


The Geodesic Edge Center of a Simple Polygon

Anna Lubiw  

David R. Cheriton School of Computer Science, University of Waterloo, Canada

Anurag Murty Naredla 

David R. Cheriton School of Computer Science, University of Waterloo, Canada

Institut für Informatik, University of Bonn, Bonn, Germany

Abstract

The *geodesic edge center* of a polygon is a point c inside the polygon that minimizes the maximum geodesic distance from c to any edge of the polygon, where *geodesic distance* is the shortest path distance inside the polygon. We give a linear-time algorithm to find a geodesic edge center of a simple polygon. This improves on the previous $O(n \log n)$ time algorithm by Lubiw and Naredla [European Symposium on Algorithms, 2021]. The algorithm builds on an algorithm to find the geodesic *vertex* center of a simple polygon due to Pollack, Sharir, and Rote [Discrete & Computational Geometry, 1989] and an improvement to linear time by Ahn, Barba, Bose, De Carufel, Korman, and Oh [Discrete & Computational Geometry, 2016].

The geodesic edge center can easily be found from the geodesic farthest-edge Voronoi diagram of the polygon. Finding that Voronoi diagram in linear time is an open question, although the geodesic *nearest* edge Voronoi diagram (the medial axis) can be found in linear time. As a first step of our geodesic edge center algorithm, we give a linear-time algorithm to find the geodesic farthest-edge Voronoi diagram restricted to the polygon boundary.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases geodesic center of polygon, farthest edges, farthest-segment Voronoi diagram

Digital Object Identifier 10.4230/LIPIcs.SoCG.2023.49

Related Version *Full Version:* <https://arxiv.org/abs/2303.09702>

Funding Research supported by the Natural Sciences and Engineering Research Council of Canada (NSERC).

Acknowledgements For helpful comments we thank Boris Aronov, Therese Biedl, John Hershberger, Joseph Mitchell, and the SoCG reviewers.

1 Introduction

The most basic “center” problem is Sylvester’s problem: given n points in the plane, find the smallest disc that encloses the points. The center of this disc is a point that minimizes the maximum distance to any of the given points. We consider a center problem that differs in two ways from Sylvester’s problem. First, the domain is a simple polygon and the distance measure is not Euclidean distance, but rather the shortest path, or “geodesic” distance inside the polygon. Second, the sites are not points but rather the edges of the polygon. More precisely, the problem is to find, given a simple polygon in the plane, the *geodesic edge center*, which is a point in the polygon that minimizes the maximum geodesic distance to a polygon edge. See Figure 1. More formally, let E be the set of edges of the polygon P , and for point $p \in P$ and edge $e \in E$, define $d(p, e)$ to be the geodesic distance from p to e . Define the *geodesic radius* of a point $p \in P$ to be $r(p) := \max\{d(p, e) : e \in E\}$. Then the *geodesic edge center* is a point $p \in P$ that minimizes $r(p)$.

Our main result is a linear-time algorithm to find the geodesic edge center of a simple n -vertex polygon. This improves our previous $O(n \log n)$ time algorithm [15]. The algorithm follows the strategies used to find the geodesic *vertex* center, which is a point in the polygon



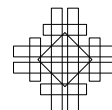
© Anna Lubiw and Anurag Murty Naredla;
licensed under Creative Commons License CC-BY 4.0
39th International Symposium on Computational Geometry (SoCG 2023).

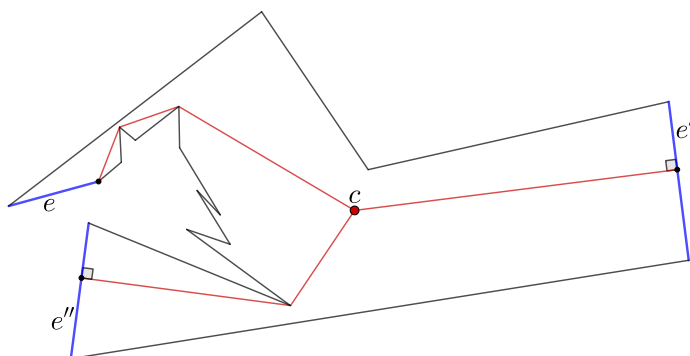
Editors: Erin W. Chambers and Joachim Gudmundsson; Article No. 49; pp. 49:1–49:15

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany





■ **Figure 1** Point c is the edge center of this polygon. Edges e, e', e'' (in blue) are geodesically farthest from c – the geodesic paths (in red) from c to these edges all have the same length.

that minimizes the maximum geodesic distance to a polygon vertex. In 1989, Pollack, Sharir and Rote [25] gave an $O(n \log n)$ time algorithm for the geodesic vertex center problem. A main tool – which is used in all subsequent algorithms – is a linear-time *chord oracle* that finds, given a chord, which side of the chord contains the center. In 2016, Ahn, Barba, Bose, De Carufel, Korman, and Oh [2] improved the runtime for the geodesic vertex center to $O(n)$. Their most important new contribution is the use of ϵ -nets to perform a divide-and-conquer search. Our algorithm follows the approach of Ahn et al., modified to deal with farthest edges rather than farthest vertices. We simplify some aspects and we repair some errors in their approach. The edge-center problem is more general than the vertex center problem via the reduction of splitting each vertex into two vertices joined by a very short edge.

In general, the center of a set of sites can be determined from the farthest Voronoi diagram of those sites, but computing the Voronoi diagram can be more costly. As the first step of our center algorithm we give a linear-time algorithm to compute the geodesic farthest-edge Voronoi diagram restricted to the boundary of the polygon. Computing the whole geodesic farthest-edge Voronoi diagram in linear time is an open problem.

Background on centers and farthest Voronoi diagrams. Megiddo [19] gave a linear-time algorithm to find the center of a set of points in the plane (Sylvester’s problem) using the “prune-and-search” technique (see also Dyer [11]), which is used in the final stages of all geodesic center algorithms. However, computing the farthest Voronoi diagram of points in the plane takes $\Theta(n \log n)$ time [26].

Our problem involves distances that are geodesic rather than Euclidean, and sites that are segments (edges) rather than points. These have been studied separately, although there is almost no work combining them.

For Euclidean distances, Megiddo’s method extends to linear-time algorithms to find the center of line segments or lines in the plane [6]. The farthest Voronoi diagram of segments in the plane was considered by Aurenhammer et al. [4], who called it a “stepchild in the vast Voronoi diagram literature”. They gave an $O(n \log n)$ time algorithm which was improved to output-sensitive time $O(n \log h)$, where h is the number of faces of the diagram [24].

For geodesic distances with point sites Ahn et al. gave a linear-time algorithm to find the geodesic center of the vertices of a polygon [2]. The corresponding farthest Voronoi diagram can be found in time $O(n \log \log n)$ [23], and in expected linear time [5]. More generally, for m points inside an n -vertex polygon, an algorithm to find their farthest Voronoi diagram was first given by Aronov et al. [3] with run-time $O((n + m) \log(n + m))$, and improved in a

sequence of papers [23, 5, 22], culminating in an optimal run time of $O(n + m \log m)$ [29]. This is also the best-known bound for finding the center of m points in a simple polygon. For sites more general than point sites inside a polygon, the only result we are aware of is our $O((n + m) \log(n + m))$ time algorithm to find the geodesic center of m *half-polygons* [15], with edges being a special case.

Finally, we mention a curious difference between nearest and farthest site Voronoi diagrams of edges in a polygon. The nearest Voronoi diagram of the edges of a polygon is the medial axis, one of the most famous and useful Voronoi diagrams. The medial axis can be found in linear time [9]. By contrast, the *farthest* Voronoi diagram of edges in a polygon has received virtually no attention, except for a convex polygon (which avoids geodesic issues) where there is an $O(n \log n)$ time algorithm [10], and a recent expected linear-time algorithm [14].

2 Overview of the algorithm

Before giving the overview of our algorithm, we outline the previous work that our algorithm builds upon, and explain what is novel about our contributions.

Pollack et al. [25] gave an $O(n \log n)$ time algorithm to find the geodesic vertex center of a simple polygon. A main ingredient is to solve the problem one dimension down. In particular, they develop an $O(n)$ time *chord oracle* that, given a chord of the polygon, finds the *relative center* restricted to the chord and from that, determines whether the center of the polygon lies to left or right of the chord. By applying the chord oracle $O(\log n)$ times, they limit the search to a convex subpolygon where Euclidean distances can be used. This reduces the problem to finding a minimum disc that encloses some disks, which Megiddo [20] solved in linear time using the same approach as for his linear programming algorithm. We extended the chord oracle to handle farthest *edges* instead of vertices [15].

The idea used in the chord oracle algorithm is central to further developments. Expressed in general terms, the goal is to find a point in a domain (either a chord or the whole polygon) that minimizes the maximum distance to a site (a vertex or edge of the polygon). The idea is to first find what we will call a *coarse cover* of the domain by a linear number of elementary regions R (intervals or triangles), each with an associated easy-to-compute convex function f_R that captures the geodesic distance to a potential farthest edge, and with the property that the upper envelope of the functions is the geodesic radius function. Thus, the goal is to find the point x that minimizes the upper envelope of the functions f_R . When the domain is a chord, the chord oracle solves this in linear time.

When the domain is the whole polygon, and the sites are vertices, Ahn et al. [2] gave a linear-time algorithm. They find a coarse cover of the whole polygon starting from Hershberger and Suri's algorithm [13] (based on matrix-searching techniques [1]) to find the farthest vertex from each vertex. They then use divide-and-conquer based on ϵ -nets – their big innovation – to reduce the domain to a triangle. After that, the vertex center is found using Megiddo-style prune-and-search techniques like those used by Pollack et al.

Our algorithm uses a similar approach, modified to deal with farthest *edges* rather than vertices. Another difference is that we give a simpler method of finding a coarse cover of the polygon by first finding the geodesic farthest-edge Voronoi diagram on the polygon boundary. There is a linear-time algorithm to find the geodesic farthest *vertex* Voronoi diagram on the polygon boundary by Oh, Barba, and Ahn [22]. Our algorithm is considerably simpler, and it is a novel idea to use the boundary Voronoi diagram to find the center.

Other differences between our approach and that of Ahn et al. are introduced in order to repair some flaws in their paper. They use ϵ -net techniques, but their range space does not have the necessary properties for finding ϵ -nets in deterministic linear time. We remedy this by using a different range space, thereby repairing and generalizing their result.

Algorithm overview**Phase I: Finding the farthest-edge Voronoi diagram restricted to the polygon boundary (Section 4)**

We first show that the linear-time algorithm of Hershberger and Suri [13] that finds the farthest *vertex* from each vertex can be modified to find the farthest *edge* from each vertex. A polygon edge e whose endpoints have the same farthest edge g is then part of the farthest Voronoi region of g . To find the Voronoi diagram on a *transition edge* e that has different farthest edges at its endpoints, we must find the upper envelope of the coarse cover of e . We use the fact that the coarse cover of e is constructed from two shortest path trees inside a smaller subpolygon called the *hourglass* of e . The hourglasses of all transition edges can be found in linear time. In each hourglass, the shortest path trees allow us to construct the upper envelope incrementally in linear time – this is a main new aspect of our work.

Phase II: Finding the geodesic edge center (Section 5)

We first find a coarse cover of the polygon by triangles, each bounded by two polygon chords plus a segment of an edge, and each with an associated convex function that captures the geodesic distance to a potential farthest edge – the potential farthest edges are those that have non-empty Voronoi regions on the boundary of P . The problem of finding the edge center is then reduced to the problem of finding the point that minimizes the upper envelope of the coarse cover functions.

To find this point we use divide-and-conquer, reducing in each step to a smaller subpolygon with a constant fraction of the coarse cover elements. There are two stages. In Stage 2, once the subpolygon is a triangle, the prune-and-search approach of Megiddo’s can be applied. In Stage 1 every coarse cover triangle that intersects the subpolygon has a boundary chord crossing the subpolygon, and ϵ -net techniques are used to reduce the number of such chords, and hence the number of coarse cover elements. Our approach follows that of Ahn et al. [2] but we repair some flaws – another main new aspect of our work. Ahn et al. recurse on subpolygons called “4-cells” that are the intersection of four half-polygons (a *half-polygon* is the subpolygon to one side of a chord). We instead recurse on “3-anchor hulls” that are the geodesic convex hulls of at most three points or subchains of the polygon boundary. These define a range space whose ground set is a set of chords and whose ranges are subsets of chords that cross a 3-anchor hull. We prove that our range space has finite VC-dimension, which repairs the faulty proof in Ahn et al. for 4-cells. Even more crucially, we give a “subspace oracle” that permits an ϵ -net to be found in *deterministic* linear time, something missing from their approach.

3 Preliminaries

Although our algorithm follows the pattern of the geodesic vertex center algorithm by Ahn et al. [2], we must re-do everything from the ground up to deal with farthest edges. In this section we summarize some basic results, deferring proofs and details to the full version [17].

Notation and definitions. A *chord* of a polygon is a straight line segment in the (closed) polygon with both endpoints on the boundary, ∂P . For a point $p \in P$ and a point or line segment s in P , $\pi(p, s)$ is the unique *shortest* (or *geodesic*) path from p to s , and $t(p, s)$ is its *terminal point*. The length of $\pi(p, s)$, denoted $d(p, s)$, is the *geodesic distance* from

p to s . For point p in P , a **farthest edge**, $F(p)$, is an edge for which $d(p, F(p)) \geq d(p, e)$, for every edge e of P . The **geodesic radius** of point p is $r(p) := d(p, F(p))$. The **geodesic edge center** is a point $p \in P$ that minimizes $r(p)$.

General position assumptions. As is standard for Voronoi diagrams of segments, e.g., see [4], we use the following tie-breaking rule to prevent 2-dimensional Voronoi regions with more than one farthest edge.

Tie-breaking rule. Suppose that p is a point of P , e and f are two edges that meet at reflex vertex u , and $\pi(p, e) = \pi(p, f) = \pi(p, u)$. Let line b be the angle bisector of u . For p not on b , break the tie $d(p, e) = d(p, f)$ by saying that the distance to the edge on the opposite side of b is greater.

We make the following general position assumptions, which we claim can be effected by perturbing vertices.

► **Assumptions 1.** (1) No three vertices of P are collinear. (2) After imposing the tie-breaking rule, no vertex is equidistant from two or more edges. (3) No point on the polygon boundary has more than two farthest edges and no point in the interior of the polygon has more than a constant number (six) of farthest edges.

It follows that the set of points with more than one farthest edge is 1-dimensional, does not contain any vertex of P , and intersects ∂P in isolated points; see the full version [17].

Properties of farthest edges. We need the following basic “triangle property” (proved in the full version [17]) about shortest paths that cross.

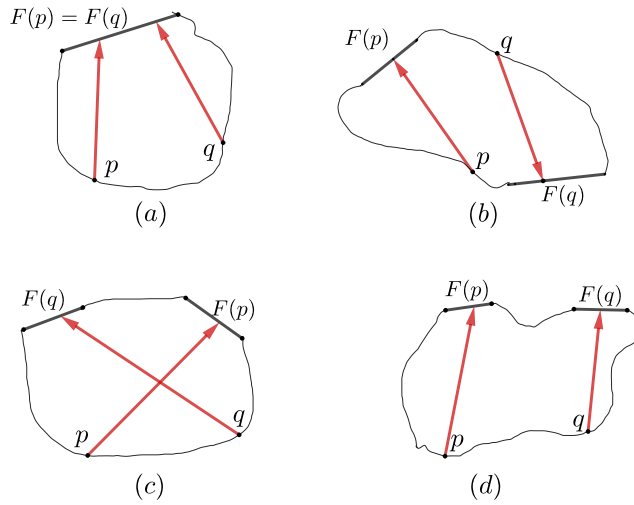
► **Lemma 1.** Suppose the points p, q and the edges e, f occur in the order p, q, e, f along the polygon boundary ∂P . Then $d(p, e) + d(q, f) \geq d(p, f) + d(q, e)$.

We then characterize what two paths to farthest edges are like, see Figure 2 and the full version [17]. In particular, we generalize the farthest-vertex Ordering Property [3] as follows.

The ordering property. As p moves clockwise around ∂P , so does $F(p)$.

Shortest paths to/from edges. As basic tools, we need linear-time algorithms to find shortest paths from a given point to all edges of the polygon, and to find shortest paths from a given edge to all vertices of the polygon. See the full version [17].

Separators and funnels. A geodesic path between two vertices of P separates ∂P into two parts, and when we focus on which vertices/edges are in opposite parts, we call the geodesic path a “separator”. Separators, first introduced by Suri [27], are a main tool for finding all farthest vertices in a polygon. In the full version [17] we extend the basic properties of separators to the case of farthest edges and prove: (1) If vertex v and edge e (e need not be farthest from v) are separated by a geodesic path $\pi(a, b)$, then the shortest path from v to e is contained, except for one edge, in the shortest path trees of a and b ; (2) A constant number of separators suffice to separate every vertex from its farthest edge.



■ **Figure 2** Schematics for possible and impossible orderings of points p, q and their farthest edges $F(p)$ and $F(q)$. (a) The only possible ordering if $F(p) = F(q)$. (b),(c) The two possible orderings if $F(p) \neq F(q)$. (d) The impossible ordering if $F(p) \neq F(q)$.

3.1 Chord oracles and coarse covers

In this section we describe the chord oracle results that we need from previous work, and we give a unified explanation of those algorithms and our current algorithm in terms of coarse covers. The basic function of a chord oracle is to decide, given a chord K , whether the center lies to the left or right (or on) the chord. Pollack et al. [25] gave a linear-time chord oracle for the geodesic vertex center, which is at the heart of all further geodesic center algorithms. We extended the chord oracle to the case of the geodesic *edge* center [15].

In both cases, a main step is the “one-dimension down” problem of finding the *relative center*, which is a point c_K on K that minimizes the geodesic radius function $r(x)$. The directions of the first segments of the paths from c_K to its farthest sites determine whether the center of P lies left/right/on K (see the full version [17]).

Algorithms to find the relative center of a chord or the center of a polygon rely on a basic convexity property of the geodesic radius function (see the full version [17]) and all follow the same pattern, which can be formalized via the concept of a *coarse cover* of the chord/polygon. The idea is that a *coarse cover* for a domain (a chord/polygon) is a set of elementary regions R (intervals/triangles) covering the domain, where each region R has an associated easy-to-compute convex function f_R , such that the upper envelope of the f_R 's is the geodesic radius function. We give a precise definition for the case of farthest edges (following [15] and specialized for our Assumptions 1).

► **Definition 2.** A *coarse cover* of chord K [or polygon P] is a set of triples (R, f, e) where

1. R is a subinterval of K [or a triangle of P], f is a function defined on domain R , and e is an edge of P .
2. For all $x \in R$, $f(x) = d(x, e)$ and either: $f(x) = d_2(x, v) + \kappa$ where d_2 is Euclidean distance, κ is a constant and v is a vertex of P ; or $f(x) = d_2(x, \bar{e})$, where d_2 is Euclidean distance and \bar{e} is the line through e .
3. For any point $x \in K$ [or P], and any edge e that is farthest from x , there is a triple (R, f, e) in the coarse cover with $x \in R$.

Condition (3) implies that the upper envelope of the functions of the coarse cover is the geodesic radius function. Thus the [relative] center problem breaks into two subproblems: (1) find a coarse cover; and (2) find the point x that minimizes the upper envelope of the coarse cover functions. The high-level idea for solving step (2) in linear time (for a chord or polygon domain) is to recursively reduce the domain (the search space) to a subinterval or subpolygon while eliminating elements of the coarse cover whose functions are strictly dominated by others. As for step (1) – constructing a coarse cover – see Section 4 for a chord and Section 5 for a polygon.

We call the chord oracle in Phase II when we use divide-and-conquer to search for the center in successively smaller subpolygons. We actually need two variations of the basic chord oracle. First, we need a *geodesic oracle* that tests which side of a geodesic contains the center. Secondly, we do not construct a coarse cover of a chord/geodesic from scratch; rather, we intersect the triangles of the coarse cover of the subpolygon with the chord/geodesic, thus avoiding runtime dependence on n . These variations are described in the full version [17].

4 Phase I: Finding the farthest-edge Voronoi diagram restricted to the polygon boundary

Based on Assumptions 1, the boundary of P consists of chains with a single farthest edge, separated by points (not vertices) that have two farthest edges (see Figure 4). Our goal is to find these points. The first step of the algorithm is to find the farthest edge from each vertex of the polygon in linear time. To do this, we extend the algorithm of Hershberger and Suri [13] that finds the farthest *vertex* from each vertex. Details are in the full version [17]. The next step is to fill in the Voronoi diagram along the polygon edges. For an edge ab where vertices a and b have the same farthest edge, i.e., $F(a) = F(b)$, all points on the edge ab have the same farthest edge, by the Ordering Property. An edge ab with $F(a) \neq F(b)$ is a **transition edge**. We will find the farthest-edge Voronoi diagram on one transition edge in linear time. To handle *all* the transition edges in linear time, we will show that for each transition edge ab we can restrict our attention to the **hourglass** $H(a, b)$ which is the subpolygon of P bounded by ab , $\pi(a, F(b))$, $\pi(b, F(a))$ and the portion of ∂P between the terminals $t(a, F(b))$ and $t(b, F(a))$. In the full version [17] we show that the hourglasses of all transition edges can be found in linear time and that the sum of their sizes is linear.

In this section, we show how to construct the farthest-edge Voronoi diagram along one polygon edge ab in time linear in the size of the polygon. We do not assume that the polygon is an hourglass. For purposes of description, imagine ab horizontal with a at the left, and the polygon interior above ab . We use the **coarse cover** (Definition 2) of the edge ab , which can be found in linear time (Lubiw and Naredla [16]). Elements of the coarse cover are triples (I, f, e) where I is a subinterval of ab and $f(x) = d(x, e)$ for any $x \in I$. By resolving overlaps of coarse cover intervals I , we find the upper envelope of the coarse cover functions f , which immediately gives the Voronoi diagram on ab . This is easy if we sort the endpoints of the intervals I , but we cannot afford to sort. Instead, we will insert the coarse cover elements one by one, maintaining a list M of [pairwise internally] disjoint subintervals of ab together with an associated distance function $f_M(x)$. An efficient insertion order depends on the fact that elements of the coarse cover of edge ab are associated with edges of the shortest path trees T_a and T_b (that consist of the shortest paths from a and b , respectively, to all the edges of P). We will use the ordering of the trees as embedded in the plane.

Oh, Barba, Ahn [23] gave a linear-time algorithm to find the farthest *vertex* Voronoi diagram on the boundary of P . The approach is similar, but they add coarse cover elements by iterating over the sites (the vertices in their case), which involves a complicated algorithm to sweep back and forth along M maintaining a shortest path to the current vertex, and a tricky amortized analysis (see [23, Lemma 7]). Our approach is simpler and more general.

4.1 Farthest-edge Voronoi diagram on one edge

In previous work [15, 16] we constructed a coarse cover (see Definition 2) of an edge ab from the shortest path trees T_a and T_b . The trees are first augmented with 0-length edges so that the paths to every polygon edge e end with a tree edge perpendicular to e . In particular, every polygon edge corresponds to a leaf in each tree.

Direct edges of T_a and T_b away from their roots. Each edge uv of $T_a \setminus T_b$ with $u \neq a$ corresponds to an ***a-side*** coarse cover element (I, f, e) where e corresponds to the farthest leaf of T_a descended from v . For example, in Figure 3, see edge a_3 of T_a and interval I_{a_3} . There are symmetrically defined ***b-side*** coarse cover elements. Each edge uv of $T_a \cap T_b$ with u visible from ab corresponds to a ***central triangle*** coarse cover element (I, f, e) where e corresponds to the farthest leaf of T_a descended from v . For example, see edge $a_5 = b_5$ and interval I_{a_5} . Each polygon edge e that has an interior point visible from ab corresponds to a ***central trapezoid*** coarse cover element (I, f, e) where I consists of the points on ab whose shortest paths to e arrive perpendicularly. For example, see edge e_4 and interval I_{e_4} .

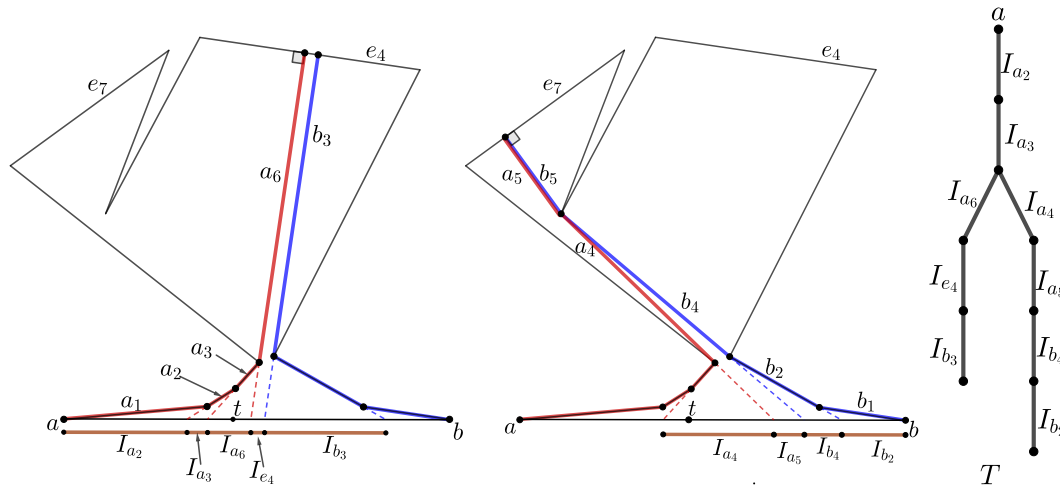
► **Lemma 3** (proved in the full version [17]). *For any edge e of P , let $C(e)$ be the set of coarse cover elements (I, f, e) for e . If $C(e)$ is nonempty, then its elements correspond to a (possibly empty) path in T_a directed towards a leaf, followed by a central triangle or trapezoid, followed by a (possibly empty) path in T_b directed towards the root. Furthermore, the corresponding intervals on ab appear in order, are [internally] disjoint, and their union is an interval.*

We next construct a single tree T whose edges correspond to coarse cover elements of ab . Then we incrementally construct the farthest-edge Voronoi diagram on ab by adding coarse cover elements in a depth first search (DFS) order of T .

Constructing tree T . Starting with T_a , attach an edge for each central trapezoid element to the associated leaf vertex of T_a ; add the path of b -side triangle elements for each polygon edge e after the central triangle or trapezoid for e ; and contract original edges of T_a that are not associated with coarse cover elements. See Figure 3(right). We give more detail of these steps in the full version [17]. The resulting tree T can be constructed in linear time and its edges are in one-to-one correspondence with the coarse cover elements.

► **Observation 4.** *If uv and vw are edges of T , then the corresponding coarse cover intervals I_1 and I_2 appear in that order along ab and intersect in a single point.*

DFS algorithm for the Voronoi diagram. We add the coarse cover elements following a DFS of T with children of a node in clockwise order. We maintain a list M of interior disjoint subintervals of ab whose union is an interval starting at a . Each subinterval in M records the coarse cover element it came from. Define f_M to be the distance function determined by the intervals of M . Initially, M is the single point a , and f_M is $-\infty$. At the end M will be the upper envelope of the coarse cover functions (though this property is not guaranteed throughout). To handle edge uv of T with associated coarse cover element (I, f, e) , we



■ **Figure 3** Coarse cover elements corresponding to some (not all) edges of T_a (red) and T_b (blue): (left) coarse cover elements for e_4 ; (middle) coarse cover elements for e_7 ; (right) the corresponding part of tree T . When I_{a_6} is handled by **Insert** it wins the comparison with I_{a_4} so it replaces I_{a_4} up to the cross-over point t , and the algorithm discards the rest of I_{a_6} , together with I_{e_4} and I_{b_3} .

compare f to f_M beginning at the left endpoint of I . We maintain a pointer p_u that gives an interval of M containing this endpoint. The recursive routine **Insert**(u, p_u) inserts into M the portions of coarse cover elements that are associated with u 's subtree and that define the upper envelope. At the top level, we call **Insert**(a, p_a), where p_a points to a .

```

Insert( $u, p_u$ )  #  $u$  is a node of  $T$  and  $p_u$  is a pointer to an interval of  $M$ 
  for each child  $v$  of  $u$  in clockwise order do
    ( $I, f, e$ ) := the coarse cover element associated with the edge  $uv$  of  $T$ 
     $l$  := left endpoint of  $I$ ;  $r$  := right endpoint of  $I$ 
    Invariant:  $p_u$  points to an interval of  $M$  that contains  $l$ 
    if  $f(l^+) > f_M(l^+)$  where  $l^+$  is just to the right of  $l$  then
      replace intervals of  $M$  starting at  $p_u$  with a subinterval of  $I$  ending at
      the ‘‘cross-over’’ point  $t < r$  where  $f_M$  starts to dominate  $f$ , or at  $r$ 
    if  $f$  dominates until  $r$  and  $v$  is not a leaf of  $T$  then
      call Insert( $v, p_v$ ), where  $p_v$  is a pointer to interval  $I$  in  $M$ 
  
```

Runtime. Each edge of T is handled once, and causes at most one new interval to be inserted into M , so the total number of endpoints inserted into M is $O(n)$. We can access $f_M(l^+)$ in constant time using the pointer p_u . Then the endpoints of intervals of M that we traverse as we do the insertion vanish from M . Thus the runtime is $O(n)$.

Correctness. The following lemma implies that the final M is the upper envelope of the coarse cover functions.

► **Lemma 5.** *The algorithm only discards pieces of coarse cover elements that do not form part of the final upper envelope.*

Proof. We examine the behaviour of the algorithm for edge uv of T with associated coarse cover element (I, f, e) , where $I = [l, r]$. We insert the subinterval $[l, t]$ into M (or no subinterval). Because $f(x) \geq f_M(x)$ for $x \in [l, t]$, any subintervals of M that are removed due to the insertion do not determine the upper envelope, so their removal is correct.

49:10 The Geodesic Edge Center of a Simple Polygon

If we insert all of interval I into M and recursively call $\text{Insert}(v, p_v)$, then this is correct by induction. So suppose we insert a proper subinterval of I or none of I . We must prove that no later part of I , and no element of the coarse cover associated with edges of the subtree rooted at v determines the upper envelope. Let t^+ be a point just to the right of t (or just to the right of l if we insert no part of I). Then $f_M(t^+) > f(t^+)$. Number the polygon edges e_1, e_2, \dots, e_m clockwise from a to b . Suppose that $e = e_i$, so $f(x) = d(x, e_i)$ for $x \in I$, in particular, $f(t^+) = d(t^+, e_i)$. Suppose that $f_M(t^+) = d(t^+, e_k)$. Then $d(t^+, e_k) > d(t^+, e_i)$.

Now consider the edges of T descended from v plus the edge uv . Consider the corresponding coarse cover elements, C_v , and let e_j be any polygon edge associated with any element in C_v . Note that the intervals on ab associated with coarse cover elements of C_v lie to the right of r , except for I associated with uv . We will prove that for any point $x \in ab$ to the right of t^+ , $d(x, e_k) > d(x, e_j)$, which implies that none of the coarse cover elements in C_v determines the upper envelope, nor does any part of I to the right of t . Thus the algorithm is correct to discard them.

We first prove the result for $x = t^+$. If uv corresponds to a central triangle/trapezoid for e_i or a b -side triangle, then T has a single path descending from v , all of whose edges are associated with e_i , i.e., $j = i$. Otherwise, by the definition of an a -side coarse cover element, e_i corresponds to the farthest leaf of T_a descended from v , which implies that $d(x, e_i) \geq d(x, e_j)$ for all $x \in I$, and in particular for $x = t^+$. Thus, in either case we have $d(t^+, e_k) > d(t^+, e_i) \geq d(t^+, e_j)$.

We next claim that $k < j$. The current f_M values arise from tree edges already processed. These consist of: (1) edges on the path from a to u ; and (2) edges of T counterclockwise from this path. Edges on the path from a to u have coarse cover intervals on ab to the left of l , by Observation 4. Thus type (1) edges do not determine $f_M(t^+)$. By the depth-first-search order, type (2) edges have coarse cover elements corresponding to polygon edges counterclockwise from e_j . Thus $k < j$.

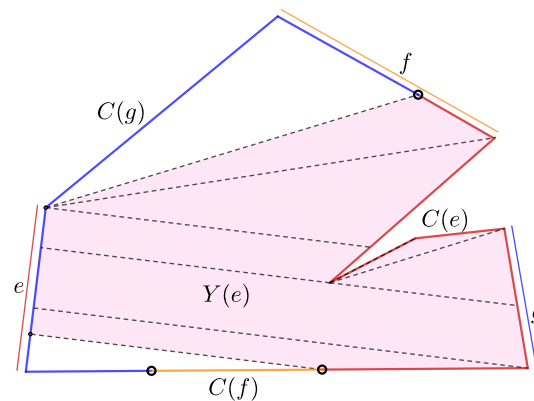
To complete the proof of Lemma 5, consider any point $x \in ab$ to the right of t^+ . The clockwise ordering around the polygon boundary is x, t^+, e_k, e_j , so by Lemma 1 and the fact that $d(t^+, e_k) > d(t^+, e_j)$, we get $d(x, e_k) > d(x, e_j)$, as required. ◀

5 Phase II: Finding the geodesic edge center

The first step of Phase II is to construct a **coarse cover** (Definition 2) of the polygon in linear time. As shown in Figure 4 the **funnel** $Y(e)$ that consists of shortest paths between a chain on ∂P with farthest edge e and e itself can be partitioned into its shortest path map. If the result includes trapezoids, we partition each one into two triangles¹. Each triangle is bounded by two polygon chords and a segment of a polygon edge, and the distance to e has the form required by Definition 2 (see the full version [17]). We seek the point inside P that minimizes the upper envelope of the functions of the coarse cover. Note that Phase I can detect if the edge center lies *on* ∂P so we may assume that the center is interior to P .

Our final divide-and-conquer algorithm follows the vertex center algorithm of Ahn et al. [2], generalized to farthest edges, and repairing flaws in their approach. At each step of the algorithm we have a subpolygon Q whose interior contains the center together with the coarse cover elements needed to compute the edge center and we shrink the subpolygon and eliminate a constant fraction of the coarse cover. Each recursive step takes time linear in the size of the subproblem (the size of Q plus the size of its coarse cover). The subpolygons

¹ Thus our triangles are not necessarily “apexed” triangles as in [2].



■ **Figure 4** The farthest-edge Voronoi diagram restricted to the polygon boundary consists of chains $C(e)$, $C(f)$, $C(g)$ farthest from edges e , f , g , respectively. To construct the coarse cover, the funnel $Y(e)$ (shaded) is partitioned (by dashed segments) into a shortest path map from e to $C(e)$.

we work with are **3-anchor hulls** defined as follows (see Figure 6). An **anchor** is a point inside P , or a subchain of ∂P . A **3-anchor hull** is the geodesic convex hull of at most three anchors. These are weakly simple in general, but we only recurse on **simple** 3-anchor hulls.

The algorithm has two stages. In Stage 1 no triangle of the coarse cover contains Q (this is true initially when $Q = P$), so every triangle has a chord crossing Q and we use ϵ -net techniques on the set of such chords to reduce to a smaller cell Q' that is crossed by a fraction of the chords, and hence by a fraction of the coarse cover triangles. Once Q is contained in a triangle of the coarse cover we show (see the full version [17]) that the size of Q , denoted $|Q|$, is at most 6. In fact, we will exit Stage 1 as soon as $|Q| \leq 6$. It is then easy to reduce Q to a triangle. After that, we switch to Stage 2, where the convexity of Q allows us to use a Megiddo-style prune-and-search technique (as Ahn et al. do) to recursively reduce the size of the subproblem. Stage 2 is deferred to the full version [17].

5.1 Stage 1: Algorithm for large Q

Consider a subproblem corresponding to a simple 3-anchor hull Q with $|Q| > 6$. We give an algorithm that either finds the edge center or reduces to a subproblem with $|Q| \leq 6$, which is handled by Stage 2. In Stage 1, no triangle of the coarse cover contains Q (as proved in the full version [17]) so each one has a chord crossing Q – we denote this set of chords by $\mathcal{K}(Q)$.

To apply ϵ -net techniques we define a **3-anchor range space** as follows. The ground set is a set \mathcal{K} of chords of P , and for each 3-anchor hull H of P there is a range $\mathcal{K}(H)$ consisting of all chords of \mathcal{K} that **cross** H . Here a chord **crosses** a set if both open half-polygons of the chord contain points of the set.

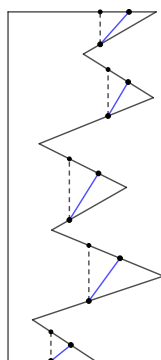
The algorithm finds a constant size ϵ -net of the 3-anchor range space on $\mathcal{K}(Q)$, which is a set $N \subseteq \mathcal{K}(Q)$ such that any 3-anchor hull not intersected by a chord of N is intersected by only a constant fraction of the chords of $\mathcal{K}(Q)$ – this is the important property that allows us to discard a fraction of the chords. The set of chords N forms an arrangement that partitions Q into cells. We use the chord oracle to determine which cell contains the center. We then add geodesic paths to subdivide this cell into a constant number of 3-anchor hulls and use a **geodesic oracle** (see the full version [17]) to find which 3-anchor hull contains the center, and to shrink it to a simple 3-anchor hull Q' . The algorithm recurses on Q' , whose coarse cover is a fraction of the size.

More details of the algorithm can be found in the full version [17]. For now, we expand on the aspects of the algorithm that differ from the approach of Ahn et al. [2]. Instead of 3-anchor hulls, their algorithm works with 4-cells, formed by taking the intersection of at most four half-polygons, where a half-polygon is the part of P to one side of a chord. The number (three versus four) is not significant, but we bound our regions by geodesics instead of chords in order to obtain the following two results.

1. The 3-anchor range space has finite VC-dimension. This implies that constant-sized ϵ -nets exist. Furthermore, there is a “subspace oracle” that allows us to find an ϵ -net N in deterministic linear time [28, Chapter 47, Theorem 47.4.3]. For further background see the full version [17], and the paper by Chazelle and Matoušek [8].

Ahn et al. claim that their range space (of chords crossing 4-cells) has finite VC-dimension but their proof is flawed. Our proof shows that their range space does in fact have finite VC-dimension. They do not mention subspace oracles, without which their algorithm runs in expected linear time rather than deterministic linear time as claimed. We expand on these aspects in Section 5.2 below.

2. A cell of the arrangement of N can be partitioned into constantly many 3-anchor hulls. The method used by Ahn et al. to subdivide a cell of N into 4-cells by adding a constant number of chords is incomplete, see Figure 5. We see how to repair their partition step but we find 3-anchor hulls more natural.



■ **Figure 5** Ahn et al. [2] subdivide a cell of N by adding vertical chords (dashed) at endpoints and intersection points of chords of N (blue), which leaves a 5-cell in this example.

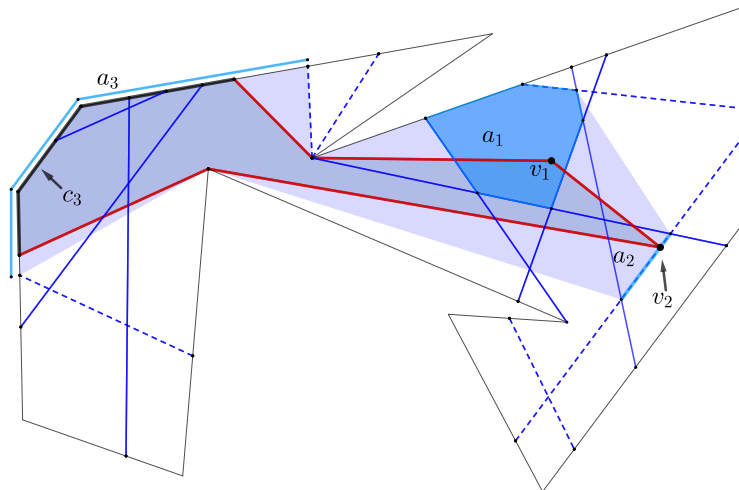
5.2 Epsilon-net results for Stage 1

In this section we expand on the ϵ -net results needed for Stage 1 of the algorithm as described above. We also give details of the flaws in the approach of Ahn et al. [2]. For an overview of ϵ -nets as used for geometric divide-and-conquer, see the full version [17] (also see Chazelle [7] or Mustafa [21]). To show that ϵ -nets of constant size exist we need the following result.

► **Lemma 6.** *The 3-anchor range space has VC-dimension less than 259.*

Our proof of Lemma 6 works equally well for 4-anchor hulls – the bound becomes 372. A 4-cell is a special case of a 4-anchor hull so our proof implies finite VC-dimension (≤ 372) for the 4-cell range space, which repairs the claim by Ahn et al.² We explain the flaw in

² In response to our enquiries, Eunjin Oh independently suggested a similar remedy.



■ **Figure 6** A simple 3-anchor hull Q (outlined in red) with anchors v_1, v_2 and the polygon chain c_3 (thick black). Solid blue chords cross Q , while dashed blue chords do not. The expanded 3-anchor hull $\psi(Q)$ (lightly shaded) has expanded anchors: a_1 , the face in the chord arrangement containing v_1 ; a_2 , the edge with v_2 in its interior; and a_3 , the polygon chain extending c_3 to chord endpoints. The same chords cross Q and $\psi(Q)$.

their proof. Let us refer to the set of chords intersecting a 4-cell as a “4-cell range”. Ahn et al. prove that the 1-cell range space has VC-dimension at most 65,535. They note that a 4-cell is the intersection of four 1-cells, and then claim in their Lemma 9.1 that this implies finite VC-dimension for the 4-cell range space. As justification, they refer to Proposition 10.3.3 of Matousek’s text [18], which states that the VC-dimension is bounded for any family whose sets can be defined by a formula of Boolean connectives (union, intersection, set difference). However, Matousek’s proposition cannot be applied in this situation because, although a 4-cell is the intersection of four 1-cells, it is not true that a 4-cell *range* is the intersection of four 1-cell *ranges*. In particular, a chord can intersect two 1-cells, but not intersect the intersection of the two 1-cells. For example, a line of slope -1 can intersect the $+x$ half-plane and the $+y$ half-plane without intersecting the $+x, +y$ quadrant.

Proof of Lemma 6. We will prove that the shattering dimension is 6 and then apply the result that a range space with shattering dimension d has VC-dimension bounded by $12d \ln(6d)$ (Lemma 5.14 from Har-Peled [12]). For $d = 6$, this is less than 259.

We must show that for a set \mathcal{K} of chords with $|\mathcal{K}| = m$, the number of distinct ranges is $O(m^6)$. We prove that the range space for \mathcal{K} is the same if we replace 3-anchor hulls by “expanded 3-anchor hulls” that are defined in terms of \mathcal{K} , more precisely, in terms of the arrangement $A(\mathcal{K})$ of the chords \mathcal{K} plus the edges of P . Define an *expanded anchor* to be an internal face, edge, or vertex of $A(\mathcal{K})$, or a polygon chain with endpoints in $V(\mathcal{K})$, the set of endpoints of chords \mathcal{K} . An *expanded 3-anchor hull* is the geodesic convex hull of at most three expanded anchors.

► **Lemma 7.** *The set of ranges $\mathcal{R} = \{\mathcal{K}(Q) \mid Q \text{ is a 3-anchor hull}\}$ is the same as the set of ranges $\overline{\mathcal{R}} = \{\mathcal{K}(Q) \mid Q \text{ is an expanded 3-anchor hull}\}$.*

Proof. To prove $\mathcal{R} \subseteq \overline{\mathcal{R}}$, consider a 3-anchor hull Q . Replace any point anchor p by the smallest (by containment) internal vertex, edge, or face of $A(\mathcal{K})$ that contains p . See Figure 6. Replace any polygon chain anchor C by the smallest chain of ∂P containing C and with

endpoints in $V(\mathcal{K})$. Let $\psi(Q)$ be the geodesic convex hull of these expanded anchors. Then $\psi(Q)$ is an expanded 3-anchor hull that contains Q , and it is straight-forward to prove that $\mathcal{K}(Q) = \mathcal{K}(\psi(Q))$ (see the full version [17]).

For the other direction, let Q be an expanded 3-anchor hull. Replace an expanded anchor that is a face, edge, or vertex of $A(\mathcal{K})$ by a point anchor in the interior of that face, edge, or vertex. An expanded anchor that is a polygon chain remains unchanged. Let $\gamma(Q)$ be the geodesic convex hull of the resulting anchors. Observe that $\gamma(Q)$ is a 3-anchor hull and $\psi(\gamma(Q)) = Q$. As above, this implies that $\mathcal{K}(Q) = \mathcal{K}(\gamma(Q))$. ◀

To complete the proof of Lemma 6 we claim that the number of expanded 3-anchor hulls of \mathcal{K} is $O(m^6)$. An expanded anchor may be an internal vertex, edge, or face of $A(\mathcal{K})$, of which there are $O(m^2)$ possibilities. Otherwise, an expanded anchor is a chain of ∂P between vertices of $V(\mathcal{K})$, also with $O(m^2)$ possibilities. Thus the number of expanded 3-anchor hulls is $O((m^2)^3) = O(m^6)$. ◀

Subspace oracle

To prove that the 3-anchor range space has a subspace oracle, we present a deterministic algorithm that, given a subset $\mathcal{K}' \subseteq \mathcal{K}$ with $|\mathcal{K}'| = m$, computes the set of ranges $\mathcal{R} = \{\mathcal{K}'(Q) \mid Q \text{ is a 3-anchor hull}\}$ in time $O(m^7)$. The idea is to use Lemma 7 and to construct $A(\mathcal{K}')$ minus the edges of P , and find, for each chord $K \in \mathcal{K}'$, which of the $O(m^2)$ expanded anchors in $A(\mathcal{K}')$ intersects each side of K , and then, for each of the $O(m^6)$ expanded 3-anchor hulls, eliminate the chords that have all three expanded anchors to one side, leaving the chords that cross the hull. For further details see [17].

References

- 1 Alok Aggarwal, Maria M Klawe, Shlomo Moran, Peter Shor, and Robert Wilber. Geometric applications of a matrix-searching algorithm. *Algorithmica*, 2(1-4):195–208, 1987. doi:10.1007/bf01840359.
- 2 Hee-Kap Ahn, Luis Barba, Prosenjit Bose, Jean-Lou De Carufel, Matias Korman, and Eunjin Oh. A linear-time algorithm for the geodesic center of a simple polygon. *Discrete & Computational Geometry*, 56(4):836–859, 2016. doi:10.1007/s00454-016-9796-0.
- 3 Boris Aronov, Steven Fortune, and Gordon Wilfong. The furthest-site geodesic Voronoi diagram. *Discrete & Computational Geometry*, 9(3):217–255, 1993. doi:10.1007/bf02189321.
- 4 Franz Aurenhammer, Robert L Scot Drysdale, and Hannes Krasser. Farthest line segment Voronoi diagrams. *Information Processing Letters*, 100(6):220–225, 2006. doi:10.1016/j.ipl.2006.07.008.
- 5 Luis Barba. Optimal algorithm for geodesic farthest-point Voronoi diagrams. In *35th International Symposium on Computational Geometry (SoCG 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019. doi:10.4230/LIPIcs.SocG.2019.12.
- 6 Binay K Bhattacharya, Shreesh Jadhav, Asish Mukhopadhyay, and J-M Robert. Optimal algorithms for some intersection radius problems. *Computing*, 52(3):269–279, 1994. doi:10.1007/bf02246508.
- 7 Bernard Chazelle. Cutting hyperplanes for divide-and-conquer. *Discrete & Computational Geometry*, 9(2):145–158, 1993. doi:10.1007/bf02189314.
- 8 Bernard Chazelle and Jiří Matoušek. On linear-time deterministic algorithms for optimization problems in fixed dimension. *Journal of Algorithms*, 21(3):579–597, 1996. doi:10.1006/jagm.1996.0060.
- 9 Francis Chin, Jack Snoeyink, and Cao An Wang. Finding the medial axis of a simple polygon in linear time. *Discrete & Computational Geometry*, 21(3):405–420, 1999. doi:10.1007/p100009429.

- 10 R L Scot Drysdale and Asish Mukhopadhyay. An $O(n \log n)$ algorithm for the all-farthest-segments problem for a planar set of points. *Information Processing Letters*, 105(2):47–51, 2008. doi:10.1016/j.ipl.2007.08.004.
- 11 Martin E Dyer. Linear time algorithms for two- and three-variable linear programs. *SIAM Journal on Computing*, 13(1):31–45, 1984. doi:10.1137/0213003.
- 12 Sariel Har-Peled. *Geometric Approximation Algorithms*. American Mathematical Society, 2011. doi:10.1090/surv/173.
- 13 John Hershberger and Subhash Suri. Matrix searching with the shortest-path metric. *SIAM Journal on Computing*, 26(6):1612–1634, 1997. doi:10.1137/s0097539793253577.
- 14 Elena Khramtcova and Evanthia Papadopoulou. An expected linear-time algorithm for the farthest-segment Voronoi diagram. arXiv, 2014. doi:10.48550/arxiv.1411.2816.
- 15 Anna Lubiw and Anurag Murty Naredla. The visibility center of a simple polygon. arXiv, 2021. doi:10.48550/arxiv.2108.07366.
- 16 Anna Lubiw and Anurag Murty Naredla. The visibility center of a simple polygon. In Petra Mutzel, Rasmus Pagh, and Grzegorz Herman, editors, *29th Annual European Symposium on Algorithms (ESA 2021)*, volume 204 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 65:1–65:14, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ESA.2021.65.
- 17 Anna Lubiw and Anurag Murty Naredla. The geodesic edge center of a simple polygon. arXiv, 2023. doi:10.48550/arXiv.2303.09702.
- 18 Jiří Matoušek. *Lectures on Discrete Geometry*, volume 212 of *Graduate Texts in Mathematics*. Springer Verlag, 2002. doi:10.1007/978-1-4613-0039-7.
- 19 Nimrod Megiddo. Linear-time algorithms for linear programming in R^3 and related problems. *SIAM Journal on Computing*, 12(4):759–776, 1983. doi:10.1137/0212052.
- 20 Nimrod Megiddo. On the ball spanned by balls. *Discrete & Computational Geometry*, 4(6):605–610, 1989. doi:10.1007/bf02187750.
- 21 Nabil H Mustafa. *Sampling in Combinatorial and Geometric Set Systems*, volume 265 of *Mathematical Surveys and Monographs*. American Mathematical Society, 2022. doi:10.1090/surv/265.
- 22 Eunjin Oh and Hee-Kap Ahn. Voronoi diagrams for a moderate-sized point-set in a simple polygon. *Discrete & Computational Geometry*, 63(2):418–454, 2020. doi:10.1007/s00454-019-00063-4.
- 23 Eunjin Oh, Luis Barba, and Hee-Kap Ahn. The geodesic farthest-point Voronoi diagram in a simple polygon. *Algorithmica*, 82(5):1434–1473, 2020. doi:10.1007/s00453-019-00651-z.
- 24 Evanthia Papadopoulou and Sandeep Kumar Dey. On the farthest line-segment Voronoi diagram. *International Journal of Computational Geometry & Applications*, 23(06):443–459, 2013. doi:10.1007/978-3-642-35261-4_22.
- 25 Richard Pollack, Micha Sharir, and Günter Rote. Computing the geodesic center of a simple polygon. *Discrete & Computational Geometry*, 4(6):611–626, 1989. doi:10.1007/bf02187751.
- 26 Michael Ian Shamos and Dan Hoey. Closest-point problems. In *16th Annual Symposium on Foundations of Computer Science (FOCS 1975)*, pages 151–162. IEEE, 1975. doi:10.1109/sfcs.1975.8.
- 27 Subhash Suri. Computing geodesic furthest neighbors in simple polygons. *Journal of Computer and System Sciences*, 39:220–235, 1989. doi:10.1016/0022-0000(89)90045-7.
- 28 Csaba D Toth, Joseph O’Rourke, and Jacob E Goodman, editors. *Handbook of Discrete and Computational Geometry*. CRC press, 2017. doi:10.1201/9781315119601.
- 29 Haitao Wang. An optimal deterministic algorithm for geodesic farthest-point Voronoi diagrams in simple polygons. *Discrete & Computational Geometry*, 2022. doi:10.1007/s00454-022-00424-6.