments

Our implementation uses CPLEX to represent and solve the presented IPs. The geometric part is based on the CGAL Arrangements package [11]. CGAL represents planar subdivision by a doubly connected edge list (DCEL), which is ideal for detecting invalid boundary cycles.

All experiments were run on an *Intel Core i7-4770* CPU clocked at 3.40 GHz with 16 GB of RAM. For each point size we considered 30 randomly generated instances, with a time limit of 30 minutes. We do not show benchmark results for BasicIP, as it performed much worse than SlabsIP and BIndexIP.

We observe that both generic IPs perform much better for the minimum boundary problems (MinBound and SMinBound, i.e., the TSP), for which we are able to solve all instances up to about 45 input points; see Figure 5. This is still much worse than the performance of custom-made approaches for the TSP, but it illustrates the greater practical difficulty of the other problems. For all problems that optimized boundary BIndexIP, performed worse than SlabsIP, as BIndexIP was not able to significantly reduce the number of separation steps. In the case of MaxBound and SMaxBound, already SlabsIP requires hardly any separation steps; see Figure 6.

In the case of problems optimizing the area, we can observe that BIndexIP indeed improves the performance significantly. The only exception is MaxArea, for which already SlabsIP usually does not require any separation step. The reason is that the optimal result is essentially the convex hull of the point set with some small inner triangles removed, which is usually found in the first round. However, for SMaxArea as well as MinArea and SMinArea we can observe a significant advantage for BIndexIP, as it is essentially able to skip the separation step.

Why are minimum boundary problems practically easier to solve? For these problem variants, the intersection constraints (4) are already implied by triangle inequality, and as such only give a slight overhead; this is not the case for the other problem variants. This can be further exploited; for problems optimizing the boundary, a more specialized IP formulation can be based on undirected edges, requiring only half the number of variables as in our generic approach.

References

[1] T. Auer and M. Held. Rpg: Heuristics for the generation of random polygons. In *8th Canadian Conference on Computational Geometry*, pages 38–44, 1996.

[2] A. I. Barvinok, S. P. Fekete, D. S. Johnson, A. Tamir, G. J. Woeginger, and R. Wodroofe. The geometric maximum Traveling Salesman Problem. *J. ACM*, 50:641–664, 2003.

[3] E. D. Demaine, J. S. B. Mitchell, and J. O'Rourke. The open problems project, 2001. `http://cs.smith.edu/~orourke/TOPP/`.

[4] A. Dumitrescu and C. D. Tóth. Long non-crossing configurations in the plane. pages 727–752, 2010.

[5] S. P. Fekete. Simplicity and hardness of the maximum Traveling Salesman Problem under geometric distances. In *Proc. 10th ACM-SIAM Sympos. Discrete Algorithms*, pages 337–345, 1999.

[6] S. P. Fekete. On simple polygonalizations with optimal area. *DCG*, 23(1):73–110, 2000.

[7] S. P. Fekete and W. R. Pulleyblank. Area optimization of simple polygons. In *Proc. 9th Annu. ACM Sympos. Sympos. Geom.*, pages 173–182, 1993.

[8] J. O'Rourke and M. Virmani. Generating random polygons. Technical Report 11, Dept. Comput. Sci., Smith College, 1991.

[9] G. Reinelt. TSPlib – A Traveling Salesman Problem library. *ORSA J. on Computing*, 3(4):376–384, 1991.

[10] A. Schrijver. *Theory of Linear and Integer Programming*. John Wiley & Sons, Inc., New York, NY, USA, 1986.

[11] R. Wein, E. Berberich, E. Fogel, D. Halperin, M. Hemmer, O. Salzman, and B. Zukerman. 2D arrangements. In *CGAL User and Reference Manual*. CGAL Editorial Board, 4.3 edition, 2014.

[12] C. Zhu, G. Sundaram, J. Snoeyink, and J. S. Mitchell. Generating random polygons with given vertices. *Computational Geometry*, 6(5):277 – 290, 1996. Sixth Canadian Conference on Computational Geometry.