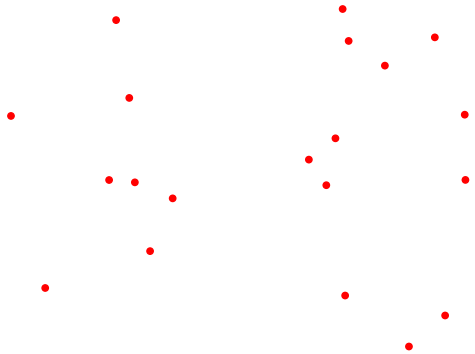
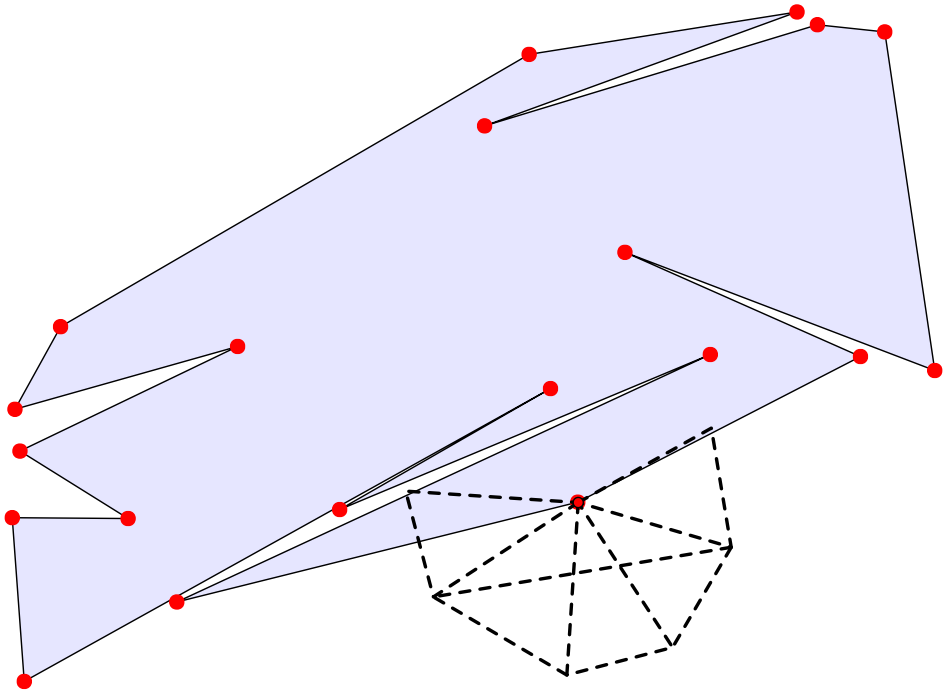
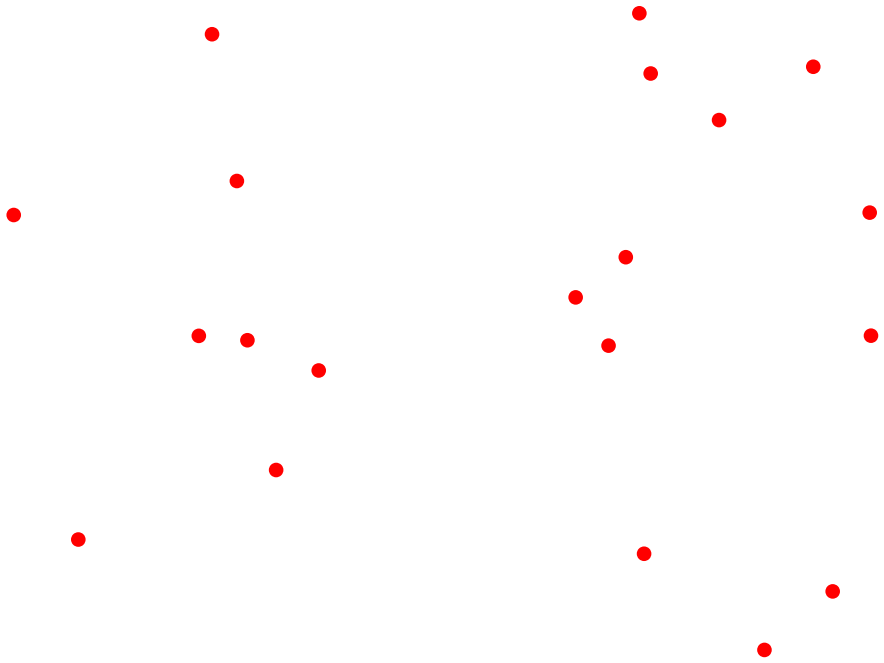




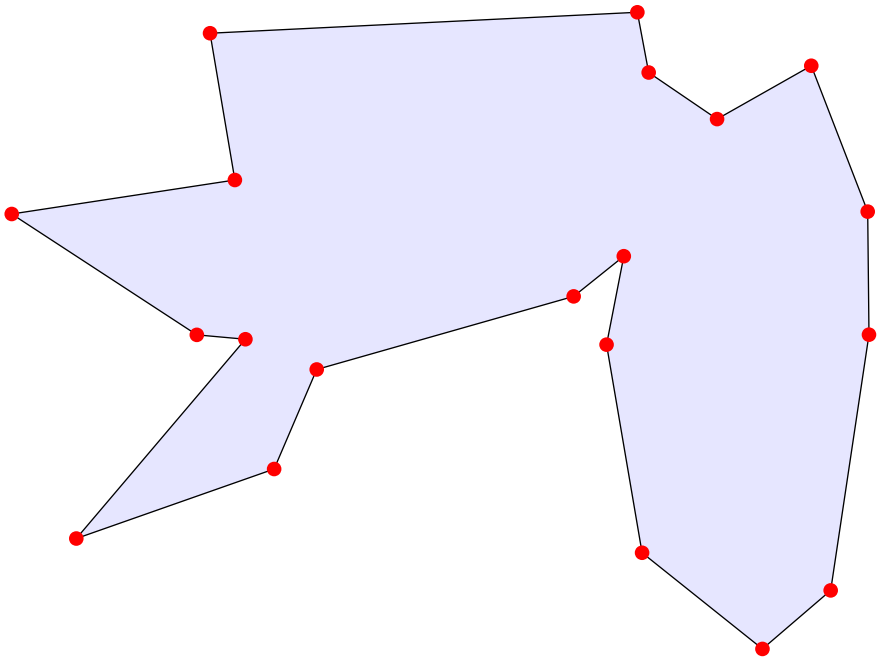
Computing Area-Optimal Simple Polygonalizations

Sándor P. Fekete, Andreas Haas, Phillip Keldenich, **Michael Perk**, and Arne Schmidt

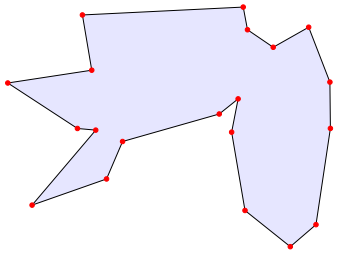
The Travelling Salesman Problem



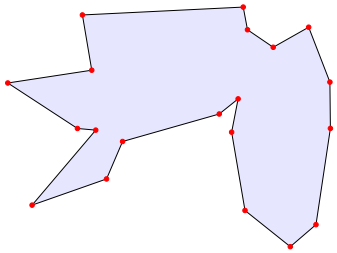
The Travelling Salesman Problem

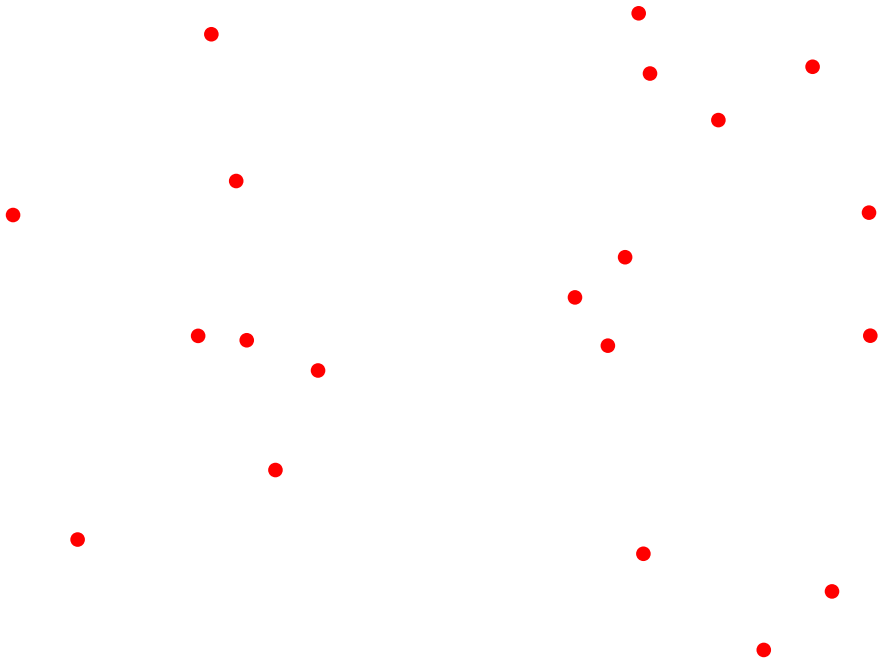


The Travelling Salesman Problem

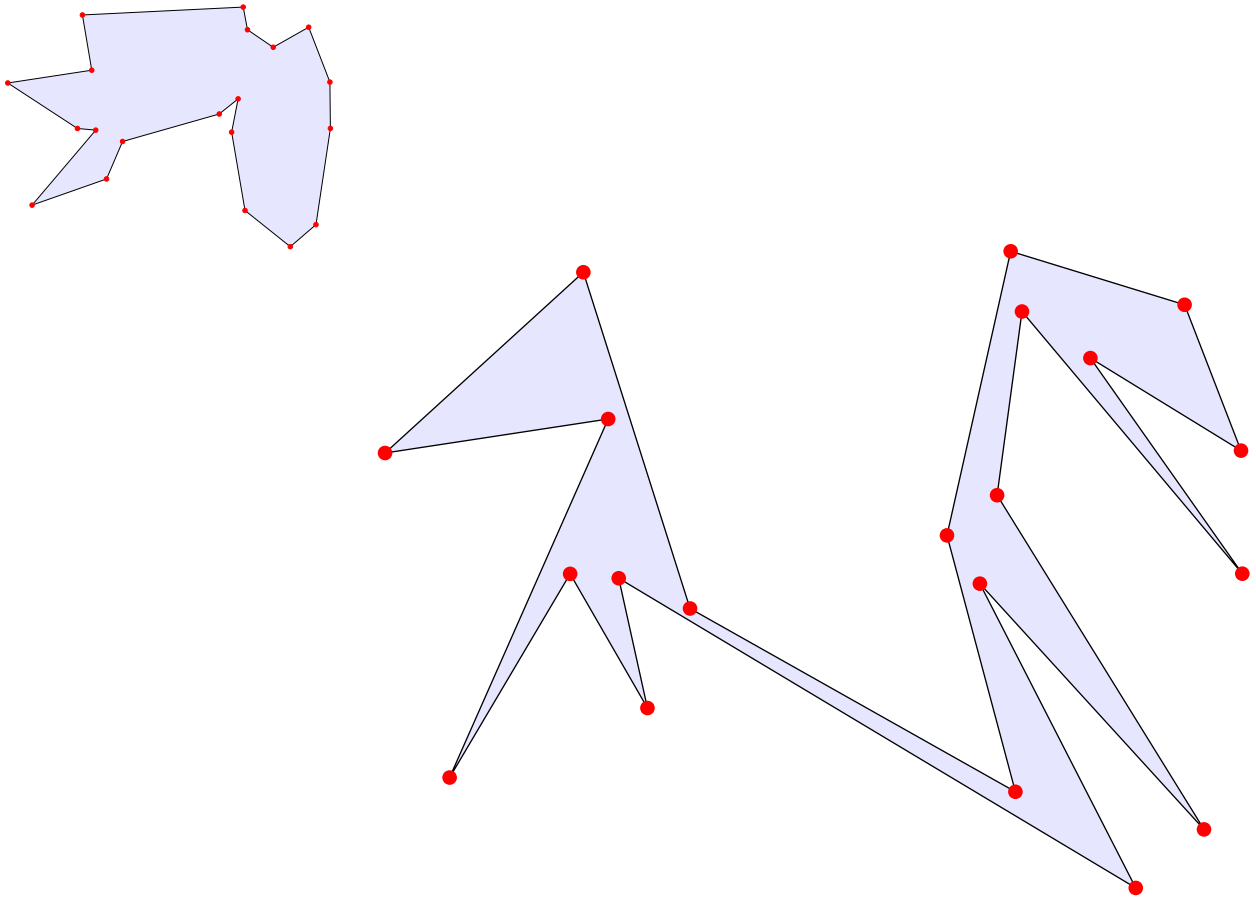


Minimum Area Polygonalization

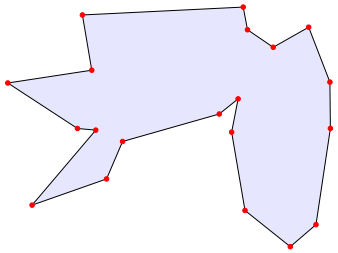


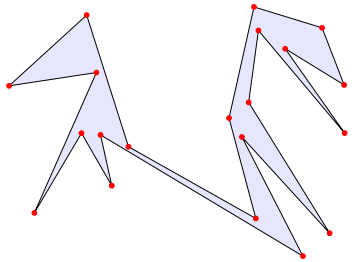


Minimum Area Polygonalization

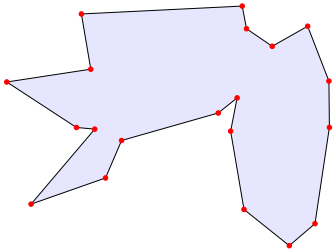


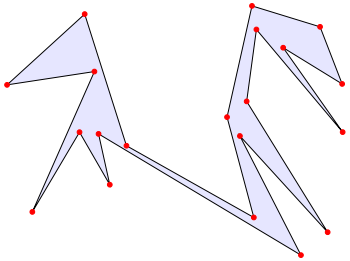
Minimum Area Polygonalization

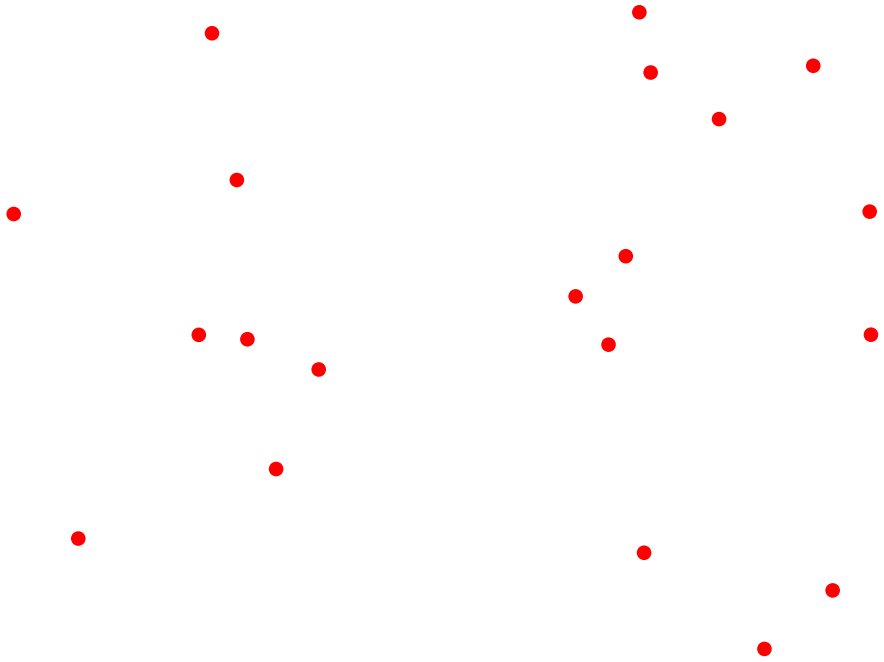




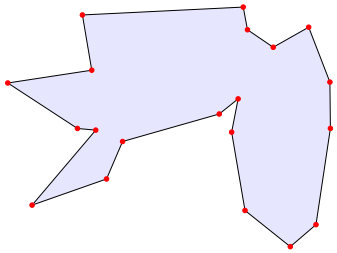
Maximum Area Polygonalization

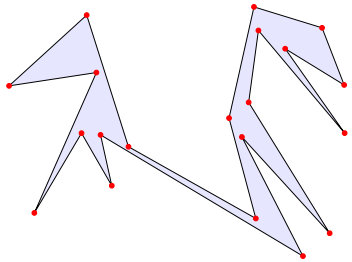


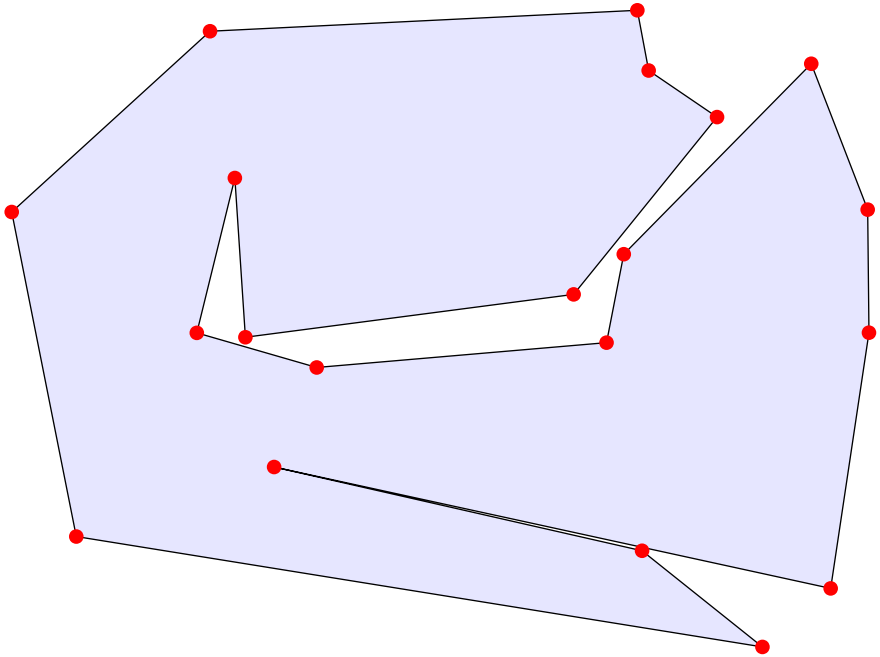




Maximum Area Polygonalization

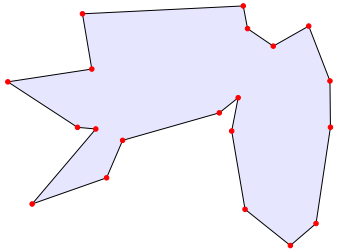




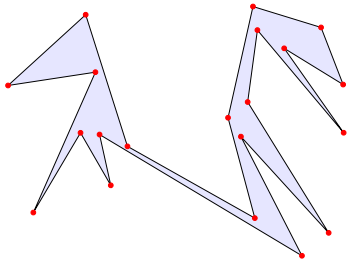


Maximum Area Polygonalization

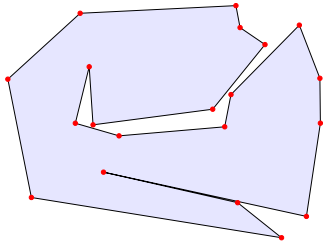
TSP



Min Area

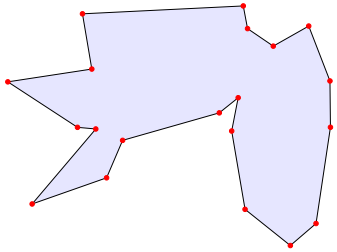


Max Area

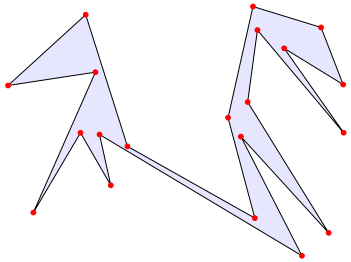


Maximum Area Polygonalization

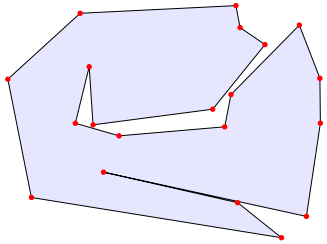
TSP



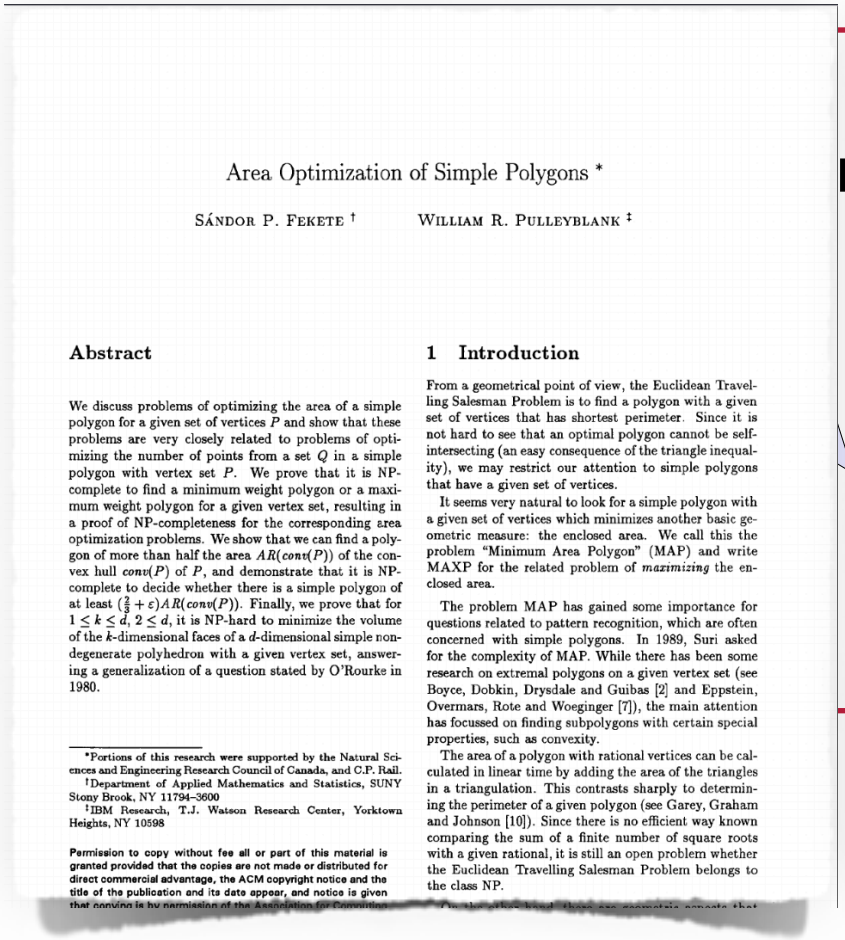
Min Area



Max Area



Maximum Area Polygonalization



Area Optimization of Simple Polygons *

SÁNDOR P. FEKETE † WILLIAM R. PULLEYBLANK ‡

Abstract

We discuss problems of optimizing the area of a simple polygon for a given set of vertices P and show that these problems are very closely related to problems of optimizing the number of points from a set Q in a simple polygon with vertex set P. We prove that it is NP-complete to find a minimum weight polygon or a maximum weight polygon for a given vertex set, resulting in a proof of NP-completeness for the corresponding area optimization problems. We show that we can find a polygon of more than half the area $AR(conv(P))$ of the convex hull $conv(P)$ of P, and demonstrate that it is NP-complete to decide whether there is a simple polygon of at least $(\frac{2}{3}+\varepsilon)AR(conv(P))$. Finally, we prove that for $1 \le k \le d$, $2 \le d$, it is NP-hard to minimize the volume of the k-dimensional faces of a d-dimensional simple nondegenerate polyhedron with a given vertex set, answering a generalization of a question stated by O'Rourke in 1980.

*Portions of this research were supported by the Natural Sciences and Engineering Research Council of Canada, and C.P. Rail.

†Department of Applied Mathematics and Statistics, SUNY Stony Brook, NY 11794-3600

‡IBM Research, T.J. Watson Research Center, Yorktown Heights, NY 10598

1 Introduction

From a geometrical point of view, the Euclidean Travelling Salesman Problem is to find a polygon with a given set of vertices that has shortest perimeter. Since it is not hard to see that an optimal polygon cannot be self-intersecting (an easy consequence of the triangle inequality), we may restrict our attention to simple polygons that have a given set of vertices.

It seems very natural to look for a simple polygon with a given set of vertices which minimizes another basic geometric measure: the enclosed area. We call this the problem "Minimum Area Polygon" (MAP) and write MAXP for the related problem of *maximizing* the enclosed area.

The problem MAP has gained some importance for questions related to pattern recognition, which are often concerned with simple polygons. In 1989, Suri asked for the complexity of MAP. While there has been some research on extremal polygons on a given vertex set (see Boyce, Dobkin, Drysdale and Guibas [2] and Eppstein, Overmars, Rote and Woeginger [7]), the main attention has focussed on finding subpolygons with certain special properties, such as convexity.

The area of a polygon with rational vertices can be calculated in linear time by adding the area of the triangles in a triangulation. This contrasts sharply to determining the perimeter of a given polygon (see Garey, Graham and Johnson [10]). Since there is no efficient way known comparing the sum of a finite number of square roots with a given rational, it is still an open problem whether the Euclidean Travelling Salesman Problem belongs to the class NP.

Discrete Comput Geom 23:73–110 (2000)
DOI: 10.1007/s004549910005

Discrete & Computational Geometry

On Simple Polygonalizations with Optimal Area*

S. P. Fekete

Department of Mathematics, TU Berlin,
Str. des 17. Juni 136,
D-10623 Berlin, Germany

Abstract. We discuss the problem of finding a simple polygonalization for a given set of vertices P that has optimal area. We show that these problems are very closely related to problems of optimizing the number of points from a set Q in a simple polygon with vertex set P and prove that it is NP-complete to find a minimum weight polygon or a maximum weight polygon for a given vertex set, resulting in a proof of NP-completeness for the corresponding area optimization problems. This answers a generalization of a question stated by Suri in 1989. Finally, we turn to higher dimensions, where we prove that, for $1 \le k \le d$, $2 \le d$, it is NP-hard to determine the smallest possible total volume of the k-dimensional faces of a d-dimensional simple nondegenerate polyhedron with a given vertex set, answering a generalization of a question stated by O'Rourke in 1980.

Maximum Area Polygonalization

Area Optimizati

SÁNDOR P. FEKETE †

Abstract

We discuss problems of optimizing the area of a si
polygon for a given set of vertices P and show that t
problems are very closely related to problems of
mizing the number of points from a set Q in a si
polygon with vertex set P. We prove that it is
complete to find a minimum weight polygon or a n
mum weight polygon for a given vertex set, resultin
a proof of NP-completeness for the corresponding
optimization problems. We show that we can find a p
gon of more than half the area $AR(conv(P))$ of the
vex hull $conv(P)$ of P, and demonstrate that it is
complete to decide whether there is a simple polyg
at least $(\frac{2}{3}+\varepsilon)AR(conv(P))$. Finally, we prove tha
$1 \le k \le d$, $2 \le d$, it is NP-hard to minimize the vol
of the k-dimensional faces of a d-dimensional simple
degenerate polyhedron with a given vertex set, ans
ing a generalization of a question stated by O'Rourk
1980.

*Portions of this research were supported by the Natura
ences and Engineering Research Council of Canada, and C.P.

†Department of Applied Mathematics and Statistics, S
Stony Brook, NY 11794-3600

‡IBM Research, T.J. Watson Research Center, Yorktown
Heights, NY 10598

ing the perimeter of a given polygon (see Garey, Graham and Johnson [10]). Since there is no efficient way known comparing the sum of a finite number of square roots with a given rational, it is still an open problem whether the Euclidean Travelling Salesman Problem belongs to the class NP.

EuroCG 2015, Ljubljana, Slovenia, March 16–18, 2015

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

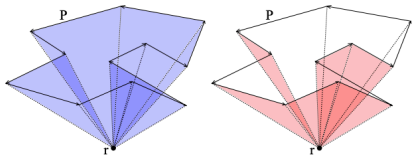


Figure 1: Calculation of the area of P: Using some reference point r, each edge forms an oriented triangle. Counterclockwise triangles contribute positive area (left), others are subtracted (right).

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-

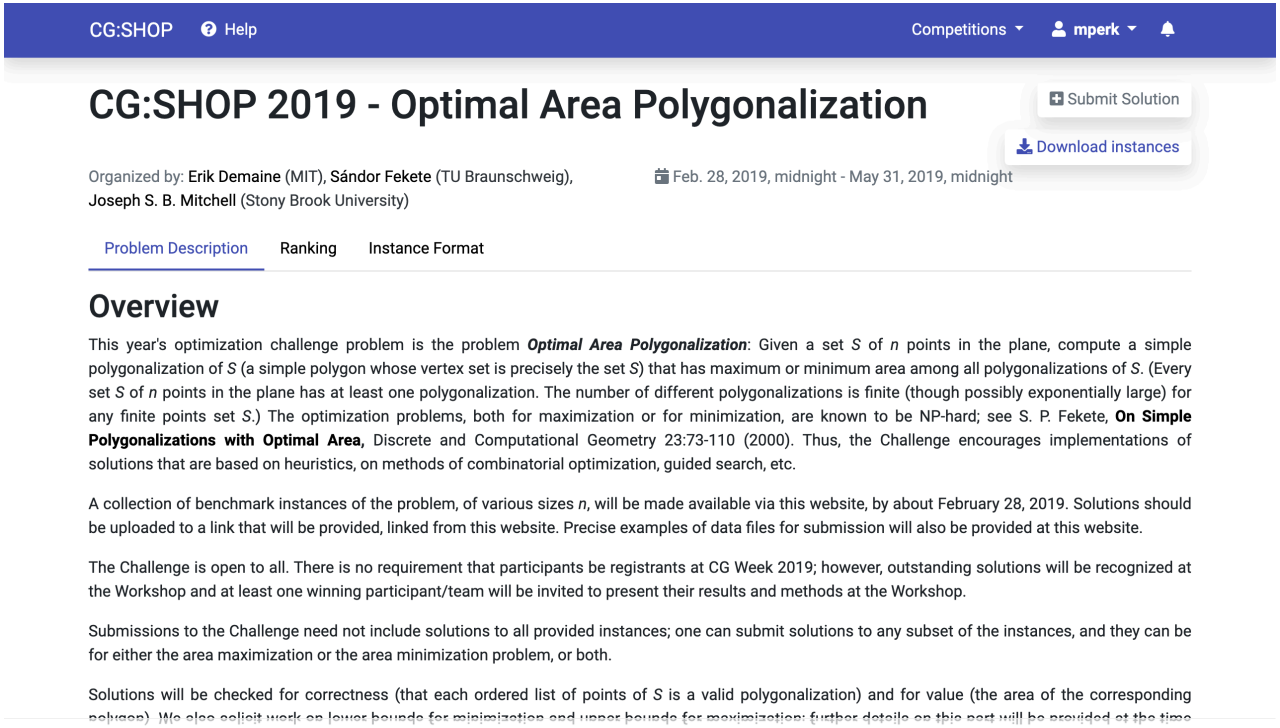
Discrete & Computational
Geometry
00 Springer-Verlag New York Inc.

rea*

gonalization for a given set of
ms are very closely related to
a simple polygon with vertex
eight polygon or a maximum
of NP-completeness for the
generalization of a question
ns, where we prove that, for
possible total volume of the
rate polyhedron with a given
vertex set, answering a generalization of a question stated by O'Rourke in 1980.

CG:SHOP 2019

- Computational Geometry: Solving Hard Optimization Problems



CG:SHOP Help Competitions mperk

CG:SHOP 2019 - Optimal Area Polygonalization

Submit Solution

Download instances

Organized by: Erik Demaine (MIT), Sándor Fekete (TU Braunschweig), Joseph S. B. Mitchell (Stony Brook University)

Feb. 28, 2019, midnight - May 31, 2019, midnight

Problem Description | Ranking | Instance Format

Overview

This year's optimization challenge problem is the problem ***Optimal Area Polygonalization***: Given a set *S* of *n* points in the plane, compute a simple polygonalization of *S* (a simple polygon whose vertex set is precisely the set *S*) that has maximum or minimum area among all polygonalizations of *S*. (Every set *S* of *n* points in the plane has at least one polygonalization. The number of different polygonalizations is finite (though possibly exponentially large) for any finite points set *S*.) The optimization problems, both for maximization or for minimization, are known to be NP-hard; see S. P. Fekete, **On Simple Polygonalizations with Optimal Area,** Discrete and Computational Geometry 23:73-110 (2000). Thus, the Challenge encourages implementations of solutions that are based on heuristics, on methods of combinatorial optimization, guided search, etc.

A collection of benchmark instances of the problem, of various sizes *n*, will be made available via this website, by about February 28, 2019. Solutions should be uploaded to a link that will be provided, linked from this website. Precise examples of data files for submission will also be provided at this website.

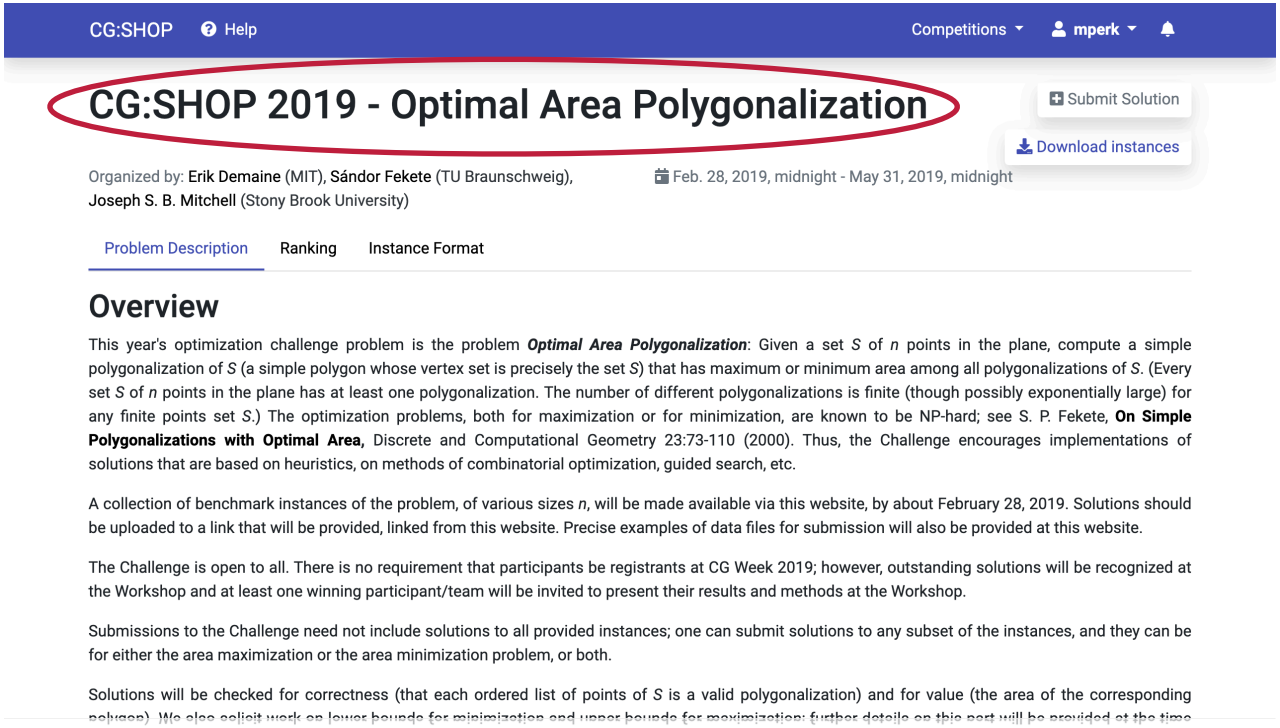
The Challenge is open to all. There is no requirement that participants be registrants at CG Week 2019; however, outstanding solutions will be recognized at the Workshop and at least one winning participant/team will be invited to present their results and methods at the Workshop.

Submissions to the Challenge need not include solutions to all provided instances; one can submit solutions to any subset of the instances, and they can be for either the area maximization or the area minimization problem, or both.

Solutions will be checked for correctness (that each ordered list of points of *S* is a valid polygonalization) and for value (the area of the corresponding polygon). We also solicit work on lower bounds for minimization and upper bounds for maximization; further details on this part will be provided at the time

CG:SHOP 2019

- Computational Geometry: Solving Hard Optimization Problems



CG:SHOP Help Competitions mperk

CG:SHOP 2019 - Optimal Area Polygonalization

Submit Solution

Download instances

Organized by: Erik Demaine (MIT), Sándor Fekete (TU Braunschweig), Joseph S. B. Mitchell (Stony Brook University)

Feb. 28, 2019, midnight - May 31, 2019, midnight

Problem Description | Ranking | Instance Format

Overview

This year's optimization challenge problem is the problem ***Optimal Area Polygonalization***: Given a set S of n points in the plane, compute a simple polygonalization of S (a simple polygon whose vertex set is precisely the set S) that has maximum or minimum area among all polygonalizations of S. (Every set S of n points in the plane has at least one polygonalization. The number of different polygonalizations is finite (though possibly exponentially large) for any finite points set S.) The optimization problems, both for maximization or for minimization, are known to be NP-hard; see S. P. Fekete, **On Simple Polygonalizations with Optimal Area,** Discrete and Computational Geometry 23:73-110 (2000). Thus, the Challenge encourages implementations of solutions that are based on heuristics, on methods of combinatorial optimization, guided search, etc.

A collection of benchmark instances of the problem, of various sizes n, will be made available via this website, by about February 28, 2019. Solutions should be uploaded to a link that will be provided, linked from this website. Precise examples of data files for submission will also be provided at this website.

The Challenge is open to all. There is no requirement that participants be registrants at CG Week 2019; however, outstanding solutions will be recognized at the Workshop and at least one winning participant/team will be invited to present their results and methods at the Workshop.

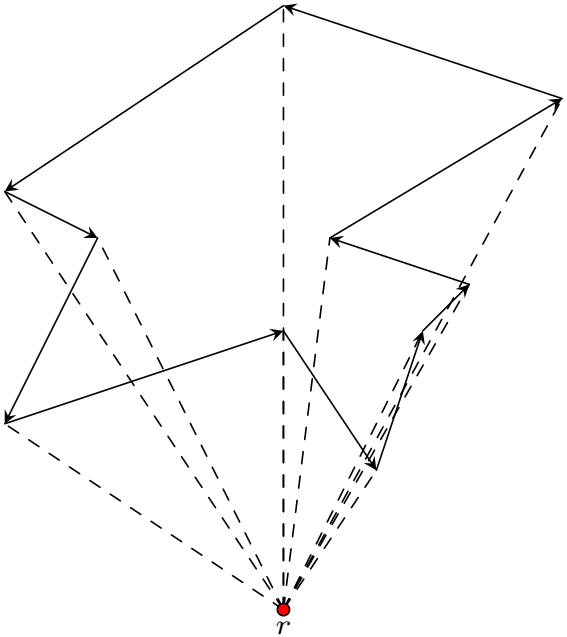
Submissions to the Challenge need not include solutions to all provided instances; one can submit solutions to any subset of the instances, and they can be for either the area maximization or the area minimization problem, or both.

Solutions will be checked for correctness (that each ordered list of points of S is a valid polygonalization) and for value (the area of the corresponding polygon). We also solicit work on lower bounds for minimization and upper bounds for maximization; further details on this part will be provided at the time

Integer Programming Approaches

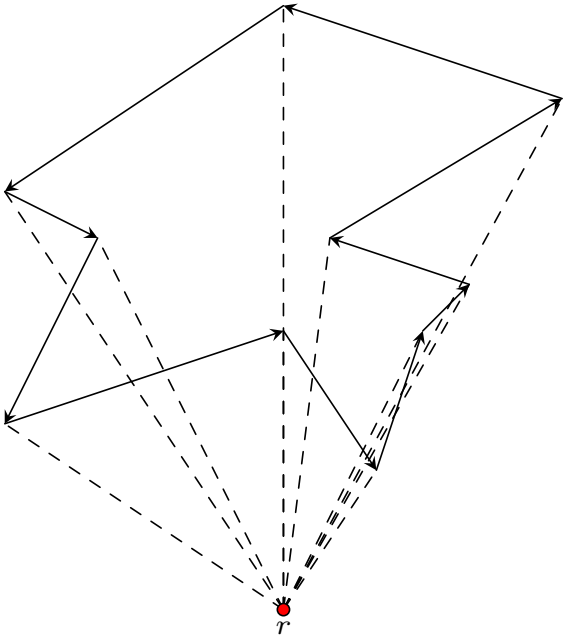
Integer Programming Approaches

edge-based

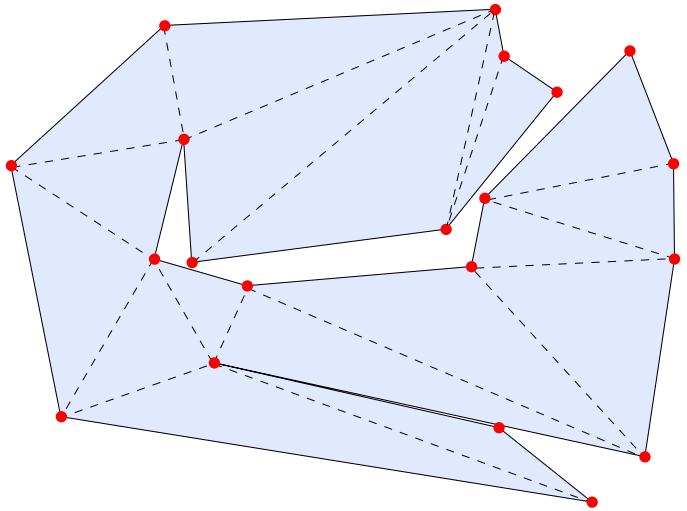


Integer Programming Approaches

edge-based

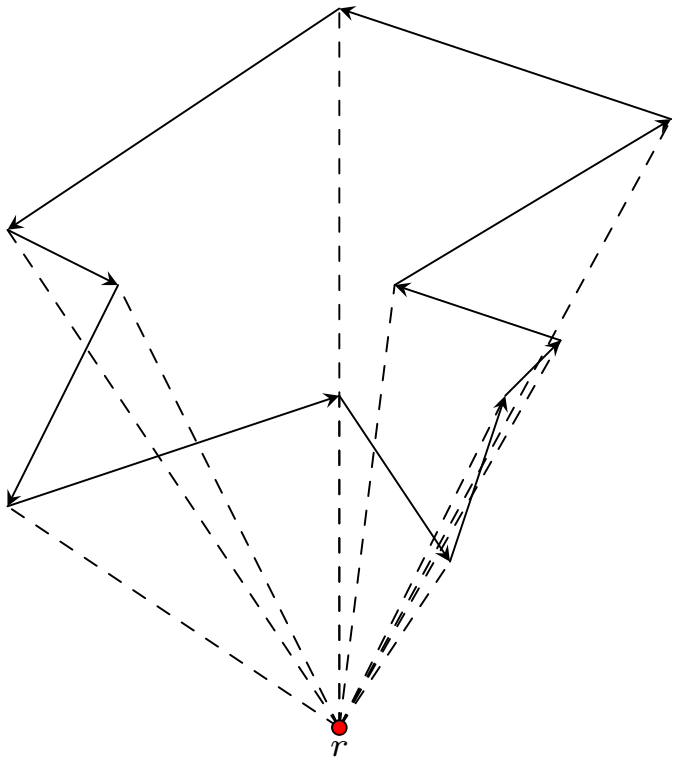


triangulation-based

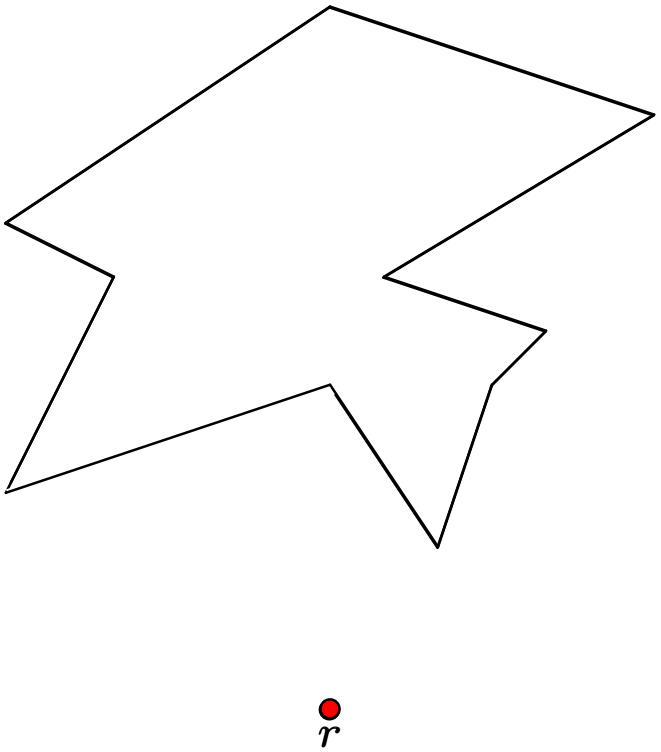


Integer Programming Approaches

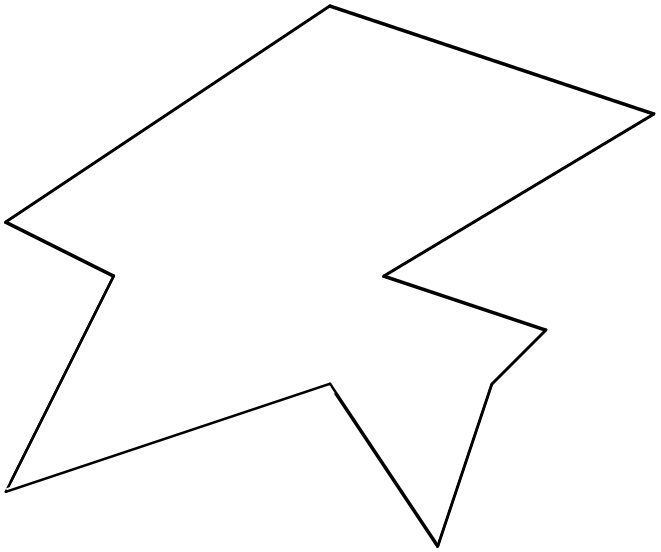
edge-based



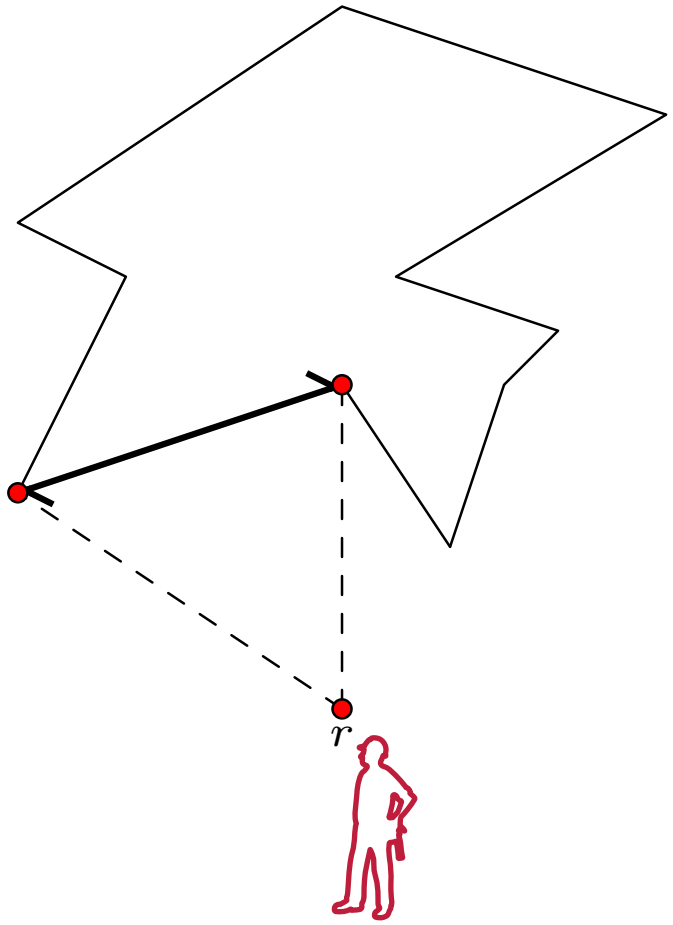
Edge-Based Area Calculation



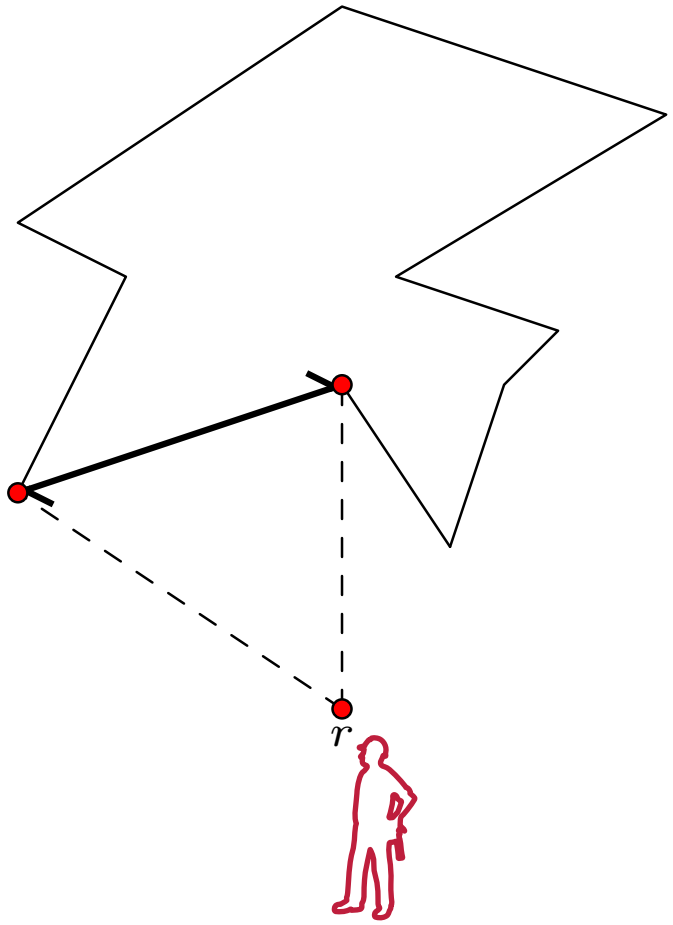
Edge-Based Area Calculation



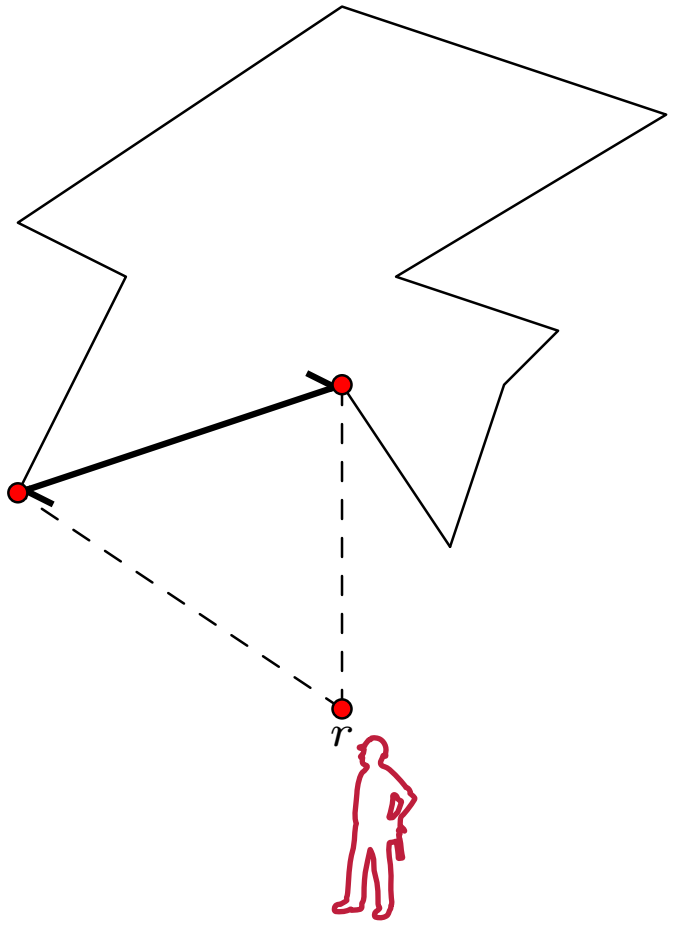
Edge-Based Area Calculation



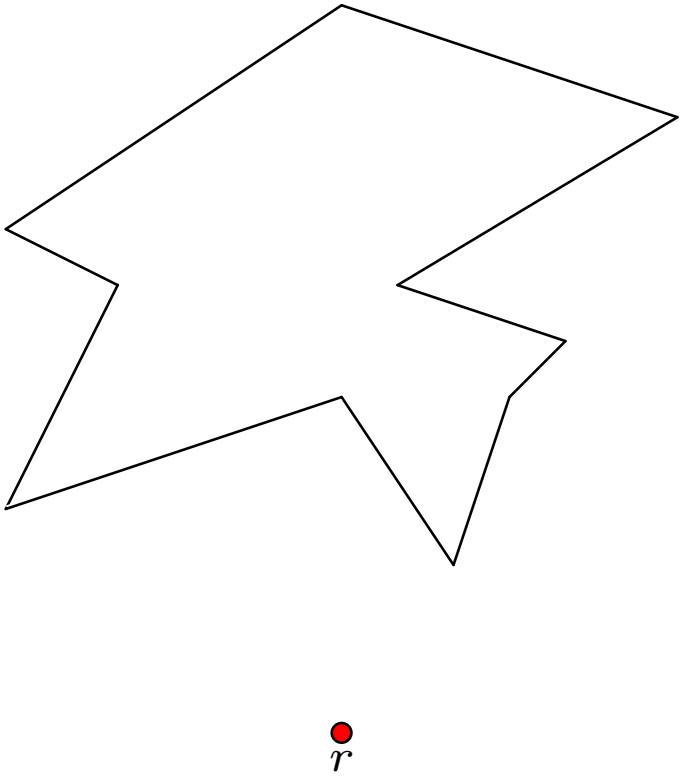
Edge-Based Area Calculation

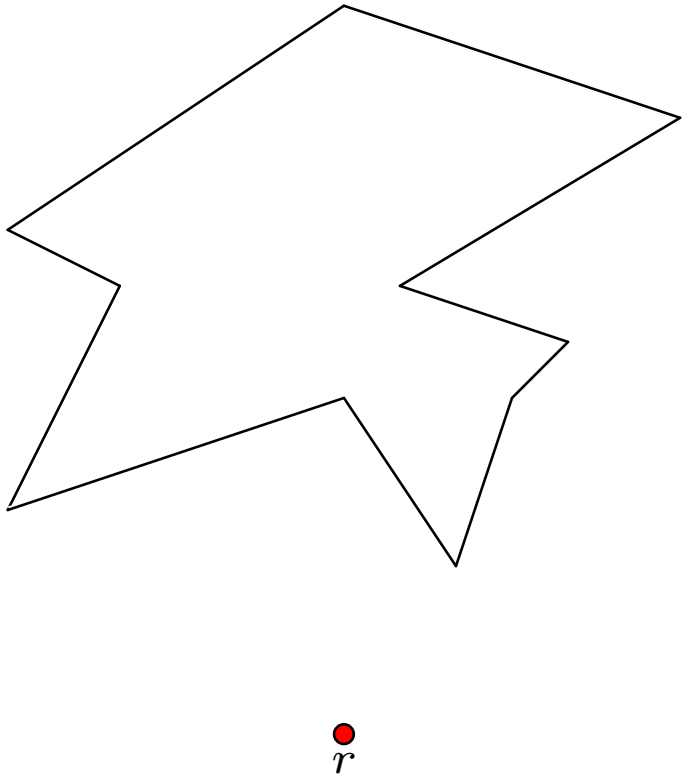


Edge-Based Area Calculation

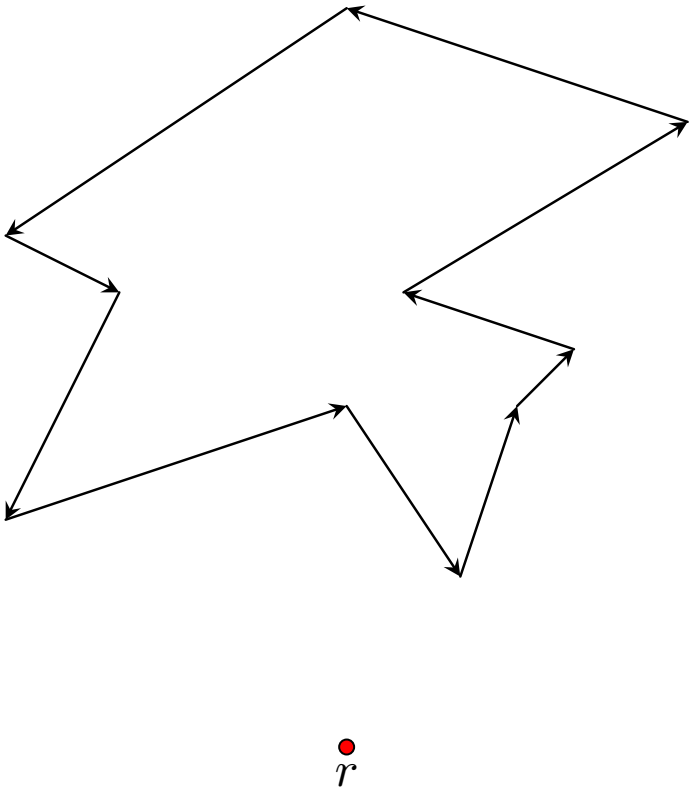


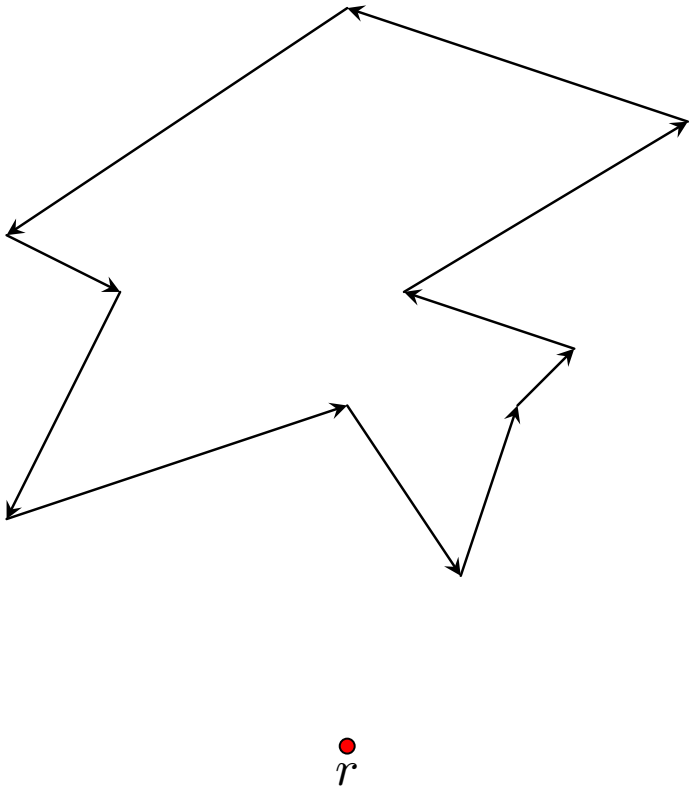
Edge-Based Area Calculation



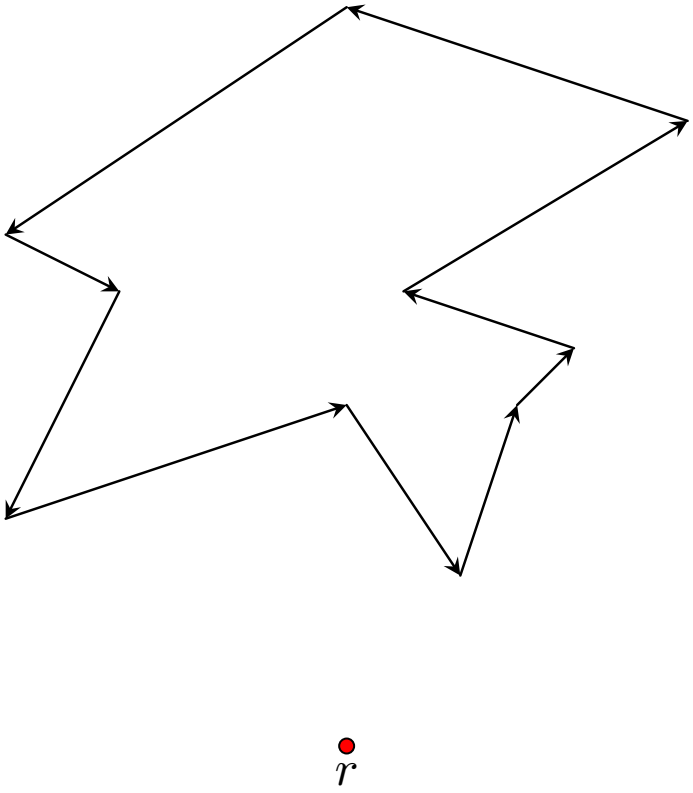


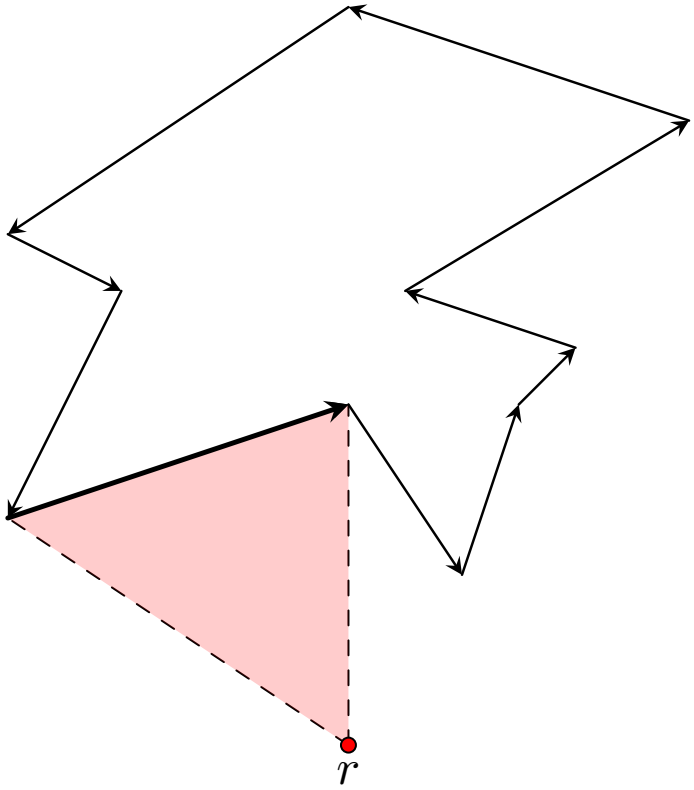
Edge-Based Area Calculation



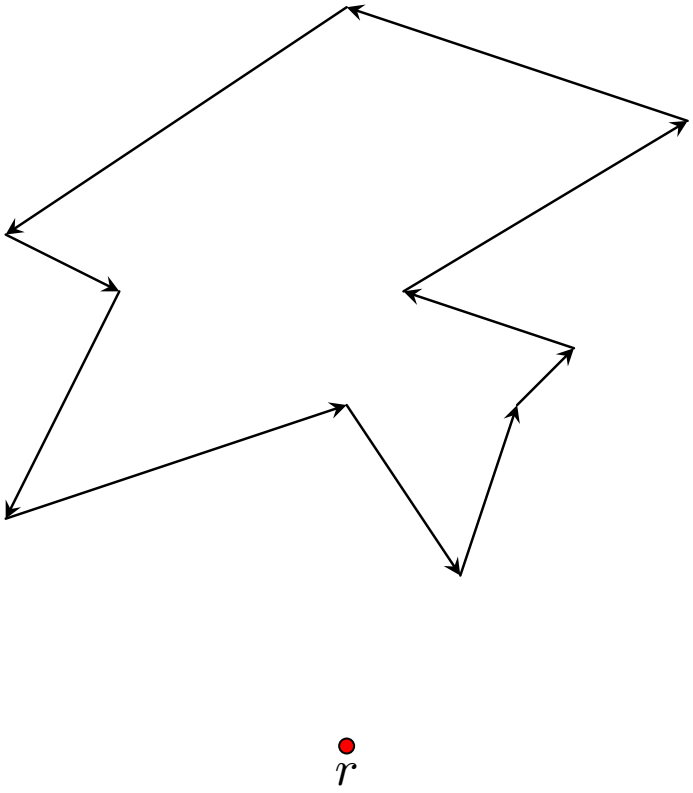


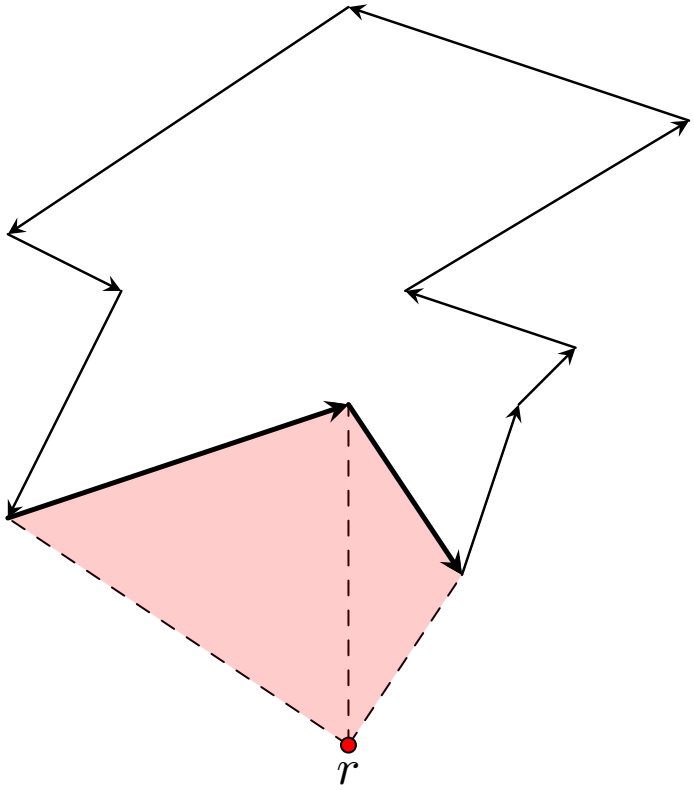
Edge-Based Area Calculation



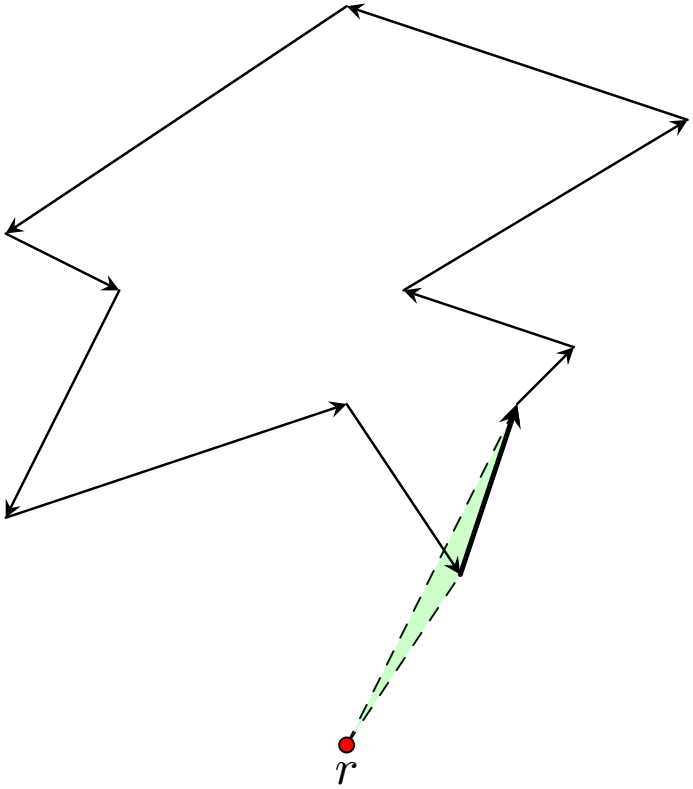


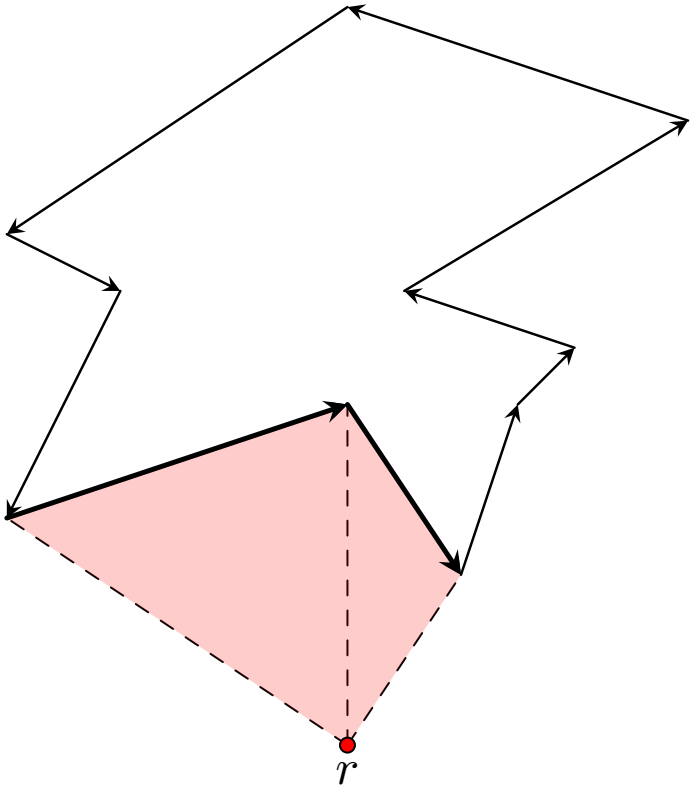
Edge-Based Area Calculation



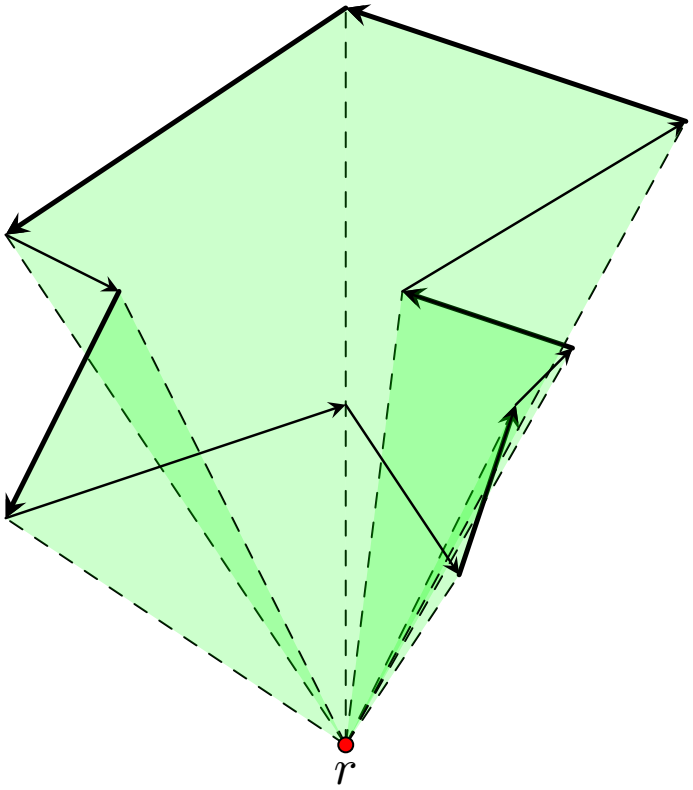


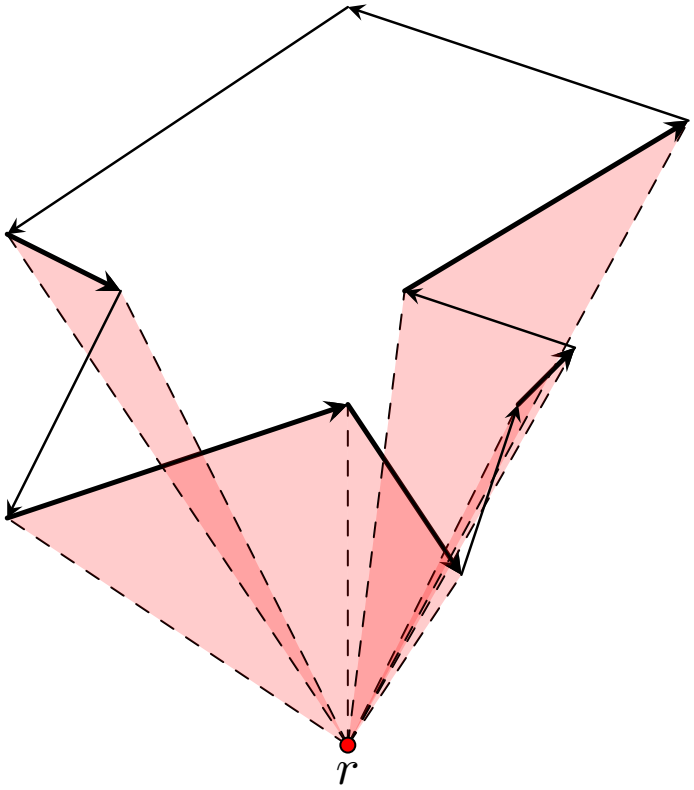
Edge-Based Area Calculation



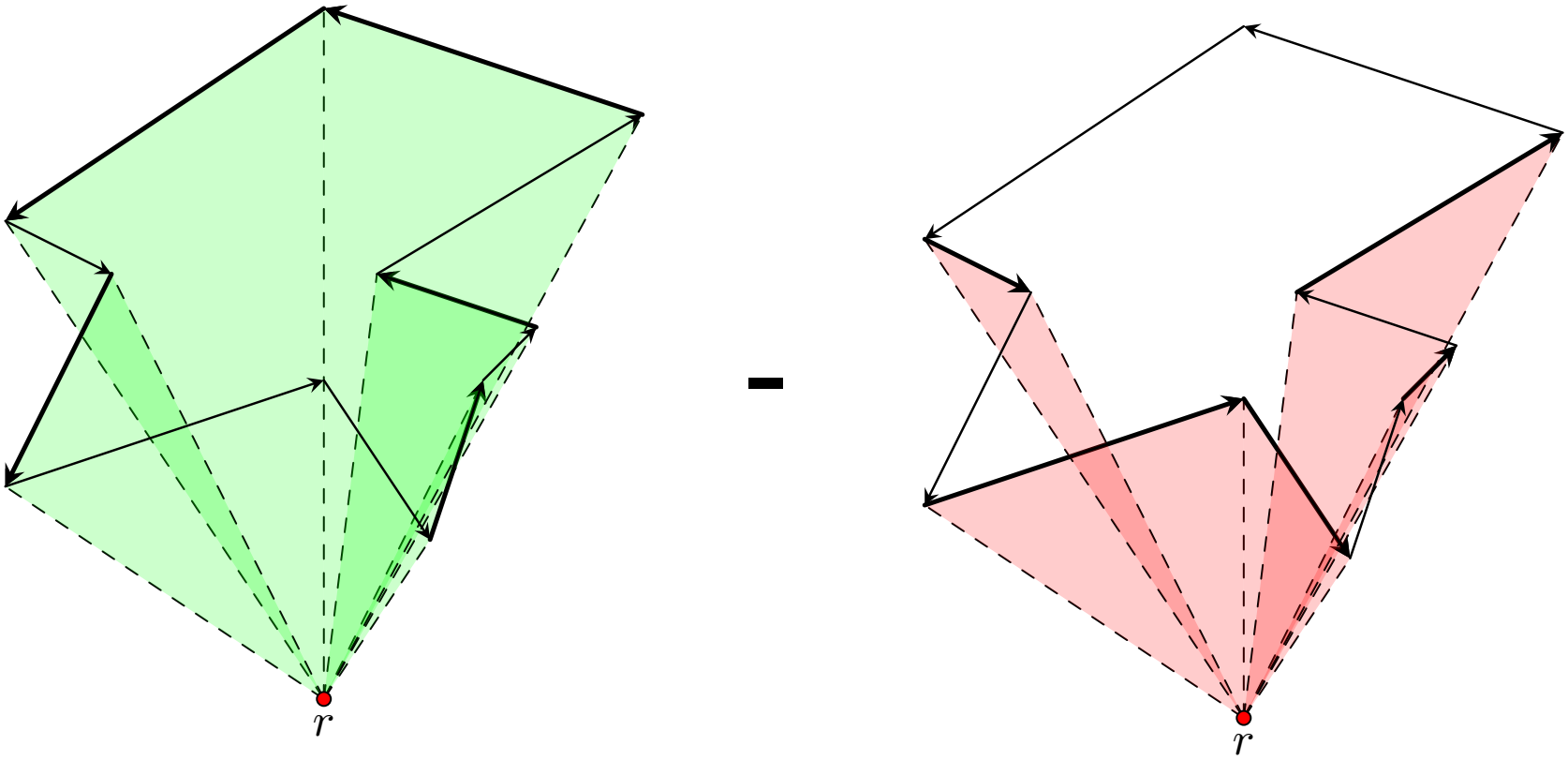


Edge-Based Area Calculation

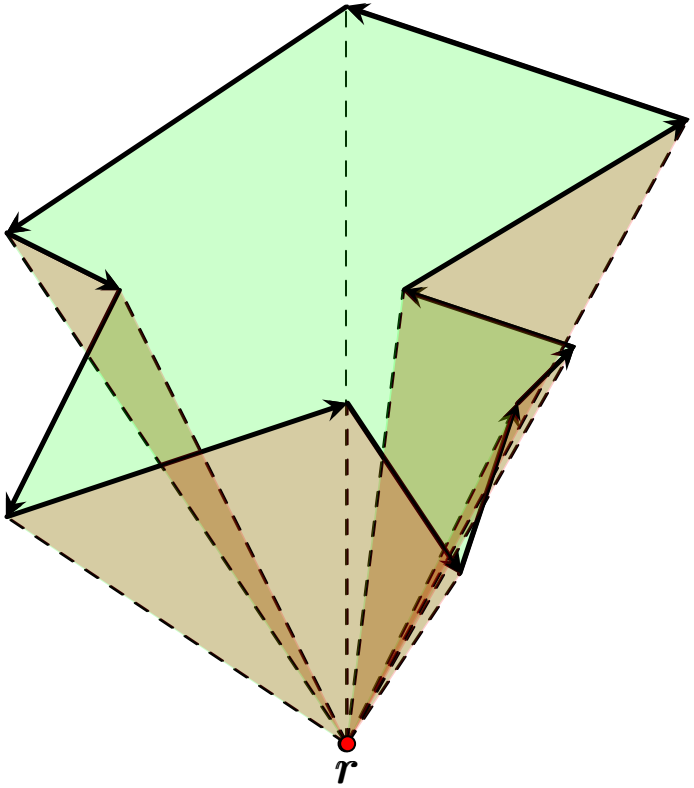




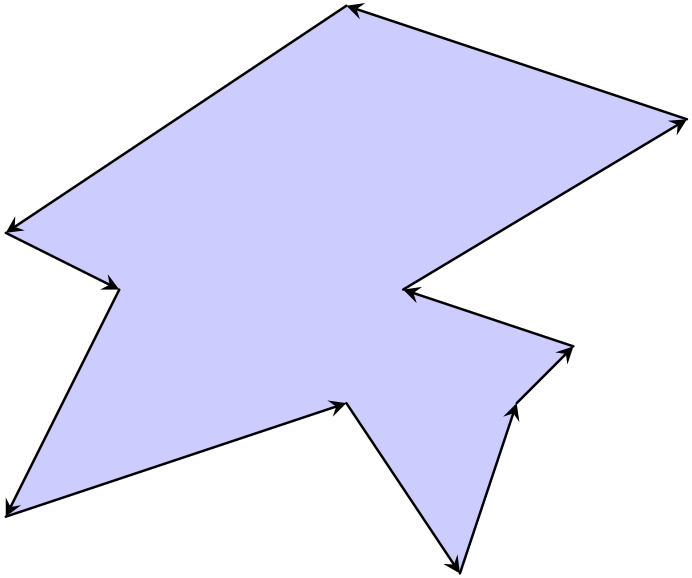
Edge-Based Area Calculation



Edge-Based Area Calculation

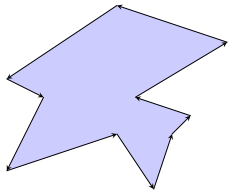


Edge-Based Integer Program



Edge-Based Integer Program

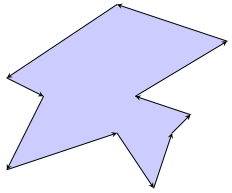
- Variables = half-edges
- Weights = signed areas of triangles



$O(n^2)$

Edge-Based Integer Program

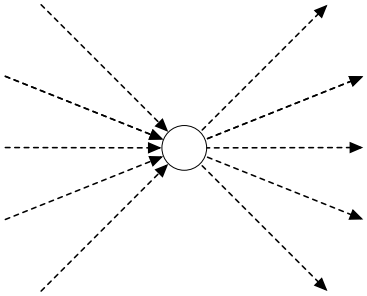
- Variables = half-edges
- Weights = signed areas of triangles



$O(n^2)$

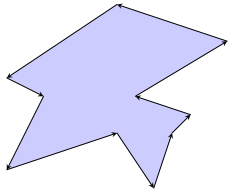
- In-degree and out-degree constraints

$O(n)$



Edge-Based Integer Program

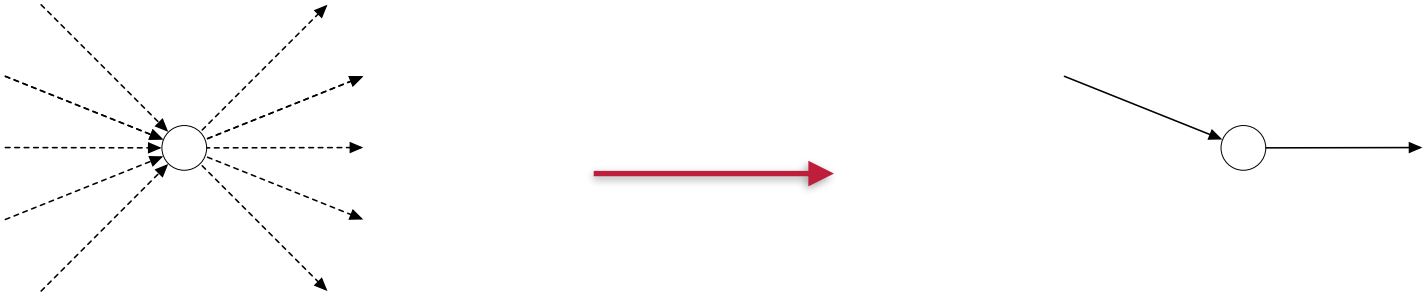
- Variables = half-edges
- Weights = signed areas of triangles



$O(n^2)$

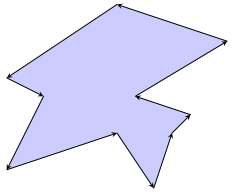
- In-degree and out-degree constraints

$O(n)$



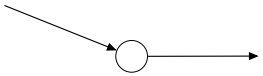
Edge-Based Integer Program

- Variables = half-edges
- Weights = signed areas of triangles



$O(n^2)$

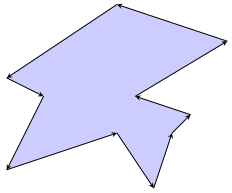
- In-degree and out-degree constraints



$O(n)$

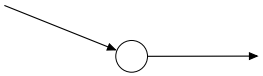
Edge-Based Integer Program

- Variables = half-edges
- Weights = signed areas of triangles



$O(n^2)$

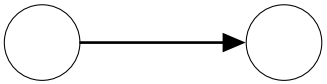
- In-degree and out-degree constraints

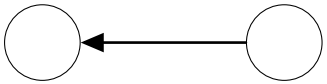


$O(n)$

- Chose at most one half-edge

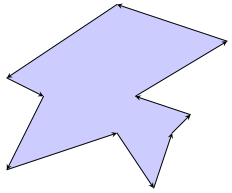
$O(n^2)$





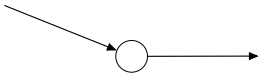
Edge-Based Integer Program

- Variables = half-edges
- Weights = signed areas of triangles



$O(n^2)$

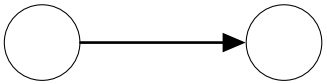
- In-degree and out-degree constraints

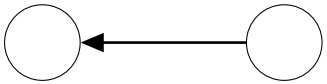


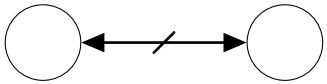
$O(n)$

- Chose at most one half-edge

$O(n^2)$

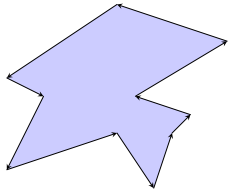






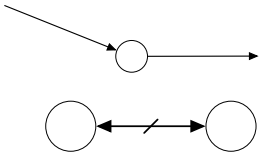
Edge-Based Integer Program

- Variables = half-edges
- Weights = signed areas of triangles



$O(n^2)$

- In-degree and out-degree constraints



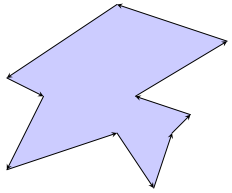
$O(n)$

- Chose at most one half-edge

$O(n^2)$

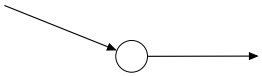
Edge-Based Integer Program

- Variables = half-edges
- Weights = signed areas of triangles



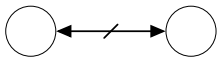
$O(n^2)$

- In-degree and out-degree constraints



$O(n)$

- Chose at most one half-edge



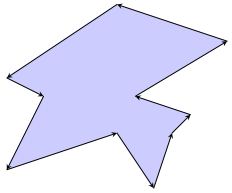
$O(n^2)$

- Avoid Intersections

$O(n^4)$

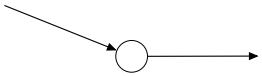
Edge-Based Integer Program

- Variables = half-edges
- Weights = signed areas of triangles



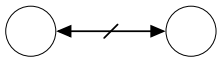
$O(n^2)$

- In-degree and out-degree constraints



$O(n)$

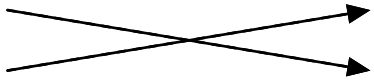
- Chose at most one half-edge



$O(n^2)$

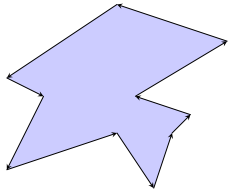
- Avoid Intersections

$O(n^4)$



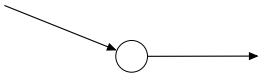
Edge-Based Integer Program

- Variables = half-edges
- Weights = signed areas of triangles



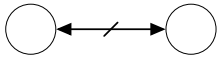
$O(n^2)$

- In-degree and out-degree constraints



$O(n)$

- Chose at most one half-edge



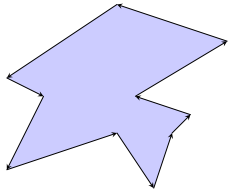
$O(n^2)$

- Avoid Intersections

$O(n^4)$

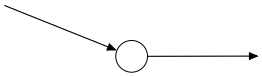
Edge-Based Integer Program

- Variables = half-edges
- Weights = signed areas of triangles



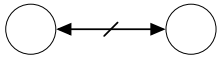
$O(n^2)$

- In-degree and out-degree constraints



$O(n)$

- Chose at most one half-edge



$O(n^2)$

- Avoid Intersections

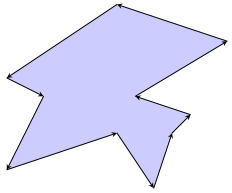
$O(n^4)$

- Counterclockwise orientation

$O(n^3)$

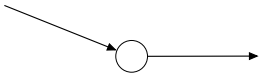
Edge-Based Integer Program

- Variables = half-edges
- Weights = signed areas of triangles



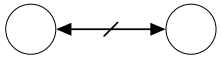
$O(n^2)$

- In-degree and out-degree constraints



$O(n)$

- Chose at most one half-edge



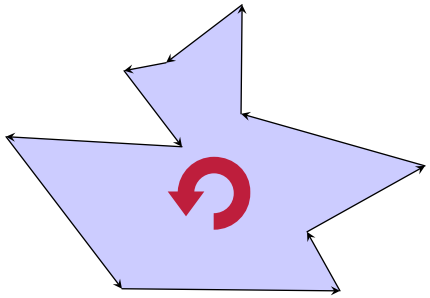
$O(n^2)$

- Avoid Intersections

$O(n^4)$

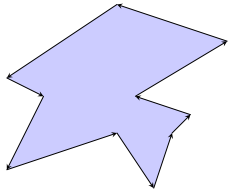
- Counterclockwise orientation

$O(n^3)$



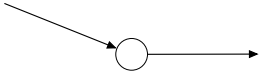
Edge-Based Integer Program

- Variables = half-edges
- Weights = signed areas of triangles



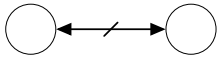
$O(n^2)$

- In-degree and out-degree constraints



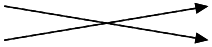
$O(n)$

- Chose at most one half-edge



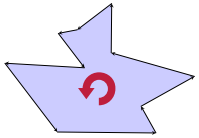
$O(n^2)$

- Avoid Intersections



$O(n^4)$

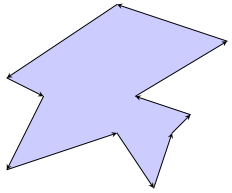
- Counterclockwise orientation



$O(n^3)$

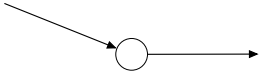
Edge-Based Integer Program

- Variables = half-edges
- Weights = signed areas of triangles



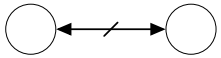
$O(n^2)$

- In-degree and out-degree constraints



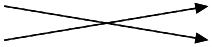
$O(n)$

- Chose at most one half-edge



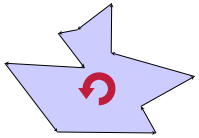
$O(n^2)$

- Avoid Intersections



$O(n^4)$

- Counterclockwise orientation



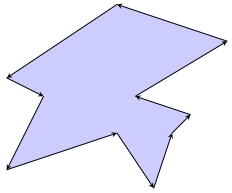
$O(n^3)$

- Subtour Constraints

$O(2^n)$

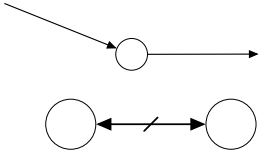
Edge-Based Integer Program

- Variables = half-edges
- Weights = signed areas of triangles



$O(n^2)$

- In-degree and out-degree constraints



$O(n)$

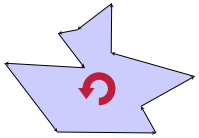
- Chose at most one half-edge

$O(n^2)$

- Avoid Intersections

$O(n^4)$

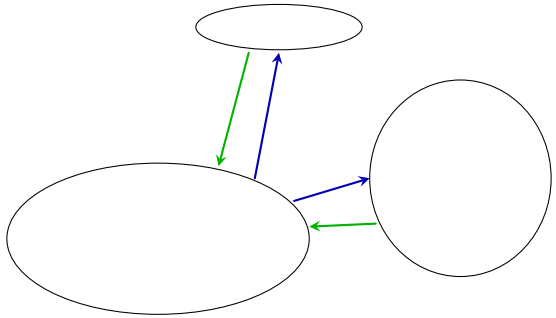
- Counterclockwise orientation



$O(n^3)$

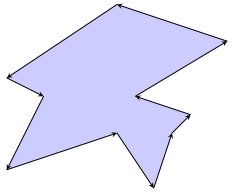
- Subtour Constraints

$O(2^n)$



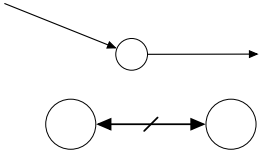
Edge-Based Integer Program

- Variables = half-edges
- Weights = signed areas of triangles



$O(n^2)$

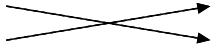
- In-degree and out-degree constraints
- Chose at most one half-edge

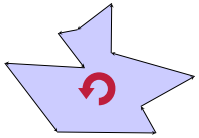


$O(n)$

$O(n^2)$

- Avoid Intersections
- Counterclockwise orientation
- Subtour Constraints

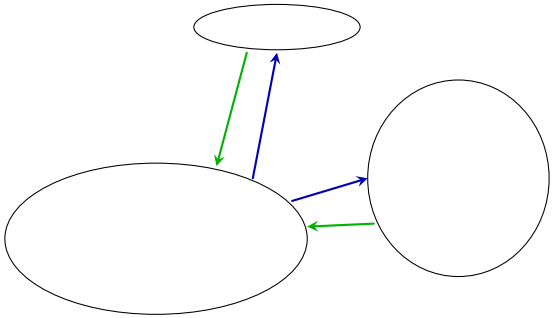




$O(n^4)$

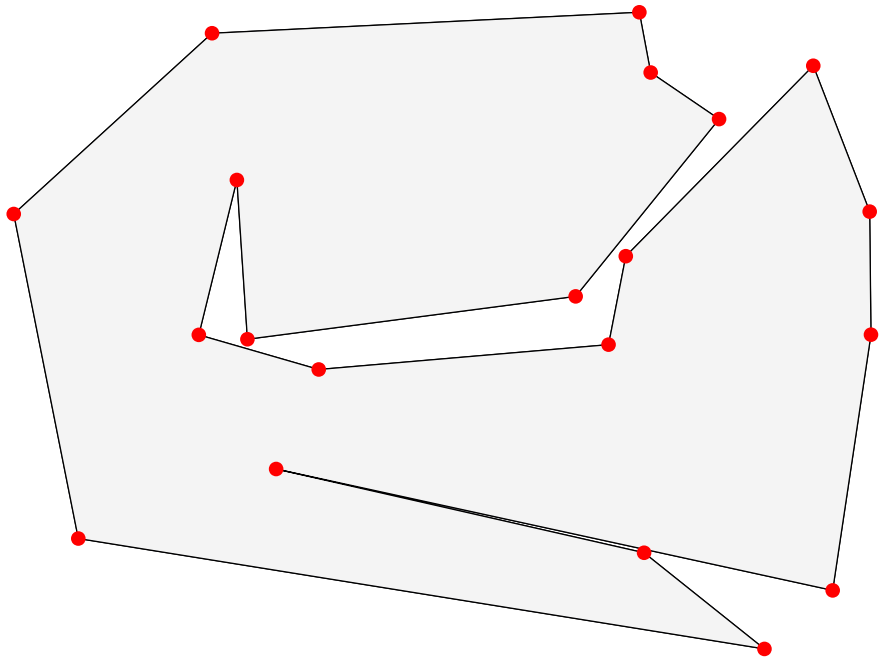
$O(n^3)$

$O(2^n)$

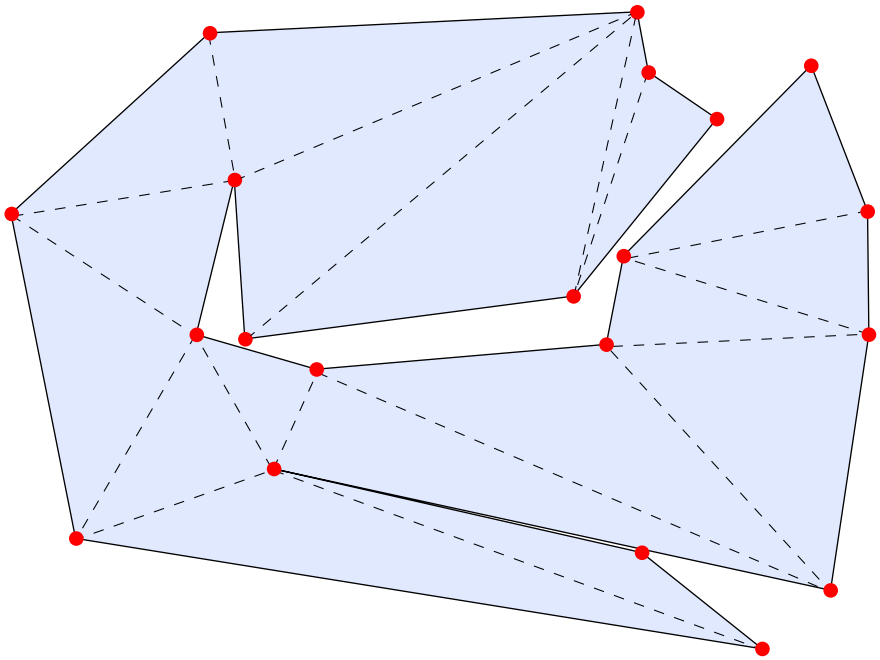


Triangulation Approach

Triangulation Approach



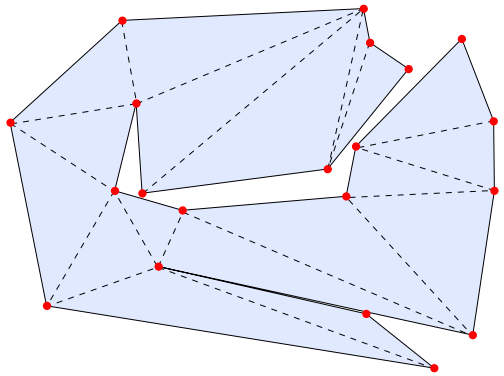
Triangulation Approach



Triangulation-Based Integer Program

- Variables = triangles
- Weights = areas of triangles

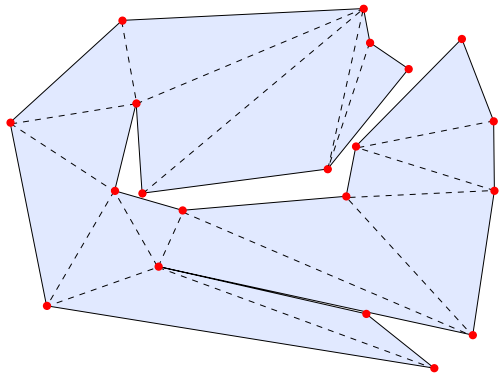
$O(n^3)$



Triangulation-Based Integer Program

- Variables = triangles
- Weights = areas of triangles

$O(n^3)$



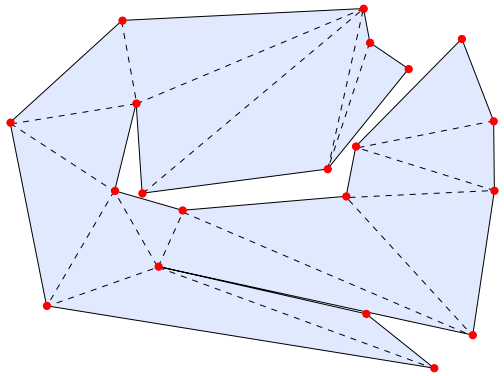
- Ensure that $n - 2$ triangles are chosen

$O(1)$

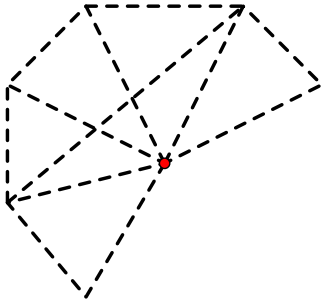
Triangulation-Based Integer Program

- Variables = triangles
- Weights = areas of triangles

$O(n^3)$



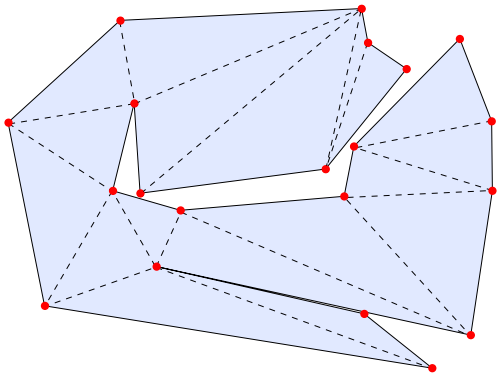
- Ensure that $n - 2$ triangles are chosen $O(1)$
- At least one triangle per point $O(n)$



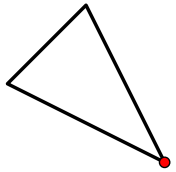
Triangulation-Based Integer Program

- Variables = triangles
- Weights = areas of triangles

$O(n^3)$



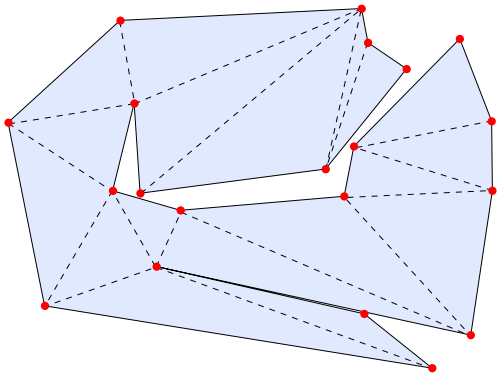
- Ensure that $n - 2$ triangles are chosen $O(1)$
- At least one triangle per point $O(n)$



Triangulation-Based Integer Program

- Variables = triangles
- Weights = areas of triangles

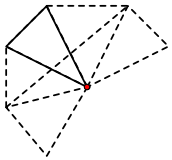
$O(n^3)$



- Ensure that $n - 2$ triangles are chosen
- At least one triangle per point

$O(1)$

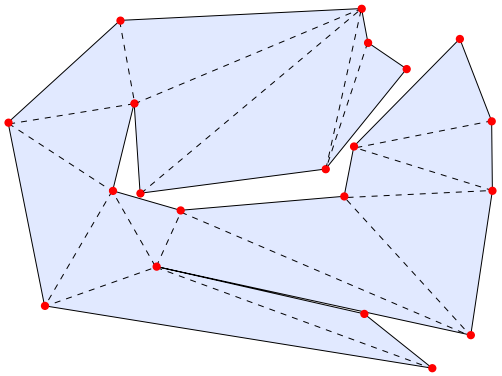
$O(n)$



Triangulation-Based Integer Program

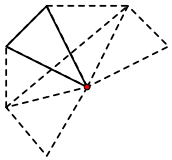
- Variables = triangles
- Weights = areas of triangles

$O(n^3)$



- Ensure that $n - 2$ triangles are chosen

$O(1)$

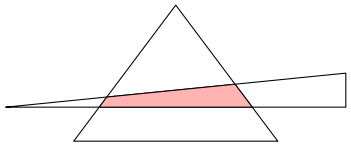


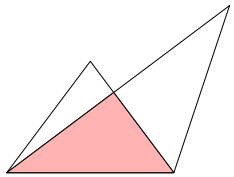
- At least one triangle per point

$O(n)$

- Avoid intersecting triangles

$O(n^6)$

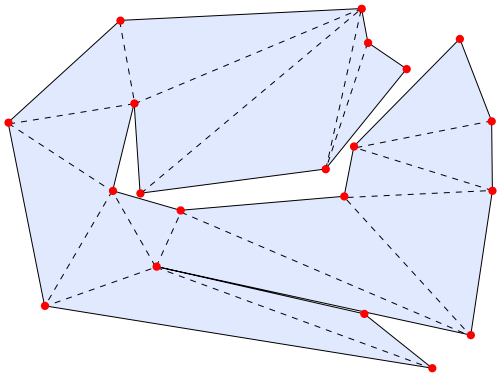




Triangulation-Based Integer Program

- Variables = triangles
- Weights = areas of triangles

$O(n^3)$

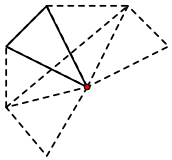


- Ensure that $n-2$ triangles are chosen

$O(1)$

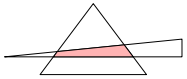
- At least one triangle per point

$O(n)$



- Avoid intersecting triangles

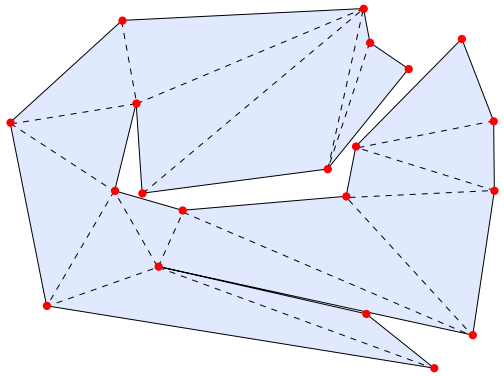
$O(n^6)$



Triangulation-Based Integer Program

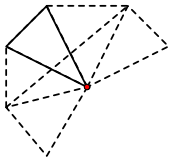
- Variables = triangles
- Weights = areas of triangles

$O(n^3)$



- Ensure that $n-2$ triangles are chosen

$O(1)$

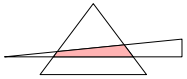


- At least one triangle per point

$O(n)$

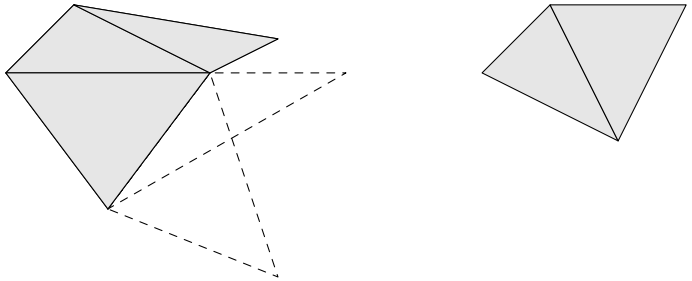
- Avoid intersecting triangles

$O(n^6)$



- Subtour Constraints

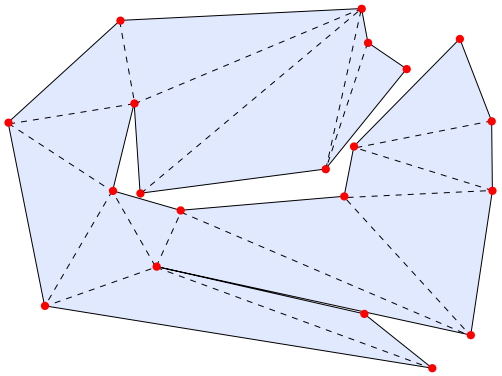
$O(2^{n^3})$



Triangulation-Based Integer Program

- Variables = triangles
- Weights = areas of triangles

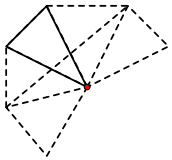
$O(n^3)$



- Ensure that $n-2$ triangles are chosen
- At least one triangle per point

$O(1)$

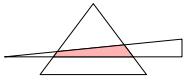
$O(n)$

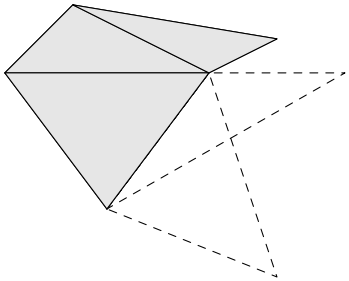


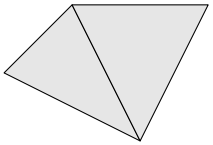
- Avoid intersecting triangles
- Subtour Constraints

$O(n^6)$

$O(2^{n^3})$



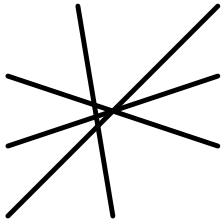




Improving the IPs

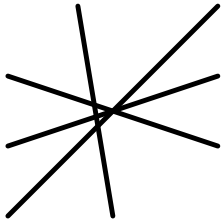
Improving the IPs

intersection cliques

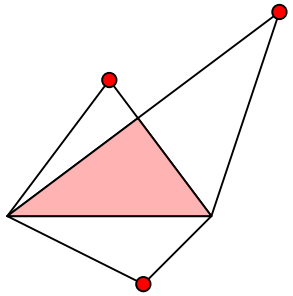


Improving the IPs

intersection cliques

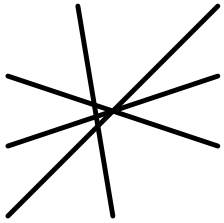


half-space constraints

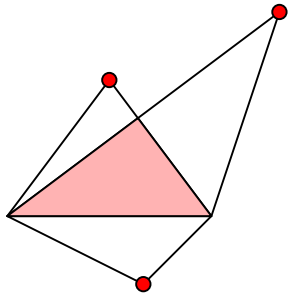


Improving the IPs

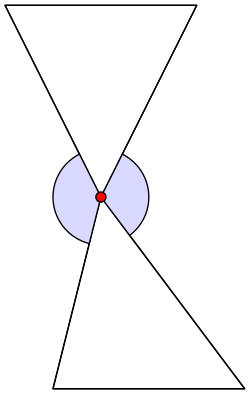
intersection cliques



half-space constraints

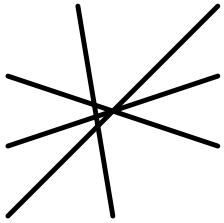


advanced subtour constraints

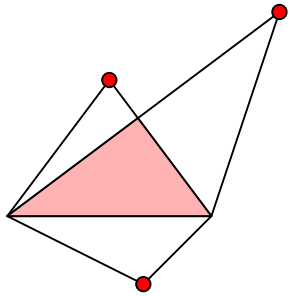


Improving the IPs

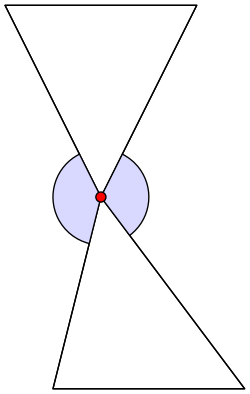
intersection cliques



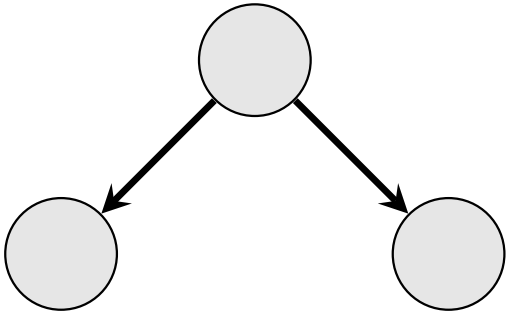
half-space constraints



advanced subtour constraints

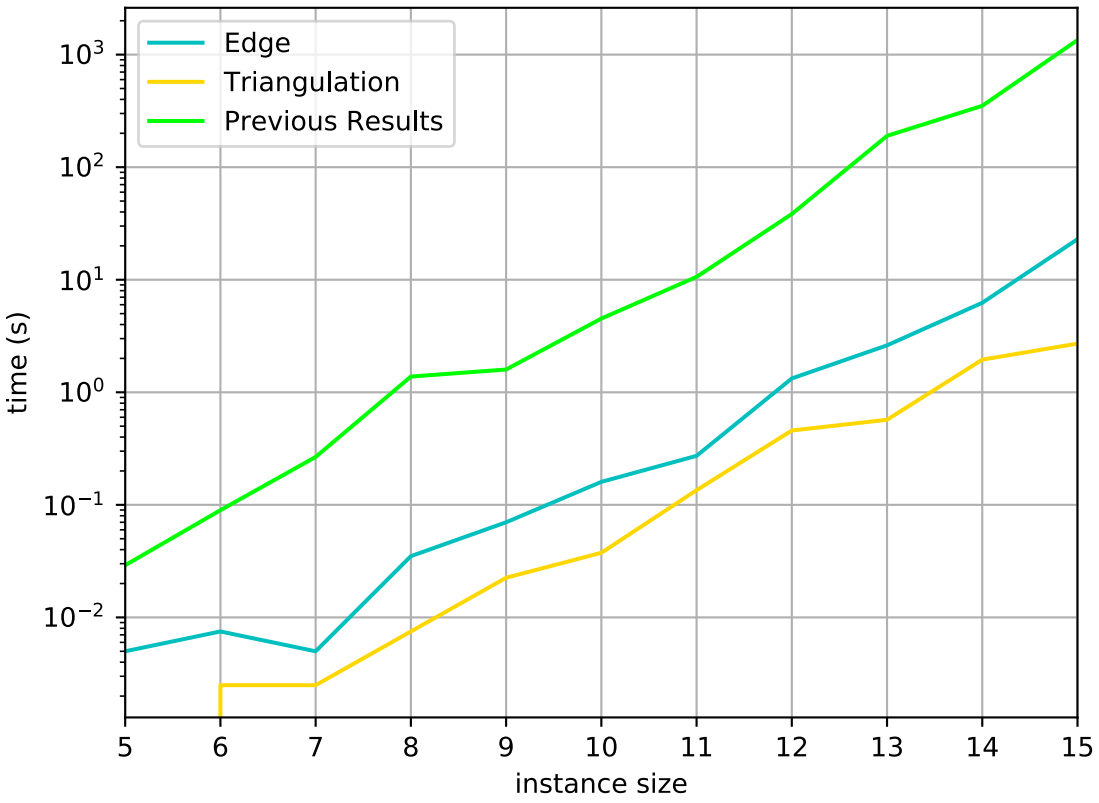


advanced branching

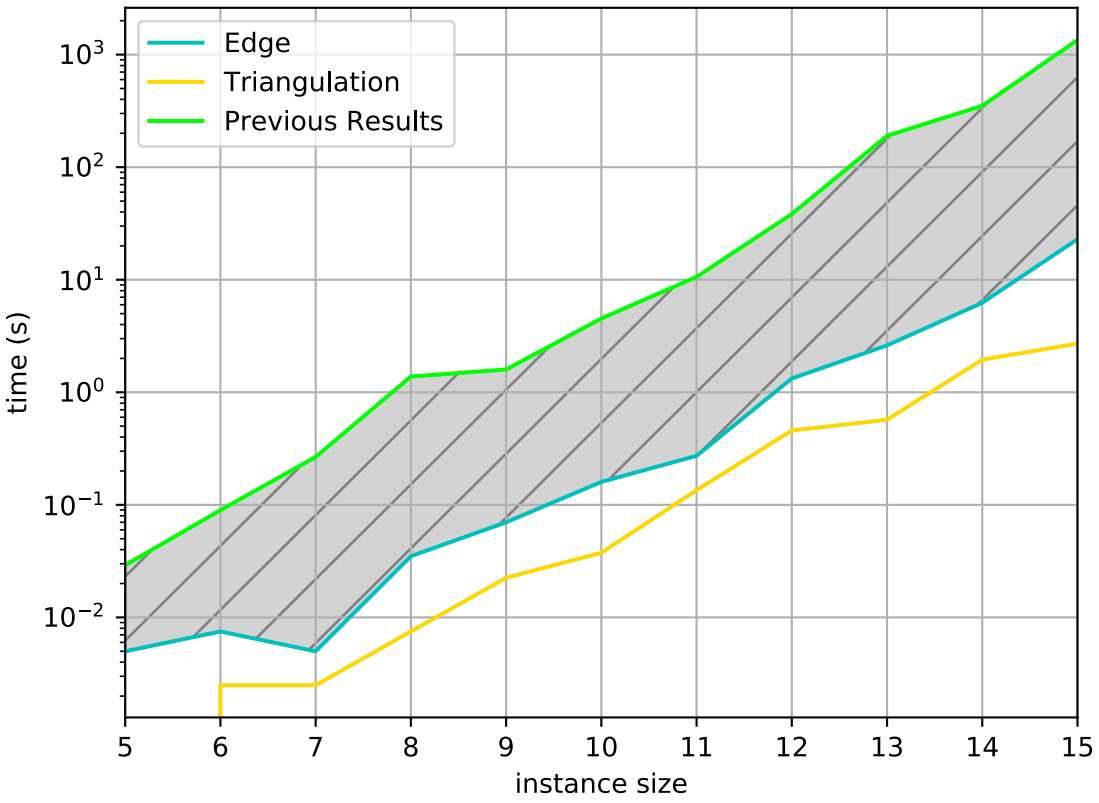


Results

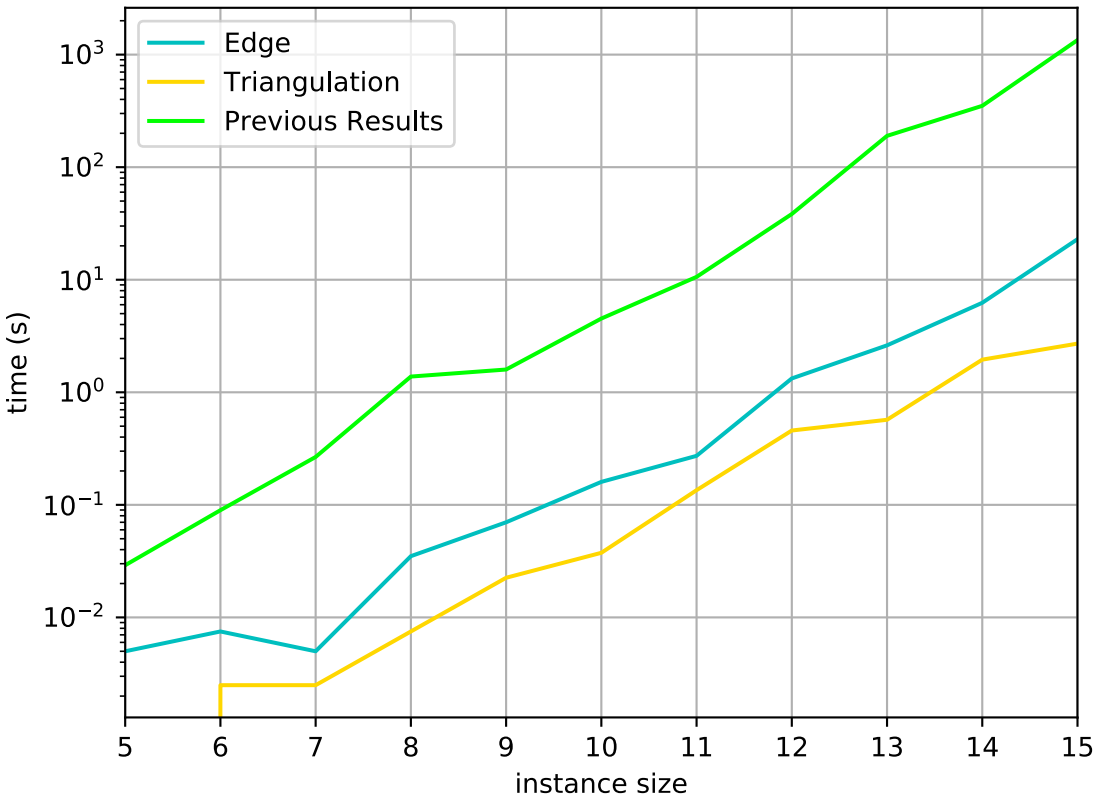
Minimizing the Area



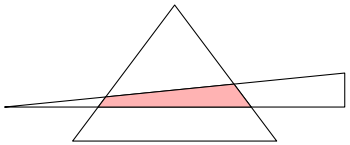
Minimizing the Area

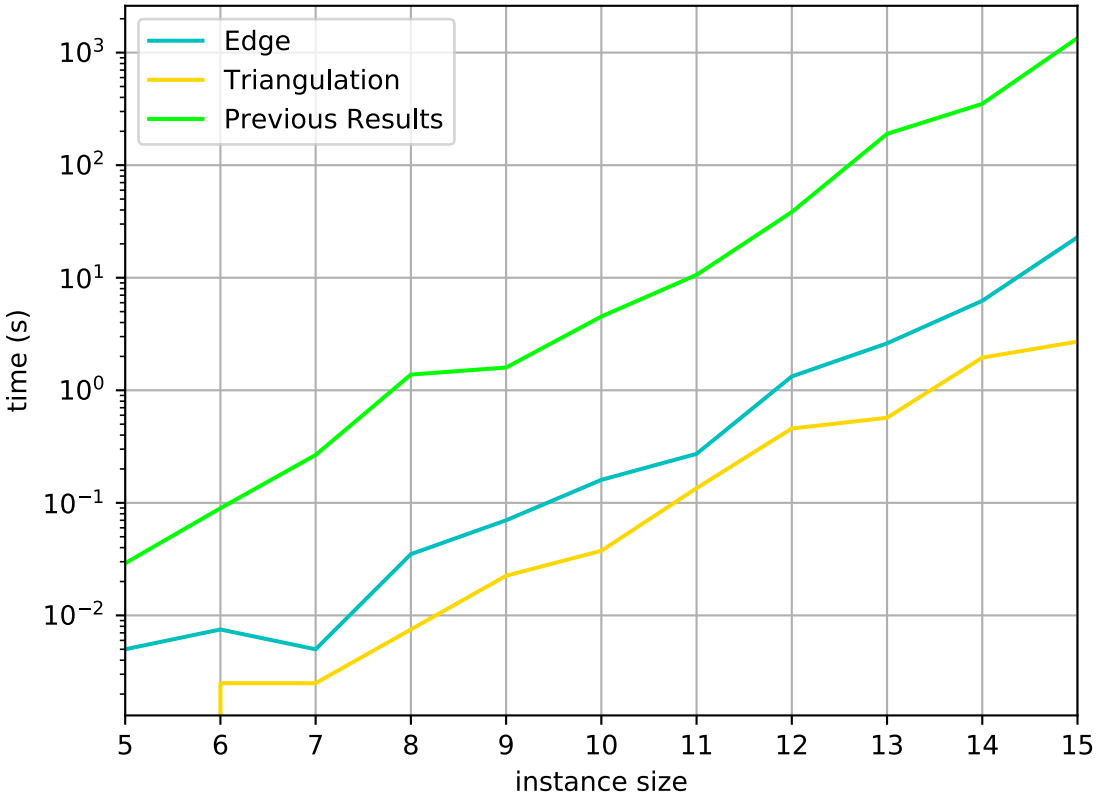


Minimizing the Area

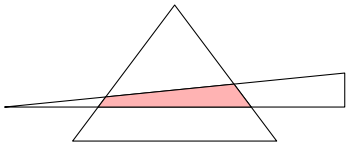


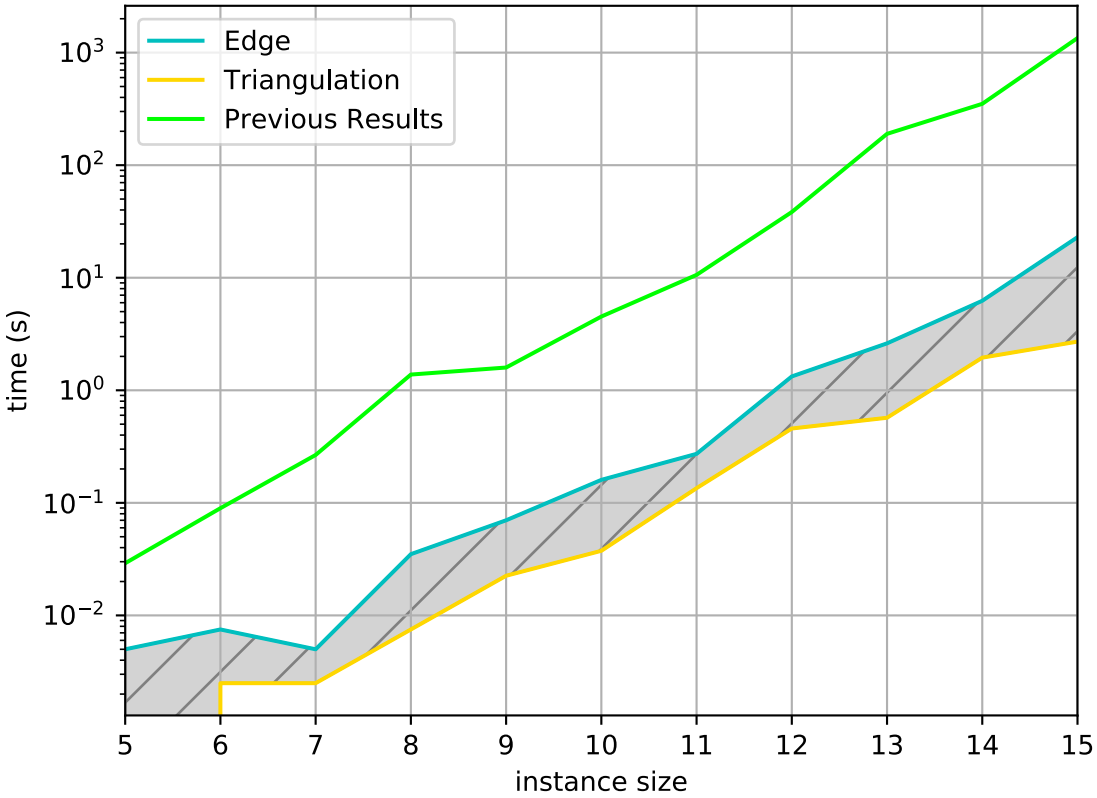
Minimizing the Area



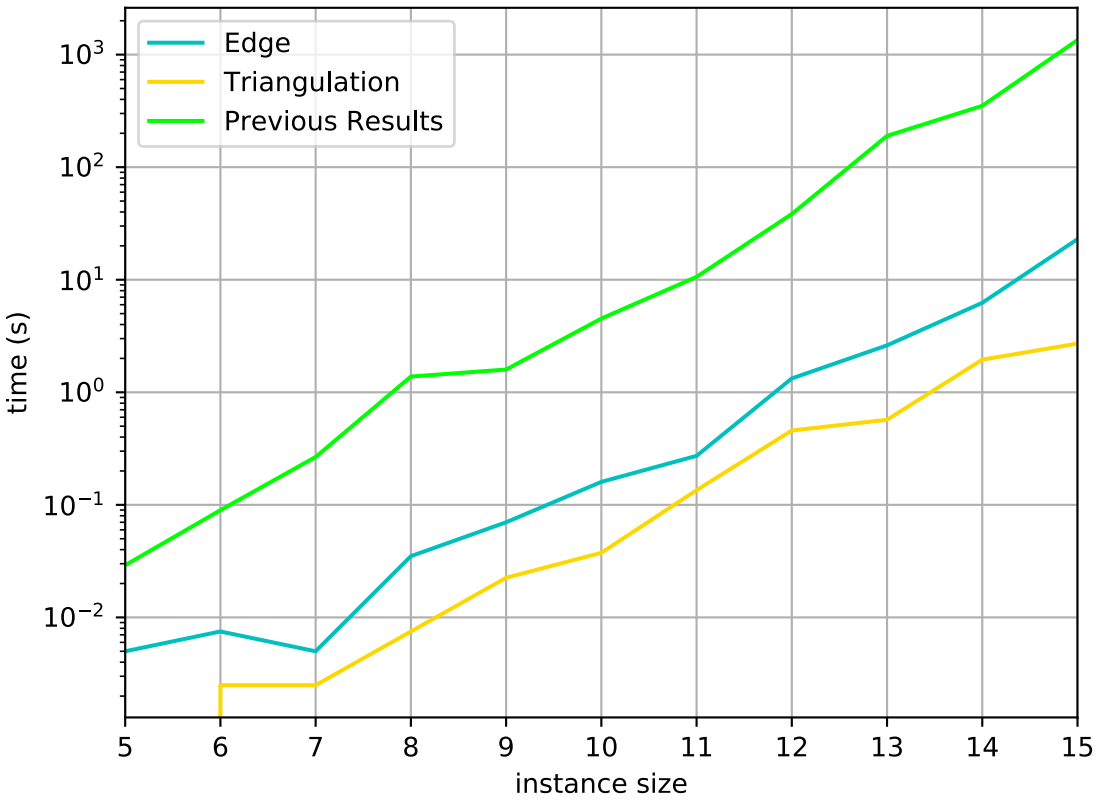


Minimizing the Area





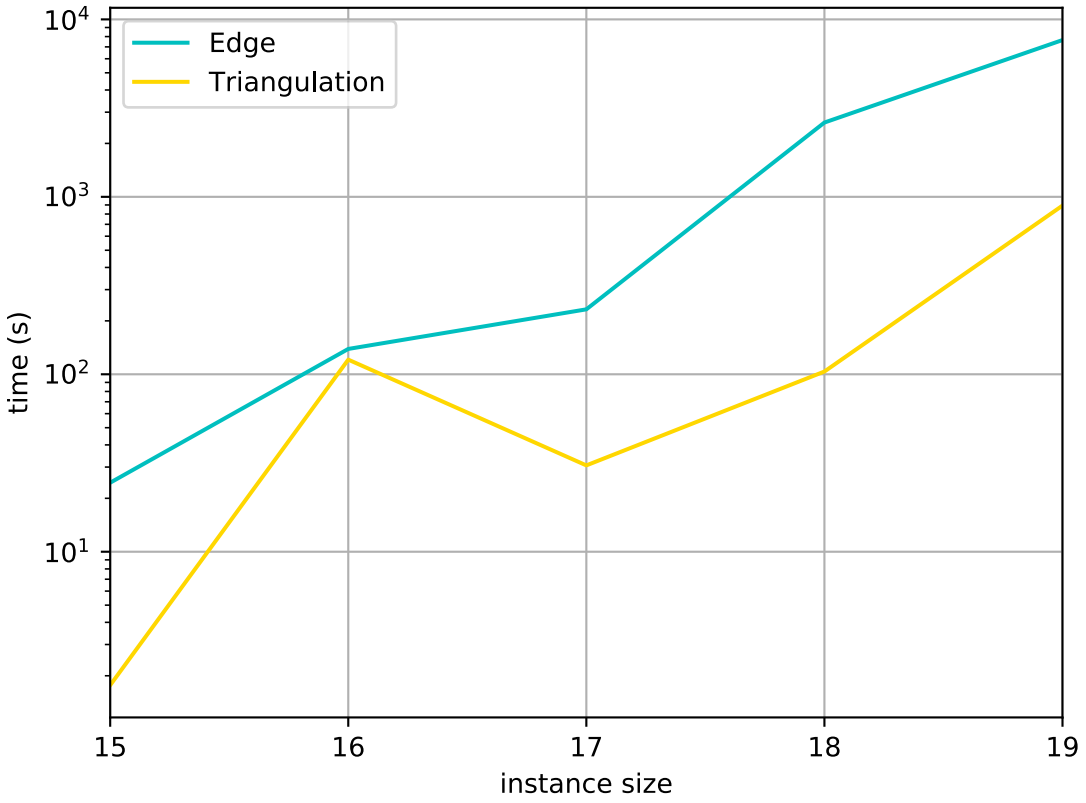
Minimizing the Area



Minimizing the Area

Minimizing the Area

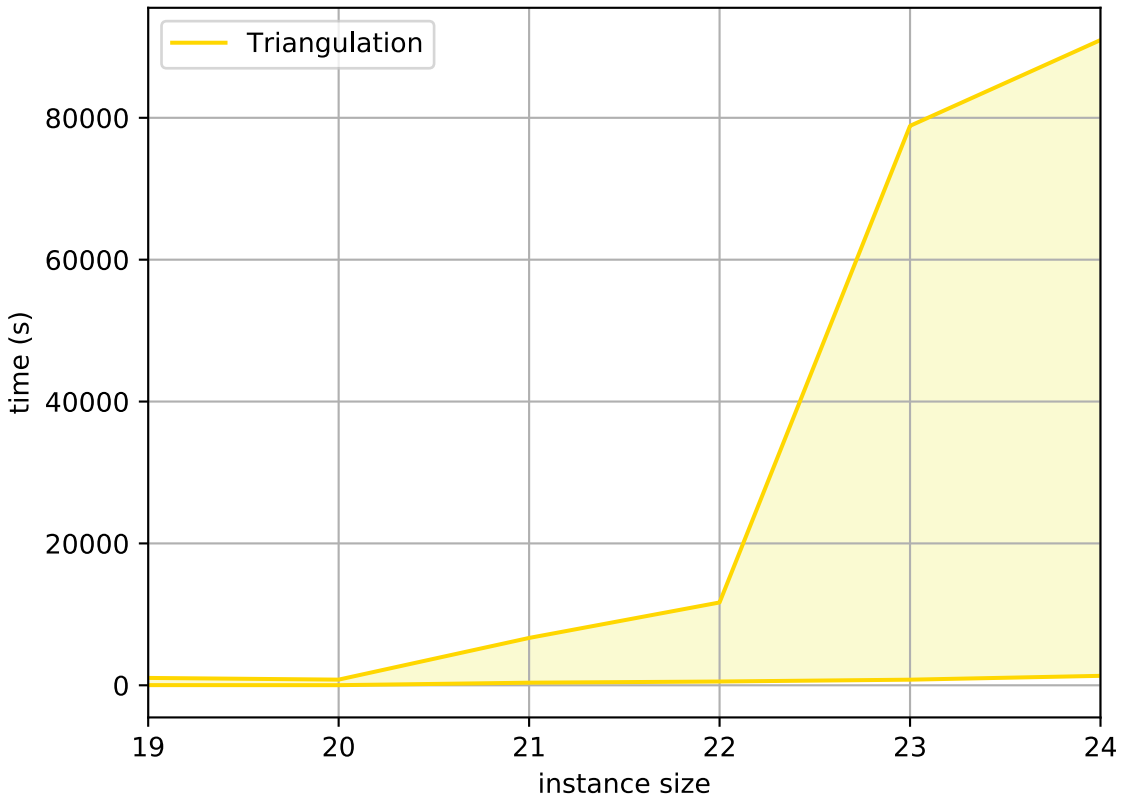
Minimizing the Area



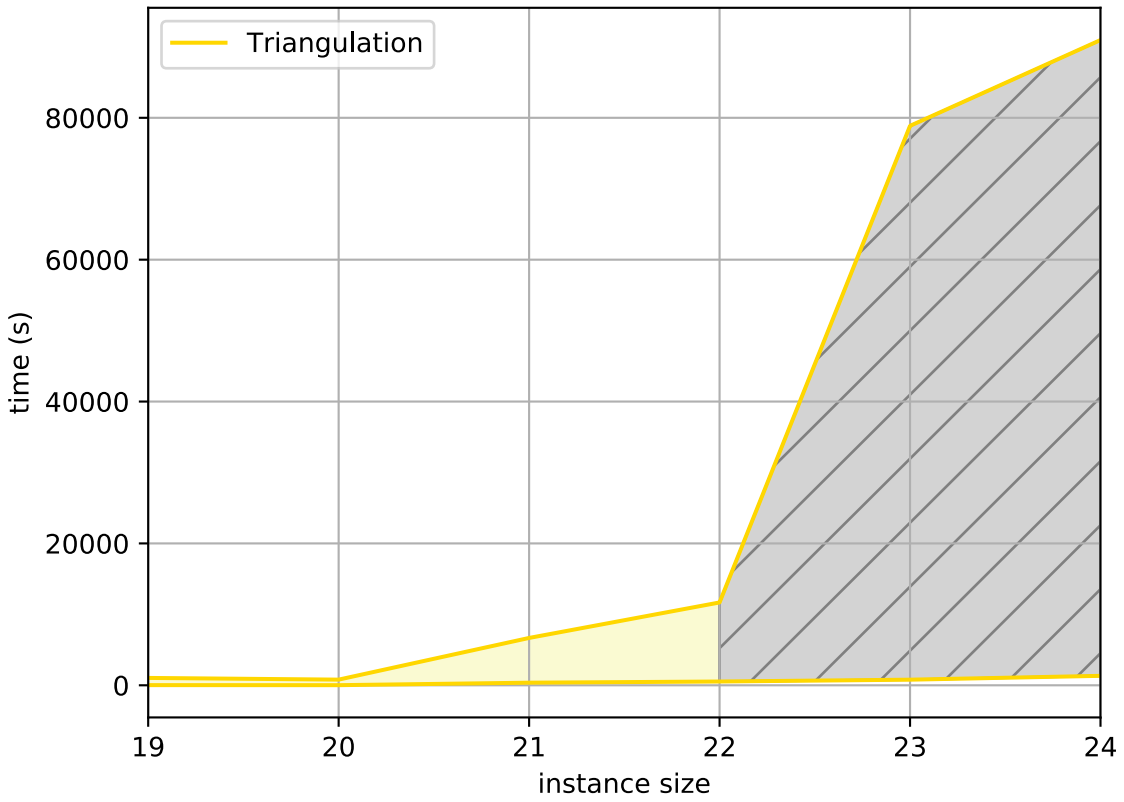
Minimizing the Area

Minimizing the Area

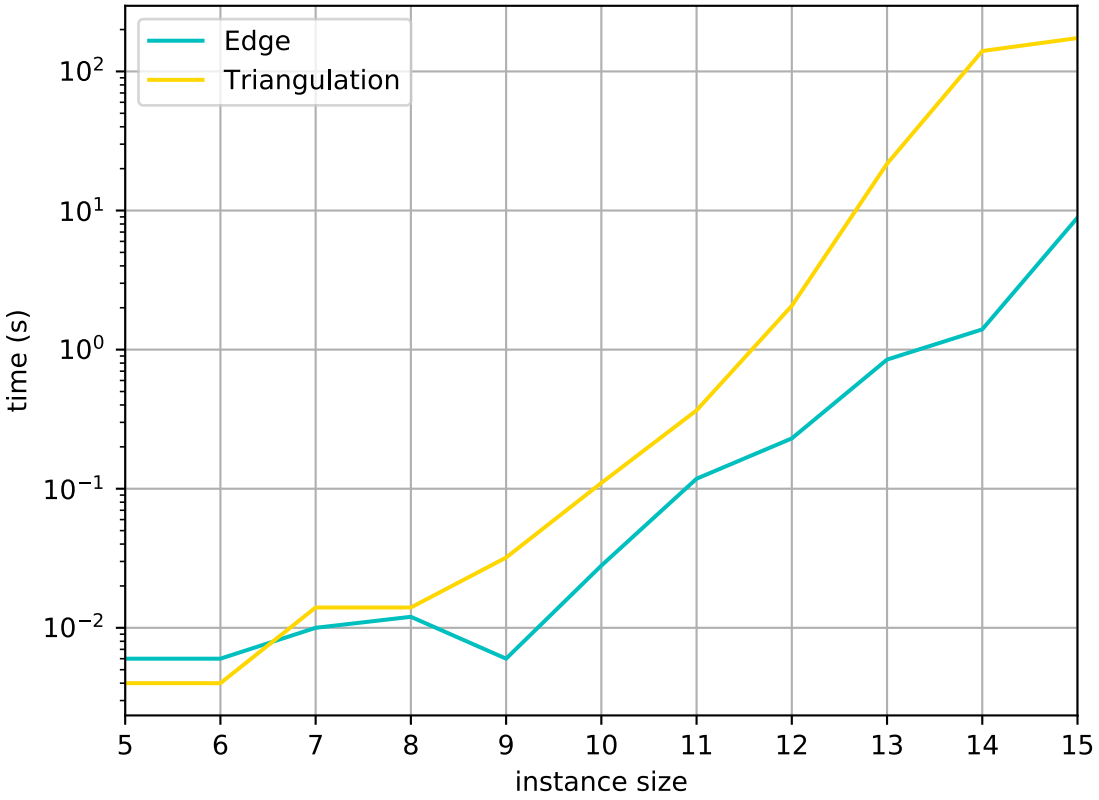
Minimizing the Area



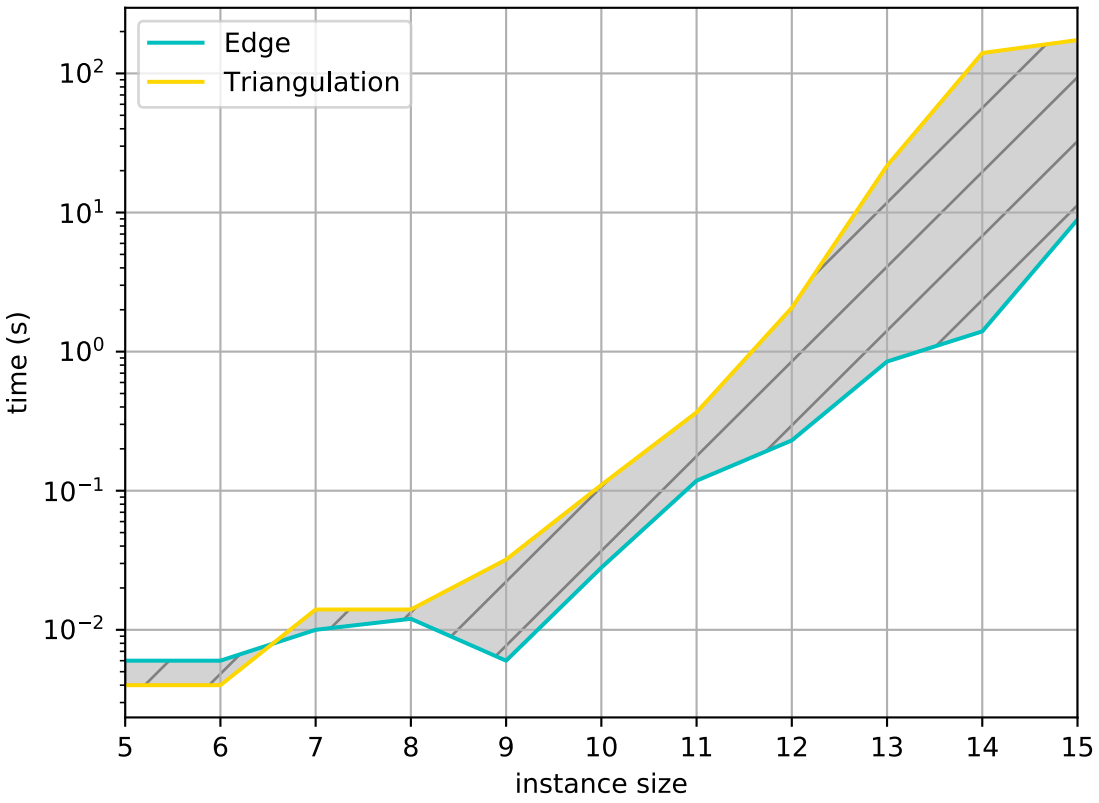
Minimizing the Area



Maximizing the Area



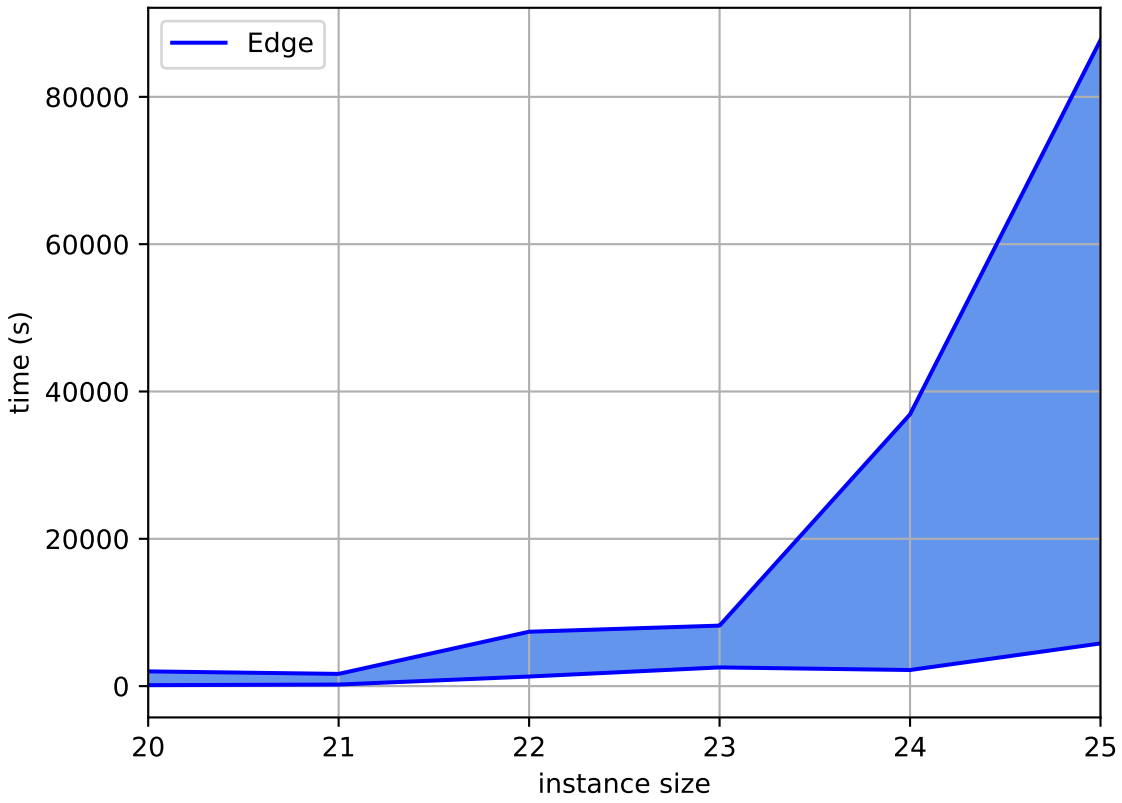
Maximizing the Area



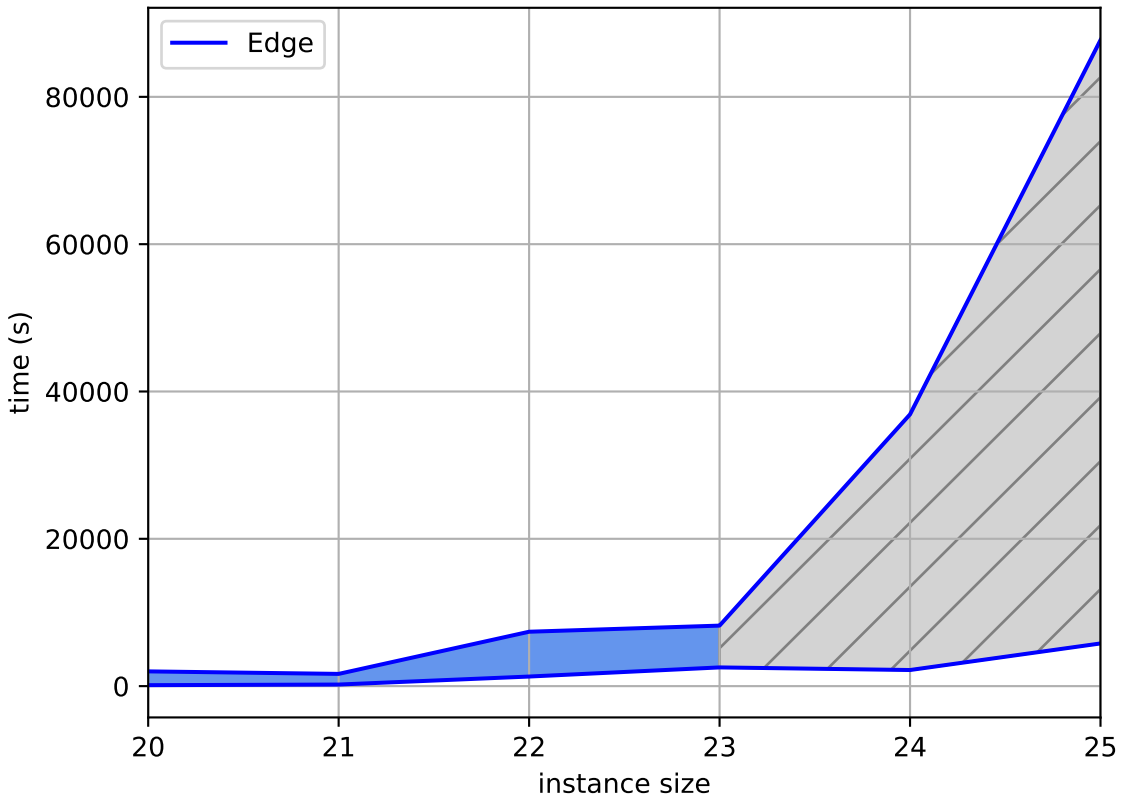
Maximizing the Area

Maximizing the Area

Maximizing the Area

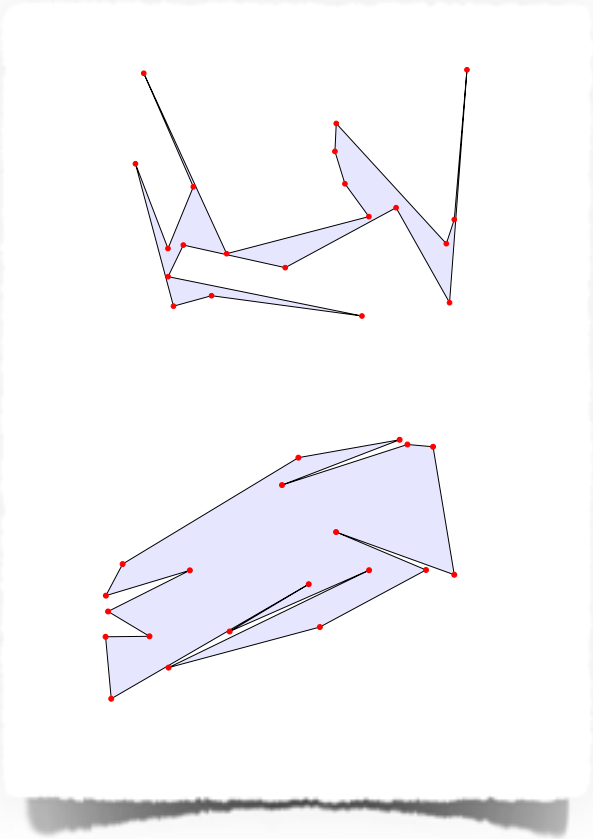


Maximizing the Area

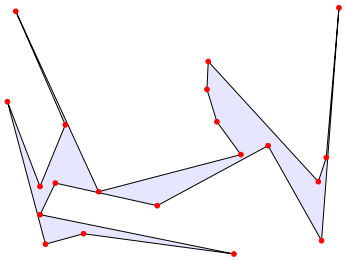


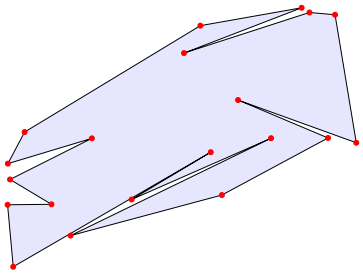
Conclusion

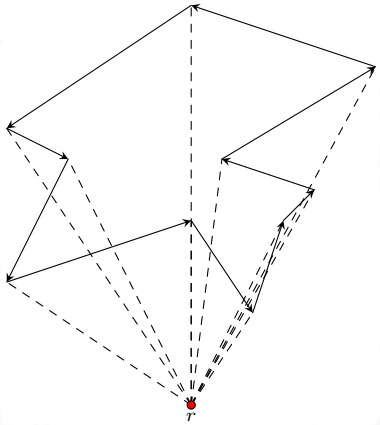
Conclusion

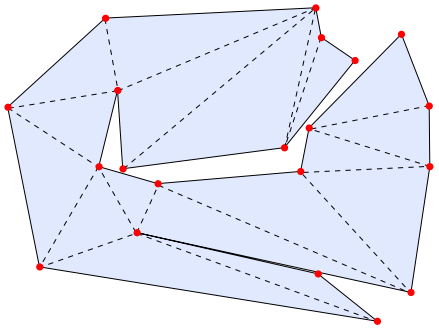


Conclusion

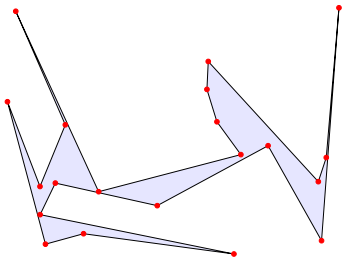


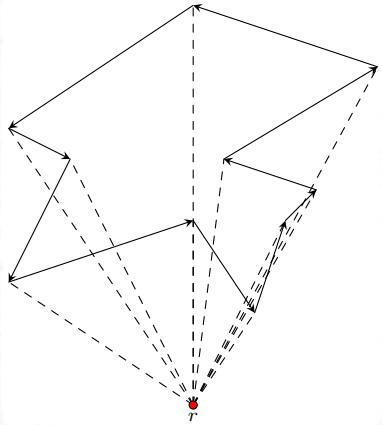


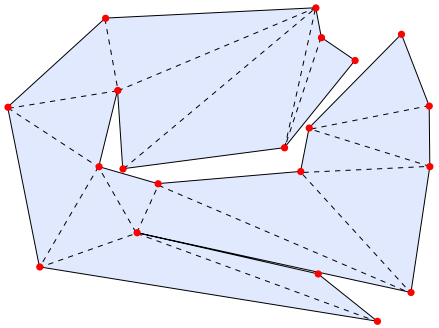


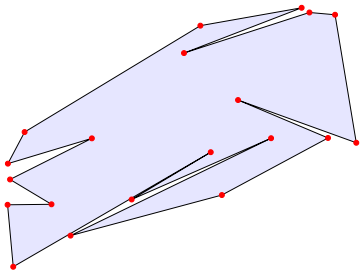


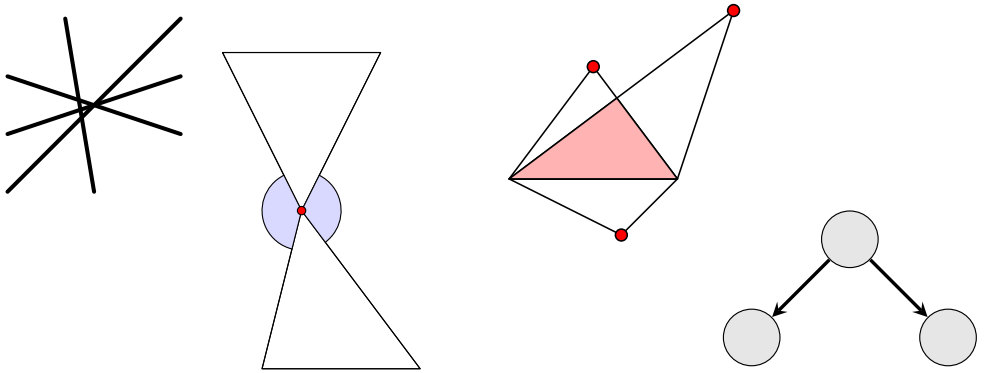
Conclusion



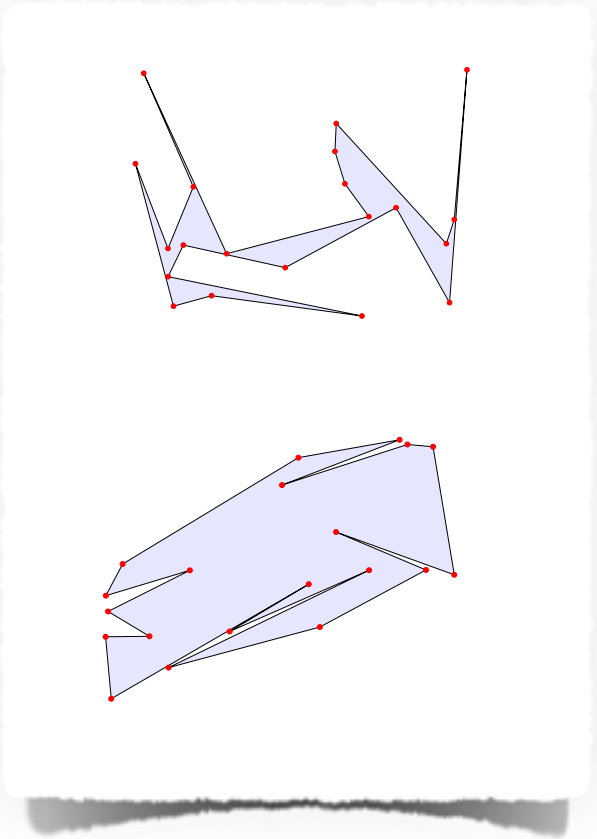


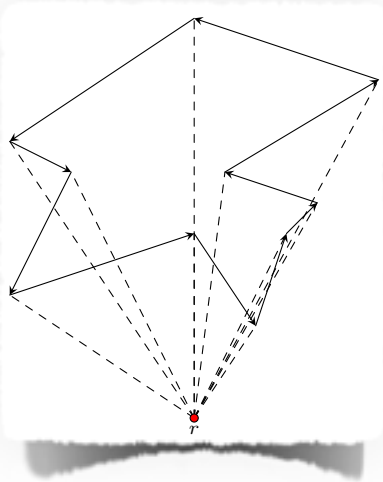


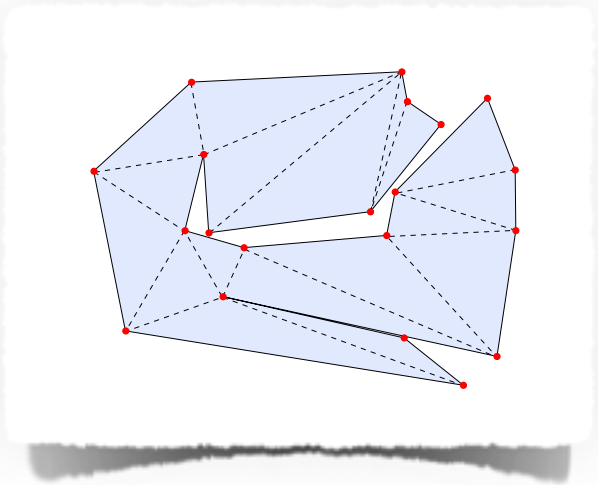


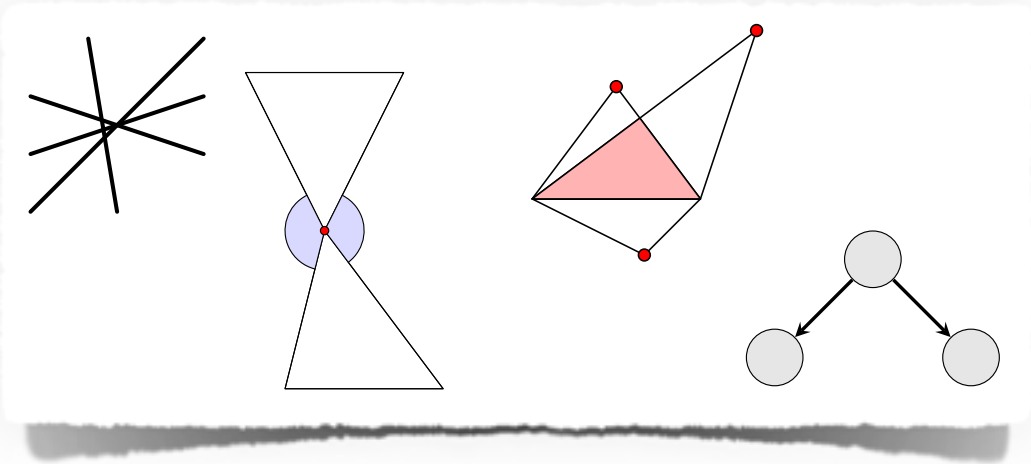


Conclusion





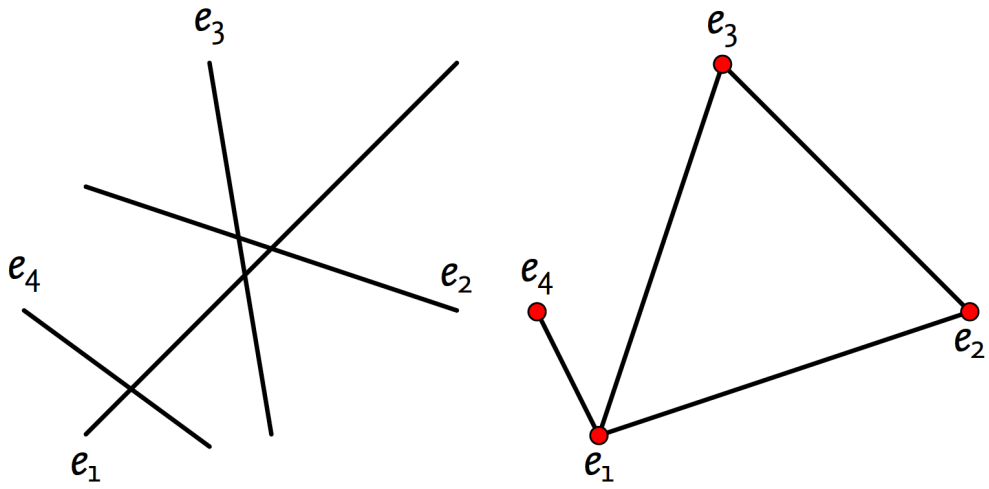




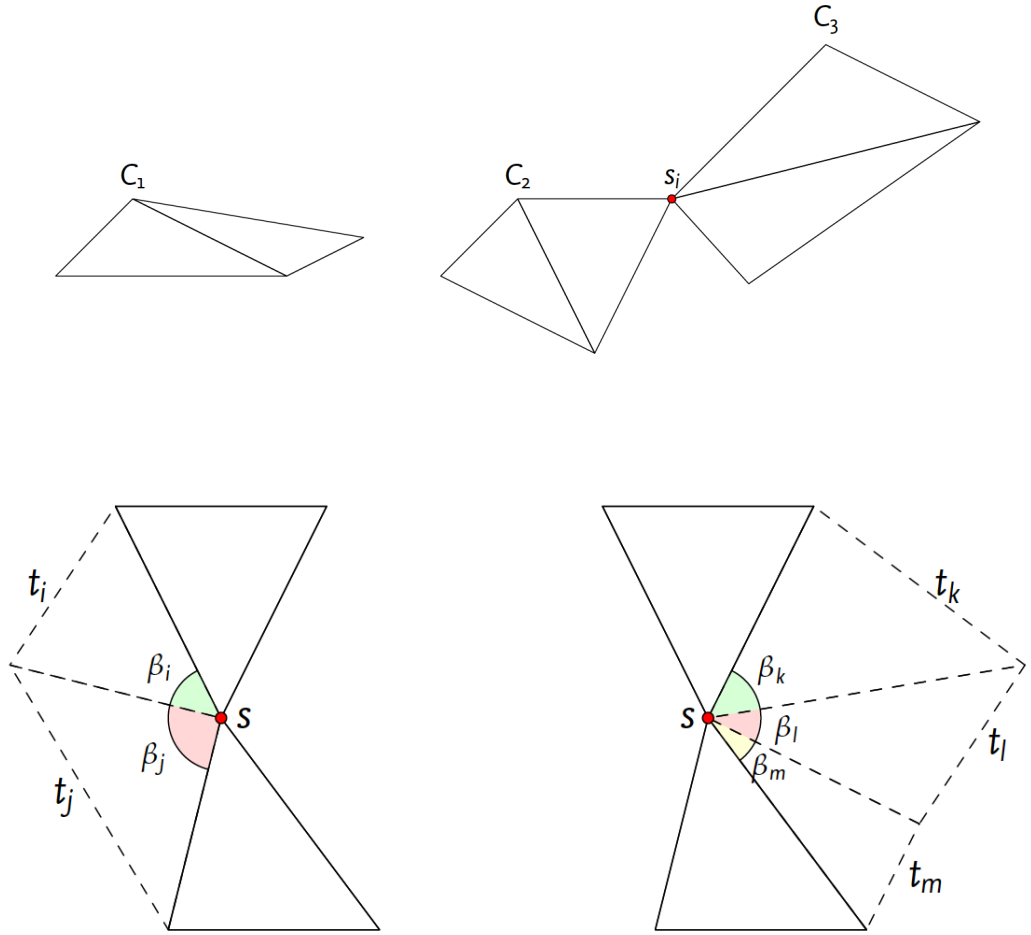
→ Increased solvable instance size by 50%

Backup

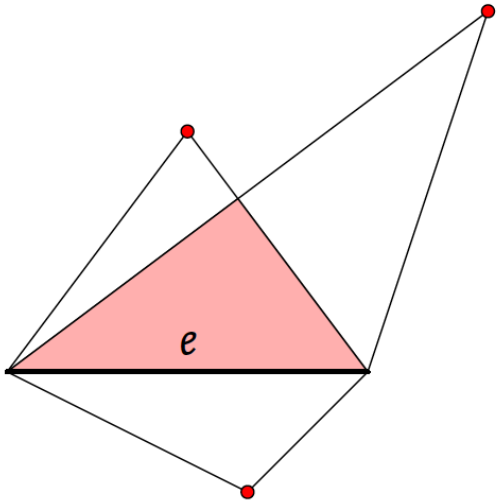
Intersection Cliques

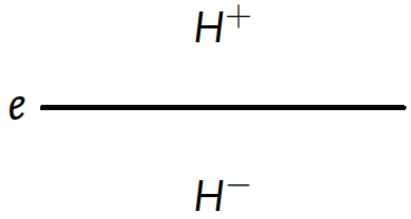


Subtour Angle Constraints

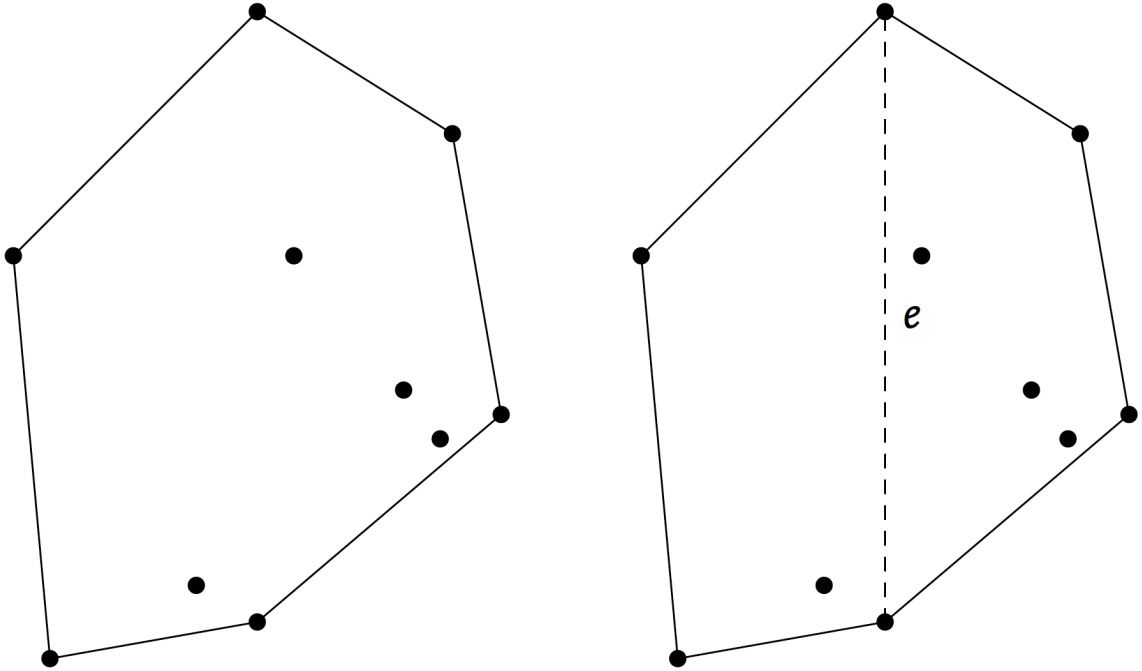


Half-space Constraints

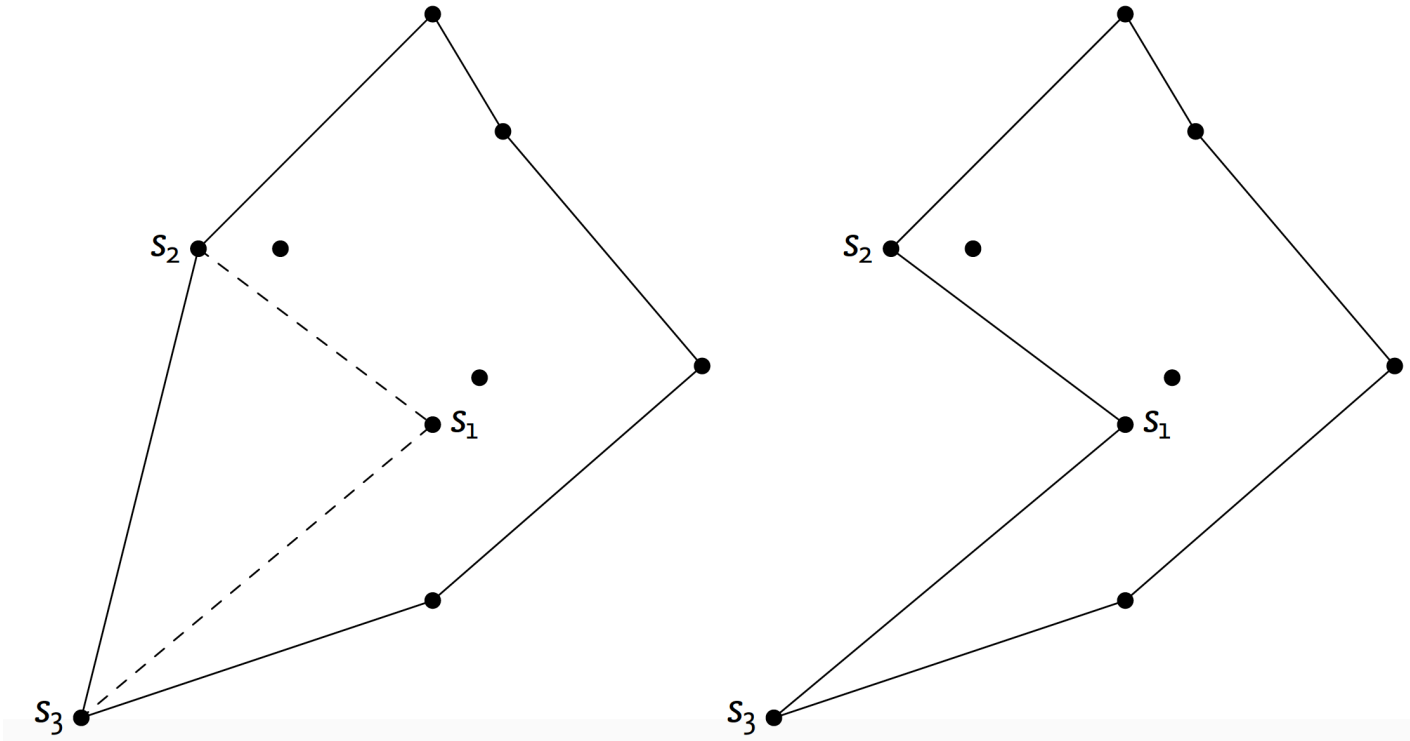




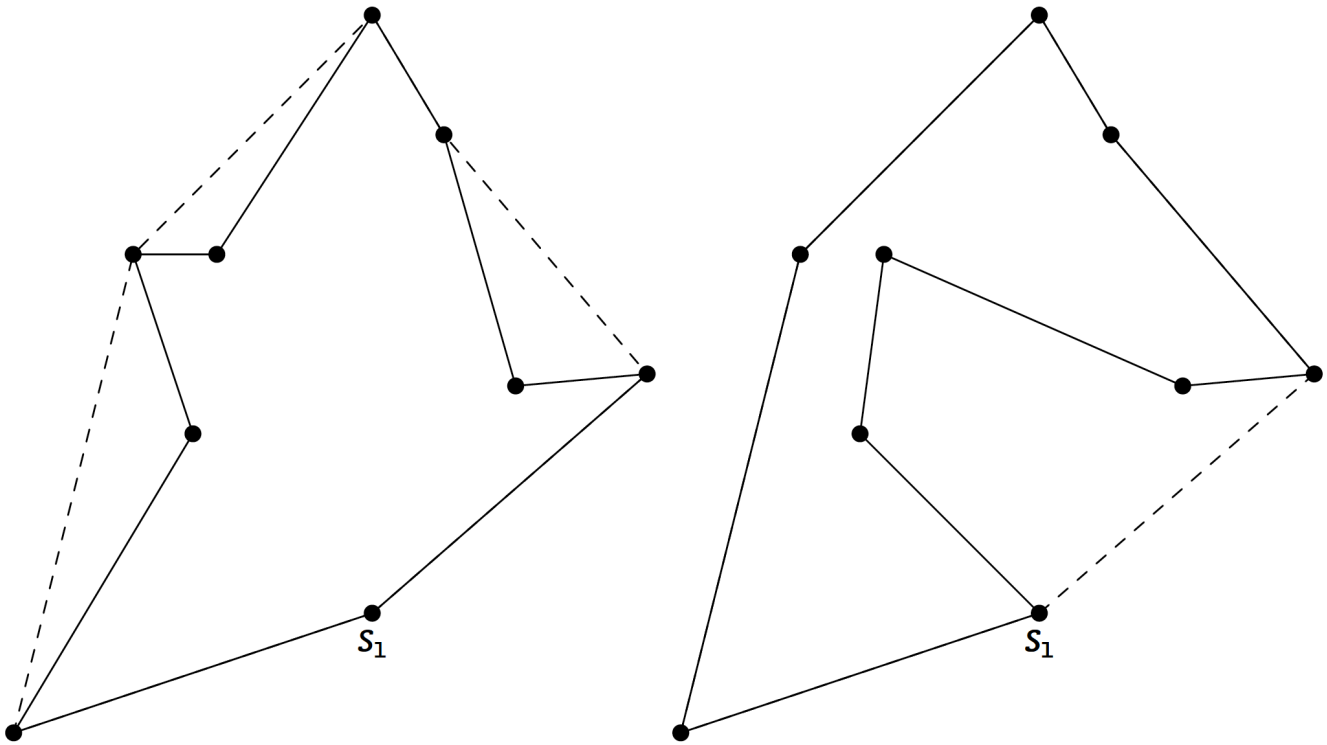
Convex Hull



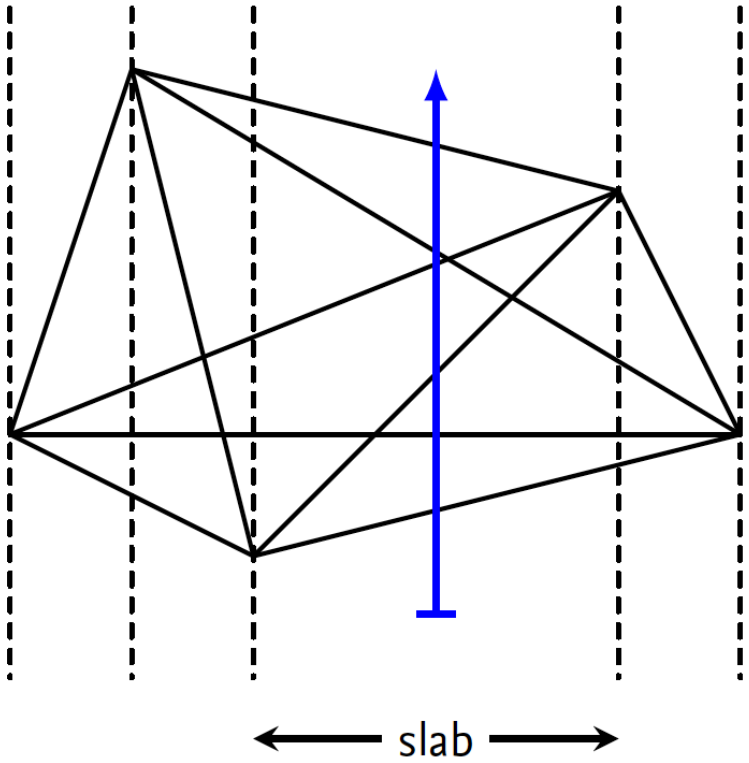
Greedy Heuristic



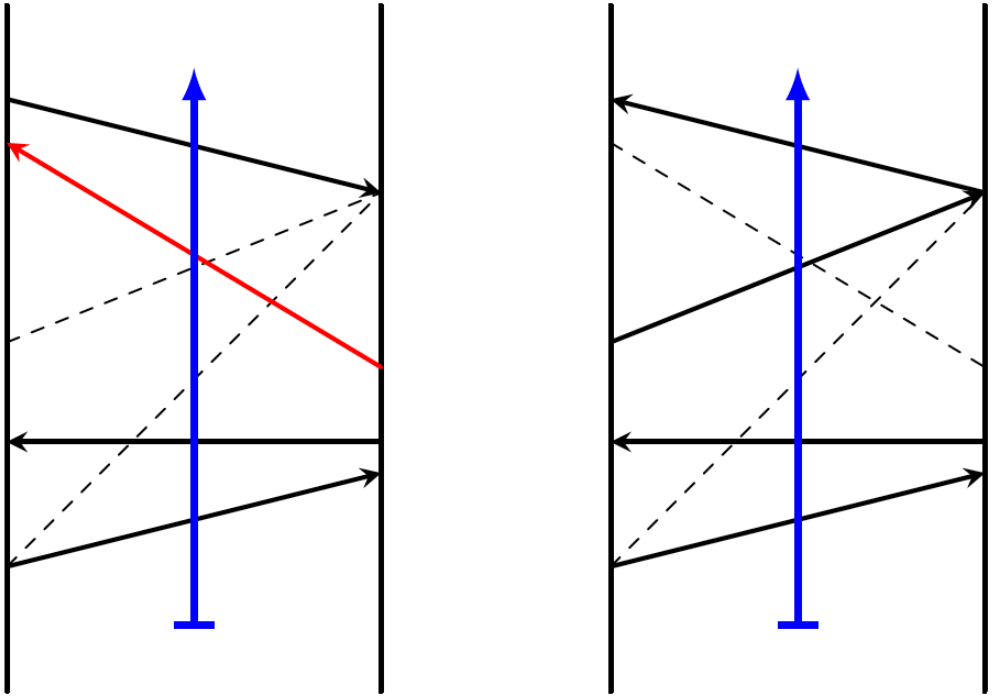
Max-Area Approximation



Slabs



Slabs



Edge-based IP

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

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

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

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

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

Edge-based IP

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

$$\forall \emptyset \neq D \subsetneq S: \qquad \sum_{(k,l)\in\delta^-(D)} z_{kl} \geq 1$$
$$\sum_{(k,l)\in\delta^+(D)} z_{kl} \geq 1$$

$$\forall e \in E^+ \dot\cup E^-: \qquad e \in \{0, 1\}$$

Triangulation IP

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

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

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

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

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

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