Optimal Algorithm for Geodesic Farthest-Point Voronoi Diagrams

Luis Barba
Department of Computer Science, ETH Zürich, Switzerland
luis.barba@inf.ethz.ch

Abstract

Let P be a 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. Given a set $\mathcal{S}$ of m sites being a subset of the vertices of P, we present the first randomized algorithm to compute the geodesic farthest-point Voronoi diagram of $\mathcal{S}$ in P running in expected $O(n + m)$ time. That is, a partition of P into cells, at most one cell per site, such that every point in a cell has the same farthest site with respect to the geodesic distance. This algorithm can be extended to run in expected $O(n + m \log m)$ time when $\mathcal{S}$ is an arbitrary set of m sites contained in P.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Geodesic distance, simple polygons, farthest-point Voronoi diagram

Digital Object Identifier 10.4230/LIPIcs.SoCG.2019.12

Related Version A full version of this paper is available at `https://arxiv.org/abs/1809.01481`.

1 Introduction

Let P be a simple n-gon. Let $\mathcal{S}$ be a set of $m \geq 3$ weighted *sites* (points) contained in $V(P)$, where $V(P)$ denotes the set of vertices of P. That is, we have a function $w : \mathcal{S} \to \mathbb{R}$ that assigns to each site of $\mathcal{S}$ a non-negative weight. We also extend the weight function to any point in P by setting $w(x) = 0$ for all $x \in P \setminus \mathcal{S}$. While we could allow the sites to lie anywhere on the boundary, ∂P, of P, as long as we know their clockwise order along ∂P, we can split the edges of P at the sites, and produce a new polygon where each site coincides with a vertex. Therefore, we assume that $\mathcal{S} \subseteq V(P)$.

Given two points x, y in P (either on the boundary or in the interior), the *geodesic path* $\pi_P(x, y)$ 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 $\pi_P(x, y)$ is the straight-line segment xy. Otherwise, $\pi_P(x, y)$ is a polygonal chain whose vertices (other than its endpoints) are reflex vertices of P. We refer the reader to [14] for more information on geodesic paths.

For a segment xy, we denote its Euclidean length by $|xy|$. For a path, its *Euclidean length* is the sum of the Euclidean length of all of its segments. Given two points x and y in P, their *geodesic distance* $\mathrm{G}^P(x, y)$ is the Euclidean length of $\pi_P(x, y)$. The *weighted geodesic distance* (or simply *w-distance*) between two points x and y in P, denoted by $\mathrm{D}^P_w(x \rightsquigarrow y)$, is the sum of $w(x)$ with the Euclidean length of $\pi_P(x, y)$, i.e., $\mathrm{D}^P_w(x \rightsquigarrow y) = w(x) + \mathrm{G}^P(x, y)$. Notice that if all weights are set to zero, then the w-distance coincides with the classical definition of geodesic distance [14]. Moreover, notice that this distance is not symmetric unless the weights of x and y coincide.

Given a point $x \in P$, an *$\mathcal{S}$-farthest site* of x in P is a site s of $\mathcal{S}$ whose w-distance to x is maximized. To ease the description, we assume that each vertex of P has a unique $\mathcal{S}$-farthest neighbor. This *general position* condition was also assumed in [1, 3, 18] and can be obtained by applying a slight perturbation [10].

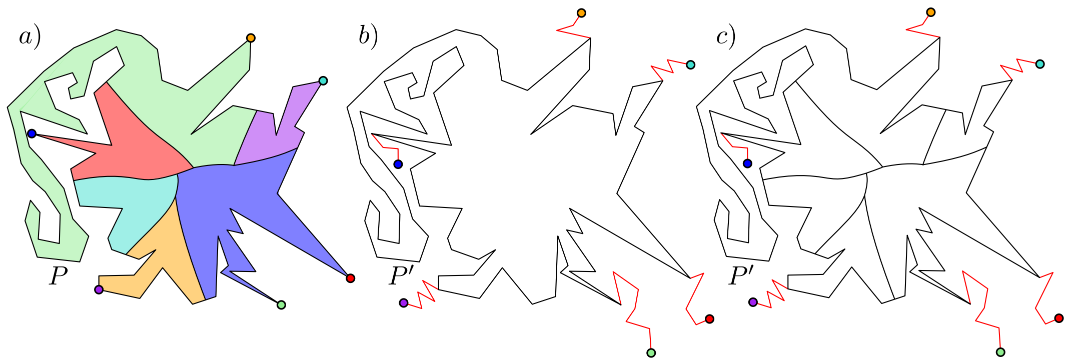

Figure 1 $a)$ A simple polygon P with a set $\mathcal{S}$ of six weighted sites and their FVD. $b)$ A new polygon P' where a path of length $w(s)$ is added at the location of each site $s \in \mathcal{S}$. Then the weight is set to zero and moved at the endpoint of its corresponding path. $c)$ The FVD of $\mathcal{S}$ in P coincides with the FVD of the new sites in the new polygon P'.

For a site $s \in \mathcal{S}$, let $\mathtt{Cell}_P(s, \mathcal{S}) = \{x \in P : \mathrm{D}^P_w(s \rightsquigarrow x) \geq \mathrm{D}^P_w(s' \rightsquigarrow x), \forall s' \in \mathcal{S}\}$ be the (weighted farthest) *Voronoi cell* of s (in P with respect to $\mathcal{S}$). That is, $\mathtt{Cell}_P(s, \mathcal{S})$ consists of all the points of P that have s as one of their $\mathcal{S}$-farthest sites. The union of all Voronoi cells covers the entire polygon P, and the closure of the set $\mathrm{int}(P) \setminus \cup_{s \in \mathcal{S}} \mathrm{int}(\mathtt{Cell}_P(s, \mathcal{S}))$ defines the (weighted farthest) *Voronoi graph* of $\mathcal{S}$ in P.

The Voronoi graph together with the set of Voronoi cells defines the *weighted geodesic farthest-point Voronoi diagram* (or simply *FVD*) of $\mathcal{S}$ in P, denoted by $\mathrm{VD}(\mathcal{S}, P)$. Thus, we indistinctively refer to $\mathrm{VD}(\mathcal{S}, P)$ as a graph or as a set of Voronoi cells; see Figure 1.

Notice that having non-zero weights on our set of sites does not make the problem harder. To see this, consider a new polygon P', where at the location of each site $s \in \mathcal{S}$, a path of length $w(s)$ is attached to the boundary of P. Additionally, the site s is given weight zero and is moved to the other endpoint of this path; see Figure 1 for an illustration. In this way we obtain a new weakly simple polygon P' and a new set of sites $\mathcal{S}'$ that defines the same FVD as $\mathcal{S}$ in P. Therefore, a weighted FVD as described in this paper has the same properties as the classical farthest-point Voronoi diagram constructed using the geodesic distance [3]. In particular, we know that the Voronoi graph is a tree with leaves on the boundary of P. Also, each edge of this graph consists of a sequence of straight lines and hyperbolic arcs that may intersect ∂P only at its endpoints [3]. Thus, we refer to the Voronoi graph as a *Voronoi tree*. While working with weighted sites might seem an unnecessary complication, we decided to work with them to ease the description of the recursive construction that our algorithm uses.

Let $F_P(x, \mathcal{S})$ be the function that maps each $x \in P$ to the w-distance to a $\mathcal{S}$-farthest neighbor of x (i.e., $F_P(x, \mathcal{S}) = \mathrm{D}^P_w(x \rightsquigarrow f_P(x, \mathcal{S}))$). Notice that $F_P(x, \mathcal{S})$ can be seen as the upper envelope of the w-distance functions from the sites in $\mathcal{S}$. Throughout the paper, we will play with this alternative way of thinking of Voronoi diagrams, either as graphs or as upper envelopes. A point $c \in P$ that minimizes $F_P(x, \mathcal{S})$ is called the *geodesic center* of P. Similarly, a point $s \in P$ that maximizes $F_P(x, \mathcal{S})$ (together with $f_P(s)$) forms a *diametral pair* and their w-distance is the *geodesic diameter*.

Related work. The problem of computing the geodesic center of a simple n-gon P (and its counterpart, the geodesic diameter) were central in the 80's in the computational geometry community. Chazelle [7] provided the first $O(n^2)$-time algorithm to compute the geodesic diameter. Suri [21] improved upon it by reducing the running time to $O(n \log n)$. Finally, Hershberger and Suri [12] introduced a matrix search technique that allowed them to obtain a linear-time algorithm for computing the diameter.

The first algorithm for computing the geodesic center of P was given by Asano and Toussaint [4], and runs in $O(n^4 \log n)$ time. This algorithm computes a super set of the vertices of the Voronoi tree of $\textsc{vd}(\mathcal{S}, P)$, where $\mathcal{S}$ is the set of vertices of P. Shortly after, Pollack et al. [20] improved the running time to $O(n \log n)$. This remained the best running time for many years until recently when Ahn et al. [1] settled the complexity of this problem by presenting a $\Theta(n)$-time algorithm to compute the geodesic center of P.

The problem of computing the FVD generalizes the problems of computing the geodesic center and the geodesic diameter. For a set $\mathcal{S}$ of $m \geq 3$ sites in a simple n-gon P, Aronov [3] presented an algorithm to compute $\textsc{vd}(\mathcal{S}, P)$ in $O((n+m)\log(n+m))$ time. While the best known lower bound is $\Omega(n + m \log m)$, it was not known whether or not the dependence on n, the complexity of P, is linear in the running time. In fact, this problem was explicitly posed by Mitchell [14, Chapter 27] in the Handbook of Computational Geometry, and solving it has become a prominent area of research in recent years. Oh et al. [18] (SoCG'16) present the first improvement to this problem in more than 20 years. Using the new tools presented by Ahn et al. [1], they introduce an $O(n \log\log n + m \log m)$-time algorithm to compute $\textsc{vd}(\mathcal{S}, P)$. As a stepping stone, they present an $O((n+m)\log\log n)$-time algorithm for the simpler case where all sites are vertices of P. In fact, any improvement on the latter algorithm translates directly to an improvement on the general problem. In particular, a linear time algorithm for the simpler case with sites on the boundary of P suffices to match the lower bound and close the problem presented by Mitchell [14, Chapter 27] in the case of farthest-point Voronoi diagrams.

Recently, not only farthest-point Voronoi diagrams have received attention. For the nearest-point geodesic Voronoi diagram, two papers have focused in finding algorithms matching the same lower bound of $\Omega(n + m \log m)$ [13, 16]. While these results work only for a limited range of m with respect to n, both papers have appeared in consecutive years in the Symposium on Computational Geometry (SoCG). In a recent breakthrough to appear in SODA'19, Oh [15] presents an optimal algorithm to compute the nearest-point geodesic Voronoi diagram running in $O(n + m \log m)$ time. However, the techniques in these papers are fundamentally based on data structures with logarithmic query time, and hence it is not conceivable to adapt them to obtain a linear-time algorithm for the case when the sites are vertices of the polygon.

Our results. In this paper, we provide an optimal, albeit randomized, algorithm to compute $\textsc{vd}(\mathcal{S}, P)$ for the important case where all sites of $\mathcal{S}$ are vertices of P. Our algorithm runs in expected $\Theta(n+m)$ time, and uses a completely different set of tools to solve the problem. Using the reduction presented by Oh et al. [18], we immediately obtain an algorithm for the general case where the sites can be arbitrary points in P. This algorithm matches the lower bound and runs in expected $\Theta(n + m \log m)$ time thereby solving the problem posed by Mitchell [14, Chapter 27] in the case of farthest-point Voronoi diagrams. It remains open to find a deterministic algorithm with the same running time.

Our approach. Let P be a simple n-gon and let $\mathcal{S}$ be a set of $m \geq 3$ sites contained in $V(P)$, where $V(P)$ is the set of vertices of P. We present a randomized $O(n+m)$-time algorithm to compute the FVD of $\mathcal{S}$ in P. We would like to use a variation of the randomized incremental construction (*RIC*) for Euclidean farthest-point Voronoi diagrams [9]. This algorithm inserts the sites, one by one, in random order, and constructs the cell of each newly inserted site in time proportional to its size. By bounding the expected size of each cell using backwards analysis, the incremental construction can be carried out in total linear time.

In the geodesic case however, the complexity of a cell depends not only on the set of sites, but also on the complexity of the polygon [2]. Already the FVD of 3 sites can have $\Omega(n)$ vertices and arcs. Moreover, there is an additional complication when using w-distances. To achieve an incremental construction, one would need to have at hand a complete description of the w-distance function $\mathrm{D}_w^P(s \rightsquigarrow x)$ inside of the newly created cell for the inserted site s. If this function is precomputed in the entire polygon, this would be too costly. Thus, one needs to define these functions only at the specific locations where they are needed. An additional problem is that for a RIC, the first inserted sites must have their w-distance defined in almost the entire polygon. Thus, already the description-size of the w-distances needed for the first batch of sites (say the first $m/100$) becomes superlinear. Therefore, it seems hopeless to try a RIC without somehow reducing the complexity of P throughout the process. Nevertheless, a RIC works well for all the sites that come after this first batch. Intuitively, the latter insertions define smaller cells, and the space needed to describe their w-distances can be nicely bounded. Thus, the main question is how to deal with this first fraction of the sites.

In this paper we overcome these difficulties with a novel approach, and manage to deal with this first fraction of the sites using pruning. First, we randomly partition the sites into B and R, where $|B| \leq \alpha m$ for some constant $0 < \alpha < 1$ (Section 3.1). Then, we construct recursively an "approximation" of the FVD of B (Section 3.2). To this end, we define a new weakly simple polygon Q containing B with only a constant fraction of the vertices of P. Essentially we prune from P all the vertices that have nothing to do with geodesic paths connecting sites in B with points in their respective Voronoi cells. Our approximation comes from recursively computing the FVD of B in Q. We show that the complexity of Q decreases sufficiently so that the recursive call leads to a linear overall running time.

However, reducing the complexity comes with a price. The w-distance from sites of B inside of Q turns out to be only "similar" to that in P. However, we make sure that these functions are accurate where it matters. After computing this "Voronoi-like" diagram for B, we need to deal with the sites of R. To this end, we turn to the RIC (Section 4). We compute the w-distance from sites in R only inside of specific parts of P, making sure that they suffice for our purpose, while their overall complexity remains linear. Another challenge comes from the fact that the w-distances from B are with respect to Q, while the ones from R are not. Thus, we need to prove that the upper envelope of these functions induces a Voronoi diagram. Once we deal with these technical details, we end up with an upper envelope of functions that we prove to coincide with the FVD of $\mathcal{S}$ in P, finishing our construction.

We show that the insertion of each site $r \in R$ can be carried out in expected $O(n/m)$ time. Thus, inserting all sites of R can be done in expected $O(n + m)$ time. After inserting the sites of R, the expected total running time of our algorithm is given by the simple recurrence $\mathrm{E}[T(n, m)] \leq T(n/2, m/2) + O(n + m) = O(n + m)$. The crucial aspect of our approach that could not be achieved before this paper, is the reduction in the complexity of the polygon. Overall, we combine many different tools, from recursion, pruning, and randomization, together with all the machinery to deal with geodesic functions. Due to space constraints, the proof of all results marked with [$*$] have been omitted and can be found in the full version of this paper [5].

2 Preliminaries

Let P be a simple n-gon and let $\mathcal{S}$ be a set of $m \geq 3$ sites contained in $V(P)$. Because $\mathcal{S} \subseteq V(P)$, we know that $m = |\mathcal{S}| \leq n$.

A subset $G \subseteq P$ is *geodesically convex* in P if for each $x, y \in G$, the geodesic path between x and y is contained in G, i.e., if $\pi_P(x, y) \subseteq G$. Given a set A of points in P, the *geodesic hull* of A in P is the minimum geodesically convex set in P that contains A. In particular, if $A \subseteq \partial P$, then the boundary of the geodesic hull of A is obtained by joining consecutive points of A along ∂P by the geodesic path between them. Note that this geodesic hull is not necessarily a simple polygon but a weakly simple polygon. Geodesic functions in weakly simple polygons behave in the exact same way as in simple polygons, and the existing machinery applies directly with no overhead [6]. Thus, while many papers state their results for simple polygons, they apply directly to weakly simple polygons. In particular, all results and tools presented in this paper apply directly to weakly simple polygons. This remark is already crucial in several recent papers [17, 18].

▶ **Lemma 1** (Restatement of Lemma 2 of [19]). *Let $A \subseteq \partial P$ be a set of $O(n)$ points sorted along ∂P. Then the geodesic hull of A in P can be computed in $O(n)$ time.*

Let $\text{VD}_\partial(\mathcal{S}, P)$ be the FVD of $\mathcal{S}$ restricted to the boundary of P. More formally, for each $s \in \mathcal{S}$, let $\texttt{bCell}_P(s, \mathcal{S}) = \texttt{Cell}_P(s, \mathcal{S}) \cap \partial P$ be the *boundary cell* of s and let $\text{VD}_\partial(\mathcal{S}, P)$ be the union of these boundary cells. The construction of $\text{VD}_\partial(\mathcal{S}, P)$ has often been a stepping stone in the computation of $\text{VD}(\mathcal{S}, P)$ [3, 18], and our algorithm follows the same approach. The following result from [18] allows us to compute it efficiently.

▶ **Theorem 2** (Theorem 9 of [18]). *Let P be an n-gon and let $\mathcal{S} \subseteq V(P)$ be a set of sites. Then, we can compute $\text{VD}_\partial(\mathcal{S}, P)$ in $O(n)$ time.*

Using this procedure, we can find out in $O(n)$ time which sites of $\mathcal{S}$ have a non-empty Voronoi cell. Therefore, we can forget about the sites with empty cells and assume without loss of generality from now on that all sites of $\mathcal{S}$ have non-empty Voronoi cells.

Given a site $s \in \mathcal{S}$ and a polygonal chain $C \subseteq \partial P$ with endpoints p and p', the *funnel* of s to C in P, denoted by $\texttt{Funnel}_P(s \to C)$, is the geodesic hull of s and C in P. It is known that $\texttt{Funnel}_P(s \to C)$ coincides with the weakly simple polygon contained in P bounded by C, $\pi_P(s, p')$ and $\pi_P(s, p)$ [1]. For ease of notation, we denote $\texttt{Funnel}_P(s \to \texttt{bCell}_P(s, \mathcal{S}))$ simply by $\texttt{Funnel}_P(s, \mathcal{S})$, i.e., the funnel with apex s that goes to $\texttt{bCell}_P(s, \mathcal{S})$. The following lemma shows the relation between Voronoi cells and their funnels.

▶ **Lemma 3** (Consequence of Lemma 4.1 of [1]). *Given a site $s \in \mathcal{S}$, the Voronoi cell $\texttt{Cell}_P(s, \mathcal{S})$ is contained in the funnel $\texttt{Funnel}_P(s, \mathcal{S})$.*

We are also interested in bounding the total complexity of the funnels of sites in $\mathcal{S}$. Given a polygon Q, let $|Q|$ denote its *combinatorial complexity* (or just *complexity*), i.e., the number of vertices and edges used to represent it.

▶ **Lemma 4** (Consequence of Corollaries 3.8 and 4.4 of [1]). *Given an n-gon P and a set $\mathcal{S} \subseteq V(P)$, $\sum_{s \in \mathcal{S}} |\texttt{Funnel}_P(s, \mathcal{S})| = O(n)$. Also, all funnels can be computed in $O(n)$ time.*

2.1 The simplification transformation

The following transformation allows us to modify the input of our problem and assume some nice structural properties without loss of generality. In this section and for this transformation, we allow the given polygon P to be weakly simple instead of a simple n-gon. The result of the transformation described in this section takes a weakly simple polygon with a set of weighted sites as input, and produces a new simple polygon with a new set of weighted sites. Moreover, this resulting polygon has a particular structure that is crucial in the recursive calls of our algorithm.

Let $\mathcal{L}_{\mathcal{S},P}$ be the set of leaves of $\text{VD}(\mathcal{S}, P)$. We first notice that we can focus on a specific geodesically convex subpolygon of P to compute $\text{VD}(\mathcal{S}, P)$.

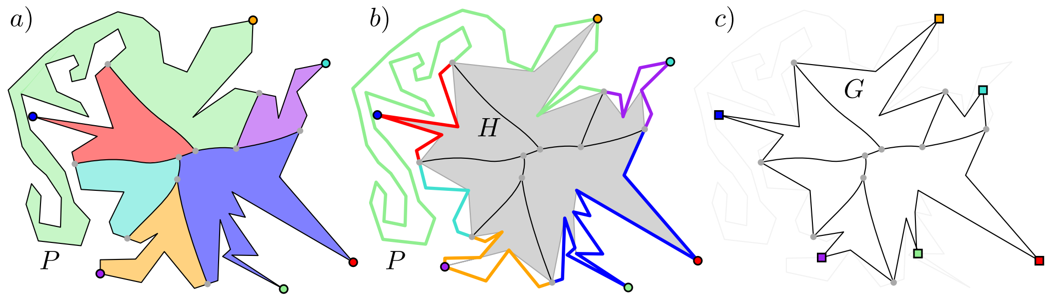

Figure 2 *a*) A simple polygon P with a set $\mathcal{S}$ of six weighted sites and their FVD. *b*) The polygon H being the geodesic hull of $\mathcal{L}_{\mathcal{S},P}$ and $\mathcal{S}$. *c*) The simplification transformation allows to redefine the problem inside a simpler polygon G with a new set of weighted sites and obtain the same FVD.

▶ **Lemma 5.** *Let H be the geodesic hull of $\mathcal{S} \cup \mathcal{L}_{\mathcal{S},P}$ in P. Then, for each $s \in \mathcal{S}$, $\texttt{Cell}_H(s, \mathcal{S}) \subseteq \texttt{Cell}_P(s, \mathcal{S})$. Moreover, the Voronoi trees of* $\text{VD}(\mathcal{S}, H)$ *and* $\text{VD}(\mathcal{S}, P)$ *coincide.*

Proof. Let s be a site of $\mathcal{S}$ and let $x \in H$. Because H is a geodesically convex subset of P, and since $x, s \in H$, we know that $\pi_H(x, s) = \pi_P(x, s)$. That is, the w-distance to x from each site in $\mathcal{S}$ is the same in P and H. Therefore, the $\mathcal{S}$-farthest sites of x are also preserved, which implies that $\texttt{Cell}_H(s, \mathcal{S}) \subseteq \texttt{Cell}_P(s, \mathcal{S})$. Because this happens for each Voronoi cell of $\mathcal{S}$ in H, and since the leaves belong also to H, the Voronoi trees coincide. ◀

While the geodesic hull H of $\mathcal{S} \cup \mathcal{L}_{\mathcal{S},P}$ in P does not necessarily have lower complexity than P, it has some nice structure. We know that its boundary consists of geodesic paths that connect consecutive points in $\mathcal{S} \cup \mathcal{L}_{\mathcal{S},P}$ along ∂P. However, this geodesic hull H is not necessarily a simple polygon; see Figure 2. To make it simple, we need to deal with "dangling paths" as follows.

We say that a vertex of H is *H-open* if it is incident to the interior of H. For each $s \in \mathcal{S}$, let a_s be the H-open vertex of $\texttt{Funnel}_P(s, \mathcal{S})$ that is geodesically closest to s. Note that all paths from s to any point in the interior of H pass through a_s. However, as long as the length of the geodesic path $\pi_P(s, a_s)$ remains the same, the shape of this path is irrelevant. In fact, this is equivalent to giving a_s a weight such that each distance measured from a_s to points in the interior of H has an added value of $\text{D}_w^P(s \rightsquigarrow a_s)$.

To formalize this intuition, we define a new polygon, a new set of sites, and a new weighted distance function as follows. Let $\mathcal{A} = \{a_s : s \in \mathcal{S}\}$ be the set of m H-open vertices defined by $\mathcal{S}$. These vertices are our new set of sites. Let $G(P, \mathcal{S})$ (or simply G if P and $\mathcal{S}$ are clear from the context) be the geodesic hull of $\mathcal{A} \cup \mathcal{L}_{\mathcal{S},P}$ in P. Note that $G \subseteq H$ is a simple polygon by the definition of each a_s in $\mathcal{A}$. We define a new weight function $w' : G \to \mathbb{R}$ so that $w'(x) = \begin{cases} \text{D}_w^P(s \rightsquigarrow a_s) & \text{if } x = a_s \in \mathcal{A} \\ 0 & \text{otherwise} \end{cases}$, if s and a_s coincide, then their weight is $w(s)$. With this new weight function, we can think of $\mathcal{A}$ as a set of weighted sites in G.

▶ **Lemma 6.** [$*$] *It holds that* $\text{VD}(\mathcal{A}, G)$ *and* $\text{VD}(\mathcal{S}, P)$ *have the same Voronoi trees.*

By Lemma 6, we can always transform the problem of computing the FVD of $\mathcal{S}$ in P as follows. Recall that $\mathcal{L}_{\mathcal{S},P}$ is the set containing each leaf of $\text{VD}(\mathcal{S}, P)$ and that $m = |\mathcal{S}|$. Compute $\text{VD}_\partial(\mathcal{S}, P)$ and the funnel $\texttt{Funnel}_P(s, \mathcal{S})$ of each site in $\mathcal{S}$ in total $O(n)$ time. Because $\mathcal{S} \cup \mathcal{L}_{\mathcal{S},P} \subseteq \partial P$ and has size $2m = O(n)$, we can compute H the geodesic hull of $\mathcal{S} \cup \mathcal{L}_{\mathcal{S},P}$ in P in $O(n)$ time using Lemma 1. After that, consider the set $\mathcal{A}$ of H-open vertices as defined above. Again, we can compute the geodesic hull G of $\mathcal{A} \cup \mathcal{L}_{\mathcal{S},P}$ in P in

$O(n)$ time using Lemma 1. By Lemma 6, $\mathrm{VD}(\mathcal{A}, G)$ and $\mathrm{VD}(\mathcal{S}, P)$ coincide, so we can forget about P and S, and focus simply on G and $\mathcal{A}$ to compute the FVD. We call this process the *simplification transformation*; see Figure 2. Note that the only convex vertices of G are the sites in $\mathcal{A}$ and the leaves in $\mathcal{L}_{\mathcal{S},P}$. We summarize the main properties of this simplification transformation in the following result.

▶ **Lemma 7.** *Let P be a simple n-gon and let $\mathcal{S} \subseteq V(P)$ be a set of $m \geq 3$ sites. The simplification transformation computes in $O(n)$ time a new simple polygon G with at most $n+m$ vertices and a new set $\mathcal{A} \subseteq V(G)$ of m weighted sites such that (1) the Voronoi trees of $\mathrm{VD}(\mathcal{A}, G)$ and $\mathrm{VD}(B, P)$ coincide, and (2) the set of convex vertices of G is exactly $\mathcal{A} \cup \mathcal{L}_{\mathcal{S},P}$.*

3 Computing the FVD

Let P be a simple polygon and let $\mathcal{S}$ be a set of $m \geq 3$ weighted sites contained in $V(P)$. Using the simplification transformation defined in Section 2.1, we can assume without loss of generality that P is a simple polygon with at most $n+m$ vertices, and among them, its convex vertices are exactly the sites in $\mathcal{S}$ and the leaves of $\mathcal{L}_{\mathcal{S},P}$ (see Lemma 7). That is, it consists of at most $2m$ convex vertices. If we consider consecutive vertices in $\mathcal{S} \cup \mathcal{L}_{\mathcal{S},P}$ along ∂P, the chain connecting them consists only of reflex vertices of P, or is a single edge. The next step explained in the following section is to randomly partition $\mathcal{S}$. Note that if $m = O(1)$, we can compute $\mathrm{VD}(\mathcal{S}, P)$ in $O(n)$ time by computing their bisectors and considering their overlay. Thus, we assume that m is larger than some predefined constant.

3.1 First phase: the partition

We compute in linear time $\mathrm{VD}_\partial(\mathcal{S}, P)$ using Lemma 2, and let $\mathcal{L}_{\mathcal{S},P}$ be the set of leaves of $\mathrm{VD}(\mathcal{S}, P)$. Note that $|\mathcal{L}_{\mathcal{S},P}| = m$. For each $s \in \mathcal{S}$, we compute the funnel $\texttt{Funnel}_P(s, \mathcal{S})$. Given a subset $R \subseteq \mathcal{S}$, let $\kappa(R) = \sum_{r \in R} |\texttt{Funnel}_P(r, \mathcal{S})|$ be the *magnitude* of R. Lemma 4 implies that $\kappa(\mathcal{S}) = O(n)$, and that all these funnels can be computed in $O(n)$ time. To be more precise, let $\tau \geq 2$ be the constant hidden by the big O notation, i.e., $\kappa(\mathcal{S}) \leq \tau n$. Next, we compute a random permutation Π of $\mathcal{S}$. Let $0 < \alpha < 1$ be some constant to be defined later. Let B and R be a partition of $\mathcal{S}$ such that B consists of the first $\lfloor \frac{\alpha}{\tau} m \rfloor$ sites according to Π, and $R = \mathcal{S} \setminus B$. We refer to B and R as the sets of *blue* and *red* sites of $\mathcal{S}$, respectively.

▶ **Observation 8.** *It holds that $\mathrm{E}[\kappa(B)] \leq \alpha n$ and $|B| = \lfloor \frac{\alpha}{\tau} m \rfloor \leq \alpha m$.*

We would like to recursively compute a Voronoi-like diagram of the sites in B while forgetting for a while about the red sites. Once we have this recursively computed diagram, we perform a randomized incremental construction of $\mathrm{VD}(\mathcal{S}, P)$ by inserting the sites of R in the random order according to permutation Π. In the next section we discuss the recursive call to compute a diagram for B, and later spend Section 4 detailing the insertion process.

3.2 A smaller polygon

While it would be great to compute $\mathrm{VD}(B, P)$, this may be too expensive as the diagram can have large complexity, and we need our recursive call to have smaller complexity (a constant fraction reduction in the size). Thus, we would not compute $\mathrm{VD}(B, P)$ exactly, but we will compute an "approximation" of it. Notice that we can see $\mathrm{VD}(B, P)$ as the upper envelope of the w-distances $\mathrm{D}^P_w(b \rightsquigarrow x)$. Because these functions have a complexity that depends on the size of the polygon, we need to simplify them. To achieve this, for a site $b \in B$, this

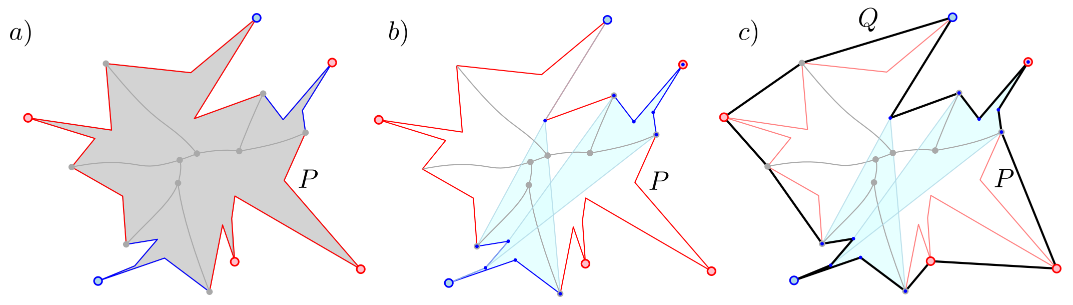

Figure 3 *a*) The polygon obtained from the simplification transformation, and the decomposition of its boundary into red and blue chains. *b*) The funnels of the sites in B are depicted, as well as all vertices in V_B. *c*) The polygon Q is obtained by taking a rubber band and keeping attached at all vertices in $V_B \cup V_C$ and letting it snap.

simpler distance function will be completely accurate inside of $\texttt{Cell}_P(b, \mathcal{S})$. However, for any point x outside of $\texttt{Cell}_P(b, \mathcal{S})$, this new distance from s to x will be only upper bounded by $\mathrm{D}_w^P(b \rightsquigarrow x)$. That is, distances from b can only get shorter, and only outside of $\texttt{Cell}_P(b, \mathcal{S})$.

To define these new distance functions, we define a new polygon Q of lower complexity than P (although $P \subseteq Q$). Let V_B be the set consisting of all vertices of P that belong to the funnel $\texttt{Funnel}_P(b, \mathcal{S})$ of some $b \in B$. By Observation 8, we know that the expected size of V_B is at most αn. Note that we could repeat the construction of B and R an expected constant number times, until we guarantee that $V_B \leq \alpha n$. Let V_C be the set of convex vertices of P. By our assumption that the simplification transformation has already been applied to P, we know by Lemma 2.1 that V_C consists of the union of $\mathcal{S}$ and $\mathcal{L}_{\mathcal{S},P}$, where $\mathcal{L}_{\mathcal{S},P}$ is the set of leaves of $\mathrm{VD}(\mathcal{S}, P)$, i.e., $|V_C| \leq 2m$. Let Q be a polygon defined as follows. Imagine the boundary of P being a rubber band, and each vertex of $V_B \cup V_C$ being a pin. By letting the rubber band free while keeping it attached at the pins, this rubber band snaps to a closed curve defining a weakly simple polygon Q; see Figure 3.

▶ **Lemma 9.** [∗] *The polygon Q contains P, is weakly simple, can be computed in expected $O(n)$ time, and has at most $\alpha n + 4m$ vertices. Moreover, for $x, y \in P$, it holds that $\mathrm{G}^Q(x, y) \leq \mathrm{G}^P(x, y)$. In particular, for each $b \in B$ and $x \in P$, $\mathrm{D}_w^Q(b \rightsquigarrow x) \leq \mathrm{D}_w^P(b \rightsquigarrow x)$, and if $x \in \texttt{Cell}_P(b, \mathcal{S})$, then $\mathrm{D}_w^Q(b \rightsquigarrow x) = \mathrm{D}_w^P(b \rightsquigarrow x)$.*

Sketch proof. The boundary of Q can be constructed by connecting consecutive points in $V_B \cup V_C$ by geodesics contained in the complement of P, i.e., in a domain that has P as a hole or obstacle (Q is also known as the relative hull of $V_B \cup V_C$ in this domain). Only convex vertices of P can be in these paths, and they can be visited only twice. Therefore, Q consists of at most $|V_B| + 2|V_C| \leq \alpha n + 4m$ vertices. ◀

Our plan is now to compute the FVD of B in the new polygon Q, and then use this as a "good" approximation of $\mathrm{VD}(B, P)$ in P. By "good" we mean that the red sites can be randomly inserted in this diagram, and that the result of this whole process is indeed $\mathrm{VD}(\mathcal{S}, P)$. We will prove these properties in the next section, but for now, we focus on describing the recursive algorithm.

Let $I(n, m)$ be the time to insert back the sites of R and obtain $\mathrm{VD}(\mathcal{S}, P)$ after having recursively computed $\mathrm{VD}(B, Q)$. Because Q consists of at most $\alpha n + 4m$ vertices by Lemma 9, and since $|B| \leq \alpha m$ by Lemma 8, we get a recursion of the form $T(n, m) = T(\alpha n + 4m, \alpha m) + I(n, m)$ for the running time of our algorithm. We claim that $I(n, m) = O(n + m)$, and we prove it in the next section. However, for $T(n, m)$ to solve to $O(n + m)$, we need to look at one more iteration of the recursion, as it can be that $\alpha n + 4m$ is not really smaller than n if m is large. Fortunately, because $T(\alpha n+4m, \alpha m) = T(\alpha(\alpha n+4m)+4\alpha m, \alpha^2 m)+I(\alpha n+4m, \alpha m)$, and by our assumption on the running time of $I(n, m)$, we get that

$$T(n, m) = T(\alpha^2 n + 8\alpha m, \alpha^2 m) + O(n + m) + O(\alpha n + 4m + \alpha m).$$

By choosing the constant α sufficiently small, and since we assume that $m \leq n$, we can guarantee that $T(n, m) \leq T(n/2, m/2) + O(n + m) = O(n + m)$ proving the main result of this paper. Therefore, it remains only to show that $I(n, m)$ is indeed $O(n + m)$, i.e., in linear time we can insert back the red sites, and obtain the FVD $\text{VD}(\mathcal{S}, P)$ from the recursively computed diagram of $\text{VD}(B, Q)$.

3.3 Preprocessing the red sites

Before going into the insertion process of the sites in R, we need to finish some preprocessing on them. To be able to insert these sites efficiently, we need to have a representation of the w-distance of each $r \in R$ defined on a sufficiently large superset of $\texttt{Cell}_P(r, \mathcal{S})$.

Note that we cannot define these distance functions in the entire polygon, otherwise we are spending already too much time and space. On the other hand, if its representation is too narrow, then during the insertion it might be that the distance information is insufficient.

Recall that $\mathcal{L}_{\mathcal{S},P}$ denotes the set of leaves of $\text{VD}(\mathcal{S}, P)$. Color the leaves in $\mathcal{L}_{\mathcal{S},P}$ *purple* if they are incident to the Voronoi cell of a site in B and a site in R; see Figure 4.

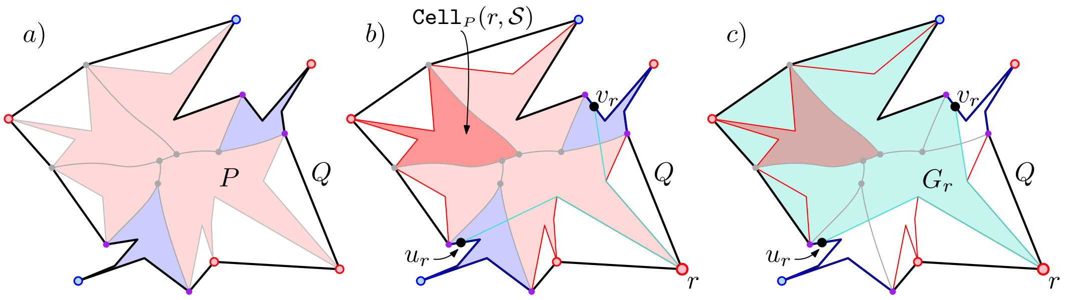

Figure 4 *a)* The coloring of the purple leaves of $\mathcal{L}_{\mathcal{S},P}$. *b)* For a site $r \in R$, the construction of u_r and v_r lying inside blue Voronoi cells. *c)* The polygon G_r where we compute the SPM of r.

Recall that given a polygon K and two points x and y on ∂K, $\partial K(x, y)$ denotes the polygonal chain that starts at x and follows the boundary of K clockwise until reaching y. For each $r \in R$, let u_r and v_r respectively be the first purple leaves of $\mathcal{L}_{\mathcal{S},P}$ reached from any point in $\texttt{bCell}_P(r, \mathcal{S})$ when walking counterclockwise and clockwise along ∂P. We then move u_r and v_r slightly clockwise and counterclockwise, respectively, so that they both sit inside of a blue Voronoi cell. Moreover, we know that $\texttt{bCell}_P(r, \mathcal{S})$ is contained in the interior of the path $\partial P(u_r, v_r)$. We define a new polygon where we define the w-function of r as follows. Because both u_r and v_r lie on ∂P and on Voronoi cells of blue sites, we know that u_r and v_r are both on ∂Q. Let G_r be a new weakly simple polygon bounded by the paths $\pi_P(r, u_r), \partial Q(u_r, v_r)$ and $\pi_P(v_r, r)$. Notice however that the paths $\pi_P(r, u_r)$ and $\pi_P(v_r, r)$, called the *walls*, are defined within the polygon P, while the other path bounding G_r is contained in ∂Q. Since $P \subset Q$, we know that $G_r \subseteq Q$. Moreover, the funnel $\texttt{Funnel}_P(r \to \partial P(u_r, v_r))$ is contained in G_r; see Figure 4.

▶ **Lemma 10.** [∗] *Given $r \in R$, it holds that $\texttt{Cell}_P(r, \mathcal{S}) \subseteq G_r$. Moreover, it holds that*

$$\mathrm{E}\left[\sum_{r \in R} |G_r|\right] = O(n) \text{ and } \bigcup_{r \in R} G_r = Q.$$

The last technical detail is the structure used to store the w-distances from the sites in R. Let $s \in \mathcal{S}$ and let $H \subseteq P$ be a subpolygon such that $s \in H$. The *shortest-path map* (or *SPM* for short) of s in H is a subdivision of H into triangles such that the geodesic path to all the points in one triangle has the same combinatorial structure. By precomputing the geodesic distance to each vertex of H, we get a constant-sized representation of the w-distance from s inside each triangle (for more information on shortest-path maps refer to [11]). We know also that the SPM of s in H can be computed in $O(|H|)$ time [8, 11].

Using these SPM's, we describe our w-distances as follows. For each $r \in R$, we compute the SPM of r in G_r. Because the complexity of this SPM is $O(|G_r|)$, we conclude that the total expected complexity of all these SPM's is $\mathrm{E}\left[\sum_{r \in R} |G_r|\right] = O(n)$ by Lemma 10.

4 Inserting back the red sites

After computing $\textsc{vd}(B, Q)$ recursively, we would like to start the randomized incremental construction of sites of R. But first, we should specify how we store $\textsc{vd}(B, Q)$.

Using the SPM's, we introduce the *refined FVD* of B in Q. This refined FVD is a decomposition of Q into constant-size cells defined as follows: For each site $b \in B$, the Voronoi cell $\texttt{Cell}_Q(b, B)$ is subdivided by the defining triangles of the SPM of b in Q. That is, we take the intersection of each defining triangle $\triangle$ of the SPM of b and intersect it with the Voronoi cell of b to obtain a *refined triangle*. As usual, for each point in a refined triangle, the geodesic distance is measured from its apex a and added with $\textsc{d}^P_w(b \rightsquigarrow a)$. Thus, each refined triangle and its distance function can be described with $O(1)$ space. We say that b *owns* these refined triangles. In other words, we have a way to describe the upper envelope of the w-distances of the sites in B within Q using a collection of constant-size refined triangles.

Assume inductively that $\textsc{vd}(B, Q)$ is represented as a refined FVD as described above. That is, for each site b of B, there is a collection of refined triangles owned by b which cover the entire Voronoi cell $\texttt{Cell}_Q(b, B)$. Let $f_b : Q \to \mathbb{R}$ such that $f_b(x) = \textsc{d}^Q_w(b \rightsquigarrow x)$ for each $x \in Q$, i.e., f_b is the function that maps each point to its w-distance to b in Q. Note that the refined triangles of b in the refined FVD of $\textsc{vd}(B, Q)$ provide a representation of this w-distance inside $\texttt{Cell}_Q(b, B)$. We say that the geodesic path $\pi_Q(b, x)$ is the *witness path* of the value of $f_b(x)$.

▶ **Observation 11.** *Let $b \in B$. Given $x \in P$, it holds that $f_b(x) \leq \textsc{d}^P_w(b \rightsquigarrow x)$. Moreover, if $x \in \texttt{Cell}_P(b, \mathcal{S})$, then $f_b(x) = \textsc{d}^P_w(b \rightsquigarrow x)$.*

Let $r \in R$ and recall that G_r is the polygon associated with r defined in Section 3.1. As a preprocessing, we have computed the SPM of r within G_r. We define a function $f_r : G_r \to \mathbb{R}$ that encodes the w-distances from r with respect to the polygon G_r, instead of Q. That is, $f_r(x) = \textsc{d}^{G_r}_w(x \rightsquigarrow r)$ for each $x \in G_r$. In this case we say that $\pi_{G_r}(r, x)$ is a *witness path* of the value of $f_r(x)$. Note that by Lemma 10, the functions f_r jointly cover polygon Q.

▶ **Lemma 12.** [∗] *Let $r \in R$. Given $x \in P$, it holds that $f_r(x) \leq \textsc{d}^P_w(r \rightsquigarrow x)$. Moreover, if $x \in \texttt{Cell}_P(r, \mathcal{S})$ and the path $\pi_P(r, x)$ contains no point of $\texttt{bCell}_P(r, \mathcal{S})$ other than its endpoints, then $f_r(x) = \textsc{d}^P_w(r \rightsquigarrow x)$.*

Note that for each site of R, we have considered their w-distances inside of G_r, while for the sites in B, their w-distances are with respect to Q. Therefore, in the intermediate steps of our incremental construction, we will not have the FVD of the sites, but some Voronoi-like structure.

4.1 The envelope

Note that $\textsc{vd}(B, Q)$ represents already the upper envelope of the functions f_b for the sites in B. We would like to complete this envelope by incrementally inserting the functions f_r for the sites in R. To deal with these upper envelopes, we introduce some definitions.

Consider the order of the sites of R according to the random permutation Π used to construct B and R. Let $\mathcal{S}_0 = B$ and for each $1 \leq i \leq |R|$, let $\mathcal{S}_i$ be the set consisting of B and the first i red sites according to the permutation Π. This is the order that we use for our randomized incremental construction. That is, on each insertion step we would like to maintain a Voronoi-like structure for the sites in $\mathcal{S}_i$.

Let s be a site of $\mathcal{S}_i$. Given a point x of Q, we say that x is *i-dominated* by s if $f_s(x) \geq f_{s'}(x)$ for all $s' \in \mathcal{S}_i$. Notice that if a point x is i-dominated, then the witness path of $f_s(x)$ must be defined.

▶ **Lemma 13.** [∗] *Let s be a site of $\mathcal{S}$, and let x be a point that is i-dominated by s. If $z \in Q$ is a point that lies on the witness path of $f_s(z)$, then z is i-dominated by s.*

Let x be a point that is i-dominated by s. Extend the last segment of the witness path of $f_s(x)$ until it touches the boundary of Q at a point x^*. We say that x^* is the *s-shadow* of x. A direct consequence of Lemma 13 is the following result.

▶ **Corollary 14.** *Let $s \in \mathcal{S}_i$. If x is i-dominated by s, then its s-shadow is also i-dominated by s and lies on ∂Q.*

The following result is crucial to guarantee the resulting structure after the incremental construction coincides with the desired FVD of $\mathcal{S}$.

▶ **Lemma 15.** [∗] *For each $0 \leq i \leq |R|$ and for each site $s \in \mathcal{S}_i$, each point in the Voronoi cell $\texttt{Cell}_P(s, \mathcal{S})$ is i-dominated by s.*

4.2 The insertion process

Let r be the i-th site of R inserted in our randomized incremental construction. To simplify our incremental construction, instead of constructing the entire set of points that are i-dominated by r, which might contain several connected components, we focus exclusively on constructing the connected component containing $\texttt{Cell}_P(r, \mathcal{S})$. This simplifies the structure of the upper envelope, and helps us to prove a bound on its complexity.

We define the *envelope-graph* of $\mathcal{S}_i$ recursively. For the base case $i = 0$, the envelope-graph of $\mathcal{S}_0$ is simply the Voronoi-tree of $\textsc{vd}(B, Q)$. This envelope-graph induces a decomposition of Q into 0-*patches*. The 0-*patch* of each site $s \in \mathcal{S}_0$ is the connected component in this decomposition that contains $\texttt{Cell}_P(s, \mathcal{S})$.

Given the envelope-graph of $\mathcal{S}_{i-1}$, the envelope-graph of $\mathcal{S}_i$ is defined as follows. We consider the set of all points of Q that are i-dominated by r and the connected components that they induce. The *i-patch* of r is the connected component that contains $\texttt{Cell}_P(r, \mathcal{S})$ induced by these points. The envelope-graph of $\mathcal{S}_i$ is then obtained by adding to it the boundary of the i-patch of r, and removing everything inside it. In this way, the $(i-1)$-patches of the envelope graph of $\mathcal{S}_{i-1}$ might shrink. However, Lemma 15 guarantees that for each site $s \in \mathcal{S}_i$, the Voronoi cell $\texttt{Cell}_P(s, \mathcal{S})$ is i-dominated by s. Therefore, $\texttt{Cell}_P(s, \mathcal{S})$ is still contained in the i-patch of s, i.e., the i-patch of s is non-empty.

▶ **Lemma 16.** [$*$] *The envelope-graph of $\mathcal{S}_i$ is a tree with at most $2|\mathcal{S}_i|$ leaves lying on the boundary of Q.*

Proof sketch. The proof is by induction, with $\textsc{vd}(B, Q)$ as base case. When inserting the i-th site $r \in R$ into the envelope-graph of $\mathcal{S}_{i-1}$, the i-patch of r is connected and intersects ∂Q in a single connected component. By adding this i-patch, the tree structure is preserved. ◀

▶ **Lemma 17.** [$*$] *The i-patch of r is contained in G_r and does not intersect the walls of G_r.*

4.3 Algorithmic description

We proceed now to describe algorithmically how to carry on the incremental construction described above, and construct the envelope-graph of $\mathcal{S}_i$. Our algorithm starts with the refined FVD of $\textsc{vd}(B, Q)$, and on each round constructs the boundary of the i-patch of a new site of R. In addition to our envelope-graph, we maintain a set of refined triangles that cover each i-patch in the same way that they cover the Voronoi cells in the refined FVD; see Figure 5. We call this representation the *refined envelope* of $\mathcal{S}_i$. We assume inductively that the envelope-graph of $\mathcal{S}_i$ is stored as a refined envelope. For the base case this holds as we assume that we have at hand the refined FVD of $\textsc{vd}(B, Q)$. In addition, we maintain the invariant that for each vertex v of Q, we know the site whose i-patch contains v. Moreover, we assume also that we have a pointer to the refined triangle of this i-patch that contains v.

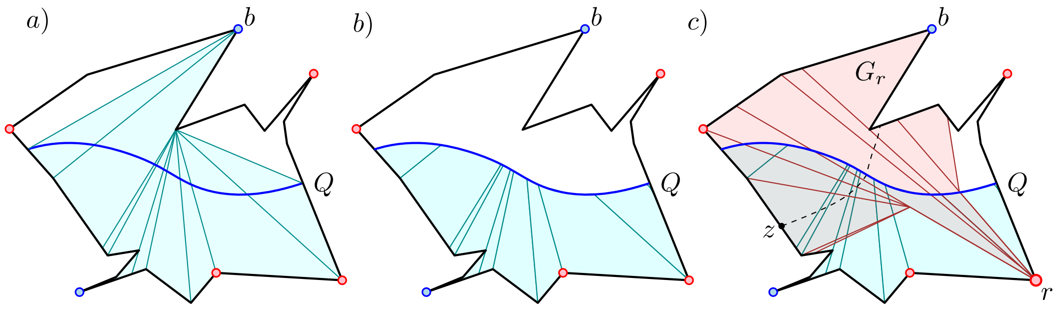

Figure 5 *a)* a site $b \in B$ and the defining triangles of f_b. *b)* The refined triangles obtained by intersecting with the 0-patch of b. *c)* The insertion of the red site r and the update of the envelope.

For each $b \in B$, let μ_b be the set of refined triangles that belong to b. For a site $r \in R$, let μ_r denote the set of triangles used to describe f_r, i.e, the set of triangles in the SPM of r inside of G_r. Thus regardless of the case μ_s denotes a set of triangles (and their associated distance function) *owned* by s.

▶ **Lemma 18.** [$*$] *Let r be the i-th site of R inserted in our randomized incremental construction. The i-patch of r can be computed in $O(M_r + D_r + |\mu_r|)$ time, where M_r is the size of the i-patch of r, and D_r is the number of arcs of the envelope-graph of $\mathcal{S}_{i-1}$ that disappear. Moreover, the refined envelope of $\mathcal{S}_i$ can be obtained within the same time from that of $\mathcal{S}_{i-1}$.*

Proof Sketch. To compute the i-patch of r, the first step is to find a point lying on its boundary, which can be done by locating the leaves of $\mathcal{L}_{\mathcal{S},P}$ bounding $\texttt{bCell}_P(r, \mathcal{S})$. Next, we walk along ∂Q until finding an endpoint z of the intersection of ∂Q with the i-patch of r. Finally, we trace the boundary of the i-patch of r inside of Q in the standard way used in the RIC of Euclidean Voronoi diagrams. That is, by walking along the overlay of the triangles in μ_r and the refined triangles of the refined envelope of $\mathcal{S}_{i-1}$, and constructing each arc of the i-patch of r at a time. The insertion time is then proportional to the size of the i-patch. ◀

The *complexity* of the envelope-graph of $\mathcal{S}_i$ is the number of vertices and arcs defining it.

▶ **Lemma 19.** [∗] *The expected complexity of the envelope-graph of $\mathcal{S}_i$ is $O(n)$.*

Proof sketch. Recall that for $b \in B$, μ_b is the set of refined triangles that belong to b, and for $r \in R$, μ_r is the set of triangles used to describe f_r inside of G_r. Since $\text{VD}(B, Q)$ is a FVD of B in Q and by Lemma 10, we know that $\mathrm{E}\left[\sum_{s\in\mathcal{S}} |\mu_s|\right] = O(n)$. Using a charging argument, we prove that the complexity of the envelope-graph of $\mathcal{S}_i$ is at most $\sum_{s\in\mathcal{S}} |\mu_s|$. ◀

We are now ready to combine these lemmas into the main result of this section.

▶ **Theorem 20.** [∗] *The envelope-graph and refined envelope of $\mathcal{S}_{|R|}$ can be computed in expected $O(n)$ time. Moreover, for each $s \in \mathcal{S}$, the envelope-graph of $\mathcal{S}_{|R|}$ coincides with the Voronoi tree of* $\text{VD}(\mathcal{S}, P)$*.*

Proof Sketch. By Lemma 15, the $|R|$-patch of each site s of $\mathcal{S}$ contains its corresponding Voronoi cell $\texttt{Cell}_P(s, \mathcal{S})$. Because the union of these Voronoi cells covers P, we conclude that the $|R|$-patch of s and $\texttt{Cell}_P(s, \mathcal{S})$ coincide inside P for each $s \in \mathcal{S}$. That is, the envelope-graph of $\mathcal{S}_{|R|}$ coincides with the Voronoi tree of $\text{VD}(\mathcal{S}, P)$.

By Lemma 18, the time needed to insert all sites of R, is $O(\sum_{r\in R}(M_r + D_r + |\mu_r|))$. Using backwards analysis, we show that $\mathrm{E}[M_r] = O(n/|S|)$. Therefore, $\mathrm{E}\left[\sum_{r\in R} M_r\right] = O(n)$. Moreover, Lemma 10 implies that $\sum_{r\in R} \mu_r = O(n)$, and since an arc is destroyed only once, we can charge this cost to the creation of the arc. Putting everything together, we conclude that the expected time needed to insert all red sites is $O(n)$. ◀

We are ready to state our main result. As mentioned in Section 3.2, the running time of our algorithm is $T(n, m) \leq T(n/2, m/2) + I(n, m)$, where $I(n, m)$ is the time to insert the red sites. By Theorem 20, and by our assumption that $m \leq n$, $I(n, m) = O(n + m)$. Therefore, by solving the recurrence we obtain the following result.

▶ **Theorem 21.** *Let P be a simple polygon and let $\mathcal{S}$ be a set of $m \geq 3$ weighted sites contained in $V(P)$. We can compute the FVD of $\mathcal{S}$ in P in expected $O(n + m)$ time.*

References

1. Hee-Kap Ahn, Luis Barba, Prosenjit Bose, Jean-Lou 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, December 2016. `doi:10.1007/s00454-016-9796-0`.
2. Boris Aronov. On the geodesic Voronoi diagram of point sites in a simple polygon. *Algorithmica*, 4(1-4):109–140, 1989.
3. Boris Aronov, Steven Fortune, and Gordon Wilfong. The furthest-site geodesic Voronoi diagram. *Discrete & Computational Geometry*, 9(1):217–255, 1993.
4. T. Asano and G.T. Toussaint. Computing the geodesic center of a simple polygon. Technical Report SOCS-85.32, McGill University, 1985.
5. Luis Barba. Geodesic farthest-point Voronoi diagram in linear time. *CoRR*, abs/1809.01481, 2018. `arXiv:1809.01481`.
6. Hsien-Chih Chang, Jeff Erickson, and Chao Xu. Detecting weakly simple polygons. In *Proceedings of the twenty-sixth annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1655–1670. SIAM, 2014.
7. Bernard Chazelle. A theorem on polygon cutting with applications. In *Proceedings of FOCS*, pages 339–349, 1982. `doi:10.1109/SFCS.1982.58`.
8. Bernard Chazelle. Triangulating a simple polygon in linear time. *Discrete & Computational Geometry*, 6(1):485–524, 1991.

9 Mark De Berg, Marc Van Kreveld, Mark Overmars, and Otfried Cheong Schwarzkopf. Computational geometry. In *Computational geometry*, pages 1–17. Springer, 2000.

10 Herbert Edelsbrunner and Ernst Peter Mücke. Simulation of simplicity: a technique to cope with degenerate cases in geometric algorithms. *ACM Transactions on Graphics*, 9(1):66–104, 1990.

11 Leonidas Guibas, John Hershberger, Daniel Leven, Micha Sharir, and Robert E Tarjan. Linear-time algorithms for visibility and shortest path problems inside triangulated simple polygons. *Algorithmica*, 2(1-4):209–233, 1987.

12 John Hershberger and Subhash Suri. Matrix Searching with the Shortest-Path Metric. *SIAM Journal on Computing*, 26(6):1612–1634, 1997.

13 Chih-Hung Liu. A Nearly Optimal Algorithm for the Geodesic Voronoi Diagram of Points in a Simple Polygon. In *LIPIcs-Leibniz International Proceedings in Informatics*, volume 99. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

14 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.

15 Eunjin Oh. Optimal Algorithm for Geodesic Nearest-point Voronoi Diagrams. In *To appear in the Proceedings of the 31st annual ACM-SIAM Symposium on Discrete Algorithms*, page TBD, 2019.

16 Eunjin Oh and Hee-Kap Ahn. Voronoi diagrams for a moderate-sized point-set in a simple polygon. In *LIPIcs-Leibniz International Proceedings in Informatics*, volume 77. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.

17 Eunjin Oh, Sang Won Bae, and Hee-Kap Ahn. Computing a geodesic two-center of points in a simple polygon. In *Latin American Symposium on Theoretical Informatics*, pages 646–658. Springer, 2016.

18 Eunjin Oh, Luis Barba, and Hee-Kap Ahn. The farthest-point geodesic Voronoi diagram of points on the boundary of a simple polygon. In *LIPIcs-Leibniz International Proceedings in Informatics*, volume 51. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2016.

19 Evanthia Papadopoulou. k-Pairs Non-Crossing Shortest Paths in a Simple Polygon. *International Journal of Computational Geometry and Applications*, 9(6):533–552, 1999.

20 Richard Pollack, Micha Sharir, and Günter Rote. Computing the geodesic center of a simple polygon. *Discrete & Computational Geometry*, 4(1):611–626, 1989.

21 Subhash Suri. Computing geodesic furthest neighbors in simple polygons. *Journal of Computer and System Sciences*, 39(2):220–235, 1989.