

A linear-time algorithm for the geodesic center of a simple polygon

Hee-Kap Ahn^{*3}, Luis Barba^{1,2}, Prosenjit Bose¹, Jean-Lou de Carufel¹, Matias Korman^{4,5}, and Eunjin Oh³

¹ School of Computer Science, Carleton University, Ottawa, Canada.

`jit@scs.carleton.ca, jdecaruf@cg.scs.carleton.ca`

² Département d'Informatique, Université Libre de Bruxelles, Brussels, Belgium.

`lbarbaf1@ulb.ac.be`

³ Department of Computer Science and Engineering, POSTECH,

77 Cheongam-Ro, Nam-Gu, Pohang, Gyeongbuk, Korea.

`{heekap, jin9082}@postech.ac.kr`

⁴ National Institute of Informatics (NII), Tokyo, Japan.

`korman@nii.ac.jp`

⁵ JST, ERATO, Kawarabayashi Large Graph Project.

Abstract

Let P be a closed simple polygon with n vertices. For any two points in P , the geodesic distance between them is the length of the shortest path that connects them among all paths contained in P . The geodesic center of P is the unique point in P that minimizes the largest geodesic distance to all other points of P . In 1989, Pollack, Sharir and Rote [Disc. & Comput. Geom. 89] showed an $O(n \log n)$ -time algorithm that computes the geodesic center of P . Since then, a longstanding question has been whether this running time can be improved (explicitly posed by Mitchell [Handbook of Computational Geometry, 2000]). In this paper we affirmatively answer this question and present a linear time algorithm to solve this problem.

1 Introduction

Let P be a simple polygon with n vertices. Given two points x, y in P , the *geodesic path* $\pi(x, y)$ is the shortest path contained in P connecting x with y . If the straight-line segment connecting x with y is contained in P , then $\pi(x, y)$ is a straight-line segment. Otherwise, $\pi(x, y)$ is a polygonal chain whose vertices (other than its endpoints) are reflex vertices of P . We refer the reader to [22] for more information on geodesic paths.

The *geodesic distance* between x and y , denoted by $|\pi(x, y)|$, is the sum of the Euclidean lengths of each segment in $\pi(x, y)$. Throughout this paper, when referring to the distance between two points in P , we mean the geodesic distance between them. To ease the description, we assume that each vertex of P has a unique farthest neighbor. This *general position* condition was also assumed by Aronov et al. [2] and can be obtained by applying a slight perturbation to the positions of the vertices [10].

Given a point $x \in P$, a (geodesic) *farthest neighbor* of x , is a point $f_P(x)$ (or simply $f(x)$) of P whose geodesic distance to x is maximized.

Let $F_P(x)$ be the function that maps each $x \in P$ to the distance to a farthest neighbor of x (i.e., $F_P(x) = |\pi(x, f(x))|$). A point $c_P \in P$ that minimizes $F_P(x)$ is called the *geodesic center* of P . Similarly, a point $s \in P$ that maximizes $F_P(x)$ (together with $f(s)$) is called a *geodesic diametral pair* and their distance is known as the *geodesic diameter*. Asano and

* The work by H.-K. Ahn and E. Oh was supported by the NRF grant 2011-0030044 (SRC-GAIA) funded by the Korea government (MSIP).



licensed under Creative Commons License CC-BY



Leibniz International Proceedings in Informatics

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

42 Toussaint [3] showed that the geodesic center is unique (whereas it is easy to see that several
 43 geodesic diametral pairs may exist).

44 In this paper, we show how to compute the geodesic center of P in $O(n)$ time.

45 1.1 Previous Work

46 Since the early 1980s the problem of computing the geodesic center (and its counterpart,
 47 the geodesic diameter) has received a lot of attention from the computational geometry
 48 community. Chazelle [7] gave the first algorithm for computing the geodesic diameter (which
 49 runs in $O(n^2)$ time using linear space). Afterwards, Suri [27] reduced it to $O(n \log n)$ -time
 50 without increasing the space constraints. Finally, Hershberger and Suri [15] presented a fast
 51 matrix search technique, one application of which is a linear-time algorithm for computing
 52 the diameter.

53 The first algorithm for computing the geodesic center was given by Asano and Toussaint [3],
 54 and runs in $O(n^4 \log n)$ -time. In 1989, Pollack, Sharir, and Rote [25] improved it to $O(n \log n)$
 55 time. Since then, it has been an open problem whether the geodesic center can be computed
 56 in linear time (indeed, this problem was explicitly posed by Pollack et al. [25] and later by
 57 Mitchell [22, Chapter 27]).

58 Several variations of these two problems have been considered. Indeed, the same problem
 59 has been studied under different metrics. For example, the L_1 geodesic distance [6], the link
 60 distance [26, 16, 9] (where we look for the path with the minimum possible number of bends
 61 or *links*), or even rectilinear link distance [23, 24] (a variation of the link distance in which
 62 only isothetic segments are allowed). The diameter and center of a simple polygon for both
 63 the L_1 and rectilinear link metrics can be computed in linear time (whereas $O(n \log n)$ time
 64 is needed for the link distance).

65 Another natural extension is the computation of the diameter and center in polygonal
 66 domains (i.e., polygons with one or more holes). Polynomial time algorithms are known for
 67 both the diameter [4] and center [5], although the running times are significantly larger (i.e.,
 68 $O(n^{7.73})$ and $O(n^{12+\varepsilon})$, respectively).

69 1.2 Outline

70 In order to compute the geodesic center, Pollack et al. [25] introduce a linear time *chord-*
 71 *oracle*. Given a chord C that splits P into two sub-polygons, the oracle determines which
 72 sub-polygon contains c_P . Combining this operation with an efficient search on a triangulation
 73 of P , Pollack et al. narrow the search of c_P within a triangle (and find the center using
 74 optimization techniques). Their approach however, does not allow them to reduce the
 75 complexity of the problem in each iteration, and hence it runs in $\Theta(n \log n)$ time.

76 The general approach of our algorithm described in Section 6 is similar: partition P
 77 into $O(1)$ cells, use an oracle to determine which cell contains c_P , and recurse within the
 78 cell. Our approach differs however in two important aspects that allows us to speed-up the
 79 algorithm. First, we do not use the chords of a triangulation of P to partition the problem
 80 into cells. We use instead a cutting of a suitable set of chords. Secondly, we compute a set Φ
 81 of $O(n)$ functions, each defined in a triangular domain contained in P , such that their upper
 82 envelope, $\phi(x)$, coincides with $F_P(x)$. Thus, we can “ignore” the polygon P and focus only
 83 on finding the minimum of the function $\phi(x)$.

84 The search itself uses ε -nets and cutting techniques, which guarantee that both the size of
 85 the cell containing c_P and the number of functions of Φ defined in it decrease by a constant
 86 fraction (and thus leads to an overall linear time algorithm). This search has however two

stopping conditions, (1) reach a subproblem of constant size, or (2) find a triangle containing c_P . In the latter case, we show that $\phi(x)$ is a convex function when restricted to this triangle. Thus, finding its minimum becomes an optimization problem that we solve in Section 7 using cuttings in \mathbb{R}^3 .

The key of this approach lies in the computation of the functions of Φ and their triangular domains. Each function $g(x)$ of Φ is defined in a triangular domain \triangle contained in P and is associated to a particular vertex w of P . Intuitively speaking, $g(x)$ maps points in \triangle to their (geodesic) distance to w . We guarantee that, for each point $x \in P$, there is one function g defined in a triangle containing x , such that $g(x) = F_P(x)$. To compute these triangles and their corresponding functions, we proceed as follows.

In Section 2, we use the matrix search technique introduced by Hershberger and Suri [15] to decompose the boundary of P , denoted by ∂P , into connected edge-disjoint chains. Each chain is defined by either (1) a consecutive list of vertices that have the same farthest neighbor v (we say that v is *marked* if it has such a chain associated to it), or (2) an edge whose endpoints have different farthest neighbors (such edge is called a *transition edge*).

In Section 3, we consider each transition edge ab of ∂P independently and compute its *hourglass*. Intuitively, the hourglass of ab , H_{ab} , is the region of P between two chains, the edge ab and the chain of ∂P that contains the farthest neighbors of all points in ab . Inspired by a result of Suri [27], we show that the sum of the complexities of all hourglasses defined on a transition edge is $O(n)$. (The *combinatorial complexity* (or simply complexity) of a geometric object is the total number of vertices and edges that define it.) In addition, we provide a new technique to compute all these hourglasses in linear time.

In Section 5 we show how to compute the functions in Φ and their respective triangles. We distinguish two cases: (1) Inside each hourglass H_{ab} of a transition edge, we use a technique introduced by Aronov et al. [2] that uses the shortest-path trees of a and b in H_{ab} to construct $O(|H_{ab}|)$ triangles with their respective functions (for more information on shortest-path trees refer to [11]). (2) For each marked vertex v we compute triangles that encode the distance from v . Moreover, we guarantee that these triangles cover every point of P whose farthest neighbor is v . Overall, we compute the $O(n)$ functions of Φ in linear time.

2 Decomposing the boundary

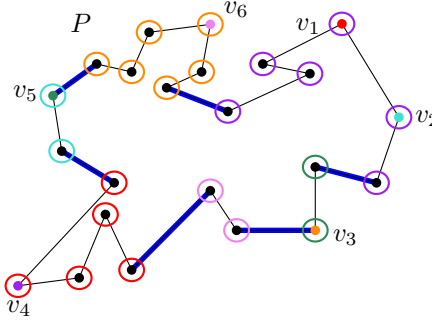
In this section, we decompose the boundary of P into chains of consecutive vertices that share the same farthest neighbor and edges of P whose endpoints have distinct farthest neighbors.

Using a result from Hershberger and Suri [15], in $O(n)$ time we can compute the farthest neighbor of each vertex of P . Recall that the farthest neighbor of each vertex of P is always a convex vertex of P [3] and is unique by our general position assumption.

We mark the vertices of P that are farthest neighbors of at least one vertex of P . Let M denote the set of marked vertices of P (clearly this set can be computed in $O(n)$ time after applying the result of Hershberger and Suri). In other words, M contains all vertices of P whose Voronoi region contains at least one vertex of P .

Given a vertex v of P , the vertices of P whose farthest neighbor is v appear contiguously along ∂P [2]. Therefore, after computing all these farthest neighbors, we effectively split the boundary into subchains, each associated with a different vertex of M ; see Figure 1.

Let a and b be the endpoints of a transition edge of ∂P such that b is the clockwise neighbor of a along ∂P . Because ab is a transition edge, we know that $f(a) \neq f(b)$. Recall that we have computed $f(a)$ and $f(b)$ in the previous step and note that $a, b, f(a), f(b)$ are



■ **Figure 1** Each vertex of the boundary of P is assigned with a farthest neighbor which is then marked. The boundary is then decomposed into vertex-disjoint chains, each associated with a marked vertex, joined by transition edges (blue) whose endpoints have different farthest neighbors.

133 in clockwise order.

134 For any vertex $v \in \partial P$ such that $f(a), v, f(b)$ are in clockwise order, we know that there
 135 cannot be a vertex u of P such that $f(u) = v$. As proved by Aronov et al. [2, Corollary 2.7.4],
 136 if there is a point x on ∂P whose farthest neighbor is v , then x must lie on the open segment
 137 (a, b) . In other words, the Voronoi region $R(v)$ restricted to ∂P is contained in (a, b) .

138 3 Hourglasses

139 Given two points x and y on ∂P , let $\partial P(x, y)$ be the polygonal chain that starts at x and
 140 follows the boundary of P clockwise until reaching y . We say that three (non-empty) disjoint
 141 sets A, B and C contained in ∂P are in *clockwise order* if $B \subset \partial P(a, c)$ for any $a \in A$ and
 142 any $c \in C$. (To ease notation, we say that three points $x, y, z \in \partial P$ are in clockwise order if
 143 $\{x\}, \{y\}$ and $\{z\}$ are in clockwise order).

144 For any polygonal chain $C = \partial P(p_0, p_k)$, the *hourglass* of C , denoted by H_C , is the simple
 145 polygon contained in P bounded by C , $\pi(p_k, f(p_0))$, $\partial P(f(p_0), f(p_k))$ and $\pi(f(p_k), p_0)$; see
 146 Figure 2. We call C and $\partial P(f(p_0), f(p_k))$ the *top* and *bottom* chains of H_C , respectively,
 147 while $\pi(p_k, f(p_0))$ and $\pi(f(p_k), p_0)$ are referred to as the *walls* of H_C .

148 We say that the hourglass H_C is *open* if its walls are vertex-disjoint. We say C is a
 149 *transition chain* if $f(p_0) \neq f(p_k)$ and neither $f(p_0)$ nor $f(p_k)$ are interior vertices of C . In
 150 particular, if an edge ab of ∂P is a transition chain, we say that it is a *transition edge* (see
 151 Figure 2).

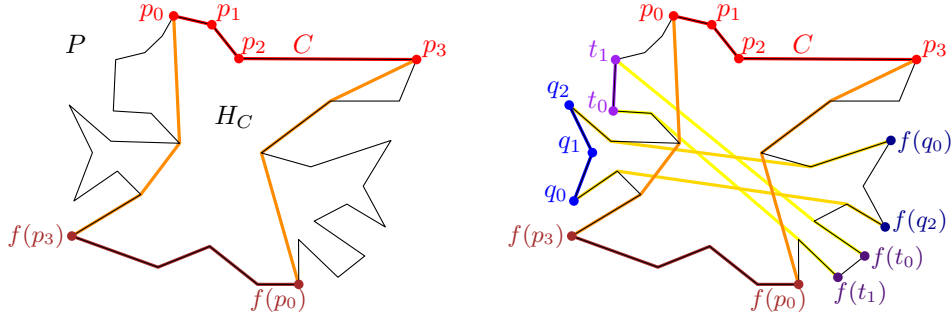
152 ► **Lemma 1.** [Restatement of Lemma 3.1.3 of [2]] If C is a transition chain of ∂P , then the
 153 hourglass H_C is an open hourglass.

154 Note that by Lemma 1, the hourglass of each transition chain is open. In the remainder
 155 of the paper, all the hourglasses considered are defined by a transition chain. That is, they
 156 are open and their top and bottom chains are edge-disjoint.

157 The following lemma is depicted in Figure 2 and is a direct consequence of the Ordering
 158 Lemma proved by Aronov et al. [2, Corollary 2.7.4].

159 ► **Lemma 2.** Let C_1, C_2, C_3 be three edge-disjoint transition chains of ∂P in clockwise order.
 160 Then, the bottom chains of H_{C_1}, H_{C_2} and H_{C_3} are also edge-disjoint and are in clockwise
 161 order.

162 Let γ be a geodesic path joining two points on the boundary of P . We say that γ *separates*
 163 two points x_1 and x_2 of ∂P if the points of $X = \{x_1, x_2\}$ and the endpoints of γ alternate



■ **Figure 2** Given two edge-disjoint transition chains, their hourglasses are open and the bottom chains of their hourglasses are also edge-disjoint. Moreover, these bottom chains appear in the same cyclic order as the top chains along ∂P .

164 along the boundary of P (x_1 and x_2 could coincide with the endpoints of γ in degenerate
 165 cases). We say that a geodesic path γ *separates* an hourglass H if it separates the points of
 166 its top chain from those of its bottom chain.

167 ► **Lemma 3.** *Let C_1, \dots, C_r be edge-disjoint transition chains of ∂P . Then, there is a set*
 168 *of $t \leq 10$ geodesic paths $\gamma_1, \dots, \gamma_t$ with endpoints on ∂P such that for each $1 \leq i \leq r$ there*
 169 *exists $1 \leq j \leq t$ such that γ_j separates H_{C_i} . Moreover, this set can be computed in $O(n)$*
 170 *time.*

171 **Proof.** Aronov et al. showed that there exist four vertices v_1, \dots, v_4 of P and geodesic paths
 172 $\pi(v_1, v_2), \pi(v_2, v_3), \pi(v_3, v_4)$ such that for any point $x \in \partial P$, one of these paths separates x
 173 from $f(x)$ [2, Lemma 2.7.6]. Moreover, they show how to compute this set in $O(n)$ time.

174 Let $\Gamma = \{\pi(v_i, v_j) : 1 \leq i < j \leq 4\}$ and note that v_1, \dots, v_4 split the boundary of P into
 175 at most four connected components. If a chain C_i is completely contained in one of these
 176 components, then one path of Γ separates the top and bottom chain of H_{C_i} . Otherwise, some
 177 vertex v_j is an interior vertex of C_i . However, because the chains C_1, \dots, C_r are edge-disjoint,
 178 there are at most four chains in this situation. For each chain C_i containing a vertex v_j ,
 179 we add the geodesic path connecting the endpoints of C_i to Γ . Therefore, Γ consists of at
 180 most 10 geodesic paths and each hourglass H_{C_i} has its top and bottom chain separated by
 181 some path of Γ . Since only $O(1)$ additional paths are computed, this can be done in linear
 182 time. ◀

183 A *chord* of P is an edge joining two non-adjacent vertices a and b of P such that $ab \subseteq P$.
 184 Therefore, a chord splits P into two sub-polygons.

185 ► **Lemma 4.** *[Restatement of Lemma 3.4.3 of [2]] Let C_1, \dots, C_r be a set of edge-disjoint*
 186 *transition chains of ∂P in clockwise order. Then each chord of P appears in $O(1)$ hourglasses*
 187 *among H_{C_1}, \dots, H_{C_r} .*

188 **Proof.** Note that chords can only appear on walls of hourglasses. Because hourglasses are
 189 open, any chord must be an edge on exactly one wall of each of these hourglasses. Assume,
 190 for the sake of contradiction, that there exist two points $s, t \in P$ whose chord st is in three
 191 hourglasses H_{C_i}, H_{C_j} and H_{C_k} (for some $1 \leq i < j < k \leq r$) such that s visited before t
 192 when going from the top chain to the bottom one along the walls of the three hourglasses.
 193 Let s_i and t_i be the points in the in the top and bottom chains of H_{C_i} , respectively, such
 194 that $\pi(s_i, t_i)$ is the wall of H_{C_i} that contains st (analogously, we define s_k and t_k)

Because C_i, C_j, C_k are in clockwise order, Lemma 2 implies that the bottom chains of C_i, C_j and C_k are also in clockwise order. Therefore, C_j lies between s_i and s_k and the bottom chain of H_{C_j} lies between t_i and t_k . That is, for each $x \in C_j$ and each y in the bottom chain of H_{C_j} , the geodesic path $\pi(x, y)$ is “sandwiched” by the paths $\pi(s_i, t_i)$ and $\pi(s_k, t_k)$. In particular, $\pi(x, y)$ contains st for each pair of points in the top and bottom chain of H_{C_j} . However, this implies that the hourglass H_{C_j} is not open—a contradiction that comes from assuming that st lies in the wall of three open hourglasses, when this wall is traversed from the top chain to the bottom chain. Analogous arguments can be used to bound the total number of walls that contain the edge st (when traversed in any direction) to $O(1)$. ◀

► **Lemma 5.** *Let x, u, y, v be four vertices of P in clockwise order. Given the shortest-path trees T_x and T_y of x and y in P , respectively, such that T_x and T_y can answer lowest common ancestor (LCA) queries in $O(1)$ time, we can compute the path $\pi(u, v)$ in $O(|\pi(u, v)|)$ time. Moreover, all edges of $\pi(u, v)$, except perhaps one, belong to $T_x \cup T_y$.*

Proof. Let X (resp. Y) be the set containing the LCA in T_x (resp. T_y) of u, y , and of v, y (resp. u, x and x, y). Note that the points of $X \cup Y$ lie on the path $\pi(x, y)$ and can be computed in $O(1)$ time by hypothesis. Moreover, using LCA queries, we can decide their order along the path $\pi(x, y)$ when traversing it from x to y . (Both X and Y could consist of a single vertex in some degenerate situations.) Two cases arise:

Case 1. If there is a vertex $x^* \in X$ lying after a vertex $y^* \in Y$ along $\pi(x, y)$, then the path $\pi(u, v)$ contains the path $\pi(y^*, x^*)$. In this case, the path $\pi(u, v)$ is the concatenation of the paths $\pi(u, y^*)$, $\pi(y^*, x^*)$, and $\pi(x^*, v)$ and that the three paths are contained in $T_x \cup T_y$. Moreover, $\pi(u, v)$ can be computed in time proportional to its length by traversing along the corresponding tree; see Figure 3 (top).

Case 2. In this case the vertices of X appear before the vertices of Y along $\pi(x, y)$. Let x' (resp. y') be the vertex of X (resp. Y) closest to x (resp. y).

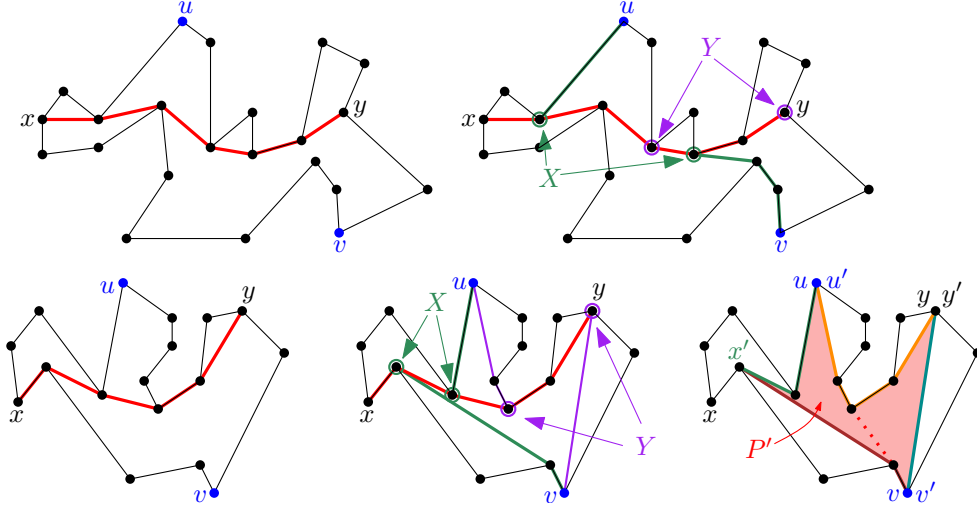
Let u' be the last vertex of $\pi(u, x)$ that is also in $\pi(u, y)$. Note that u' can be constructed by walking from u' towards x until the path towards y diverges. Thus, u' can be computed in $O(|\pi(u, u')|)$ time. Define v' analogously and compute it in $O(|\pi(v, v')|)$ time.

Let P' be the polygon bounded by the geodesic paths $\pi(x', u')$, $\pi(u', y')$, $\pi(y', v')$ and $\pi(v', x')$. Because the vertices of X appear before those of Y along $\pi(x, y)$, P' is a simple polygon; see Figure 3 (bottom).

In this case the path $\pi(u, y)$ is the union of $\pi(u, u')$, $\pi(u', v')$ and $\pi(v', v)$. Because $\pi(u, u')$ and $\pi(v', v)$ can be computed in time proportional to their length, it suffices to compute $\pi(u', v')$ in $O(|\pi(u', v')|)$ time.

Note that P' is a simple polygon with only four convex vertices x', u', y' and v' , which are connected by chains of reflex vertices. Thus, the shortest path from x' to y' can have at most one diagonal edge connecting distinct reflex chains of P' . Since the rest of the points in $\pi(u', v')$ lie on the boundary of P' and from the fact that each edge of P' is an edge of $T_x \cup T_y$, we conclude all edges of $\pi(u, v)$, except perhaps one, belong to $T_x \cup T_y$.

We want to find the common tangent between the reflex paths $\pi(u', x')$ and $\pi(v', y')$, or the common tangent of $\pi(u', y')$ and $\pi(v', x')$ as one of them belongs to the shortest path $\pi(u', v')$. Assume that the desired tangent lies between the paths $\pi(u', x')$ and $\pi(v', y')$. Since these paths consist only of reflex vertices, the problem can be reduced to finding the common tangent of two convex polygons. By slightly modifying the linear time algorithm to compute this tangents, we can make it run in $O(|\pi(u', v')|)$ time.



■ **Figure 3** (top) Case 1 of the proof of Lemma 5 where the path $\pi(u, v)$ contains a portion of the path $\pi(x, y)$. (bottom) Case 2 of the proof of Lemma 5 where the path $\pi(u, v)$ has exactly one edge being the tangent of the paths $\pi(u', y')$ and $\pi(v', x')$.

241 Since we do not know if the tangent lies between the paths $\pi(u', x')$ and $\pi(v', y')$, we
 242 process the chains $\pi(u', y')$ and $\pi(v', x')$ in parallel and stop when finding the desired tangent.
 243 Consequently, we can compute the path $\pi(u, v)$ in time proportional to its length. ◀

► **Lemma 6.** Let P be a simple polygon with n vertices. Given k disjoint transition chains C_1, \dots, C_k of ∂P , it holds that

$$\sum_{i=1}^k |H_{C_i}| = O(n).$$

244 **Proof.** Because the given transition chains are disjoint, Lemma 2 implies that the bottom
 245 chains of their respective hourglasses are also disjoint. Therefore, the sum of the complexities
 246 of all the top and bottom chains of these hourglasses is $O(n)$. To bound the complexity of
 247 their walls we use Lemma 4. Since no chord is used more than a constant number of times,
 248 it suffices to show that the total number of chords used by all these hourglasses is $O(n)$.

To prove this, we use Lemma 3 to construct $O(1)$ *splitting chains* $\gamma_1, \dots, \gamma_t$ such that for each $1 \leq i \leq k$, there is a splitting chain γ_j that separates the top and bottom chains of H_{C_i} . For each $1 \leq j \leq t$, let

$$\mathcal{H}^j = \{H_{C_i} : \text{the top and bottom chain of } H_{C_i} \text{ are separated by } \gamma_j\}.$$

Since the complexity of the shortest-path trees of the endpoints of γ_j is $O(n)$ [11], and from the fact that the chains C_1, \dots, C_k are disjoint, Lemma 5 implies that the total number of edges in all the hourglasses of \mathcal{H}^j is $O(n)$. Moreover, because each of these edges appears in $O(1)$ hourglasses among C_1, \dots, C_k , we conclude that

$$\sum_{H \in \mathcal{H}^j} |H| = O(n).$$

249 Since we have only $O(1)$ splitting chains, our result follows. ◀

3.1 Building hourglasses

Let E be the set of transition edges of ∂P . Given a transition edge $ab \in E$, we say that H_{ab} is a *transition hourglass*. In order to construct the triangle cover of P , we construct the transition hourglass of each transition edge of E . By Lemma 6, we know that $\sum_{ab \in E} |H_{ab}| = O(n)$. Therefore, our aim is to compute the cover in time proportional to the size of H_{ab} .

By Lemma 3 we can compute a set of $O(1)$ separating paths such that for each transition edge ab , the transition hourglass H_{ab} is separated by one (or more) paths in this set. For each endpoint of the $O(1)$ separating paths we compute its shortest-path tree in linear time [8, 11]. In addition, we preprocess these trees in linear time to support LCA queries [14]. Both computations need linear time per endpoint and use $O(n)$ space. Since we do this process for a constant number of endpoints, overall this preprocessing takes $O(n)$ time.

Let γ be a separating path. Note that γ separates the boundary of P into two chains S and S' such that $S \cup S' = \partial P$. Let $\mathcal{H}(\gamma)$ be the set of transition hourglasses separated by γ whose transition edge is contained in S (whenever an hourglass is separated by more than one path, we pick one arbitrarily). Note that we can classify all transition hourglasses into the sets $\mathcal{H}(\gamma)$ in $O(n)$ time (since $O(1)$ separating paths are considered).

We claim that we can compute all transition hourglasses of $\mathcal{H}(\gamma)$ in $O(n)$ time. By construction, the wall of each of these hourglasses consists of a (geodesic) path that connects a point in S with a point in S' . Let $u \in S$ and $v \in S'$ be two vertices such that $\pi(u, v)$ is the wall of a hourglass in $\mathcal{H}(\gamma)$. Because LCA queries can be answered in $O(1)$ time [14], Lemma 5 allows us to compute this path in $O(|\pi(u, v)|)$ time. Therefore, we can compute all hourglasses of $\mathcal{H}(\gamma)$ in $O(\sum_{H \in \mathcal{H}(\gamma)} |H| + n) = O(n)$ time by Lemma 6. Because only $O(1)$ separating paths are considered, we obtain the following result.

► **Lemma 7.** *The added complexity of the transition hourglasses of all transition edges of P is $O(n)$. Moreover, all these hourglasses can be constructed in $O(n)$ time.*

4 Funnels

Let $C = (p_0, \dots, p_k)$ be a chain of ∂P and let v be a vertex of P not in C . The *funnel* of v to C , denoted by $S_v(C)$, is the simple polygon bounded by C , $\pi(p_k, v)$ and $\pi(v, p_0)$; see Figure 4 (a). Note that the paths $\pi(v, p_k)$ and $\pi(v, p_0)$ may coincide for a while before splitting into disjoint chains. See Lee and Preparata [17] or Guibas et al. [11] for more details on funnels.

A subset $R \subset P$ is *geodesically convex* if for every $x, y \in R$, the path $\pi(x, y)$ is contained in R . This funnel $S_v(C)$ is then the minimum geodesically convex set that contains v and C .

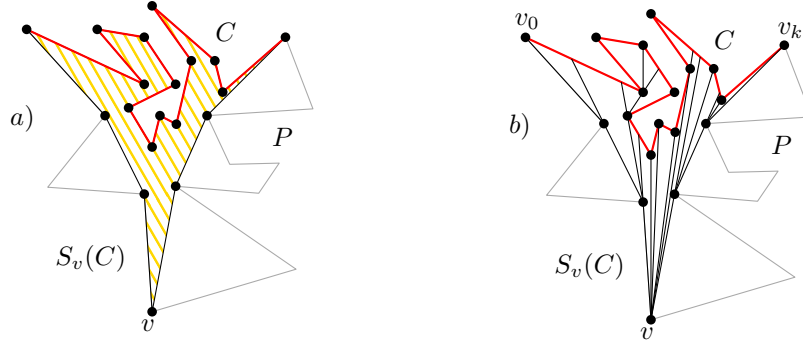
Given two points $x, y \in P$, the (geodesic) *bisector* of x and y is the set of points contained in P that are equidistant from x and y . This bisector is a curve, contained in P , that consists of straight-line segments and hyperbolic arcs. Moreover, this curve intersects ∂P only at its endpoints [1, Lemma 3.22].

The (farthest) *Voronoi region* of a vertex v of P is the set of points $R(v) = \{x \in P : F_P(x) = |\pi(x, v)|\}$ (including boundary points).

► **Lemma 8.** *Let v be a vertex of P and let C be a transition chain such $R(v) \cap \partial P \subseteq C$ and $v \notin C$. Then, $R(v)$ is contained in the funnel $S_v(C)$*

Proof. Let a and b be the endpoints of C such that $a, b, f(a)$ and $f(b)$ are in clockwise order. Because $R(v) \cap \partial P \subseteq C$, we know that $f(a), v$ and $f(b)$ are in clockwise order.

Let α (resp. β) be the bisector of v and $f(a)$ (resp. $f(b)$). Let h_a (resp. h_b) be the set of points of P that are farther from v than from $f(a)$ (resp. $f(b)$). Note that α is the boundary of h_a while β bounds h_b .



■ **Figure 4** a) The funnel $S_v(C)$ of a vertex v and a chain C contained in ∂P are depicted. b) The decomposition of $S_v(C)$ into apexed triangles produced by the shortest-path map of v .

295 By definition, we know that $R(v) \subseteq h_a \cap h_b$. Therefore, it suffices to show that $h_a \cap h_b \subset$
 296 $S_v(C)$. Assume for a contradiction that there is a point of $h_a \cap h_b$ lying outside of $S_v(C)$.
 297 By continuity of the geodesic distance, the boundaries of $h_a \cap h_b$ and $S_v(C)$ must intersect.
 298 Because $a \notin h_a$ and $b \notin h_b$, both bisectors α and β must have an endpoint on the edge
 299 ab . Since the boundaries of $h_a \cap h_b$ and $S_v(C)$ intersect, we infer that $\beta \cap \pi(v, b) \neq \emptyset$ or
 300 $\alpha \cap \pi(v, a) \neq \emptyset$. Without loss of generality, assume that there is a point $w \in \beta \cap \pi(v, b)$, the
 301 case where w lies in $\alpha \cap \pi(v, a)$ is analogous.

Since $w \in \beta$, we know that $|\pi(w, v)| = |\pi(w, f(b))|$. By the triangle inequality and since w cannot be a vertex of P as w intersects ∂P only at its endpoints, we get that

$$|\pi(b, f(b))| < |\pi(b, w)| + |\pi(w, f(b))| = |\pi(b, w)| + |\pi(w, v)| = |\pi(b, v)|.$$

302 Which implies that b is farther from v than from $f(b)$ —a contradiction that comes from
 303 assuming that $h_a \cap h_b$ is not contained in $S_v(C)$. ◀

304 4.1 Funnels of marked vertices

305 Recall that for each marked vertex $v \in M$, we know at least of one vertex on ∂P such that v
 306 is its farthest neighbor.

307 ▶ **Lemma 9.** *Let x be a point in P . If $f(x) = v$ for some marked vertex $v \in M$, then*
 308 *$x \in S_v(C_v)$.*

309 **Proof.** Since $f(u_0) \neq f(u_k)$, C_v is a transition chain. Moreover, C_v contains $R(v) \cap \partial P$ by
 310 definition. Therefore, Lemma 8 implies that $R(v) \subset S_v(C_v)$. Since $v = f(x)$, we know that
 311 $x \in R(v)$ and hence that $x \in S_v(C_v)$. ◀

312 For any marked vertex v , let u_1, \dots, u_{k-1} be the vertices of P such that $v = f(u_i)$ and
 313 assume that u_1, \dots, u_{k-1} are in clockwise order. Let u_0 and u_k be the neighbors of u_1 and
 314 u_{k-1} other than u_2 and u_{k-2} , respectively. Note that both u_0u_1 and $u_{k-1}u_k$ are transition
 315 edges of P . Thus, we can assume that their transition hourglasses have been computed.

316 Let $C_v = (u_0, \dots, u_k)$ and consider the funnel $S_v(C_v)$. We call C_v the *main chain*
 317 of $S_v(C_v)$ while $\pi(u_k, v)$ and $\pi(v, u_0)$ are referred to as the *walls* of the funnel. Because
 318 $v = f(u_1) = f(u_{k-1})$, we know that v is a vertex of both $H_{u_0u_1}$ and $H_{u_{k-1}u_k}$. By definition,
 319 we have $\pi(v, u_0) \subset H_{u_0u_1}$ and $\pi(v, u_k) \subset H_{u_{k-1}u_k}$. Thus, we can explicitly compute both
 320 paths $\pi(v, u_0)$ and $\pi(v, u_k)$ in $O(|H_{u_0u_1}| + |H_{u_{k-1}u_k}|)$ time. So, overall, the funnel $S_v(C_v)$

can be constructed in $O(k + |H_{u_0 u_1}| + |H_{u_{k-1} u_k}|)$ time. Recall that, by Lemma 6, the total sum of the complexities of the transition hourglasses is $O(n)$. In particular, we can bound the total time needed to construct the funnels of all marked vertices by $O(n)$.

Since the complexity of the walls of these funnels is bounded by the complexity of the transition hourglasses used to compute them, we get that

$$\sum_{v \in M} |S_v(C_v)| = O\left(n + \sum_{ab \in E} |H_{ab}|\right) = O(n).$$

► **Lemma 10.** *The added complexity of the funnels of all marked vertices of P is $O(n)$. Moreover, all these funnels can be constructed in $O(n)$ time.*

5 Covering the polygon with apexed triangles

An *apexed triangle* $\triangle = (a, b, c)$ with *apex* a is a triangle contained in P with an associated distance function $g_\triangle(x)$, called the *apex function* of \triangle , such that (1) a is a vertex of P , (2) $b, c \in \partial P$, and (3) there is a vertex w of P , called the *definer* of \triangle , such that

$$g_\triangle(x) = \begin{cases} -\infty & \text{if } x \notin \triangle \\ |xa| + |\pi(a, w)| = |\pi(x, w)| & \text{if } x \in \triangle \end{cases}$$

In this section, we show how to find a set of $O(n)$ apexed triangles of P such that the upper envelope of their apex functions coincides with $F_P(x)$. To this end, we first decompose the transition hourglasses into apexed triangles that encode all the geodesic distance information inside them. For each marked vertex $v \in M$ we construct a funnel that contains the Voronoi region of v . We then decompose this funnel into apexed triangles that encode the distance from v .

5.1 Inside the transition hourglass

Let ab be a transition edge of P such that b is the clockwise neighbor of a along ∂P . Let B_{ab} denote the open bottom chain of H_{ab} . As noticed above, a point on ∂P can be farthest from a vertex in B_{ab} only if it lies in the open segment ab . That is, if v is a vertex of B_{ab} such that $R(v) \neq \emptyset$, then $R(v) \cap \partial P \subset ab$.

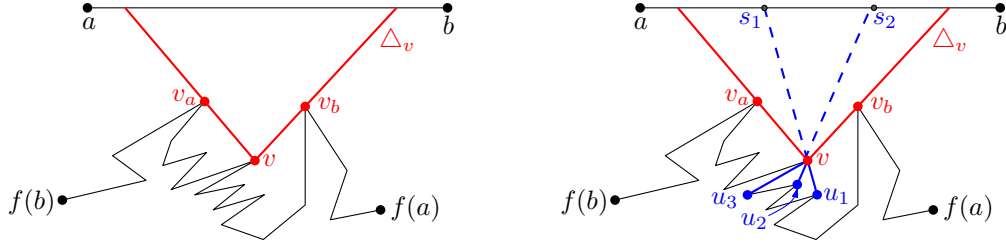
In fact, not only this Voronoi region is inside H_{ab} when restricted to the boundary of P , but we can further bound its location and show that $R(v) \subset H_{ab}$. The next result follows trivially from Lemma 8.

► **Corollary 11.** *Let v be a vertex of B_{ab} . If $R(v) \neq \emptyset$, then $R(v) \subset H_{ab}$.*

Our objective is to compute $O(|H_{ab}|)$ apexed triangles contained in H_{ab} , each with its distance function, such that the upper envelope of these apex functions coincides with $F_P(x)$ restricted to H_{ab} where it “matters”.

The same approach was already used by Pollack et al. in [25, Section 3]. Given a segment contained in the interior of P , they show how to compute a linear number of apexed triangles such that $F_P(x)$ coincides with the upper envelope of the corresponding apex functions in the given segment.

While the construction we follow is analogous, we use it in the transition hourglass H_{ab} instead of the full polygon P . Therefore, we have to specify what is the relation between the upper envelope of the computed functions and $F_P(x)$. We will show that the upper envelope



■ **Figure 5** (left) A vertex v visible from the segment ab lying on the bottom chain of H_{ab} , and the triangle Δ_v which contains the portion of ab visible from v . (right) The children u_1 and u_2 of v are visible from ab while w_3 is not. The triangle Δ_v is split into apexed triangles by the rays going from u_1 and u_2 to v .

of the apex functions computed in H_{ab} coincides with $F_P(x)$ inside the Voronoi region $R(v)$ of every vertex $v \in B_{ab}$.

Let T_a and T_b be the shortest-path trees in H_{ab} from a and b rooted at a and b , respectively. We can compute these trees in $O(|H_{ab}|)$ time [11]. For each vertex v such that $f(a), v$ and $f(b)$ are in clockwise order, let v_a and v_b be the neighbors of v in the paths $\pi(v, a)$ and $\pi(v, b)$, respectively. We say that a vertex v is *visible* from ab if $v_a \neq v_b$. Note that if a vertex is visible, then the extension of these segments must intersect the top segment ab . Therefore, for each visible vertex v , we obtain a triangle Δ_v as shown in Figure 5.

We further split Δ_v into a series of triangles with apex at v as follows: Let u be a child of v in either T_a or T_b . As noted by Pollack et al., v can be of three types, either (1) u is not visible from ab (and is hence a child of v in both T_a and T_b); or (2) u is visible from ab , is a child of v only in T_b , and v_bvu is a left turn; or (3) u is visible from ab , is a child of v only in T_a , and v_avu is a right turn.

Let u_1, \dots, u_{k-1} be the children of v of type (2) sorted in clockwise order around v . Let $c(v)$ be the maximum distance from v to any invisible vertex in the subtrees of T_a and T_b rooted at v ; if no such vertex exists, then $c(v) = 0$. Define a function $d_l(v)$ on each vertex v of H_{ab} in a recursive fashion as follows: If v is invisible from ab , then $d_l(v) = c(v)$. Otherwise, let $d_l(v)$ be the maximum of $c(v)$ and $\max\{d_l(u_i) + |u_iv| : u_i \text{ is a child of } v \text{ of type (2)}\}$. Symmetrically, we define a function $d_r(v)$ using the children of type (3) of v .

For each $1 \leq i \leq k-1$, extend the segment u_iv past v until it intersects ab at a point s_i . Let s_0 and s_k be the intersections of the extensions of vv_a and vv_b with the segment ab . We define then k apexed triangles contained in Δ_v as follows. For each $0 \leq i \leq k-1$, consider the triangle $\Delta(s_i, v, s_{i+1})$ whose associated apexed (left) function is

$$f_i(x) = \begin{cases} |xv| + \max_{j>i} \{c(v), |vu_j| + d_l(u_j)\} & \text{if } x \in \Delta(s_i, v, s_{i+1}) \\ -\infty & \text{otherwise} \end{cases}$$

In a symmetric manner, we define a set of apexed triangles induced by the type (3) children of v and their respective apexed (right) functions.

Let g_1, \dots, g_r and $\Delta_1, \dots, \Delta_r$ respectively be an enumeration of all the generated apex functions and apexed triangles such that g_i is defined in the triangle Δ_i . Because each function is determined uniquely by a pair of adjacent vertices in T_a or in T_b , and since these trees have $O(|H_{ab}|)$ vertices, we conclude that $r = O(|H_{ab}|)$.

Note that for each $1 \leq i \leq r$, the apexed triangle Δ_i has two vertices on the segment ab and a third vertex, say a_i , being its apex such that for each $x \in \Delta_i$, $g_i(x) = |\pi(x, w_i)|$ for some vertex w_i of H_{ab} . Recall that w_i is called the *definer* of Δ_i . Intuitively, Δ_i defines a portion of the geodesic distance function from w_i in a constant complexity region.

► **Lemma 12.** *Given a transition edge ab of P , we can compute a set \mathcal{A}_{ab} of $O(|H_{ab}|)$ apexed triangles in $O(|H_{ab}|)$ time with the property that for any point $p \in P$ such that $f(p) \in B_{ab}$, there is an apexed triangle $\triangle \in \mathcal{A}_{ab}$ with apex function g and definer equal to $f(p)$ such that*

1. $p \in \triangle$ and
2. $g(p) = F_P(p)$.

Proof. Because $p \in R(f(p))$, Lemma 11 implies that $p \in H_{ab}$. Consider the path $\pi(p, f(p))$ and let v be the neighbor of p along this path. By construction of \mathcal{A}_{ab} , there is a triangle $\triangle \in \mathcal{A}_{ab}$ apexed at v with definer w that contains p . The apex function $g(x)$ of \triangle encodes the geodesic distance from x to w . Because $F_P(x)$ is the upper envelope of all the geodesic functions, we know that $g(p) \leq F_P(p)$.

To prove the other inequality, note that if $v = f(p)$, then trivially $g(p) = |pv| + |\pi(v, w)| \geq |pv| = |\pi(p, f(p))| = F_P(p)$. Otherwise, let z be the next vertex after v in the path $\pi(p, f(p))$. Three cases arise:

(a) If z is invisible from ab , then so is $f(p)$ and hence,

$$|\pi(p, f(p))| = |pv| + |\pi(v, f(p))| \leq |pv| + c(v) \leq g(p).$$

(b) If z is a child of type (2), then z plays the role of some child u_j of v in the notation used during the construction. In this case:

$$|\pi(p, f(p))| = |pv| + |vz| + |\pi(z, f(p))| \leq |pv| + |vu_j| + d_l(u_j) \leq g(p).$$

(c) If z is a child of type (3), then analogous arguments hold using the (right) distance d_r . Therefore, regardless of the case $F_P(p) = |\pi(p, f(p))| \leq g(p)$.

To bound the running time, note that the recursive functions d_l , d_r and c can be computed in $O(|T_a| + |T_b|)$ time. Then, for each vertex visible from ab , we can process it in time proportional to its degree in T_a and T_b . Because the sum of the degrees of all vertices in T_a and T_b is $O(|T_a| + |T_b|)$ and from the fact that both $|T_a|$ and $|T_b|$ are $O(|H_{ab}|)$, we conclude that the total running time to construct \mathcal{A}_{ab} is $O(|H_{ab}|)$. ◀

In other words, Lemma 12 says that no information on farthest neighbors is lost if we only consider the functions of \mathcal{A}_{ab} within H_{ab} . In the next section we use a similar approach to construct a set of apexed triangles (and their corresponding apex functions), so as to encode the distance from the vertices of M .

5.2 Inside the funnels of marked vertices

We now proceed to split a given funnel into $O(|S_v(C_v)|)$ apexed triangles that encode the distance function from v . To this end, we use the algorithm described by Guibas et al. [12, Section 2] to compute the shortest-path map of v in $S_v(C_v)$ in $O(|S_v(C_v)|)$ time. This algorithm produces a partition of $S_v(C_v)$ into $O(|S_v(C_v)|)$ interior disjoint triangles with vertices on ∂P , such that each triangle consists of all points in $S_v(C_v)$ whose shortest path to v consists of the same sequence of vertices; see Figure 4 (b). Let \triangle be a triangle in this partition and let a be its apex, i.e., the first vertex found along each path $\pi(x, v)$, where $x \in \triangle$. We define the apex function $g_\triangle(x)$ of \triangle as follows:

$$g_\triangle(x) = \begin{cases} |xa| + |\pi(a, v)| & \text{if } x \in \triangle \\ -\infty & \text{otherwise} \end{cases}$$

Therefore, for each $x \in \triangle$, $g_\triangle(x) = |\pi(x, v)|$.

► **Lemma 13.** *The shortest-path map of v in $S_v(C_v)$ can be computed in $O(|S_v(C_v)|)$ time and produces $O(|S_v(C_v)|)$ interior disjoint apexed triangles such that their union covers $S_v(C_v)$. Moreover, for each point $x \in R(v)$, there is an apexed triangle Δ with apex function $g(x)$ such that (1) $x \in \Delta$ and (2) $g(x) = F_P(x)$.*

Proof. The above procedure splits $S_v(C_v)$ into $O(|S_v(C_v)|)$ apexed triangles, such that the apex function in each of them is defined as the geodesic distance to v . By Lemma 9, if $x \in R(v)$, then $x \in S_v(C_v)$. Therefore, there is an apexed triangle Δ with apex function $g(x)$ such that $x \in \Delta$ and $g(x) = |\pi(x, v)| = F_P(x)$. Thus, we obtain properties (1) and (2). ◀

6 Prune and search

With the tools introduced in the previous sections, we can describe the prune and search algorithm to compute the geodesic center. The idea of the algorithm is to partition P into $O(1)$ cells, determine in which cell of P the center lies and recurse on that cell as a new subproblem with smaller complexity.

We can discard all apexed triangles that do not intersect the new cell containing the center. Using cuttings to produce this partition of P , we can show that both the complexity of the cell containing the center, and the number of apexed triangles that intersect it decrease by a constant fraction in each iteration of the algorithm. This process is then repeated until either of the two objects has constant descriptive size.

Let τ be the set of all apexed triangles computed in previous sections. Lemmas 6 and 12 bound the number of apexed triangles constructed inside the transition hourglasses, while Lemmas 10 and 13 do so inside the funnels of the marked vertices. We obtain the following.

► **Corollary 14.** *The set τ consists of $O(n)$ apexed triangles.*

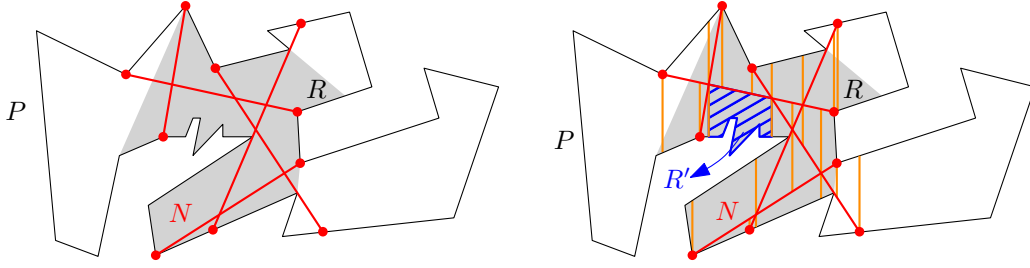
Let $\phi(x)$ be the upper envelope of the apex functions of the triangles in τ (i.e., $\phi(x) = \max\{g(x) : \Delta \in \tau \text{ and } g(x) \text{ is the apex function of } \Delta\}$). The following result is a direct consequence of Lemmas 12 and 13, and shows that the $O(n)$ apexed triangles of τ not only cover P , but their apex functions suffice to reconstruct the function $F_P(x)$.

► **Lemma 15.** *The functions $\phi(x)$ and $F_P(x)$ coincide in the domain of points of P , i.e., for each $p \in P$, $\phi(p) = F_P(p)$.*

Given a chord C of P a *half-polygon* of P is one of the two simple polygons in which C splits P . A *k-cell* of P is a simple polygon obtained as the intersection of at most k half-polygons. Because a *k-cell* is the intersection of geodesically convex sets, it is also geodesically convex.

The recursive algorithm described in this section takes as input a 4-cell (initially equal to P) containing the geodesic center of P and the set of apexed triangles of τ that intersect R . In each iteration, it produces a new 4-cell of smaller complexity that intersects just a fraction of the apexed triangles and contains the geodesic center of P . By recursing on this new cell, the complexity of the problem is reduced in each iteration.

Let R be a 4-cell of P containing the geodesic center of P and let τ_R be the set of apexed triangles of τ that intersect R . Let $m_R = \max\{|R|, |\tau_R|\}$, where $|R|$ denotes the combinatorial complexity of R . Recall that, by construction of the apexed triangles, for each triangle of τ_R at least one and at most two of its boundary segments are chords of P . Let \mathcal{C} be the set containing all chords that belong to the boundary of a triangle of τ_R . Therefore, $|\mathcal{C}| \leq 2|\tau_R| \leq 2m_R$.



■ **Figure 6** The ϵ -net N splits R into $O(1)$ sub-polygons that are further refined into a 4-cell decomposition using $O(1)$ ray-shooting queries from the vertices of the arrangement defined by N .

450 To construct ϵ -nets, we need some definitions (for more information on ϵ -nets refer to [20]).
 451 Let φ be the set of all open 4-cells of P . For each $t \in \varphi$, let $\mathcal{C}_t = \{C \in \mathcal{C} : C \cap t \neq \emptyset\}$ be the
 452 set of chords of \mathcal{C} induced by t . Finally, let $\varphi_{\mathcal{C}} = \{\mathcal{C}_t : t \in \varphi\}$ be the family of subsets of \mathcal{C}
 453 induced by φ . Consider the set system $(\mathcal{C}, \varphi_{\mathcal{C}})$ (denoted by (\mathcal{C}, φ) for simplicity) defined by
 454 \mathcal{C} and φ . The proof of the next lemma is deferred to the Appendix for ease of readability.

455 ► **Lemma 16.** *The set system (\mathcal{C}, φ) has constant VC-dimension.*

456 Let $\epsilon > 0$ (the exact value of ϵ will be specified later). Because the VC-dimension
 457 of the set system (\mathcal{C}, φ) is finite by Lemma 16, we can compute an ϵ -net N of (\mathcal{C}, φ) in
 458 $O(|\mathcal{C}|/\epsilon) = O(m_R)$ time [20]. The size of N is $O(\frac{1}{\epsilon} \log \frac{1}{\epsilon}) = O(1)$ and its main property is
 459 that any 4-cell that does not intersect a chord of N will intersect at most $\epsilon|\mathcal{C}|$ chords of \mathcal{C} .

460 Observe that N partitions R into $O(1)$ sub-polygons (not necessarily 4-cells). We further
 461 refine this partition to obtain 4-cells. That is, we shoot vertical rays up and down from
 462 each endpoint of N , and from the intersection point of any two segments of N , see Figure 6.
 463 Overall, this partitions R into $O(1)$ 4-cells such that each either (i) is a convex polygon
 464 contained in P of at most four vertices, or otherwise (ii) contains some chain of ∂P . Since
 465 $|N| = O(1)$, the whole decomposition can be computed in $O(m_R)$ time (the intersections
 466 between segments of N are done in constant time, and for the ray shooting operations we
 467 walk along the boundary of R once).

468 In order to determine which 4-cell contains the geodesic center of P , we extend each edge
 469 of a 4-cell to a chord C . This can be done with two ray-shooting queries (each of which
 470 takes $O(m_R)$ time). We then use the chord-oracle from Pollack et al. [25, Section 3] to decide
 471 which side of C contains c_P . The only requirement of this technique is that the function
 472 $F_P(x)$ coincides with the upper envelope of the apex functions when restricted to C . Which
 473 is true by Lemma 15 and from the fact that τ_R consists of all the apexed triangles of τ that
 474 intersect R .

475 Because the chord-oracle described by Pollack et al. [25, Section 3] runs in time linear in
 476 the number of functions defined on C , we can decide in total $O(m_R)$ time in which side of C
 477 the geodesic center of P lies. Since our decomposition into 4-cells has constant complexity,
 478 we need to perform $O(1)$ calls to the oracle before determining the 4-cell R' that contains
 479 the geodesic center of P .

480 The chord-oracle computes the minimum of $F_P(x)$ restricted to the chord before determin-
 481 ing the side containing the minimum. In particular, if c_P lies on any chord bounding R' ,
 482 then the chord-oracle will find it. Therefore, we can assume that c_P lies in the interior of R' .
 483 Moreover, since N is a ϵ -net, we know that at most $\epsilon|\mathcal{C}|$ chords of \mathcal{C} intersect R' .

484 We can show that the complexity of R' also decreases: since $|\mathcal{C}| \leq 2|\tau_R| \leq 2m_R$, at most
 485 $2\epsilon m_R$ apexed triangles intersect R' . Because $F_P(x)$ is defined in each point of R' , Lemma 15

implies that each vertex of R' is covered by at least one apexed triangle of τ_R . Since each apexed triangle can cover at most three vertices, by the pigeonhole principle we conclude that R' can have at most $6\varepsilon m_R$ vertices. Otherwise, an apexed triangle would contain at least four vertices of R' . Thus, if we choose $\varepsilon = 1/12$, we guarantee that both the size of the 4-cell R' and the number of apexed triangles in $\tau_{R'}$ are at most $m_R/2$.

In order to proceed with the algorithm on R' recursively, we need to compute the set $\tau_{R'}$ with the at most $\varepsilon|\mathcal{C}|$ apexed triangles of τ_R that intersect R' (i.e., prune the apexed triangles that do not intersect with R'). For each apexed triangle $\triangle \in \tau_R$, we can determine in constant time if it intersects R' (either one of the endpoints is in $R' \cap \partial P$ or the two boundaries have non-empty intersection in the interior of P). Overall, we need $O(m_R)$ time to compute the at most $\varepsilon|\mathcal{C}|$ triangles of τ_R that intersect R' .

By recursing on R' , we guarantee that after $O(\log m_R)$ iterations, we reduce the size of either τ_R or R' to constant. In the former case, the minimum of $F_P(x)$ can be found by explicitly constructing function ϕ in $O(1)$ time. In the latter case, we triangulate R' and apply the chord-oracle to determine which triangle will contain c_P . The details needed to find the minimum of $\phi(x)$ inside this triangle are given in the next section.

► **Lemma 17.** *In $O(n)$ time we can find either the geodesic center of P or a triangle containing the geodesic center.*

7 Finding the center within a triangle

In order to complete the algorithm it remains to show how to find the geodesic center of P for the case in which R' is a triangle. If this triangle is in the interior of P , it may happen that several apexed triangles of τ fully contain R' . Thus, the pruning technique used in the previous section cannot be further applied. We solve this case with a different approach.

Recall that $\phi(x)$ denotes the upper envelope of the apex functions of the triangles in τ , and the geodesic center is the point that minimizes ϕ . The key observation is that, as it happened with chords, the function $\phi(x)$ restricted to R' is convex.

Let $\triangle_1, \triangle_2, \dots, \triangle_m$ be the set of $m = O(n)$ apexed triangles of τ that intersect R' . Let a_i and w_i be the apex and the definer of \triangle_i , respectively. Let $g_i(x)$ be the apex function of \triangle_i such that

$$g_i(x) = \begin{cases} |xa_i| + \kappa_i & \text{if } x \in \triangle_i \\ -\infty & \text{otherwise} \end{cases},$$

where $\kappa_i = |\pi(a_i, w_i)|$ is a constant.

By Lemma 15, $\phi(x) = F_P(x)$. Therefore, the problem of finding the center is equivalent to the following optimization problem in \mathbb{R}^3 :

(P1). Find a point $(x, r) \in \mathbb{R}^3$ minimizing r subject to $x \in R'$ and

$$g_i(x) \leq r, \text{ for } 1 \leq i \leq m.$$

Thus, we need only to find the solution to (P1) to find the geodesic center of P . We use some remarks described by Megiddo in order to simplify the description of (P1) [21].

To simplify the formulas, we square the equation $|xa_i| \leq r - \kappa_i$:

$$\|x\|^2 - 2x \cdot a_i + \|a_i\|^2 = |xa_i|^2 \leq (r - \kappa_i)^2 = r^2 - 2r\kappa_i + \kappa_i^2.$$

And finally for each $1 \leq i \leq m$, we define the function $h_i(x, r)$ as follows:

$$h_i(x, r) = \begin{cases} \|x\|^2 - 2x \cdot a_i + \|a_i\|^2 - r^2 + 2r\kappa_i - \kappa_i^2 & \text{if } x \in \triangle_i \\ -\infty & \text{otherwise} \end{cases}.$$

Therefore, our optimization problem can be reformulated as:

(P2). Find a point $(x, r) \in \mathbb{R}^3$ such that r is minimized subject to $x \in R'$ and

$$h_i(x, r) \leq 0 \text{ and } r > \max\{\kappa_i\}, \text{ for } 1 \leq i \leq m.$$

Let $h'_i(x, r) = \|x\|^2 - 2x \cdot a_i + \|a_i\|^2 - r^2 + 2r\kappa_i - \kappa_i^2$ be a function defined in the entire plane and let (P2') be an optimization problem analogous to (P2) where every instance of $h_i(x, r)$ is replaced by $h'_i(x, r)$. The optimization (P2') was studied by Megiddo in [21]. We provide some of the intuition used by Megiddo to solve this problem.

Although the functions $h'_i(x, r)$ are not linear, they all have the same non-linear terms. Therefore, for $i \neq j$, we get that $h'_i(x, r) = h'_j(x, r)$ defines a *separating plane*

$$\gamma_{i,j} = \{(x, r) \in \mathbb{R}^3 : 2(\kappa_i - \kappa_j)r - 2(a_i - a_j) \cdot x + \|a_i\|^2 - \|a_j\|^2 - \kappa_i^2 + \kappa_j^2 = 0\}$$

As noted by Megiddo [21], this separating plane has the following property: If the solution (x, r) to (P2') is known to lie to one side of $\gamma_{i,j}$, then we know that one of the constraints is redundant.

Thus, to solve (P2') it sufficed to have a *side-decision oracle* to determine in which side of a plane $\gamma_{i,j}$ the solution lies. Megiddo showed how to implement this oracle in a way that the running time is proportional to the number of constraints [21].

Once we have such an oracle, Megiddo's problem can be solved using a prune and search approach: pair the functions arbitrarily, and consider the set of $m/2$ separating planes defined by these pairs. For some constant t , compute a $1/t$ -cutting in \mathbb{R}^3 of the separating planes. A $1/t$ -cutting is a partition of the plane into $O(t^3) = O(1)$ convex regions each of which is of constant complexity and intersects at most $m/2t$ separating planes. A cutting of planes can be computed in linear time in \mathbb{R}^3 for any $t = O(1)$ [18]. After computing the cutting, determine in which of the regions the minimum lies by performing $O(1)$ calls to the side-decision oracle. Because at least $(t - 1)m/2t$ separating planes do not intersect this constant complexity region, for each of them we can discard one of the constraints as it becomes redundant. Repeating this algorithm recursively we obtain a linear running time.

To solve (P2) we follow a similar approach, but our set of separating planes needs to be extended in order to handle apex functions as they are only defined in the same way as in (P2') in a triangular domain. Note that no vertex of an apexed triangle can lie inside R' .

7.1 Optimization problem in a convex domain

In this section we describe our algorithm to solve the optimization problem (P2). To this end, we pair the apexed triangles arbitrarily to obtain $m/2$ pairs. By identifying the plane where P lies with the plane $Z_0 = \{(x, y, z) : z = 0\}$, we can embed each apexed triangle in \mathbb{R}^3 . A *plane-set* is a set consisting of at most five planes in \mathbb{R}^3 . For each pair of apexed triangles $(\triangle_i, \triangle_j)$ we define its plane-set as follows: For each chord of P bounding either \triangle_i or \triangle_j (at most two chords on each triangle), consider the line extending this chord and the vertical extrusion of this line in \mathbb{R}^3 , i.e., the plane containing this chord orthogonal to Z_0 . Moreover, consider the separating plane $\gamma_{i,j}$. The set containing these planes is the plane-set of the pair $(\triangle_i, \triangle_j)$.

Let Γ be the union of all the plane-sets defined by the $m/2$ pairs of apexed triangles. Because the plane-set of each pair $(\triangle_i, \triangle_j)$ consists of at most five planes and contains at least one plane unique to this pair, say $\gamma_{i,j}$, we infer that $m/2 \leq |\Gamma| \leq 5m/2$.

Compute a $1/t$ -cutting of Γ in $O(m)$ time for some constant t to be specified later. Because t is constant, this $1/t$ -cutting splits the space into $O(1)$ convex regions, each bounded by a

constant number of planes [18]. Using a side-decision algorithm (to be specified later), we can determine the region Q of the cutting that contains the solution to (P2). Because Q is the region of a $1/t$ -cutting of Γ , we know that at most $|\Gamma|/t$ planes of Γ intersect Q . In particular, at most $|\Gamma|/t$ plane-sets intersect Q and hence, at least $(t-1)|\Gamma|/t$ plane-sets do not intersect Q . Since $|\Gamma| \geq m/2$, at least $(t-1)m/2t$ plane-sets do not intersect Q .

Let (Δ_i, Δ_j) be a pair such that its plane-set does not intersect Q . Let Q' be the projection of Q on the plane Z_0 . Because the plane-set of this pair does not intersect Q , we know that Q' intersects neither the boundary of Δ_i nor that of Δ_j . Two cases arise:

Case 1. If either Δ_i or Δ_j does not intersect Q' , then we know that their apex function is redundant and we can drop the constraint associated with this apexed triangle.

Case 2. If $Q' \subset \Delta_i \cap \Delta_j$, then we need to decide which constrain to drop. To this end, we consider the separating plane $\gamma_{i,j}$. Notice that inside the vertical extrusion of $\Delta_i \cap \Delta_j$ (and hence in Q), the plane $\gamma_{i,j}$ has the property that if we know its side containing the solution, then one of the constraints can be dropped. Since $\gamma_{i,j}$ does not intersect Q as $\gamma_{i,j}$ belongs to the plane-set of (Δ_i, Δ_j) , we can decide which side of $\gamma_{i,j}$ contains the solution to (P2) and drop one of the constraints.

Regardless of the case, if the plane-set of a pair (Δ_i, Δ_j) does not intersect Q , then we can drop one of its constraints. Since at least $(t-1)m/2t$ plane-sets do not intersect Q , we can drop at least $(t-1)m/2t$ constraints. By choosing $t = 2$, we are able to drop at least $(t-1)m/2t = m/4$ constraints. Consequently, after $O(m)$ time, we are able to drop $m/4$ apexed triangles. By repeating this process recursively, we end up with a constant size problem in which we can compute the upper envelope of the functions explicitly and find the solution to (P2) using exhaustive search. Thus, the running time of this algorithm is bounded by the recurrence $T(m) = T(3m/4) + O(m)$ which solves to $O(m)$. Because $m = O(n)$, we can find the solution to (P2) in $O(n)$ time.

The last detail is the implementation of the side-decision algorithm. Given a plane γ , we want to decide in which side lies the solution to (P2). To this end, we solve (P2) restricted to γ , i.e., with the additional constraint of $(x, r) \in \gamma$. This approach was used by Megiddo [21], the idea is to recurse by reducing the dimension of the problem. Another approach is to use a slight modification of the chord-oracle described by Pollack et al. [25, Section 3].

Once the solution to (P2) restricted to γ is known, we can follow the same idea used by Megiddo [21] to find the side of γ containing the global solution to (P2). Find the apex functions that define the minimum restricted to γ . Since $\phi(x) = F_P(x)$ is locally defined by these functions, we can decide in which side the minimum lies using convexity. We obtain the following result.

► **Lemma 18.** *Let R' be a convex trapezoid contained in P such that R' contains the geodesic center of P . Given the set of all apexed triangles of τ that intersect R' , we can compute the geodesic center of P in $O(n)$ time.*

The following theorem summarizes the result presented in this paper.

► **Theorem 19.** *We can compute the geodesic center of any simple polygon P of n vertices in $O(n)$ time.*

References

- 1 B. Aronov. On the geodesic Voronoi diagram of point sites in a simple polygon. *Algorithmica*, 4(1-4):109–140, 1989.

- 600 **2** B. Aronov, S. Fortune, and G. Wilfong. The furthest-site geodesic Voronoi diagram. *Discrete & Computational Geometry*, 9(1):217–255, 1993.
- 601
- 602 **3** T. Asano and G. Toussaint. Computing the geodesic center of a simple polygon. Technical Report SOCS-85.32, McGill University, 1985.
- 603
- 604 **4** S. W. Bae, M. Korman, and Y. Okamoto. The geodesic diameter of polygonal domains. *Discrete & Computational Geometry*, 50(2):306–329, 2013.
- 605
- 606 **5** S. W. Bae, M. Korman, and Y. Okamoto. Computing the geodesic centers of a polygonal domain. In *Proceedings of CCCG*, 2014.
- 607
- 608 **6** S. W. Bae, M. Korman, Y. Okamoto, and H. Wang. Computing the L_1 geodesic diameter and center of a simple polygon in linear time. In *Proceedings of LATIN*, pages 120–131, 2014.
- 609
- 610
- 611 **7** B. Chazelle. A theorem on polygon cutting with applications. In *Proceedings of FOCS*, pages 339–349, 1982.
- 612
- 613 **8** B. Chazelle. Triangulating a simple polygon in linear time. *Discrete & Computational Geometry*, 6(1):485–524, 1991.
- 614
- 615 **9** H. Djidjev, A. Lingas, and J.-R. Sack. An $O(n \log n)$ algorithm for computing the link center of a simple polygon. *Discrete & Computational Geometry*, 8:131–152, 1992.
- 616
- 617 **10** H. Edelsbrunner and E. P. Mücke. Simulation of simplicity: a technique to cope with degenerate cases in geometric algorithms. *ACM Transactions on Graphics*, 9(1):66–104, 1990.
- 618
- 619
- 620 **11** L. Guibas, J. Hershberger, D. Leven, M. Sharir, and R. E. Tarjan. Linear-time algorithms for visibility and shortest path problems inside triangulated simple polygons. *Algorithmica*, 2(1-4):209–233, 1987.
- 621
- 622
- 623 **12** L. J. Guibas and J. Hershberger. Optimal shortest path queries in a simple polygon. *Journal of computer and system sciences*, 39(2):126–152, 1989.
- 624
- 625 **13** H. Harborth and M. Möller. *The Esther-Klein-problem in the projective plane*. Inst. für Mathematik, TU Braunschweig, 1993.
- 626
- 627 **14** D. Harel and R. E. Tarjan. Fast algorithms for finding nearest common ancestors. *SIAM Journal on Computing*, 13(2):338–355, 1984.
- 628
- 629 **15** J. Hershberger and S. Suri. Matrix searching with the shortest-path metric. *SIAM Journal on Computing*, 26(6):1612–1634, 1997.
- 630
- 631 **16** Y. Ke. An efficient algorithm for link-distance problems. In *Proceedings of SoCG*, pages 69–78, 1989.
- 632
- 633 **17** D.-T. Lee and F. P. Preparata. Euclidean shortest paths in the presence of rectilinear barriers. *Networks*, 14(3):393–410, 1984.
- 634
- 635 **18** J. Matoušek. Approximations and optimal geometric divide-and-conquer. *Journal of Computer and System Sciences*, 50(2):203–208, 1995.
- 636
- 637 **19** J. Matoušek. *Lectures on discrete geometry*, volume 108. Springer New York, 2002.
- 638 **20** J. Matoušek. Construction of epsilon nets. In *Proceedings of SoCG*, pages 1–10, New York, 1989. ACM.
- 639
- 640 **21** N. Megiddo. On the ball spanned by balls. *Discrete & Computational Geometry*, 4(1):605–610, 1989.
- 641
- 642 **22** J. S. B. Mitchell. Geometric shortest paths and network optimization. In J.-R. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*, pages 633–701. Elsevier, 2000.
- 643
- 644 **23** B. Nilsson and S. Schuierer. Computing the rectilinear link diameter of a polygon. In *Proceedings of CG*, pages 203–215, 1991.
- 645
- 646 **24** B. Nilsson and S. Schuierer. An optimal algorithm for the rectilinear link center of a rectilinear polygon. *Computational Geometry: Theory and Applications*, 6:169–194, 1996.
- 647
- 648 **25** R. Pollack, M. Sharir, and G. Rote. Computing the geodesic center of a simple polygon. *Discrete & Computational Geometry*, 4(1):611–626, 1989.
- 649

- 650 26 S. Suri. *Minimum Link Paths in Polygons and Related Problems*. PhD thesis, Johns Hopkins
 651 Univ., 1987.
- 652 27 S. Suri. Computing geodesic furthest neighbors in simple polygons. *Journal of Computer
 653 and System Sciences*, 39(2):220–235, 1989.
- 654 28 P. Turán. On an extremal problem in graph theory. *Mat. Fiz. Lapok*, 48(436-452):137,
 655 1941.

656 A Bounding VC dimension

657 In this section we provide the proof of Lemma 16. That is, we want to prove that the set
 658 system (\mathcal{C}, φ) has constant VC-dimension. Recall that \mathcal{C} is a set of chords of P and φ is the
 659 set of all open 4-cells of P .

660 Let $A \subseteq \mathcal{C}$ be a subset of chords. We say that A is *shattered* by φ if each of the subsets
 661 of A can be obtained as the intersection of some $S \in \varphi$ with A , i.e., if for each $\sigma \subseteq A$, there
 662 exists $S \in \varphi$ such that $\sigma = S \cap A$. The *VC-dimension* of (\mathcal{C}, φ) is the supremum of the sizes
 663 of all finite shattered subsets of \mathcal{C} .

664 Let $\mathcal{H} = \{H : H \text{ is a half-polygon of } P\}$. Because each 4-cell of P is the intersection of
 665 at most four half-polygons of P , the following result is a direct consequence of Proposition
 666 10.3.3 of [19, Chapter 10].

667 ► **Lemma 20.** *If $(\mathcal{C}, \mathcal{H})$ has VC-dimension d , then (\mathcal{C}, φ) has VC-dimension $O(d)$.*

668 Let A be an arbitrary subset of \mathcal{C} such that $|A| = \kappa$ for some constant $\kappa > 6$ to
 669 be determined later. Recall that if no subset of \mathcal{C} of size κ is shattered by \mathcal{H} , then the
 670 VC-dimension of $(\mathcal{C}, \mathcal{H})$ is at most κ .

671 By Lemma 20, it suffices to show that A is not shattered by \mathcal{H} to bound the VC-dimension
 672 of $(\mathcal{C}, \mathcal{H})$ and hence of (\mathcal{C}, φ) . We spend the remainder of this section proving that A is not
 673 shattered by \mathcal{H} .

674 For any $B \subseteq A$, notice that B partitions P into sub-polygons.

675 ► **Lemma 21.** *Let $B \subseteq A$. If there is a 6-cell σ among the sub-polygons in which B
 676 partitions P , then A is not shattered by \mathcal{H} .*

677 **Proof.** Let C_1, \dots, C_6 be the chords supporting the six half-polygons whose intersection
 678 defines σ . Assume that C_1, \dots, C_6 appear in these order when walking clockwise along the
 679 boundary of σ . Note that any half-polygon that intersects C_1, C_3 and C_5 must intersect
 680 either C_2, C_4 or C_6 . Therefore, the set $\{C_1, C_3, C_5\}$ cannot be obtained as the intersection
 681 of some half-polygon $H \in \mathcal{H}$ with A . Because $\{C_1, C_3, C_5\}$ is also a subset of A , we conclude
 682 that A is not shattered by \mathcal{H} . ◀

683 Given two chords C_1 and C_2 of A , we say that C_1 and C_2 are *separated* if there exists a
 684 chord $S \in A$ such that C_1 and C_2 lie on different open half-polygons supported by S .

685 Note that if A contains two chords C_1 and C_2 that are separated, then the subset $\{C_1, C_2\}$
 686 cannot be obtained as the intersection of a half-polygon $H \in \mathcal{H}$ with A , i.e., A is not shattered
 687 by \mathcal{H} . Therefore, we assume from now on that no two chords of A are separated

688 Let G be the intersection graph of A , i.e., the graph with vertex set A and an edge
 689 between two chords if they intersect. An Erdős-Szekeres type result from Harborth and
 690 Möller [13] shows that every arrangement of nine pseudo-lines determines a subarrangement
 691 with a hexagonal face. Thus, if G has a clique of size nine, then that subset of chords is a set
 692 of pseudo-lines. Therefore, it contains a subset of chords that splits P into sub-polygons,

one of which is a 6-cell. In this case, Lemma 21 implies that A is not shattered by \mathcal{H} . Consequently, we assume from now on that G has no clique of size nine.

Turán's Theorem [28] states that if G has no clique of size nine, then it has at most $(7/16)\kappa^2$ edges. Let C_1 be the chord in A with the smallest degree in G . Notice that C_1 has degree at most $7\kappa/16$. Therefore, there are at least $9\kappa/16$ chords of A that do not intersect C_1 . Consider the two half-polygons supported by C_1 and note that one of them, say P' , contains at least $9\kappa/32$ chords of A that do not intersect C_1 . Let A' be the set containing these chords.

Let G' be subgraph of G induced by A' and let C_2 be the chord of A' with smallest degree in G' . Because G' has no clique of size nine, we infer that C_2 has degree at most $7|A'|/16$. Repeating the same argument, there is a set A'' of at least $9|A'|/16$ chords of A' that intersect neither C_2 nor C_1 . Because no two chords of A are separated, all chords in A'' are contained in the half-polygon supported by C_2 that contains C_1 . Otherwise, C_1 and some of these chords are separated.

Let G'' be the subgraph of G induced by A'' . Repeating the above procedure recursively on G'' and A'' four more times, we obtain a set C_1, \dots, C_6 of pairwise disjoint chords such that for each $1 \leq i \leq 6$, C_1, \dots, C_{i-1} are contained in the same half-polygon supported by C_i . Consequently, the set $\{C_1, \dots, C_6\} \subseteq A$ partitions P into sub-polygons one of which is a 6-cell bounded by the chords C_1, \dots, C_6 .

Note that the above construction can be applied as long as

$$\left(\frac{9}{32}\right) \left(\frac{9}{16}\right)^4 \kappa \geq 1.$$

That is, as long as $|A| \geq 36$ we can always find a subset $B \subseteq A$ such that there is a 6-cell among the sub-polygons in which B partitions P . Therefore, as long as $|A| \geq 36$, A is not shattered by \mathcal{H} , i.e., the set system $(\mathcal{C}, \mathcal{H})$ has VC-dimension at most 36. By Lemma 20, we obtain the following result.

► **Lemma 16.** *The set system (\mathcal{C}, φ) has constant VC-dimension.*