Computing Area-Optimal Simple Polygonalizations

Sándor P. Fekete[1], Andreas Haas[1], Phillip Keldenich[1], Michael Perk[1], and Arne Schmidt[1]

1 **Department of Computer Science, TU Braunschweig, Germany**
{s.fekete, a.haas, p.keldenich, m.perk, arne.schmidt}@tu-bs.de

Abstract

We consider methods for finding a simple polygon of minimum (Min-Area) or maximum (Max-Area) possible area for a given set of points in the plane. Both problems are known to be NP-hard; at the center of the recent CG Challenge, practical methods have received considerable attention. However, previous methods focused on heuristic methods, with no proof of optimality. We develop exact methods, based on a combination of geometry and integer programming. As a result, we are able to solve instances of up to $n = 25$ points to provable optimality. While this extends the range of solvable instances by a considerable amount, it also illustrates the practical difficulty of both problem variants.

1 Introduction

While the classic geometric Traveling Salesman Problem (TSP) is to find a (simple) polygon with a given set of vertices that has shortest perimeter, it is natural to look for a simple polygon with a given set of vertices that minimizes another basic geometric measure: the enclosed area. The problem Min-Area asks for a simple polygon with minimum enclosed area, while Max-Area demands one of maximum area; see Figure 1 for an illustration.

Both problem variants were shown to be NP-complete by Fekete [2,3,6], who also showed that no polynomial-time approximation scheme (PTAS) exists for Min-Area problem and gave a $\frac{1}{2}$-approximation algorithm for Max-Area.

1.1 Related Work

Most previous practical work has focused on finding heuristics for both problems. Taranilla et al. [11] proposed three different heuristics. Peethambaran [9,10] later proposed randomized and greedy algorithms on solving Min-Area as well as the d-dimensional variant of both Min-Area and Max-Area. Considerable recent attention and progress was triggered by the 2019 CG Challenge, which asked contestants to find good solutions for a wide spectrum of benchmark instances with up to 1,000,000 points; details will be described in a forthcoming special issue of the *Journal of Experimental Algorithms* [1].

Despite this focus, there has only been a limited amount of work on computing provably optimal solutions for instances of interesting size. The only notable exception is by Fekete et al. [4], who were able to solve all instances of Min-Area with up to $n = 14$ and some with up to $n = 16$ points, as well as all instances of Max-Area with up to $n = 17$ and some with up to $n = 19$ points. This illustrates the inherent practical difficulty, which differs considerably from the closely related TSP, for which even straightforward IP-based approaches can yield provably optimal solutions for instances with hundreds of points, and sophisticated methods can solve instances with tens of thousands of points.

36th European Workshop on Computational Geometry, Würzburg, Germany, March 16–18, 2020.
This is an extended abstract of a presentation given at EuroCG'20. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

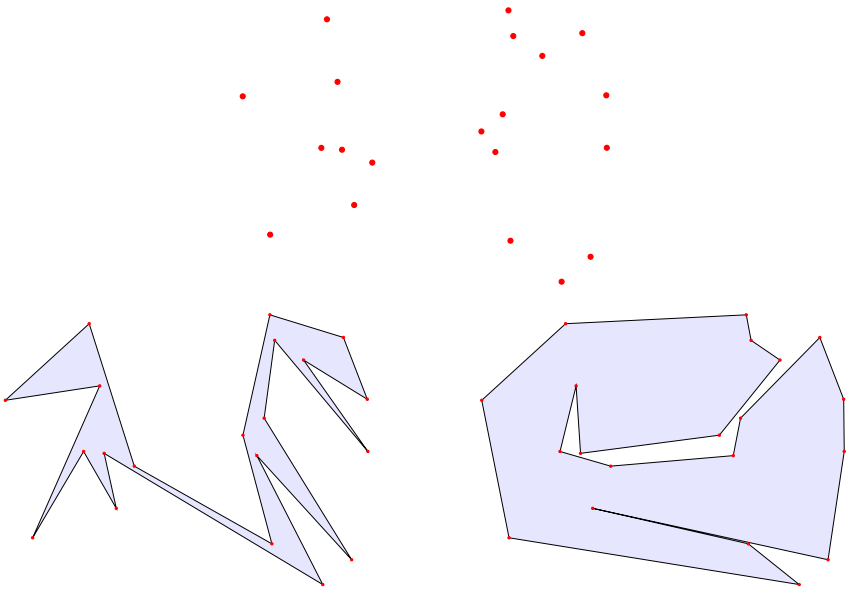


Figure 1 (Top) A set of 20 points. (Bottom Left) Min-Area solution. (Bottom Right) Max-Area solution.

1.2 Our Results

We present a systematic study of exact methods for Min-Area and Max-Area polygonizations. We show that a number of careful enhancements can help to extend the range of instances that can be solved to provable optimality, with different approaches working better for the two problem variants. On the other hand, our work shows that the problems appear to be practically harder than other geometric optimization problems such as the Euclidean TSP.

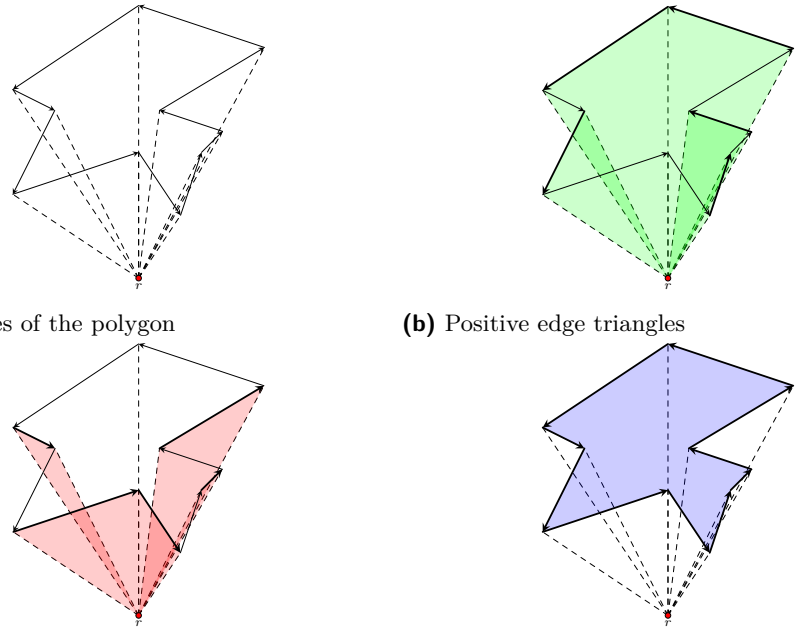
2 Tools

We considered two models based on integer programming: an *edge-based formulation* (described in Section 2.1) and a *triangle-based formulation* (described in Section 2.2). In addition, we developed a number of further refinements and improvements (described in Section 2.3).

2.1 Edge-Based Formulation

The first formulation is based on considering *directed* edges of the polygon boundary. As shown in Figure 2, the area $A_{\mathcal{P}}$ of a polygon $\mathcal{P}$ can be computed by adding the (signed) triangle areas f_e that are formed by edges e and an arbitrary, fixed reference point r.

This gives rise to an integer program in which the choice of half-edges $e = (i, j)$ is modeled by 0-1 variables $z_e = z_{ij}$. In contrast to Euclidean TSP, intersections between edges must be prevented with *intersection constraints* (5). The *slab inequalities* (6) ensure that the polygon is oriented in a counterclockwise manner and thus the area calculation yields the correct result. A *slab* D is a vertical strip bounded by the x-coordinates of two consecutive points in the order of x-coordinates of points. The edges of slab D get ordered by the y-coordinate at the intersection with the (centered) halving line between the points. Now the bottommost chosen edge has to be oriented from left to right and the topmost one from right to left,



(a) Triangles of the polygon

(b) Positive edge triangles

(c) Negative edge triangles

(d) Calculated difference between (b) and (c)

Figure 2 Area computation of a polygon using a reference point r

while chosen edges inbetween have to alternate in their direction. Furthermore, we introduce *subtour constraints* (7) that enforce a polygonization that visits all points in S.

$$\{\min, \max\} \sum_{e^+ \in E^r} z_{e^+} \cdot f_e - \sum_{e^- \in E^r} z_{e^-} \cdot f_e \tag{1}$$

$$\forall s_i \in S: \quad \sum_{(j,i) \in \delta^+(s_i)} z_{ji} = 1 \tag{2}$$

$$\forall s_i \in S: \quad \sum_{(i,j) \in \delta^-(s_i)} z_{ij} = 1 \tag{3}$$

$$\forall e = \{i,j\} \in E: \quad z_{ij} + z_{ji} \leq 1 \tag{4}$$

$$\forall \text{ intersecting } \{i,j\}, \{k,l\} \in E: \quad z_{ij} + z_{ji} + z_{kl} + z_{lk} \leq 1 \tag{5}$$

$$(\forall \text{ slabs } D)(\forall m = 1, \ldots, |D|): \quad \sum_{i=1}^{m} z_{e^{lr}_{i_D}} - z_{e^{rl}_{i_D}} \tag{6}$$

$$\forall D \subsetneq S, D \neq \emptyset: \quad \begin{matrix} \sum_{(k,l) \in \delta^-(D)} z_{kl} \geq 1 \\ \sum_{(k,l) \in \delta^+(D)} z_{kl} \geq 1 \end{matrix} \tag{7}$$

$$\forall e = \{i,j\} \in E: \quad z_{ij}, z_{ji} \in \{0,1\} \tag{8}$$

As there are $\Theta(n^2)$ possible edges, the number of intersection constraints may be as big as $\Theta(n^4)$. Moreover, the number of subtour constraints (7) may be exponential, so they are only added when necessary in an incremental fashion.

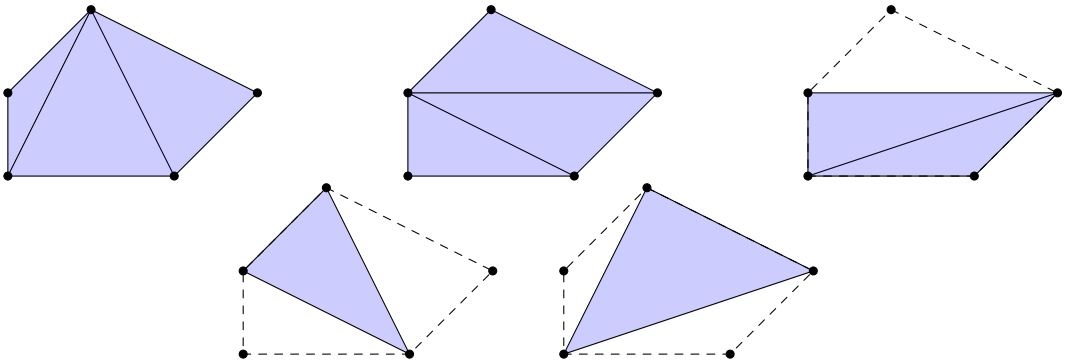


Figure 3 A set of five points and its ten empty triangles.

2.2 Triangle-Based Formulation

An alternative is the triangle-based formulation, which considers the set $T(P)$ of possibly $\binom{n}{3}$ many empty triangles of a point set P; see Figure 3 for an illustration. Making use of the fact that a simple polygon with n vertices consists of $(n-2)$ empty triangles with non-intersection interiors, we get the following IP formulation, in which the presence of an empty triangle $\triangle$ is described by a 0-1 variable $x_\triangle$.

The objective function (9) is the sum over the chosen triangles areas. *Triangle constraint* (10) ensures that we choose exactly $n-2$ triangles, which is the number of triangles in a triangulation of a simple polygon. Furthermore, *point constraints* (11) guarantee that a solution has at least one adjacent triangle at each point $s_i \in S$. Moreover, *intersection constraints* (12) ensure that we only select triangles with disjoint interiors. Finally, the *subtour constraints* (13) ensure that the set of selected triangles forms a simple polygon.

$$\{\min, \max\} \sum_{\triangle \in T} f_\triangle \cdot x_\triangle \tag{9}$$

$$\sum_{\triangle \in T} x_\triangle = n - 2 \tag{10}$$

$$\forall s_i \in S: \quad \sum_{\triangle \in \delta(s_i)} x_\triangle \geq 1 \tag{11}$$

$$\forall \text{intersecting} \quad \triangle_i, \triangle_j \in T: \quad x_{\triangle_i} + x_{\triangle_j} \leq 1 \tag{12}$$

$$\forall D \subsetneq T, D \neq \emptyset, |D| \leq n-3: \quad \sum_{\triangle \in D} x_\triangle \leq \sum_{\triangle \in \delta(D)} x_\triangle + |D| - 1 \tag{13}$$

$$\forall \triangle \in T: \quad x_\triangle \in \{0, 1\} \tag{14}$$

As there are $\Theta(n^3)$ possible empty triangles, the number of intersection constraints may be as big as $\Theta(n^6)$. Again, the number of subtour constraints (13) may be exponential, so they are only added when necessary in an incremental fashion.

2.3 Enhancing the Integer Programs

Given the considerable size of the described IP formulations, we developed a number of enhancements to improve efficiency. For points on the **convex hull**, only a reduced number of neighbors need to be considered. Employing good **initial solutions** improves the performance in branch-and-bound searching; we used a number of greedy heuristics, as well as the $\frac{1}{2}$-approximation of Fekete. The large number of corresponding inequalities made it particularly important to deal with **intersections** in an efficient manner: we condensed the constraints for cliques of intersecting objects into single inequalities, and introduced special *halfspace inequalities* for the triangle-based approach. Further increases in efficiency were obtained by careful choices of how to **branch on variables** and careful maintenance of **subtour constraints**.

3 Experiments

Based on the described approaches, we ran experiments on some machines with some specifications and parameters. We used CPLEX on an Intel(R) Core(TM) i7-6700K CPU 4.00GHz with four cores and 8 threads utilizing an L3 Cache with 8MB. The solver was able to use a maximum amount of 64GB RAM.

3.1 Edge-Based Solvers

EdgeV1 is a basic integer program of the edge-based approach. It adds all intersection constraints before starting the solving process and adds subtour constraints in every integer solution. This integer program is an improvement to the edge-based MinArea integer program presented by Papenberg et al. [4,8]. In the former approach cycle based subtour constraints were added after an optimal solution has been found. This resulted in poor computing times even for small point sets. EdgeV2 extends the previous version by adding intersection constraints at interim solutions. Moreover, this version includes a branching extension where branching on a variable z_e results in intersecting edges getting branched to zero. We also utilize properties of the convex hull to exclude certain variables, i.e., edges that connect two non-adjacent points on the convex hull, from the computation. EdgeV2 makes use of this concept by setting these variables to zero. Fekete et al. [4] introduced the concept of a boundary index. Their results indicate small improvements in computation time when adding the constraints. EdgeBIV2 extends the previous version by adding boundary index constraints. The upcoming sections will show that the boundary index constraints will increase the computation time of our integer program. Because of this, we removed the concept in favor of version three. In EdgeV3 we additionally search for subtours in fractional interim solutions and add slab constraints during the solving process. Furthermore, we pass a start solution to the solver which was generated by an abstraction of the Greedy Min-Area heuristic of Taranilla et al. [11].

3.2 Triangle-Based Solvers

TriangulationV1 is the first version of the triangle-based approach. Compared to the basic triangulation approach of Papenberg [8], we have fewer variables and different subtour constraints (13). Similar to the edge-based approaches, we pass a start solution obtained from Greedy Min-Area as an input to the solver. We added further *halfspace inequalities* as well as equalities for edges which connect non-adjacent vertices of the convex hull.

In TriangulationV1 we add subtour constraints and intersection constraints in every integer solution. TriangulationV2 extends the first version with so-called *subtour angle constraints*. These are added at every integer solution. We are able to reuse the connected components we need to compute along the way. This allows us to add constraints (13) without much additional computation time. TriangulationV3 makes use of additional results on ineffective subtour constraints. In addition to the constraints of TriangulationV2, we add point-based subtour constraints to every intermediate integer solution.

3.3 Results for Minimization

As Figure 4a shows, our various enhancements result in a considerable reduction of the computation times, compared to the approach by Papenberg et al. [4, 8]. Furthermore, it turned out that the triangle-based approach was able to compute optimal solutions for larger instances, as shown in Figure 5.

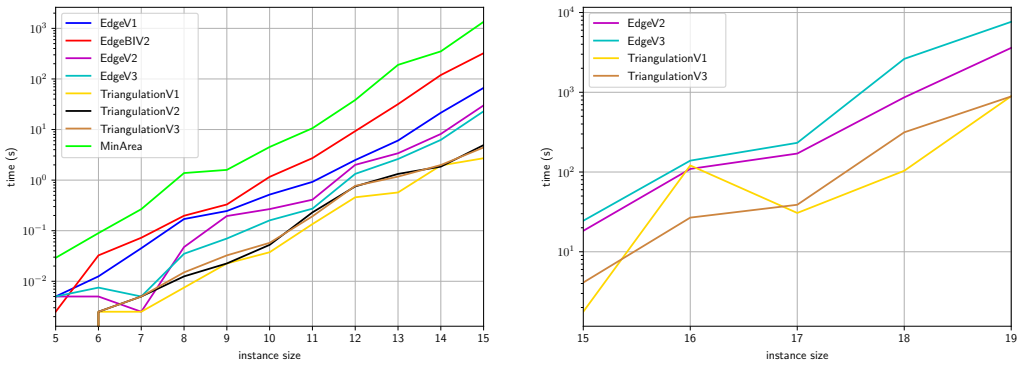


Figure 4 Computation times of Min-Area of the implemented solver versions. The computing time values are the average over 5 instances for each size. (a) Comparison with the MinArea version of Papenberg [8] for random instances of size $5 - 15$. MinArea operated on different instances than the rest. (b) Comparison of the best solver versions of both approaches for random instances of size $16 - 19$.

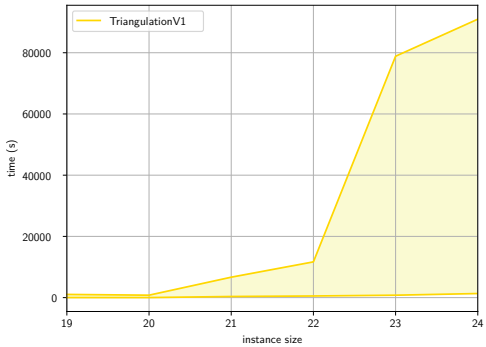


Figure 5 Computation time of TriangulationV1 of both approaches for random instances of size $19 - 24$. Shown are the minimum and maximum computation time needed to optimality.

3.4 Results for Maximization

For Max-Area, the edge-based approach turned out to be more useful: As Figure 6a shows, the runtime for the triangle-based solvers grew significantly faster. This seems to be mostly due to the fact that for the maximization version, intersections of "fat" intermediate subpolygons occur more frequently than for the "skinny" ones in the minimization version. Furthermore, we were able to expand the size of solvable instances in reasonable time to 23, as shown in Figure 7b.

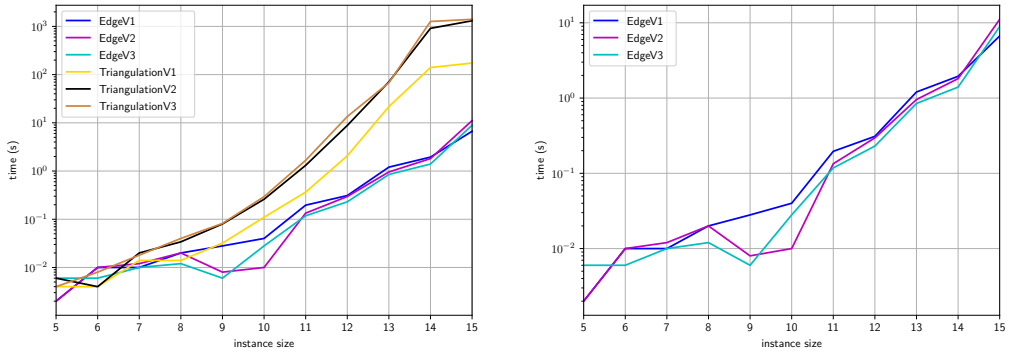


Figure 6 Computation times of all solver versions of Max-Area using both approaches for random instances of size $5 - 15$. The computing time values are the average over 10 instances for each size. (a) Comparison of solver version from both approaches. (b) Comparison of all edge-based solver versions

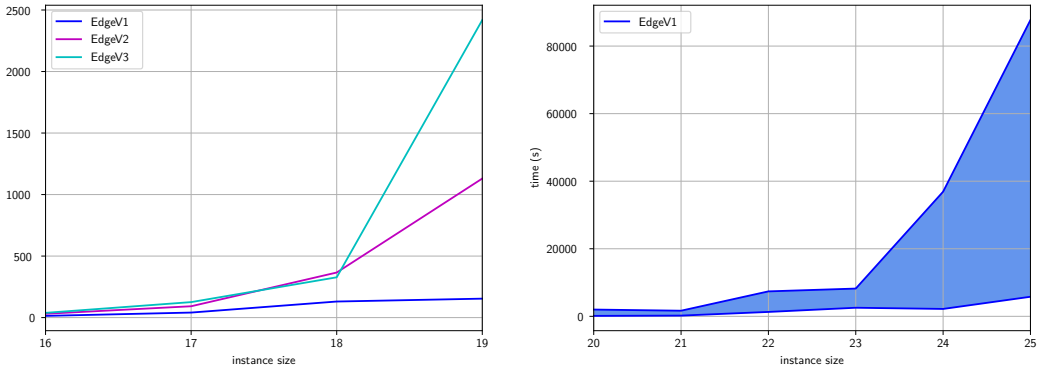


Figure 7 Computation time of Max-Area using different edge-based solver versions. (a) The computing time of all edge-based solver versions on random instances of size $16 - 19$. The values are the average over 5 instances for each size. (b) Range of computation time for EdgeV1 for random instances of size $20 - 25$.

4 Conclusions

While our work shows that with some amount of algorithm engineering, it is possible to extend the range of instances that can be solved to provable optimality, it also illustrates the practical difficulty of the problem. This reflects the limitations of such IP-based methods:

The edge-based approach makes use of an asymmetric variant of the TSP, which is known to be harder than the symmetric TSP, while the triangle-based approach suffers from its inherently large number of variables and constraints. Furthermore, the non-local nature of Min-Area and Max-Area polygons (which may contain edges that connect far-away points) makes it difficult to reduce the set of candidate edges.

As a result, Min-Area and Max-Area turn out to be prototypes of geometric optimization problems that are difficult both in theory and practice. This differs fundamentally from a problem such as Minimum Weight Triangulation, for which provably optimal solutions to huge point sets can be found [7], and practically difficult instances seem elusive [5].

References

1 Erik D. Demaine, Sándor P. Fekete, and Joseph S.B. Mitchell. The 2019 CG Challenge: Area-optimal polygonalizations. *Manuscript*, 2020.

2 Sándor P. Fekete. *Geometry and the Travelling Salesman Problem.* Ph.D. thesis, Department of Combinatorics and Optimization, University of Waterloo, Waterloo, ON, 1992.

3 Sándor P. Fekete. On simple polygonizations with optimal area. *Discrete & Computational Geometry*, 23(1):73–110, 2000.

4 Sándor P. Fekete, Stephan Friedrichs, Michael Hemmer, Melanie Papenberg, Arne Schmidt, and Julian Troegel. Area-and boundary-optimal polygonalization of planar point sets. In *European Workshop on Computational Geometry (EuroCG)*, pages 133–136, 2015.

5 Sándor P. Fekete, Andreas Haas, Dominik Krupke, Yannic Lieder, Eike Niehs, Michael Perk, Victoria Sack, and Christian Scheffer. Hard instances of the minimum-weight triangulation problem. Submitted to *European Workshop on Computational Geometry (EuroCG 2020).*

6 Sándor P. Fekete and William R. Pulleyblank. Area optimization of simple polygons. In *Symposium on Computational Geometry (SoCG)*, pages 173–182, 1993.

7 Andreas Haas. Solving large-scale minimum-weight triangulation instances to provable optimality. In *Symposium on Computational Geometry (SoCG)*, pages 44:1–44:14, 2018.

8 Melanie Papenberg. Exact Methods for area-optimal Polygons. Master's thesis, University of Technology Braunschweig, 2014.

9 Jiju Peethambaran, Amal Dev Parakkat, and Ramanathan Muthuganapathy. A randomized approach to volume constrained polyhedronization problem. *Journal of Computing and Information Science in Engineering*, 15(1):011009, 2015.

10 Jiju Peethambaran, Amal Dev Parakkat, and Ramanathan Muthuganapathy. An empirical study on randomized optimal area polygonization of planar point sets. *Journal of Experimental Algorithmics (JEA)*, 21:1–10, 2016.

11 Maria Teresa Taranilla, Edilma Olinda Gagliardi, and Gregorio Hernández Peñalver. Approaching minimum area polygonization. In *Congreso Argentino de Ciencias de la Computación (CACIC)*, pages 161–170, 2011.