Antoniadis, Antonios; De Berg, Mark; Kisfaludi-Bak, Sándor; Skarlatos, Antonis

Computing Smallest Convex Intersecting Polygons

Published in:
30th Annual European Symposium on Algorithms, ESA 2022

DOI:
10.4230/LIPIcs.ESA.2022.9

Published: 01/09/2022

Document Version
Publisher's PDF, also known as Version of record

Computing Smallest Convex Intersecting Polygons

Antonios Antoniadis
Department for Applied Mathematics, University of Twente, Enschede, The Netherlands

Mark de Berg
Department of Mathematics and Computer Science, TU Eindhoven, The Netherlands

Sándor Kisfaludi-Bak
Department of Computer Science, Aalto University, Espoo, Finland

Antonis Skarlatos
Department of Computer Science, Universiät Salzburg, Austria

Abstract

A polygon C is an *intersecting polygon* for a set $\mathcal{O}$ of objects in $\mathbb{R}^2$ if C intersects each object in $\mathcal{O}$, where the polygon includes its interior. We study the problem of computing the minimum-perimeter intersecting polygon and the minimum-area convex intersecting polygon for a given set $\mathcal{O}$ of objects. We present an FPTAS for both problems for the case where $\mathcal{O}$ is a set of possibly intersecting convex polygons in the plane of total complexity n.

Furthermore, we present an exact polynomial-time algorithm for the minimum-perimeter intersecting polygon for the case where $\mathcal{O}$ is a set of n possibly intersecting segments in the plane. So far, polynomial-time exact algorithms were only known for the minimum perimeter intersecting polygon of lines or of disjoint segments.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms

Keywords and phrases convex hull, imprecise points, computational geometry

Digital Object Identifier 10.4230/LIPIcs.ESA.2022.9

Related Version *Full Version*: `https://arxiv.org/abs/2208.07567`

Funding Mark de Berg is supported by the Dutch Research Council (NWO) through Gravitation-grant NETWORKS-024.002.003.

Antonis Skarlatos: Part of the work was done during an internship at the Max Planck Institute for Informatics in Saarbrücken, Germany.

1 Introduction

Convex hulls are among the most fundamental objects studied in computational geometry. In fact, the problem of designing efficient algorithms to compute the convex hull of a planar point set $\mathcal{O}$ – the smallest convex set containing $\mathcal{O}$ – is one of the problems that started the field [5, 11]. Since the early days, the problem has been studied extensively, resulting in practical and provably efficient algorithms, in the plane as well as in higher dimensions; see the survey by Seidel [13, Chapter 26] for an overview.

A natural generalization is to consider convex hulls for a collection $\mathcal{O}$ of geometric objects (instead of points) in $\mathbb{R}^2$. Note that the convex hull of a set of polygonal objects is the same as the convex hull of the vertices of the objects. Hence, such convex hulls can be computed using algorithms for computing the convex hull of a point set. A different generalization, which leads to more challenging algorithmic questions, is to consider the smallest convex set that *intersects* all objects in $\mathcal{O}$. Thus, instead of requiring the convex set to fully contain each object from $\mathcal{O}$, we only require that it has a non-empty intersection with each object.

Notice that in case of points, the "smallest" set is well-defined: if convex sets C_1 and C_2 both contain a point set $\mathcal{O}$, then $C_1 \cap C_2$ also contains $\mathcal{O}$. Hence, the convex hull of a point set $\mathcal{O}$ can be defined as the intersection of all convex sets containing $\mathcal{O}$. When $\mathcal{O}$ consists of objects, however, this is no longer true, and the term "smallest" is ambiguous. In the present

paper we consider two variants: given a set $\mathcal{O}$ of possibly intersecting convex polygons in $\mathbb{R}^2$ of total complexity n, find a convex set of minimum perimeter that intersects all objects in $\mathcal{O}$, or a convex set of minimum area that intersects all objects in $\mathcal{O}$.

Observe that a minimum-perimeter connected intersecting set C for $\mathcal{O}$ must be a convex polygon. To see this, observe that for any object $o \in \mathcal{O}$ we can select a point $p_o \in o \cap C$, and take the convex hull of these points; the result is a feasible convex polygon whose perimeter is no longer than that of C. Thus the convexity of the solution could be omitted from the problem statement. This contrasts with the minimum-area problem, where there is always an intersecting polygon of zero area, namely, a tree. The convexity requirement is therefore essential in the problem statement. Note that it is still true that the minimum-area convex intersecting set is a polygon: given a convex solution, we can again take the convex hull of the points p_o and get a feasible solution whose area is not greater than the area of the initial convex solution. We also remark that the two problems typically have different optima. If $\mathcal{O}$ consists of the three edges of an equilateral triangle, then the minimum-area solution is a line segment (that is, a degenerate polygon of zero area), whereas the minimum-perimeter solution is the triangle whose vertices are the midpoints of the edges.

The problem of computing minimum-area or minimum-perimeter convex intersecting polygons, as well as several related problems, have already been studied. Dumitrescu and Jiang [4] considered the minimum-perimeter intersecting polygon problem. They gave a constant-factor approximation algorithm as well as a PTAS for the case when the objects in $\mathcal{O}$ are segments or convex polygons. They achieved a running time of $n^{O(1)}/\varepsilon + 2^{O(1/\varepsilon^{2/3})}n$. They also prove that computing a minimum-perimeter intersecting polygon for a set $\mathcal{O}$ of non-convex polygons (or polygonal chains) is NP-hard. For convex input objects, however, the hardness proof fails. Hence, Dumitrescu and Jiang ask the following question.

Question 1. Is the problem of computing a minimum-perimeter intersecting polygon of a set of segments NP-hard?

In case of *disjoint* segments, a minimum-perimeter intersecting polygon can be found in polynomial time [6, 7], but for intersecting segments the question is still open.

The problem of computing smallest intersecting polygons for a set $\mathcal{O}$ of objects has also been studied in works on *imprecise points*. Now the input is a set of points, but the the exact locations of the points are unknown. Instead, for each point one is given a region where the point can lie. One can then ask questions such as: what is the largest possible convex hull of the imprecise points? And what is the smallest possible convex hull? If we consider the objects in our input set $\mathcal{O}$ as the regions for the imprecise points, then the latter question is exactly the same as our problem of finding smallest intersecting convex sets. In this setup both the minimum-perimeter and minimum-area problem have been considered, for sets $\mathcal{O}$ consisting of convex regions of total complexity n. There are exact polynomial-time algorithms for minimum (and maximum) perimeter and area, for the special case where $\mathcal{O}$ consists of horizontal line segments or axis-parallel squares [10]. Surprisingly, some of these problems are NP-hard, such as the *maximum*-area/perimeter problems for segments. This gave rise to the study of approximation algorithms and approximation schemes [9].

In some cases, the minimum-perimeter problem can be phrased as a travelling salesman problem with neighborhoods (TSPN). Here the goal is to find the shortest closed curve intersecting all objects from the given set $\mathcal{O}$. In general, an optimal TSPN tour need not be convex, but one can show that in the case of lines or rays, an optimal tour is always convex: if a convex polygon intersects a line (or a ray) then its boundary intersects the line (resp. the ray). Therefore, computing a minimum-perimeter intersecting polygon of lines (or rays) is the same problem as TSPN with line neighborhoods (resp. ray neighborhoods). TSPN of

lines in $\mathbb{R}^2$ admits a polynomial-time algorithm [2]. In higher dimensions, TSPN has a PTAS for hyperplane neighborhoods [1], but notice that this is not the natural generalization of the minimum-intersecting polygon problem. Tan [12] proposed an exact algorithm for TSPN of rays in $\mathbb{R}^2$, but there seems to be an error in the argument; see the full version for details. At the time of writing this article, we believe that a polynomial-time algorithm for TSPN of rays is not known, but there is a constant-factor approximation algorithm due to Dumitrescu [3], as well as a PTAS [4].

Our results. In order to resolve Question 1, we first need to establish a good structural understanding and a dynamic programming algorithm. It turns out that the algorithm can also be used for approximation. We give dynamic-programming-based approximation schemes for the minimum-perimeter and minimum-area convex intersecting polygon problems. Our first algorithm is a fully polynomial time approximation scheme (FPTAS) for the minimum-perimeter problem of arbitrary convex objects of total complexity n.

▶ **Theorem 1.** *Let $\mathcal{O}$ be a set of convex polygons of total complexity n in $\mathbb{R}^2$ and let* OPT *be the minimum perimeter of an intersecting convex polygon for $\mathcal{O}$. For any given $\varepsilon > 0$, we can compute an intersecting polygon for $\mathcal{O}$ whose perimeter is at most $(1+\varepsilon)\cdot$ OPT, in $O(n^{2.373}/\varepsilon + n/\varepsilon^8)$ time.*

This is a vast improvement over the PTAS given by Dumitrescu and Jiang [4], as the dependence on $1/\varepsilon$ is only polynomial in our algorithm. Our approximation algorithms work in a word-RAM model, where input polygons are defined by the coordinates of their vertices, and where each coordinate is a word of $O(\log n)$ bits.

We also get a similar approximation scheme for the minimum area problem, albeit with a slower running time. Here we rely more strongly on the fact that the objects of $\mathcal{O}$ are convex *polygons*, and an extension to (for example) disks is an interesting open question. The full version details how the minimum-perimeter FPTAS needs to be adapted to the minimum-area setting.

▶ **Theorem 2.** *Let $\mathcal{O}$ be a set of convex polygons of total complexity n in $\mathbb{R}^2$ and let* OPT *be the minimum area of an intersecting convex polygon for $\mathcal{O}$. For any given $\varepsilon > 0$, we can compute a convex intersecting polygon for $\mathcal{O}$ whose area is at most $(1+\varepsilon)\cdot$ OPT, in $O(n^{17}\log(1/\varepsilon) + n^{11}/\varepsilon^{24})$ time.*

We remark that both Theorem 1 and Theorem 2 work if the input has *polytopes* instead of polygons, that is, when each object is the intersection of some half-planes.

While the dynamic programming algorithm developed above is crucial to get an exact algorithm, we are still several steps from being able to resolve Question 1. The main challenge here is that the vertices of the optimum intersecting polygon can be located at arbitrary boundary points in $\mathcal{O}$, and there is no known way to discretize the problem. We introduce a subroutine that uses an algorithm of Dror et al. [2] to compute parts of the minimum-perimeter intersecting polygon that contain no vertices of input objects. We are able to achieve a polynomial-time algorithm (on a real-RAM machine) for the minimum perimeter intersecting polygon problem only when the objects are line segments.

▶ **Theorem 3.** *Let $\mathcal{O}$ be a set of n line segments in the plane. Then we can compute a minimum-perimeter intersecting polygon for $\mathcal{O}$ in $O(n^9 \log n)$ time.*

If $\mathtt{P} \neq \mathtt{NP}$, then this gives a direct negative answer to Question 1. The theorem also extends to the case of rays (this is the scenario studied by Tan [12]; see the discussion in the full version.

Our techniques. Our approximation algorithms both compute an approximate solution whose vertices are from some fine grid. To determine a suitable grid resolution, we need to be able to compute lower bounds on OPT, which is non-trivial. It is also non-trivial to know where to place the grid, such that it is guaranteed to contain an approximate solution. The problem is that our lower bound gives us the location of a solution that is a constant-factor approximation, but this can be far from the location of a $(1+\varepsilon)$-approximation. Hence, for the minimum-area problem we generate a collection of grids, one of which is guaranteed to contain a $(1+\varepsilon)$-approximate solution. Finally, we face some further difficulties since a square grid may be insufficient: the optimum intersecting polygon may be extremely (exponentially) thin and long, and of area close to zero. In such cases there is no square grid of polynomial size that would contain a good solution. These problems are resolved in Section 2.

Section 3 presents our dynamic programming algorithm for minimum perimeter. In the dynamic programming the main technical difficulty lies in the fact that it is not clear what subset of objects should be visited in each subproblem. A portion of the optimum's boundary could in principle be tasked with intersecting an arbitrary subset of $\mathcal{O}$, while some of the objects in $\mathcal{O}$ need not be intersected by the optimum boundary and will simply be covered by the interior of the optimum intersecting polygon: a naïve approach therefore would not yield a polynomial-time algorithm. Our carefully designed subproblems have a clear corresponding set of objects to "visit", using orderings of certain tangents of input objects for this purpose. The minimum area problem uses a similar dynamic program, see the full version for its details.

Finally, in order to present our exact algorithm in Section 4, we need to modify our dynamic program to deal with subproblems where the vertices of a convex chain do not come from a discretized set. In such cases, we have to find the order in which the objects of $\mathcal{O}$ are visited by the chain. We are able to prove a specific ordering only in the case when the objects are line segments. The order then allows us to invoke the algorithm of Dror et al. [2] in a black-box manner.

2 Locating an optimal solution

The algorithms to be presented in subsequent sections need to approximately know the size and location of a smallest intersecting polygon. We use an algorithm from [4] to locate the minimum-perimeter intersecting polygon. With respect to the minimum-area intersecting polygon we prove that either there is a solution with a constant number of vertices (that can be computed with a different algorithm), or it is sufficient to consider polygons whose vertices are from a grid which comes from a polynomial collection of different grids.

Locating the minimum-perimeter optimum. For the minimum-perimeter intersecting polygon of a set $\mathcal{O}$ of convex objects, Dumitrescu and Jiang [4] present an algorithm $A1$ that, for a given $\varepsilon_1 > 0$, outputs a rectangle R intersecting all input objects $\mathcal{O}$ and with perimeter at most $\frac{4}{\pi}(1+\varepsilon_1)\text{OPT}$. At a high level, $A1$ guesses an orientation of the rectangle among $\lceil \frac{\pi}{4\varepsilon_1} \rceil$ many discrete orientations and then uses a linear program to identify the smallest perimeter rectangle of that orientation that intersects $\mathcal{O}$. In [4] it is described how Algorithm $A1$ is used to locate an optimal solution if the input objects are convex polygons. In particular, for any $\varepsilon > 0$ running $A1$ with $\varepsilon_1 = \frac{\varepsilon}{2+\varepsilon}$ gives a rectangle R. Let σ be the square that is concentric and parallel to R and has a side length of $3 \cdot \text{per}(R)$. Then the following holds.

▶ **Lemma 4** (Lemma 3 in [4]). *Suppose that* $\text{per}(R) \geq (1+\varepsilon)\text{OPT}$. *Then there is an optimum polygon* C_{opt} *that is covered by* σ.

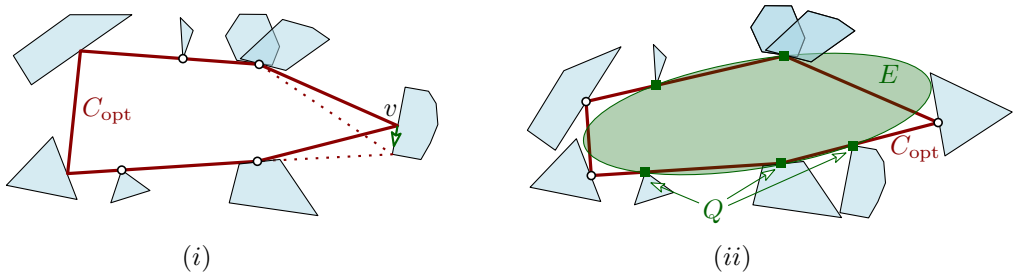

Figure 1 (i) If v and adjacent sides of C_{opt} are disjoint from X, then we can slide v without increasing the area of C_{opt}. (ii) The points of Q (green) and their circumscribed ellipse E.

Algorithm $A1$ needs to solve $O(1/\varepsilon)$ many linear programs with $O(n)$ variables and $O(n)$ constraints each. Thus R and σ can be found in $O(T_{\text{LP}}(n)/\varepsilon)$ time, where $O(T_{\text{LP}}(n)$ is the running time of an LP solver with $O(n)$ variables and $O(n)$ constraints. The state-of-the-art LP solver by Jiang et al. [8] achieves a running time better than $O(n^{2.373})$. Lemma 4 directly implies that if $\text{per}(R) \geq (1+\varepsilon)\text{per}(C_{\text{opt}})$, then

$$\text{diam}(C_{\text{opt}}) \leq \text{diam}(\sigma) = 3\sqrt{2}\text{per}(R) \leq 3\sqrt{2}\frac{4}{\pi}(1+\varepsilon)\text{per}(C_{\text{opt}}) = O(\text{diam}(C_{\text{opt}})). \tag{1}$$

The shape and location of the minimum-area optimum. For the rest of this section, let X denote the set of vertices in the planar arrangement given by $\mathcal{O}$.

▶ **Lemma 5.** *Let C_{opt} be a minimum-area intersecting polygon for the input $\mathcal{O}$ that has the minimum number of vertices, and among such polygons has the maximum number of points from X on its boundary. Then for any vertex v of C_{opt} that is not in X, the relative interior of at least one side of C_{opt} adjacent to v contains a point of X.*

Proof. Suppose for a contradiction that $v \notin X$ and that the relative interior of the sides in C_{opt} adjacent to v are disjoint from X. Observe that v must be on the boundary of an input object, so it is in the relative interior of an edge e of an input object, see Fig. 1(i). Then there exists a vector parallel to e along which we can move v while fixing its neighboring vertices in C_{opt}, without increasing the area of C_{opt}. This movement can be continued until we hit a point in X, or the angle of the polygon becomes π at v_1 or v_2. As a result, we end up with a feasible polygon S whose area is no greater than that of C_{opt}, and it has one less vertex or at least one more point of X on its boundary. This contradicts the properties of C_{opt}. ◀

The following lemma is more involved. Its full proof can be found in the full version.

▶ **Lemma 6.** *For any given set of input polygons $\mathcal{O}$ and $0 < \varepsilon < 1$ there is an intersecting polygon C of area $(1+\varepsilon)$OPT which either has at most 8 vertices, or its vertices are in a rectangular grid G of size $O(1/\varepsilon^3) \times O(1/\varepsilon^3)$ where G belongs to a collection $\mathcal{G}$ of grids that can be generated in polynomial time.*

Proof sketch. Let $Q = X \cap \partial C_{\text{opt}}$. By Lemma 5 we can show that $|Q| \geqslant \lceil |V(C_{\text{opt}})|/2 \rceil$, so if $|V| \geqslant 9$, then $|Q| \geqslant 4$. We guess the circumscribed ellipse of Q, and use an affinity on the entire instance and C_{opt} which transforms the ellipse into a circle E. Wlog. we assume that E is the unit circle centered at the origin. It is known that the disk with boundary $\frac{1}{2}E$ is covered by $\text{conv}(Q)$ and thus by C_{opt}.

Consider the division of ∂C_{opt} defined by the points of Q. A *section* of ∂C_{opt} between two consecutive points of Q is a *spike* if it has a vertex $v \in V(C_{\text{opt}})$ that is at distance $\Omega(1/\varepsilon)$ from the origin. One can show that if such a spike exists, then $\text{area}(C_{\text{opt}})$ is $\Omega(1/\varepsilon)$.

On the other hand, we can show that C_{opt} cannot have more than 2 spikes. We then claim that if C_{opt} has a spike, then there is a polygon on at most 8 vertices that has area at most $(1+\varepsilon)\text{area}(C_{\text{opt}})$. This is because C_{opt} can be covered by the intersection of the "cones" defined by the spike(s), which defines a polygon C. We can show that the extra area in the intersection of the (at most two) spike cones is $O(1)$, thus $\text{area}(C) \leqslant (1+\varepsilon)\text{area}(C_{\text{opt}})$, and C has at most 8 vertices, as desired.

If C_{opt} has no spikes, then all of its vertices are within distance $O(1/\varepsilon)$ from the origin. We define a fine grid G (that depends on the guessed ellipse E) in the radius $O(1/\varepsilon)$ square around the origin, and use standard arguments to show that the minimum area intersecting polygon whose vertices are from G has area at most $(1+\varepsilon)\text{OPT}$. We can therefore return all the grids G for each guess of the circumscribed ellipse of Q. ◀

3 An FPTAS for the minimum-perimeter problem of convex objects in the plane

Let $\mathcal{O}$ be a set of n convex objects in the plane for which we want to compute a minimum-perimeter convex intersecting polygon. We assume that $\mathcal{O}$ cannot be be stabbed by a single point – this is easy to test without increasing the total running time. Since a minimum-perimeter intersecting polygon is necessarily convex, we will from now on drop the adjective "convex" from our terminology. We do this even when referring to convex intersecting polygons that are not necessarily of minimum perimeter.

In the previous section we have seen that for any $\varepsilon > 0$, we can find a feasible rectangle R and a square σ with the following property: Either $\text{per}(R) \leq (1+\varepsilon)\text{OPT}$, or $C_{\text{opt}} \subseteq \sigma$ with $\text{diam}(C_{\text{opt}}) = \Omega(1)\text{diam}(\sigma)$. Next we describe an algorithm that, given a parameter $\varepsilon > 0$ and a corresponding square σ, outputs an intersecting polygon $C^* \subseteq \sigma$ for $\mathcal{O}$ such that if $\text{per}(R) \geq (1+\varepsilon)\text{per}(C_{\text{opt}})$ then $\text{per}(C^*) \leqslant (1+\varepsilon)\text{OPT}$, where $\text{OPT} = \text{per}(C_{\text{opt}})$ (cf. Lemma 4 and Equation 1). Finally, we output either R or C^*, whichever has smaller perimeter.

Our algorithm starts by partitioning σ into a regular grid $G(\sigma)$ of $O(1/\varepsilon^2)$ cells of edge length at most $(\varepsilon/8)\cdot\text{OPT}$. We say that a convex polygon is a *grid polygon* if its vertices are grid points from $G(\sigma)$. The following observation is standard, but a full proof can be found in the full version.

▶ **Observation 7.** *Suppose σ contains an optimal solution C_{opt}. Let $C(\sigma)$ be a minimum-perimeter grid polygon that is an intersecting polygon for $\mathcal{O}$. Then* $\text{per}(C(\sigma)) \leqslant (1+\varepsilon)\cdot\text{OPT}$.

Next we describe an algorithm to compute a minimum-perimeter grid polygon $C(\sigma)$ that is an intersecting polygon for $\mathcal{O}$.

First, we "guess" the lexicographically smallest vertex v_{bot} of $C(\sigma)$, see Figure 2(i). We can guess v_{bot} in $O(1/\varepsilon^2)$ different ways. For each possible guess we will find the best solution (if it exists), and then we report the best solution found over all guesses.

Now consider a fixed guess for the lexicographically smallest vertex v_{bot} of $C(\sigma)$. With a slight abuse of notation we will use $C(\sigma)$ to denote a minimum-perimeter grid polygon that is an intersecting polygon of $\mathcal{O}$ and that has v_{bot} as lexicographically smallest vertex. (If the polygon $C(\sigma)$ does not exist, the algorithm described below will detect this.) We will compute $C(\sigma)$ by dynamic programming.

The vertices of $C(\sigma)$ are grid points in the region $h^+ \setminus \rho_0$, where h^+ is the closed half-plane above the horizontal line through v_{bot} and ρ_0 is the horizontal ray emanating from v_{bot} and pointing to the left. Let V be the set of such grid points, excluding v_{bot}. We first order the points from V in angular order around v_{bot}. More precisely, for a point $v \in V$, let $\phi(v)$ denote

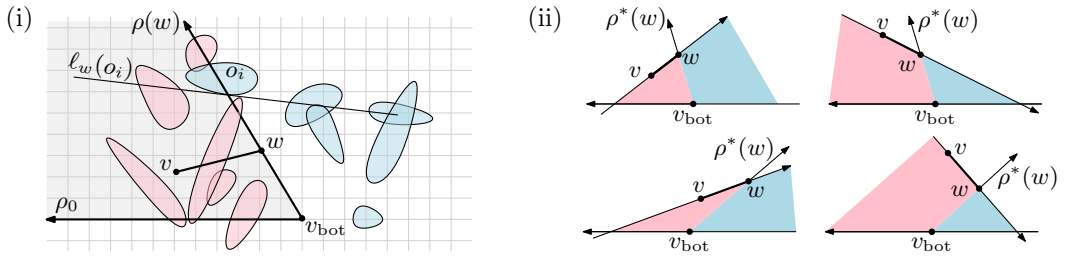

Figure 2 (i) The wedge defined by ρ_0 and $\rho(w)$ is shown in light grey. Objects in $\mathcal{O}(v, w)$ are red, objects in $\mathcal{O} \setminus \mathcal{O}(v, w)$ are blue. (ii) The partial solution $C(\Gamma)$ must be contained in the red region, while $C \setminus C(\Gamma)$ must lie in the blue region. There are four situations, depending on whether the angle between ρ_0 and $\rho(w)$ is acute or not, and whether the line containing vw intersects ρ_0 or not.

the angle over which we have to rotate ρ_0 in clockwise direction until we hit v. For two points $v, w \in V$, we write $v \prec w$ if $\phi(v) < \phi(w)$. Let $V^+ := V \cup \{v_{\text{bot}}, \overline{v}_{\text{bot}}\}$, where $\overline{v}_{\text{bot}}$ is a copy of v_{bot}, and define $v_{\text{bot}} \prec v$ and $v \prec \overline{v}_{\text{bot}}$ for all $v \in V$. The copy $\overline{v}_{\text{bot}}$ serves to distinguish the start and the end vertex of the clockwise circular sequence of vertices of $C(\sigma)$. Note that if $v_{\text{bot}}, v_1, \ldots, v_k, \overline{v}_{\text{bot}}$ denotes this circular sequence then $v_{\text{bot}} \prec v_1 \prec \cdots \prec v_k \prec \overline{v}_{\text{bot}}$ (since $C(\sigma)$ will never have two vertices that make the same angle with v_{bot}).

We now describe our dynamic-programming algorithm. Consider a polyline from v_{bot} to some point $v \in V$. We say that this polyline is a *convex chain* if, together with the line segment vv_{bot}, it forms a convex polygon. We denote the convex polygon induced by such a chain Γ by $C(\Gamma)$. The problem we now wish to solve is as follows:

> Compute a minimum-length convex chain Γ^* from v_{bot} to $\overline{v}_{\text{bot}}$ such that $C(\Gamma^*)$ is an intersecting polygon for $\mathcal{O}$.

Our dynamic-programming algorithm uses the partial order $\prec$ defined above. We now want to define a subproblem for each point $v \in V^+$, which is to find the "best" chain Γ_v ending at v. For this to work, we need to know which objects from $\mathcal{O}$ should be covered by the partial solution $C(\Gamma_v)$. This is difficult, however, because objects that intersect the ray from v_{bot} and going through v could either be intersected by $C(\Gamma_v)$ or by the part of the solution that comes after v. To overcome this problem we let the subproblems be defined by the last edge on the chain, instead of by the last vertex. This way we can decide which objects should be covered by a partial solution, as explained next.

Consider a convex chain from v_{bot} to a point $w \in V^+$ whose last edge is vw. Let $\rho(w)$ be the ray emanating from v_{bot} in the direction of w, and let $\rho^*(w)$ be the part of the ray starting at w. For $w = \overline{v}_{\text{bot}}$ we define $\rho(w)$ to be the horizontal ray emanating from v_{bot} and going to the right, and we define $\rho^*(w) = \rho(w)$. For an object $o_i \in \mathcal{O}$ that intersects $\rho^*(w)$, let $\ell_w(o_i)$ be a line that is tangent to o_i at the first intersection point of $\rho^*(w)$ with o_i. We now define the set $\mathcal{O}(v, w)$ to be the subset of objects $o_i \in \mathcal{O}$ such that one of the following conditions is satisfied; see also Figure 2(i).

- **(i)** o_i intersects the wedge defined by ρ_0 and $\rho(w)$, but not $\rho(w)$ itself; or
- **(ii)** o_i intersects $v_{\text{bot}}w$; or
- **(iii)** o_i intersects $\rho^*(w)$ but not $v_{\text{bot}}w$, and the tangent line $\ell_w(o_i)$ intersects the half-line containing vw and ending at w.

The next lemma shows that we can use the sets $\mathcal{O}(v, w)$ to define our subproblems.

▶ **Lemma 8.** *Let C be any convex polygon that is an intersecting polygon for $\mathcal{O}$ and that has v_{bot} as lexicographically smallest vertex and vw as one of its edges. Let Γ_w be the part of ∂C from v_{bot} to w in clockwise direction. Then all objects in $\mathcal{O}(v, w)$ intersect $C(\Gamma_w)$ and all objects in $\mathcal{O} \setminus \mathcal{O}(v, w)$ intersect $C \setminus C(\Gamma_w)$.*

We can now state our dynamic program. To this end we define, for two points $v, w \in V^+$ with $v \prec w$, a table entry $A[v, w]$ as follows.

$A[v, w] :=$ the minimum length of a convex chain Γ from v_{bot} to w whose last edge is vw and such that all objects in $\mathcal{O}(v, w)$ intersect $C(\Gamma)$,

where the minimum is ∞ if no such chain exists. Lemma 8 implies the following.

▶ **Observation 9.** *Let Γ^* be a shortest convex chain from v_{bot} to $\overline{v}_{\text{bot}}$ such that $C(\Gamma^*)$ is an intersecting polygon for $\mathcal{O}$. Then* $\text{length}(\Gamma^*) = \min\{A[v, \overline{v}_{\text{bot}}] : v \in V$ *and* $\mathcal{O}(v, \overline{v}_{\text{bot}}) = \mathcal{O}\}$.

Hence, if we can compute all table entries $A[v, w]$ then we have indeed solved our problem. (The lemma only tells us something about the value of an optimal solution, but given the table entries $A[v, w]$ we can compute the solution itself in a standard way.)

The entries $A[v, w]$ can be computed using the following lemma. Define $\Delta(v_{\text{bot}}, v, w)$ to be the triangle with vertices v_{bot}, v, w.

▶ **Lemma 10.** *Let $v, w \in V^+$ with $v \prec w$. Let $V(v, w)$ be the set of all points $u \in V \cup \{v_{\text{bot}}\}$ with $u \prec v$ such that u lies below the line $\ell(v, w)$ through v and w and such that all objects in $\mathcal{O}(v, w) \setminus \mathcal{O}(u, v)$ intersect $\Delta(v_{\text{bot}}, v, w)$. Then*

$$A[v, w] = \begin{cases} |v_{\text{bot}}w| & \text{if } v = v_{\text{bot}} \text{ and all objects in } \mathcal{O}(v, w) \text{ intersect } v_{\text{bot}}w \\ \infty & \text{if } v = v_{\text{bot}} \text{ and not all objects in } \mathcal{O}(v, w) \text{ intersect } v_{\text{bot}}w \\ |vw| + \min\limits_{u \in V(v,w)} A[u, v] & \text{otherwise} \end{cases}$$

Putting everything together, we can finish the proof of Theorem 1.

Proof of Theorem 1. We first use algorithm $A1$ from [4] to compute the rectangle R and the square σ, which as discussed can be done in $O(n^{2.373}/\varepsilon)$ time. For a square σ we guess the vertex v_{bot} in $O(1/\varepsilon^2)$ different ways.

For each guess we run the dynamic-programming algorithm described above. There are $O(1/\varepsilon^4)$ entries $A[u, v]$ in the dynamic-programming table. The most time-consuming computation of a table entry is in the third case of Lemma 10. Here we need to compute the set $\mathcal{O}(v, w)$, which can be done in $O(n)$ time by checking every $o_i \in \mathcal{O}$. For each of the $O(1/\varepsilon^2)$ points with $u \prec v$ such that u lies below the line $\ell(v, w)$ we then check in $O(n)$ time if all objects in $\mathcal{O}(v, w) \setminus \mathcal{O}(u, v)$ intersect $\Delta(v_{\text{bot}}, v, w)$, so that we can compute $A[v, w]$. Hence, computing $A[v, w]$ takes $O(n/\varepsilon^2)$ time, which implies that the whole dynamic program needs $O(n/\varepsilon^6)$ time.

Thus the algorithm takes $O(n^{2.373}/\varepsilon) + O(1/\varepsilon^2) \cdot O(n/\varepsilon^6) = O(n^{2.373}/\varepsilon + n/\varepsilon^8)$ time. ◀

▶ Remark 11. Although Theorem 1 is stated only for the case where $\mathcal{O}$ is a set of convex polygons, it is not too hard to extend it to other convex objects, for example disks: one just needs to replace the approximate rectangle-finding linear program of Dumitrescu and Jiang [4] with some other polynomial-time algorithm to find an (approximate) minimum perimeter intersecting rectangle in each of the $O(1/\varepsilon)$ orientations.

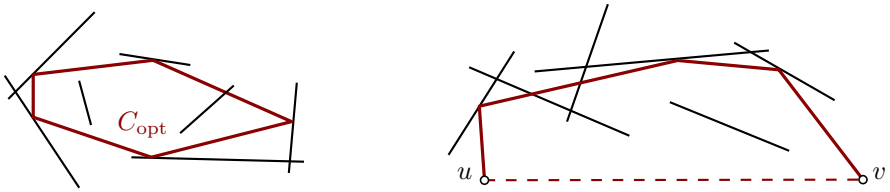

Figure 3 (i) Subroutine I computes a minimum perimeter intersecting polygon whose vertices are not segment endpoints. (ii) Subroutine II computes a polygon with fixed edge uv for some (sub)set of segments. With the exception of u and v, the polygon's vertices are not allowed to be segment endpoints.

4 An exact algorithm for the minimum-perimeter intersecting polygon of segments

We describe an exact algorithm to compute a minimum-perimeter intersecting object for a set $\mathcal{O}$ of line segments[1] in the plane. Consider an optimal solution C_{opt}, and let Y be the set of all endpoints of the segments in $\mathcal{O}$. The exact algorithm is based on two subroutines for the following two problems. As before, whenever we talk about intersecting polygons, we implicitly require them to be convex. Figure 3 illustrates the polygons computed by these subroutines.

- *Subroutine I*: If $\mathcal{O}$ admits a minimum-perimeter intersecting polygon with none of its vertices being in Y, then compute such a polygon. Otherwise compute a feasible intersecting polygon, or report $+\infty$.
- *Subroutine II*: Given two points $u, v \in \mathbb{R}^2$ and a subset $\mathcal{O}' \subseteq \mathcal{O}$, decide if $\mathcal{O}'$ admits a minimum-perimeter intersecting polygon that has uv as one of its edges and none of whose other vertices belongs to Y and, if so, compute a minimum-perimeter such intersecting polygon. If no such minimum-perimeter intersecting polygon exists, report $+\infty$. Note that we allow $u = v$, in which case the edge uv degenerates to a point.

In the full version we show:

▶ **Theorem 12.** *There exist exact algorithms for* Subroutine I *and* Subroutine II *that run in time* $O(n^6 \log n)$ *and* $O(n^3 \log n)$*, respectively.*

Setting the stage for the dynamic program

With Subroutine I available, it remains to find the minimum-perimeter intersecting polygon that has at least one vertex from Y. To this end we will develop an algorithm that, for a given point $p_{\text{bot}} \in Y$, finds a minimum-perimeter convex intersecting polygon that has p_{bot} as a vertex (if it exists). We will run this algorithm for each choice of $p_{\text{bot}} \in Y$.

Let $p_{\text{bot}} \in Y$ be given, and assume that $\mathcal{O}$ admits an intersecting polygon that has p_{bot} as a vertex (this can be tested in $O(n \log n)$ time). In contrast to Section 3, p_{bot} need not be the lexicographically smallest point and our algorithm therefore relies on guessing the

[1] Although we describe our algorithm for non-degenerate segments, all our arguments also work if some (or all) of the segments are in fact lines, rays, or even points.

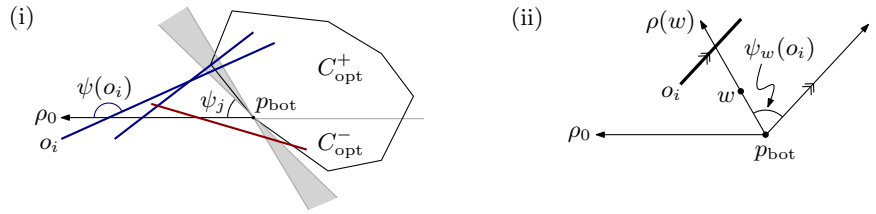

Figure 4 (i) The grey double wedge indicates the region containing the tangent at p_{bot}. (Note that then there must be more segments than the blue and red segments that are shown. In particular, there must be segments parallel to the lines delimiting the double wedge.) The blue segments are in $\mathcal{O}(\rho_0)^+$ so they must be intersected by C^+_{opt}, while the red segment is in $\mathcal{O}(\rho_0)^-$ and must be intersected by C^-_{opt}. (ii) The definition of $\psi_w(o_i)$.

orientation of a line tangent to C_{opt} at p_{bot}. More specifically, we are able to find a set $\Psi = \{\psi_1, \psi_2, \dots, \}$, of $O(n)$ possible angles with $\psi_j < \psi_{j+1}$ for all $0 \leq j \leq |\Psi|$ (where we set $\psi_0 := 0$ and $\psi_{|\Psi|+1} = \pi$) and reduce the problem to:

> Given a point p_{bot} and value j with $0 \leqslant j \leqslant |\Psi|$, find a minimum-perimeter intersecting polygon C_{opt} for $\mathcal{O}$ such that
> - p_{bot} is a vertex of C_{opt},
> - the horizontal ray ρ_0 going from p_{bot} to the left does not intersect C_{opt},
> - C_{opt} has a tangent line ℓ at p_{bot} such that the clockwise angle from ρ_0 to ℓ lies in the range $[\psi_j, \psi_{j+1}]$.

In the above, each angle ψ_j corresponds to the angle over which we have to rotate ρ_0 clockwise so that it becomes parallel with some segment $o_j \in \mathcal{O}$. An extensive description of this reduction can be found in the full version.

The dynamic program

We will now develop our dynamic program for the problem that we just stated, for a given point $p_{\text{bot}} \in Y$, a ray ρ_0, and range $[\psi_j, \psi_{j+1}]$; see Figure 4(i).

Let $\mathcal{O}(\rho_0) \subseteq \mathcal{O}$ denote the set of segments that intersect the ray ρ_0, and let ℓ_0 be the line containing ρ_0. The line ℓ_0 may split the optimal solution C_{opt} into two parts: a part C^+_{opt} above ℓ_0 and a part C^-_{opt} below ℓ_0. Let $\psi(o_i)$ denote the angle over which we have to rotate ρ_0 in clockwise direction until it becomes parallel to o_i. Since we have fixed the range of the tangent at p_{bot} to lie in the range $[\psi_j, \psi_{j+1}]$, we can split $\mathcal{O}(\rho_0)$ into two subsets, $\mathcal{O}(\rho_0)^+ := \{o_i \in \mathcal{O}(\rho_0) : \psi(o_i) \geqslant \psi_{j+1}\}$ and $\mathcal{O}(\rho_0)^- := \{o_i \in \mathcal{O}(\rho_0) : \psi(o_i) \leqslant \psi_j\}$.

Note that $\mathcal{O}(\rho_0) = \mathcal{O}(\rho_0)^+ \cup \mathcal{O}(\rho_0)^-$, because ψ_j and ψ_{j+1} are consecutive angles in Ψ. Because the orientation of the tangent at p_{bot} lies in the range $[\psi_j, \psi_{j+1}]$, we know that the segments in $\mathcal{O}(\rho_0)^+$ must be intersected by C^+_{opt}, while the segments in $\mathcal{O}(\rho_0)^-$ must be intersected by C^-_{opt}; see Figure 4(i). Intuitively, the segments in $\mathcal{O}(\rho_0)^+$ must be intersected by "the initial part" of C_{opt}, while the segments in $\mathcal{O}(\rho_0)^-$ are intersected by "the later part". We will use this when we define the subproblems in our dynamic program.

In Section 3 we defined subproblems for pairs of grid points v, w. The goal of such a subproblem was to find the minimum-length convex chain Γ such that $C(\Gamma)$ intersects a certain subset $\mathcal{O}(v, w)$ and whose last edge is vw. The fact that we knew the last edge vw was crucial to define the set $\mathcal{O}(v, w)$, since the slope of vw determined which objects should be intersected by $C(\Gamma)$. In the current setting this does not work: we could define a subproblem for pairs $v, w \in Y$, but "consecutive" vertices v, w from Y along Γ are now connected by a

polyline Z_{vw} whose inner vertices are disjoint from Y. The difficulty is that the polyline Z_{vw} depends on the segments that need to be intersected by $C(\Gamma)$. Hence, there is a cyclic dependency between the set of segments to be intersected by $C(\Gamma)$ (which depends on the slope of zw, where z is the vertex of Z_{vw} preceding w) and the vertex z (which depends on the segments that need to be intersected by $C(\Gamma)$). We overcome this problem as follows.

Similarly to the previous section, we call a polyline Γ from p_{bot} to some point $v \in Y$ a *convex chain* if, together with the line segment vp_{bot}, it forms a convex polygon. We denote this polygon by $C(\Gamma)$. Consider a convex chain Γ_w ending in a point $w \in Y$. Let $\rho(w)$ denote the ray from p_{bot} through w and let $\rho^*(w)$ be the part of this ray starting at w. Let $\mathcal{O}(w)$ be the set of input segments that intersect $\rho^*(w)$. Of those segments, $C(\Gamma_w)$ must intersect the ones such that the line $\ell(o_i)$ containing[2] the segment o_i intersects the half-line containing zw and ending at w, where z is the (unknown) vertex preceding w. For a segment $o_i \in \mathcal{O}(w)$, let $\psi_w(o_i)$ be the angle over which we have to rotate $\rho(w)$ in clockwise direction to make it parallel to o_i; see Figure 4(ii). Let $\Psi(w) = \{\psi_1, \psi_2, \ldots\}$, for all $1 \leqslant j < |\Psi(w)|$, be the sorted set of (distinct) angles $\psi_w(o_i)$ defined by the segments in $\mathcal{O}(w)$. For an index j with $1 \leqslant j \leqslant |\Psi(w)|$, define $\mathcal{O}(w, j) := \{o_i \in \mathcal{O}(w) : \psi_w(o_i) \leqslant \psi_j\}$, and define $\mathcal{O}(w, 0) = \varnothing$. We call $\mathcal{O}(w, j)$ a *prefix set*. The key observation is that $C(\Gamma_w)$ must intersect the segments from some prefix set $\mathcal{O}(w, j)$, where j depends on the unknown vertex z preceding w. So our dynamic program will try all possible prefix sets $\mathcal{O}(w, j)$, and make sure that subproblems are combined in a consistent manner.

We now have everything in place to describe our dynamic-programming table. It consists of entries $A[w, j]$, where w ranges over all points in Y, and j ranges over all values for which $\mathcal{O}(w, j)$ is defined. For convenience add two special entries, $A[p_{\text{bot}}, 0]$ and $A[\overline{p}_{\text{bot}}, 0]$; the former will serve as the base case, and the latter will contain (the value of) the final solution. Note that these are the only ones for p_{bot} and $\overline{p}_{\text{bot}}$, and that we have at most $|Y| \cdot n = O(n^2)$ entries. We define the set $\mathcal{O}^*(w, j)$ of segments to be covered in a subproblem.

> For a point $w \in Y$, the set $\mathcal{O}^*(w, j)$ consists of the segments $o_i \in \mathcal{O}$ that satisfy one of the following conditions:
>
> - **(i)** o_i intersects the clockwise wedge from ρ_0 to $\rho(w)$ – note that this wedge need not be convex – but not $\rho(w)$ itself, and $o_i \notin \mathcal{O}^-(\rho_0)$; or
> - **(ii)** o_i intersects $p_{\text{bot}}w$; or
> - **(iii)** $o_i \in \mathcal{O}(w, j)$.
>
> Furthermore, $\mathcal{O}^*(p_{\text{bot}}, 0) := \varnothing$ and $\mathcal{O}^*(\overline{p}_{\text{bot}}, 0) := \mathcal{O}$.

We would like now to define $A[w, j]$ to be the minimum length of a convex chain Γ from p_{bot} to w such that all objects in $\mathcal{O}^*(w, j)$ intersect $C(\Gamma)$. There is, however, a technicality to address: the minimum-perimeter polygon that intersects all segments from $\mathcal{O}$ need not be convex when we require it to have p_{bot} as a vertex. Such a non-convex polygon cannot be the final solution – if the optimum for a given choice of p_{bot} is non-convex, then p_{bot} was not the correct choice – but it makes a clean definition of our subproblems awkward. Therefore, instead of first defining the subproblems and then giving the recursive formula, we will immediately give the recursive formula and then prove that it computes what we want.

For two points $v, w \in Y^+$ (where $Y^+ = Y \cup \{\overline{p}_{\text{bot}}\}$) with $v \prec w$ and a set $\mathcal{O}' \subseteq \mathcal{O}$, let $L(v, w, \mathcal{O}')$ be the minimum length of a convex chain Γ from v to w such that the convex polygon defined by Γ and vw is an intersecting set for $\mathcal{O}'$ and all inner vertices of Γ are disjoint from Y. Recall that we can compute $L(v, w, \mathcal{O}')$ using subroutine II. As before, let $\Delta(p_{\text{bot}}, v, w)$ denote the triangle with vertices p_{bot}, v, w.

[2] Since the input objects are now segments, the tangent line $\ell_w(o_i)$ is just the line containing o_i.

▶ **Definition 13.** *Let $w \in Y^+$ and j be a value for which $\mathcal{O}[w,j]$ is defined. Thus $0 \leqslant j \leqslant |\Psi(w)|$, where we set $|\Psi(w)| := 0$ for $w \in \{p_{\text{bot}}, \overline{p}_{\text{bot}}\}$. For $v \prec w$ and $0 \leqslant j' \leqslant |\Psi(v)|$, let*

$$\mathcal{O}^*(w,j,v,j') := \mathcal{O}^*(w,j) \ \setminus \ \Big(\mathcal{O}^*(v,j') \cup \{o_i \in \mathcal{O} : o_i \text{ intersects } \Delta(p_{\text{bot}}, v, w)\}\Big)$$

and define

$$A[w,j] \ := \ \begin{cases} 0 & \text{if } w = p_{\text{bot}} \\ \min\limits_{\substack{v \prec w \\ 0 \leqslant j' \leqslant |\Psi(v)|}} L(v, w, \mathcal{O}^*(w,j,v,j')) + A[v,j'] & \text{otherwise.} \end{cases}$$

The next lemma implies that the table entry $A[\overline{p}_{\text{bot}}, 0]$ defined by this recursive formula gives us what we want. Part (a) implies that $A[\overline{p}_{\text{bot}}, 0]$ will never return a value that is too small, while part (b) implies that for the correct choice of p_{bot} and range of orientations for the tangent to C_{opt} at p_{bot}, the entry $A[\overline{p}_{\text{bot}}, 0]$ gives us (the value of) the optimal solution.

▶ **Lemma 14.** *Consider the table entry $A[\overline{p}_{\text{bot}}, 0]$ defined by Definition 13 for a given point p_{bot} and range $[\psi_i, \psi_{i+1}]$.*

(a) *There exists a convex intersecting polygon for $\mathcal{O}$ of perimeter at most $A[\overline{p}_{\text{bot}}, 0]$.*

(b) *If p_{bot} is a vertex of the minimum-perimeter convex intersecting polygon C_{opt} for $\mathcal{O}$, and ρ_0 does not intersect C_{opt}, and there is a tangent line ℓ at p_{bot} whose orientation is in the range $[\psi_i, \psi_{i+1}]$, then $\text{per}(C_{\text{opt}}) = A[\overline{p}_{\text{bot}}, 0]$.*

Putting everything together

Lemma 14 implies that after solving the dynamic programs for all choices of p_{bot} and the range $[\psi_i, \psi_{i+1}]$, we have found the minimum perimeter intersecting set for $\mathcal{O}$. (Computing the intersecting set itself, using the relevant dynamic-program table, is then routine.) This leads to the proof of Theorem 3.

Proof of Theorem 3. The number of dynamic programs solved is $O(|Y| \cdot n) = O(n^2)$. The dynamic-programming tables have $O(n^2)$ entries. Computing an entry takes $O(|Y| \cdot n) = O(n^2)$ calls to Subroutine II, at $O(n^3 \log n)$ time each. The dynamic programs thus take $O(n^2) \cdot O(n^2) \cdot O(n^2) \cdot O(n^3 \log n) = O(n^9 \log n)$ time. If the optimal solution does not go through any point of Y, then by Theorem 12 it will be found in $O(n^6 \log n)$ time. The optimum of these two algorithms is the global optimum. ◀

5 Conclusion

We gave fully polynomial time approximation schemes for the minimum perimeter and minimum area convex intersecting polygon problems for convex polygons. Additionally, we developed a polynomial-time algorithm for the minimum perimeter problem of segments.

It is likely that the running times of our algorithms can be improved further. One could also try to generalize the set of objects, for example, adapting the minimum area algorithm to arbitrary convex objects. We propose the following open questions for further study.

- Is there a polynomial-time exact algorithm for the minimum area convex intersecting polygon of segments?
- Is there a polynomial-time exact algorithm for minimum perimeter or minimum area convex intersecting polygon of convex polygons, or are these problems NP-hard?

- Is there a polynomial-time approximation scheme for the minimum volume or minimum surface area convex intersecting polytope of convex polytopes in $\mathbb{R}^3$? Can we at least approximate the diameter of the optimum solution to these problems?

It would be especially interesting to see an NP-hardness proof for minimum volume or surface area convex intersecting set of convex objects in higher dimensions.

References

1. Antonios Antoniadis, Krzysztof Fleszar, Ruben Hoeksma, and Kevin Schewior. A PTAS for Euclidean TSP with hyperplane neighborhoods. *ACM Trans. Algorithms*, 16(3):38:1–38:16, 2020.
2. Moshe Dror, Alon Efrat, Anna Lubiw, and Joseph SB Mitchell. Touring a sequence of polygons. In *STOC 2003: Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, pages 473–482, 2003.
3. Adrian Dumitrescu. The traveling salesman problem for lines and rays in the plane. *Discrete Mathematics, Algorithms and Applications*, 4(04):1250044, 2012.
4. Adrian Dumitrescu and Minghui Jiang. Minimum-perimeter intersecting polygons. *Algorithmica*, 63(3):602–615, 2012. doi:10.1007/s00453-011-9516-3.
5. Ray A. Jarvis. On the identification of the convex hull of a finite set of points in the plane. *Information processing letters*, 2(1):18–21, 1973.
6. Ahmad Javad, Ali Mohades, Mansoor Davoodi, and Farnaz Sheikhi. Convex hull of imprecise points modeled by segments in the plane, 2010.
7. Yiyang Jia and Bo Jiang. The minimum perimeter convex hull of a given set of disjoint segments. In *International Conference on Mechatronics and Intelligent Robotics*, pages 308–318. Springer, 2017.
8. Shunhua Jiang, Zhao Song, Omri Weinstein, and Hengjie Zhang. A faster algorithm for solving general lps. In *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing 2021*, pages 823–832. ACM, 2021. doi:10.1145/3406325.3451058.
9. Marc J. van Kreveld and Maarten Löffler. Approximating largest convex hulls for imprecise points. *J. Discrete Algorithms*, 6(4):583–594, 2008. doi:10.1016/j.jda.2008.04.002.
10. Maarten Löffler and Marc J. van Kreveld. Largest and smallest convex hulls for imprecise points. *Algorithmica*, 56(2):235–269, 2010. doi:10.1007/s00453-008-9174-2.
11. Franco P. Preparata and Se June Hong. Convex hulls of finite sets of points in two and three dimensions. *Communications of the ACM*, 20(2):87–93, 1977.
12. Xuehou Tan. The touring rays and related problems. *Theoretical Computer Science*, 2021.
13. Csaba D. Tóth, Joseph O'Rourke, and Jacob E Goodman. *Handbook of discrete and computational geometry*. CRC press, 2017.