Subquadratic Algorithms for the Diameter and the Sum of Pairwise Distances in Planar Graphs

Sergio Cabello*

FULL VERSION
July 13, 2016

Abstract
We show that how to compute in $O(n^{11/6} \text{polylog}(n))$ expected time the diameter and the sum of the pairwise distances in a planar graph with $n$ vertices. These are the first algorithms for this problem using time $O(n^c)$ for $c < 2$.

Keywords: planar graph, diameter, Wiener index, distances in graphs, Voronoi diagram.

1 Introduction

Let $G$ be an undirected graph with $n$ vertices and abstract, positive edge lengths $\lambda : E(G) \rightarrow \mathbb{R}_{> 0}$. The length of a path in $G$ is the sum of the edge lengths along the path. We define the distance between two vertices $x$ and $y$ of $G$, denoted by $d_G(x, y)$, as the minimum length over all paths in $G$ from $x$ to $y$. The diameter of $G$ is

$$\text{diam}(G) = \max\{d_G(x, y) \mid x, y \in V(G)\},$$

and the sum of the pairwise distances of $G$ is

$$\Sigma(G) = \sum_{(x, y) \in V(G)^2} d_G(x, y).$$

The concept is essentially equivalent to the average distance in the graph and the Wiener index, defined as $\Sigma(G)/2$, a parameter commonly used in mathematical chemistry.

Computing the diameter and the sum of the pairwise distances of a graph is a fundamental problem in graph algorithms. The obvious way to compute them is via solving the all-pairs shortest paths problem (APSP) explicitly and then extract the relevant information. A key question is whether one can avoid the explicit computation of all the pairwise distances.

Roditty and Vassilevska Williams [31] show that, for arbitrary graphs with $n$ vertices and $O(n)$ edges, one cannot compute the diameter in $O(n^{2-\delta_0})$ time, for some constant $\delta_0 > 0$, unless the strong exponential time hypothesis (SETH) fails. In fact, their proof shows that for unweighted graphs we cannot distinguish in $O(n^{2-\delta_0})$ time between sparse graphs that have diameter 2 or 3, assuming the SETH. This implies the same conditional lower bound for computing the sum of the pairwise distances in sparse graphs. Indeed, an unweighted graph of $n$ vertices has diameter 2 if and only if

$$\Sigma(G) = \frac{1}{2} \sum_{x \in V(G)} \left( \deg_G(x) + 2(n - 1 - \deg_G(x)) \right) = n(n - 1) - |E(G)|.$$

*Department of Mathematics, IMFM, and Department of Mathematics, FME University of Ljubljana, Slovenia. Supported by the Slovenian Research Agency, program P1-0297. Email address: sergio.cabello@fmf.uni-lj.si
Thus, if we could compute the sum of pairwise distances in $O(n^{2-\delta_0})$ time, we could also distinguish in the same time whether the graph has diameter 2 or larger, and the SETH fails.

Given such conditional lower bounds, it is natural to shift the interest towards identifying families of sparse graphs where one can compute the diameter or the sum of pairwise distances in truly subquadratic time. Here we show that the diameter and the sum of pairwise distances in planar graphs can be computed in $O(n^{11/6} \log \log(n))$ expected time. There are efficient algorithms for computing all the distances in a planar graph [12] or a specified subset of the distances; see for example [5, 27]. However, none of these tools seem fruitful for computing the diameter or the sum of the pairwise distances.

Related work. For graphs of bounded treewidth one can compute the diameter and the sum of pairwise distances in near-linear time [1, 7]. For planar graphs, Wulff-Nilsen [34] gives an algorithm to compute the diameter and the sum of pairwise distances in $O(n^2 \log \log(n)/\log(n))$ time, which is slightly subquadratic.

Researchers have also looked into near-optimal approximations. In particular, Weimann and Yuster [33] provide a $(1+\epsilon)$-approximation to the diameter of undirected planar graphs in $O((n/\epsilon^4) \log \log n + 2^{O(1/\epsilon^2)} n)$ time. As it was mentioned in [13], a near-linear time randomized $(1 + \epsilon)$-approximation to the sum of pairwise distances in undirected planar graphs can be obtained using random sampling and an oracle for $(1 + \epsilon)$-approximate distances [17, 32]. See [16] for the average distance in arbitrary discrete metric spaces.

Our approach. Let us describe the high-level idea of our approach. The main new ingredient is the use of additively-weighted Voronoi diagrams in pieces of the graph: we make a quite expensive preprocessing step in each piece that permits the efficient computation of such Voronoi diagrams in each piece for several different weights.

To be more precise, let $G$ be a planar graph with $n$ vertices. We first compute an $r$-division: this is a decomposition of $G$ into $O(n/r)$ pieces, each of them with $O(r)$ vertices and $O(\sqrt{r})$ boundary vertices. This means that all the interaction between a piece $P$ and the complement goes through the $O(\sqrt{r})$ boundary vertices of $P$.

Consider a piece $P$ and a vertex $x$ outside $P$. We would like to break $P$ into regions according to the boundary vertex of $P$ that is used in the shortest path from $x$. This can be modeled as an additively-weighted Voronoi diagram in the piece: each boundary vertex is a weighted site whose weight equals the distance from $x$. Thus, we have to compute several such Voronoi diagrams for each piece.

Assuming that a piece is embedded, one can treat such a Voronoi diagram as an abstract Voronoi diagram an encode it using the dual graph. In particular, a bisector corresponds to a cycle in the dual graph. We can precompute all possible Voronoi diagrams for $O(1)$ sites, and that information suffices to compute the Voronoi diagram using a randomized incremental construction. Once we have the Voronoi diagram, encoded as a subgraph of the dual graph, we have to extract the information from each Voronoi region. Although this is the general idea, several technical details appear. For example, the technology of abstract Voronoi diagrams can be used only when the sites are cofacial.

Assumptions. We will assume that the distance between each pair of vertices is distinct and there is a unique shortest path between each pair of vertices. This can be enforced with high probability using infinitesimal perturbations or deterministically using lexicographic comparison; see for example the discussion in [6]. Since we are aiming to a randomized algorithm and our running times are barely subquadratic, the actual method that is used does not seem very relevant at this point.
Roadmap. We assume that the reader is familiar with planar graphs. In the next section we explain the notation we use and the type of r-division that we will employ. In Section 3 we explain how to extract information about the vertices contained in a dual cycle. In Section 4 we explain the concept of abstract Voronoi diagrams. In Section 5 we deal with different definitions of Voronoi diagrams in plane graphs, show that they are equivalent, and discuss their algorithmic aspects. In Section 6 we give the final algorithm. We conclude with a discussion on alternative ideas.

2 Notation and divisions

A plane graph is a planar graph together with a fixed embedding. In the arguments we will use the geometry of the embedding and the plane quite often. For example, we will talk about the faces enclosed by a cycle of the graph. However, all the computations can be done assuming a combinatorial embedding, described as the circular order of the edges incident to each vertex.

Let $G^*$ be the dual graph of a plane graph $G$. We keep in $G^*$ any parallel edges that may occur. When $G$ is 2-connected, the graph $G^*$ has no loops. For each vertex $v$, edge $e$ and face $f$ of $G$, we use $v^*$, $e^*$ and $f^*$ to denote their dual counterparts, respectively. We assume natural embeddings of $G$ and $G^*$ where each dual edge $e^*$ of $G^*$ crosses $G$ exactly once and does so at $e$. There are no other types of intersections between $G$ and $G^*$. For any set of edges $A \subseteq E(G)$, we use the notation $A^* = \{e^* \mid e \in A\}$.

Quite often we identify a graph object and its geometric representation in the embedding. In particular, (closed) walks in the graph define (closed) curves in the plane. For each simple closed curve $\gamma$ in the plane, let $\text{int}(\gamma)$ be the bounded domain of $\mathbb{R}^2 \setminus \gamma$, and let $\text{ext}(\gamma)$ be the unbounded one. For each simple closed curve $\gamma$, in particular for a cycle in the dual graph $G^*$, let $V_{\text{int}}(\gamma, G) = \text{int}(\gamma) \cap V(G)$ and $V_{\text{ext}}(\gamma, G) = \text{ext}(\gamma) \cap V(G)$.

Vertices of $G$ are usually denoted by $x, y, z, u, v$. Faces of $G$ are usually denoted by symbols like $f$ and $g$. The dual vertices are usually denoted as $f^*$ or $g^*$, or using early letters of the Latin alphabet, like $a$ and $b$. We will denote cycles and paths in the dual graph with Greek letters, like $\gamma$ or $\pi$. Sets of cycles and paths in the dual graph are with capital Greek letters, like $\Gamma$ or $\Pi$.

For each set $A \subset \mathbb{R}^2$, we use $\overline{A}$ for its closure and $A^o$ for its interior.

Divisions. The concept of r-division for planar graphs was introduced by Frederickson [11], and then refined and used by several authors; see for example [5, 14, 20, 28] for a sample. For us it is most convenient to use the construction of Klein, Mozes and Sommer [19]. We first state the definitions carefully, almost verbatim from [19].

Let $G$ be a plane graph. A piece $^1$ $P$ of $G$ is an edge-induced subgraph of $G$. In each piece we assume the embedding inherited from $G$. A boundary vertex of a piece $P$ is a vertex of $P$ that is incident to some edge in $E(G) \setminus E(P)$. A hole of a piece $P$ is a face of $P$ that is not a face of $G$. Note that all boundary vertices of a piece $P$ are incident to a hole of $P$. An r-division with a few holes of $G$ is a collection $\{P_1, \ldots, P_k\}$ of pieces of $G$ such that

- there are $O(n/r)$ pieces, that is, $k = O(n/r)$;
- each edge of $G$ is in at least one piece;
- each piece has $O(r)$ vertices;
- each piece has $O(\sqrt{r})$ boundary vertices;

---

$^1$They use the term “region”, which in our opinion is more suitable. However, we are using such a term for so many things that in our context we prefer to use some other term.
each piece has $O(1)$ holes.

**Theorem 1** (Klein, Mozes and Sommer [19]). There is a linear-time algorithm that, for any biconnected triangulated planar embedded graph $G$, outputs an $r$-division of $G$ with with few holes.

Consider an $r$-division with a few holes $\mathcal{P} = \{P_1, \ldots, P_k\}$. For each piece $P_i$, let $K_i$ be the complete graph on the boundary vertices of $P_i$ where each edge $uv$ has length $d_{P_i}(u,v)$. Each single graph $K_i$ can be constructed in $O(r \log r)$ time using the multiple-source shortest-path algorithms of [6, 18].

The **dense distance graph** of $G$ with respect to $\mathcal{P}$, denoted by $\text{DDG}(\mathcal{P}, G)$, is $\bigcup_i K_i$. For each piece $P_i$ of $\mathcal{P}$, the distances in $P_i \cup \text{DDG}(\mathcal{P}, G)$ and in $G$ match, for the vertices that appear in $P_i \cup \text{DDG}_e(\mathcal{P}, G)$.

As used by several works [4, 28, 10, 26, 27], a shortest path tree in $P_i \cup \text{DDG}(\mathcal{P}, G)$ can be computed in $O((r + nr^{-1/2}) \text{polylog } r)$ time after $O(n \log r)$ preprocessing time. We will actually need the ancestor-descendant relation between boundary vertices in the shortest path tree.

**Lemma 2.** Assume that we are given any given $r$-division $\mathcal{P}$ of $G$. After spending $O(n \log r)$ preprocessing time, for each vertex $x$ in $P \in \mathcal{P}$, we can compute in $O((r + nr^{-1/2}) \text{polylog } r)$ time the shortest path tree from $x$ in $P \cup \text{DDG}(\mathcal{P}, G)$.

A particular consequence of this result is that we can compute all the pairwise distances between the vertices of a piece $P$ in $O((r^2 + nr^{1/2}) \text{polylog } r)$ time.

## 3 Information within a dual cycle

Let $G$ be a plane graph with $n$ vertices. Assume that each vertex $x$ of $G$ has a weight $w(x) > 0$. For each subset $U$ of vertices we define $\sigma(U) = \sum_{x \in U} w(x)$ and $\mu(U) = \max_{x \in U} w(x)$. In particular, $\sigma(V(G))$ is the sum of all weights and $\mu(V(G))$ is the largest weight.

Let $\gamma$ be a cycle in $G^*$. We are interested in a way to compute $\sigma(V_{\text{int}}(\gamma, G))$ and $\mu(V_{\text{int}}(\gamma, G))$ locally, after some preprocessing of $G$. Here, locally means that we would like to just look at the edges $\gamma$ and a small neighborhood. For the sum of the weights, $\sigma(\cdot)$, this can be achieved as explained by Park and Phillips [29] and Patel [30], which we summarize now.

Let $E(G^*)$ be the edges of $E(G^*) = E(G)^* \text{ oriented}$. Thus, each dual edge $ab$ gives rise to two edges, $\overrightarrow{a \rightarrow b}$ and $\overrightarrow{b \rightarrow a}$. In the following, we assume that any cycle $\gamma$ in $G^*$ is traversed clockwise. Thus, when we write $\overrightarrow{a \rightarrow b} \in \gamma$, it means that we are following the clockwise orientation of $\gamma$.

**Lemma 3.** Let $x_0$ be a fixed vertex in $G$. In linear time we can compute a weight function $\chi_\sigma : E(G^*) \rightarrow \mathbb{R}$ with the following property. For every dual cycle $\gamma$ that is oriented clockwise and contains $x_0$ in its interior, we have

$$\sigma(V_{\text{ext}}(\gamma, G)) = \sum_{\overrightarrow{a \rightarrow b} \in \gamma} \chi_\sigma(\overrightarrow{a \rightarrow b}).$$

**Proof.** The proof for the case when $w(x) = 1$ for all $x$ is given in [30]. The modification for the general case is trivial. We include a sketch of the proof for the sake of self-containment.

Consider any spanning tree $T$ of $G$ and root it at the distinguished vertex $x_0$. We orient the edges of $T$ as follows: if $xy \in E(T)$ and $y$ is a child of $x$, we orient it as $x \rightarrow y$. Let $\overrightarrow{T}$ be the resulting oriented edges from $T$. Let $T_\gamma$ be the subtree of $\overrightarrow{T} - (x \rightarrow y)$ that contains $\gamma$. See Figure 1, left. Assume that $ab$ is the edge dual to $xy$ and $\overrightarrow{a \rightarrow b}$ crosses $x \rightarrow y$ from left to right. We assign $\chi_\sigma(\overrightarrow{a \rightarrow b}) = \sigma(V(T_\gamma))$ and $\chi_\sigma(\overrightarrow{b \rightarrow a}) = -\chi_\sigma(\overrightarrow{a \rightarrow b})$. For any dual edge $ab$ of $E(G^*) \setminus E(T)$ we set $\chi_\sigma(\overrightarrow{a \rightarrow b}) = \chi_\sigma(\overrightarrow{b \rightarrow a}) = 0$. This finishes the description of the function $\chi_\sigma$. It is easy to see that we can compute $\chi_\sigma$ in linear time.
Figure 1: Proof of Lemma 3. Left: orientation of the edges of $T$. The dart $a \rightarrow b$ crosses $x \rightarrow y$ from left to right and thus $\chi_\sigma(a \rightarrow b) = w(V(T_y))$. Right: the crossings of $\gamma$ and $P_z$ alternate between left-to-right and right-to-left, as we walk along $P_z$.

From the definition of $\chi_\sigma$ we have

$$\sum_{a \rightarrow b \in \gamma} \chi_\sigma(a \rightarrow b) = \sum_{a \rightarrow b \in E(\gamma) \cap E(T)} \chi_\sigma(a \rightarrow b) = \sum_{x \rightarrow y \in T, \gamma \text{ crosses } x \rightarrow y \text{ left-to-right}} \sigma(V(T_y)) - \sum_{x \rightarrow y \in T, \gamma \text{ crosses } x \rightarrow y \text{ right-to-left}} \sigma(V(T_y)). \quad (1)$$

Consider any vertex $z$ of $V(G)$ and let $P_z$ be the path in $T$ from $x_0$ to $z$. Since $x_0$ is in $\text{int}(\gamma)$ and $\gamma$ describes a simple curve, the crossings between $P_z$ and $\gamma$, as we walk along along $P_z$, alternate between left-to-right and right-to-left crossings. See Figure 1, right. Since $\gamma$ defines a simple curve, the number of crossings is even if $z$ is in $\text{int}(\gamma)$ and odd otherwise. It follows that $w(z)$ contributes to the sum on the right side of equation (1) either once, if $z$ is in $\text{ext}(\gamma)$, or zero times, if $z$ is in $\text{int}(\gamma)$. The result follows. 

The previous lemma can be used to compute $\sigma(V_{\text{int}}(\gamma, G))$ because $\sigma(V_{\text{int}}(\gamma, G)) + \sigma(V_{\text{ext}}(\gamma, G)) = \sigma(V(G))$.

The proof of Lemma 3 heavily uses that the sum has an inverse operation. We are not aware of any such result for computing the maximum weight, $\mu(V_{\text{int}}(\gamma, G))$ or $\mu(V_{\text{ext}}(\gamma, G))$. However, in our very specific case, we will only deal with very special cycles, and we can do something similar.

Let $\gamma$ be a cycle of $G^*$ and let $x_0$ be a vertex of $G$. We say that $\gamma$ is $x_0$-star-shaped if $x_0$ is in $\text{int}(\gamma)$ and, for each vertex $y$ in $V_{\text{int}}(\gamma, G)$, the shortest path in $G$ from $x_0$ to $y$ is contained in $\text{int}(\gamma)$. We define the following family of dual cycles:

$$\Xi(G, x_0) = \{ \gamma \mid \gamma \text{ is a } x_0\text{-star-shaped cycle of } G^* \}. \quad (2)$$

Assume that $G$ is a triangulated graph. For any two consecutive edges $ab$ and $bc$ of $G^*$, we define $\text{left}(a, b, c) = 1$ if the faces $a^*, b^*$ and $c^*$ are counterclockwise around the common vertex of $(ab)^*$, and $\text{left}(a, b, c) = 0$ otherwise. See Figure 2 for an example.

**Lemma 4.** Let $G$ be a triangulated plane graph and let $x_0$ be a vertex of $G$. In linear time we can compute a weight function $\chi_\mu : E(G^*) \rightarrow \mathbb{R}$ with the following property. For every dual cycle $\gamma = a_0a_1 \ldots a_k$ of $\Xi(G, x_0)$ that is oriented clockwise, where $a_k = a_0$, we have

$$\mu(V_{\text{int}}(\gamma, G)) = \max \left\{ w(x_0) \cup \{ \chi_\mu(a_i, a_{i+1}) \mid i = 1 \ldots k, \text{ left}(a_{i-1}, a_i, a_{i+1}) = 1 \} \right\} \quad \text{(indices modulo } k)$$

**Proof.** Orient all the edges of $G$ away from $x_0$: an edge $xy$ gets oriented as $x \rightarrow y$ if and only if $d_G(x_0, x) < d_G(x_0, y)$. Each dual edge $ab$ is oriented in such a way that it crosses the corresponding oriented primal edge from left to right. Consider an oriented dual edge $b \rightarrow c$. Since $G$ is a triangulation, there is a unique dual vertex $a$ such that $\text{left}(a, b, c) = 1$. We define the primal edge $\varphi(bc)$ as the unique
edge of the triangle $b^*$ that is not $(ab)^*$ or $(bc)^*$. Consider the region $R(bc)$ of the plane enclosed by the shortest paths from $x_0$ to the endpoints of $\varphi(bc)$ and the edge $\varphi(bc)$ itself. We include in $R(bc)$ the two shortest paths and the edge used to define it. In particular, $R(bc)$ is a closed set. See Figure 2. We define $\chi_{\mu}(bc)$ as the maximum weight of $V(G) \cap R(bc)$.

Consider any cycle $\gamma = a_0 a_1 \ldots a_k$ of $\Xi(G, x_0)$ that is oriented clockwise. Note that, since $\gamma$ is $x_0$-star-shaped, it crosses each oriented edge of $G$ from left to right. The intuition is to build the “cycle of $G$ immediately to the right” of $\gamma$. Define the set of edges

$$F = F(\gamma) = \{ \varphi(a_i a_{i+1}) \mid \text{left}(a_{i-1}, a_i, a_{i+1}) = 1 \}.$$ 

If $F$ is empty, then $\gamma$ has only right turns. In that case, $\gamma$ only encloses $x_0$ and the lemma holds because we did include $w(x_0)$ among the values to compute the maximum. See the left of Figure 3 for this scenario.

If $F$ is not empty, the concatenation of the edges of $F$, in the same order as produced by the order of the index $i$ gives a non-crossing closed walk $C = C(\gamma)$, as it can be shown by induction. Whenever left($a_{i-1}, a_i, a_{i+1}$) = 0, the endpoint of the walk we are building does not change, while each left turn of $\gamma$ prolongs the current walk by one edge. See the right of Figure 3 and Figure 4 for a couple of examples. The walk $C$ may be a cycle, or, when several edges of $F$ have $x_0$ as an endpoint, it consists of several portions through $x_0$. It has a petal-like structure. Some of the edges in the walk may be walked in both directions. In any case, these walks can be infinitesimally perturbed to simple, closed, planar curves that enclose $F$. We use \text{int}(C)$ for the interior of any such infinitesimal perturbation of $C$.

Because there is no vertex of $G$ between $\gamma$ and $C$, we have $\text{int}(C) \cap V(G) = V_{\text{int}}(\gamma, C)$. Since $\gamma$ is
A closed walk in the proof of Lemma 4 looks like. In the right we show some of the regions $R(a_i a_{i+1})$ whose union is $\text{int}(C(\gamma))$.

\[ \text{int}(C) = \bigcup_{\text{left}(a_{i-1}, a_i, a_{i+1}) = 1} R(a_i a_{i+1}). \]

To see Figure 4 for an example of the relevant regions. We conclude that

\[
\mu(V_{\text{int}}(\gamma, G)) = \mu\left(\text{int}(C) \cap V(G)\right)
= \max\{\mu(R(a_i a_{i+1}) \cap V(G)) \mid \text{left}(a_{i-1}, a_i, a_{i+1}) = 1\}
= \max\{\chi(\mu(a_i a_{i+1}) \mid \text{left}(a_{i-1}, a_i, a_{i+1}) = 1\}.
\]

Since $x_0 \in R(a_i a_{i+1})$ for all dual edges $a_i a_{i+1}$, the correctness of the formula to compute $\mu(V_{\text{int}}(\gamma, G))$ follows.

We next argue that $\chi(\mu)$ can be computed in linear time. For each face of $G$ we compute and store the maximum weight over its incident vertices. Let $T_0$ be the unique single source shortest path of $G$ from $x_0$ and let $C = E(G) \setminus E(T_0)$ be the remaining edges. The dual of the edges of $G$ not in $T_0$, $C^*$, form a spanning tree of the dual graph $G^*$. We root $C^*$ at the dual vertex representing the outer face of $G$. Each edge $e \in C$ defines a region $A_e$ of the plane. Each edge $e \in C$ defines a dual subtree, denoted by $C^*_e$, which is the component of $C^* - e^*$ without the root. The region $A_e$ corresponds to the faces of $G$ that dualize to vertices of $C^*_e$. Using a bottom-up traversal of the tree $C$ and the values stored for each face, we can compute the values $\mu(A_e \cap V(G))$ in linear time for all edges $e \in C$. From this we can compute $\chi(\mu)$ easily because such value corresponds the maximum over $\mu(A_{\varphi(bc)} \cap V(G))$ and over weights of the vertices in the shortest paths from $x_0$ to the endpoints of $\varphi(bc)$. (We have to add those shortest path because the shortest paths from $x_0$ to the endpoints of $\varphi(bc)$ may share some prefix that does not enclose any face.)

**Lemma 5.** Let $G$ be a triangulated plane graph and let $x_0$ be a vertex of $G$. Let $\Pi = \{\pi_1, \ldots, \pi_\ell\}$ be a family of simple paths in $G^*$ with a total of $m$ edges, counted with multiplicity. After $O(n + m)$ preprocessing time, we can answer the following type of queries in $O(k)$ time.

- Given a cycle $\gamma$ that encloses $x_0$, described as a concatenation of $k$ paths from $\Pi$, return $\sigma(V_{\text{int}}(\gamma, G))$.
- Given a cycle $\gamma$ of $\Xi(G, x_0)$, described as a concatenation of $k$ paths from $\Pi$, return $\mu(V_{\text{int}}(\gamma, G))$. 


Proof. We compute for $G$ the functions $\chi_\sigma$ of Lemma 3 and the function $\chi_\mu$ of Lemma 4. For each path $\pi$ of $\Pi$ we compute the sum $\check{\chi}_\sigma(\pi)$ of $\chi_\sigma(e)$ over the edges $e$ of $\pi$. Also, for each path $\pi = a_1 a_2 \ldots a_t$ of $\Pi$ we compute and store

$$\check{\chi}_\mu(\pi) = \max\{\chi(a_i a_{i+1}) | i = 2 \ldots t-1, \text{ left}(a_{i-1}, a_i, a_{i+1}) = 1\}.$$  

Consider a cycle $\gamma$ of $G^*$ described as a concatenation of paths $\pi^1, \ldots, \pi^k$ of $\Pi$. Because of the properties of $\chi_\sigma$ in Lemma 3 we have

$$\sigma(\mathsf{ext}(\gamma, G)) = \sum_{a \rightarrow b \in \gamma} \chi_\sigma(a \rightarrow b) = \sum_{i=1}^k \sum_{a \rightarrow b \in \pi^i} \chi_\sigma(a \rightarrow b) = \sum_{i=1}^k \check{\chi}_\sigma(\pi^i).$$

It follows that we can compute $\sigma(\mathsf{ext}(\gamma, G))$ in $O(k)$ time with the precomputed sums, and thus we can obtain also $\sigma(\mathsf{int}(\gamma, G))$ in the same time bound.

For the maximum weight, we do something similar. For each path $\pi^i$, let $a^i b^i$ be the first edge and let $c^i d^i$ be the last edge of $\pi^i$. Thus, $d^i = a^{i+1}$. Using the properties of $\chi_\mu$ in Lemma 4 we then have that $\mu(\mathsf{int}(\gamma, G))$ is the maximum in

$$\{\chi_\mu(\pi^i) | i = 1 \ldots k\} \cup \{\chi_\mu(c^i, d^i, b^{i+1}) | i = 1 \ldots k-1, \text{ left}(c^i, d^i, b^i) = 1\} \cup$$

$$\{\chi_\mu(c^k, d^k, b^1) \cdot \text{left}(c^k, d^k, b^1)\}.$$

The first set of values is precomputed and available. The second and third sets of values arise because we have to check the concatenation of two consecutive paths for left turns, and it takes additional $O(k)$ time. The result follows.

We want to achieve something similar for cycles defined by subpaths. Using binary search trees to represent subpaths we achieve the following.

**Lemma 6.** Let $G$ be a triangulated plane graph and let $x_0$ be a vertex of $G$. Let $\Pi = \{\pi_1, \ldots, \pi_\ell\}$ be a family of simple paths in $G^*$ with a total of $m$ edges, counted with multiplicity. After $O(n + m)$ preprocessing time, we can answer the following type of queries in $O(k \log n)$ time.

- Given a cycle $\gamma$ that encloses $x_0$, described as a concatenation of $k$ subpaths from $\Pi$, return $\sigma(\mathsf{int}(\gamma, G))$.
- Given a cycle $\gamma$ of $\mathcal{E}(G, x_0)$, described as a concatenation of $k$ subpaths from $\Pi$, return $\mu(\mathsf{int}(\gamma, G))$.

**Proof.** For each path $\pi_i \in \Pi$ of length $m_i$ we build a family $\Pi_i$ of $O(m_i)$ subpaths recursively, as it is usually done for binary search trees. Starting with $\Pi_i = \{\pi_i\}$, for each path contained in $\Pi_i$, we split it into 2 subpaths of roughly the same number of edges, and add them to $\Pi_i$. We then compute a binary search tree that represents this decomposition. Any given subpath $\pi'_i$ of $\pi_i$ can be expressed as the concatenation of $O(\log m_i) = O(\log n)$ subpaths of $\Pi_i$. Moreover, such a representation can be found in $O(\log n)$ time using the standard binary search tree. The sum of the lengths of the paths in $\Pi_i$ is $O(m_i \log m_i)$.

We can then use Lemma 5 with the family $\bigcup_i \Pi_i$. The preprocessing takes $O(n + \sum_i m_i \log m_i) = O(n + m \log n)$. 

8
Consider a cycle \( \gamma \) given as the concatenation of \( k \) paths \( \pi^1, \ldots, \pi^k \), each of them a subpath of some path in \( \Pi \). For each \( \pi^i \), we spend \( O(\log n) \) time to find the \( O(\log n) \) paths of \( \bigcup_i \Pi_i \) whose concatenation is \( \pi^i \). Thus we have expressed \( \gamma \) as the concatenation of \( O(k \log n) \) paths in \( \bigcup_i \Pi_i \), and the time bound follows from the guarantees of Lemma 5.

For our application we will have to deal with pieces. Let \( P \) be a piece of \( G \). Each cycle of path \( \gamma \) in \( G^* \) projects onto a non-crossing closed walk in \( P^* \), denoted by \( P^*(\gamma) \). See Figure 5 for an example. Although the projection \( P^*(\gamma) \) may be a (closed) walk with repeated vertices, it can be infinitesimally perturbed to a non-crossing (closed) curve. Note that \( \text{int}(\gamma) \cap V(P) = V_{\text{int}}(P^*(\gamma), P) \). We define

\[
\Xi_{P^*}(G, x0) = \{ P^*(\gamma) \mid \gamma \in \Xi(G, x0) \}.
\]

Although the preprocessing time in the following result can probably be improved, it is enough for our purposes because currently the bottleneck is somewhere else.

**Proposition 7.** Let \( G \) be a triangulated plane graph with \( n \) vertices, let \( P \) be a piece of \( G \) with \( r \) vertices and \( O(1) \) holes, and let \( x_0 \) be a vertex of \( P \). Let \( \Pi_{P^*} = \{ P^*(\pi^1), \ldots, P^*(\pi^t) \} \) be a family of paths in \( P^* \) obtained by projections of simple paths in \( G^* \), and let \( m \) be the total number of edges in \( \Pi_{P^*} \), counted with multiplicity. After \( O(r^2 n + m) \) preprocessing time, we can answer the following type of queries in \( O(k \log n) \) time.

- Given a cycle \( P^*(\gamma) \) of \( P^* \) that encloses \( x_0 \), described as a concatenation of \( k \) subpaths from \( \Pi_{P^*} \), return \( \sigma(V_{\text{int}}(P^*(\gamma), G)) \).
- Given a cycle \( P^*(\gamma) \) of \( \Xi_{P^*}(G, x_0) \), described as a concatenation of \( k \) subpaths from \( \Pi_{P^*} \), return \( \mu(V_{\text{int}}(P^*(\gamma), P)) \).

**Proof.** We change the weight of each vertex of \( V(G) \setminus V(P) \) to 0. For the vertices of \( P \) we use the original weights.

For each hole \( h \) of \( P \) and each edges \( e_1 \) and \( e_2 \) on the boundary of \( h \), we compute a dual path \( \pi_{e_1, e_2} \) in \( G^* \) between \( e_1^* \) and \( e_2^* \). Now, in each path \( P^*(\pi_i) \) of \( \Pi_{P^*} \) where there is an appearance of \( e_1^* e_2^* \) we replace it with \( \pi_{e_1, e_2} \). There are \( O(r^2) \) such possible replacements per hole and thus \( O(r^2) \) such paths in total.

Thus, we have spent \( O(r^2 n + m) \) time for all the replacements together.

Now we can apply the method of Lemma 6 treating each subpath \( \pi_{e_1, e_2} \) as an elementary, indivisible piece, equivalent to a single edge of \( G^* \).

\[ \Box \]
4 Abstract Voronoi diagrams

Abstract Voronoi diagrams were introduced by Klein [21] as a way to handle together several of the
different types of Voronoi diagrams that were appearing. The concept is restricted to the plane $\mathbb{R}^2$. They
are defined using the concept of bisectors and dominant regions. We will use the definition by Klein,
Langetepe and Nilforoushan [23], as it seems the most recent and general. For the construction, we use
the randomized incremental construction of Klein, Mehlhorn and Meiser [24], also discussed in [23] for
their framework. In our notation, we will introduce an $A$ in front to indicate we are talking about objects
in the abstract Voronoi diagram.

Let $S$ be a finite set, which we refer to as abstract sites. For each ordered $(p, q) \in S^2$ of distinct sites,
we have a simple planar curve $AJ(p, q)$ and an open domain $AD(p, q)$ whose boundary is $AJ(p, q)$. We
refer to the pair $(AJ(p, q), AD(p, q))$ as an abstract bisector. Define for each $p \in S$ the abstract Voronoi
region $AVR(p, S) = \bigcap_{q \in S \setminus \{p\}} AD(p, q)$. Then the abstract Voronoi diagram of $S$, denoted by $AVD(S)$, is
defined as $AVD(S) = \mathbb{R}^2 \setminus \bigcup_{p \in S} AVR(p, S)$.

The intuition is that the set $AD(p, q)$ is the set of points that are closer to $p$ than to $q$ and that
$AJ(p, q)$ plays the role of bisector. Then, $AVR(p, S)$ stands for the points that are dominated by $p$, when
compared against all $q \in S \setminus \{p\}$. Note that $AVR(p, S)$ is an open set because it is the intersection of open
sets. The abstract Voronoi diagram, $AVD(S)$ would then be the set of points where no site dominates,
meaning that at least two sites are “equidistant” from the point. However, the theory does not rely on
any such interpretations. This makes it very powerful but less intuitive: some arguments become more
cumbersome.

While these concepts can be considered in all generality the theory is developed assuming that certain
axioms are satisfied. The system of abstract bisectors $\{(AJ(p, q), AD(p, q)) \mid p, q \in S, p \neq q\}$ is admissible
if it satisfies the following properties:

(A1) For all distinct $p, q \in S$, $J(p, q) = J(q, p)$.

(A2) For all distinct $p, q \in S$, the plane $\mathbb{R}^2$ is the disjoint union of $D(p, q)$, $J(p, q)$ and $D(q, p)$.

(A3) There exists a special point in the plane, which we call $p_\infty$, such that, for all distinct $p, q \in S$, the
curve $J(p, q)$ passes through $p_\infty$.2

(A4) For each subset $S'$ of $S$ with 3 elements and each $p \in S'$, the abstract Voronoi region $AVR(p, S')$ is
path connected.

(A5) For each subset $S'$ of $S$ with 3 elements we have $\mathbb{R}^2 = \bigcup_{p \in S} AVR(p, S')$.

For the rest of the discussion on abstract Voronoi diagrams, we assume that these axioms are satisfied.
Note that axioms (A4)-(A5) are not the ones given in the definition of [23] but, as they show in their
Theorem 15, they are equivalent. In this regard, our definition is closer to the one given in [22]. Since
we are going to work with very natural, no-pathological Voronoi diagrams, any of the sets of axioms used
in any of the other papers we have encountered also works in our case. Assuming these axioms, one can
show that the abstract Voronoi diagram $AVD(S)$ is a plane graph [23, Theorem 10]. This brings a natural
concept of abstract Voronoi edge and abstract Voronoi vertex as those being vertices (of degree $\geq 3$)
and edges in the plane graph $AVD(S)$.

Klein, Mehlhorn and Meiser provide a randomized incremental construction of abstract Voronoi
diagrams. One has to be careful about what it means to compute an abstract Voronoi diagrams, since it is

2Usually the axiom tells that the stereographic projection to the sphere of the curve $J(p, q)$ can be completed to a closed
Jordan curve passing through the north pole. For us it will be more convenient to project from a different point and complete
all curves within the plane to make them pass through $p_\infty$. 

not even clear how the input is specified. For their construction, they assume as primitive operation that one can compute the abstract Voronoi diagram of any five abstract sites. The output is combinatorially described with a plane graph $H$ and telling the description of each vertex and edge of $H$. The description of a vertex or an edge is a pointer to a vertex or an edge, respectively, in the abstract Voronoi diagram for at most four abstract sites. Thus, we tell that an edge $e$ of $H$ corresponds to some precise abstract edge $e'$ of AVD($S'$), where $|S'| \leq 4$. Whether AVD($S'$) can be computed explicitly or not, it depends on how the input bisectors can be manipulated.

Klein, Mehlhorn and Meiser consider a special case, which is the one we will be using, where the basic operation requires the abstract Voronoi diagram of only four sites. (This particular case is not discussed in [23], but they discuss the general case.)

**Theorem 8** (Klein, Mehlhorn and Meiser [24]). Assume that we have an admissible system of abstract bisectors for a set $S$ of $m$ sites. The abstract Voronoi diagram of $S$ can be computed in expected time $O(m \log m)$ using an expected number of $O(m \log m)$ elementary operations. If the abstract Voronoi diagram of any three sites contains at most one abstract Voronoi vertex, besides the special point $p_\infty$, then an elementary operation is the computation of an abstract Voronoi diagram for four sites.

5 Voronoi diagrams in planar graphs

We will need additively weighted Voronoi diagrams in plane graphs. We first define Voronoi diagrams for arbitrary graphs. Then we discuss a representation using the dual graphs that works only for plane graphs and discuss some folklore properties. See for example the papers of Marx and Pilipczuk [25] or Colin de Verdière [8] for similar intuitions. The dual representation is the key to be able to use the machinery of abstract Voronoi diagrams as a black box.

5.1 Arbitrary graphs

Let $G$ be an arbitrary graph, not necessarily planar, with positive edge lengths. A site $s$ is a pair $(v_s, w_s)$, where $v_s \in V(G)$ is its location, and $w_s \in \mathbb{R}_{\geq 0}$ is its weight. With a slight abuse of notation, we will use $s$ instead of $v_s$ as the vertex. For example, for a site $s$ we will write $s \in V(G)$ instead of $v_s \in S$ and $d_G(x, s)$ instead $d_G(x, v_s)$.

Let $S$ be a set of sites in $G$. For each $s \in S$, its **graphic Voronoi region**, denoted GV$_G(s, S)$, is defined by

$$GV_G(s, S) = \{x \in V(G) \mid \forall t \in S \setminus \{s\} : d_G(x, s) + w_s \leq d_G(x, t) + w_t\}.$$ 

See Figure 6 for an example. Even assuming that all distances in $G$ are distinct, we may have $d_G(x, s) + w_s = d_G(x, t) + w_t$ for some vertex $x$. Also, some Voronoi cells may be empty. In our case, we will only deal with cases where these two things cannot happen. We say that the set $S$ of sites is **generic** when for each $x \in V(G)$, we have $d_G(x, s) + w_s \neq d_G(x, t) + w_t$, and it is **independent** when each Voronoi cell is nonempty. It is easy to see that, if $S$ is a generic, independent set of sites, then $s \in GV_G(s, S)$ and each vertex $x$ of $V(G)$ belongs to precisely one graphic Voronoi cell $GV_G(s, S)$ over all $s \in S$.

The **graphic Voronoi diagram** of $S$ (in $G$) is the collection of graphic Voronoi regions:

$$GV_D_G(S) = \{GV_G(s, S) \mid s \in S\}.$$

The following property is standard.

**Lemma 9.** Let $S$ be a generic, independent set of sites. Then for each $s \in S$ it holds the following:

- For each $x$ in $GV_G(s, S)$, the shortest path from $x$ to $s$ is contained in $GV_G(s, S)$. 

11
Figure 6: A Voronoi diagram for four sites. An abstract Voronoi edge is marked with thicker pen.

- \( GVR_G(s, S) \) induces a connected subgraph of \( G \).

**Proof.** Let \( x \) be a vertex of \( GVR_G(s, S) \) and let \( P(s, x) \) be the shortest path in \( G \) from \( s \) to \( x \). Assume, for the sake of reaching a contradiction, that some vertex \( y \) on \( P(s, x) \) is contained in some other Voronoi cell \( GVR_G(t, S) \), where \( t \neq s \). Because of uniqueness of shortest paths, this means that \( d_G(t, y) < d_G(s, y) \).

However, this implies that

\[
d_G(t, x) \leq d_G(t, y) + d_G(y, x) < d_G(s, y) + d_G(y, x) = d_G(s, x),
\]

where in the last equality we have used that \( y \) lies in the shortest path \( P(s, x) \). The obtained inequality \( d_G(t, x) < d_G(s, x) \) contradicts the property that \( x \in GVR_G(s, S) \). This proves the first item.

To show the second item, note that subgraph of \( G \) induced \( GVR_G(s, S) \) contains (shortest) paths from \( s \) to all vertices of \( GVR_G(s, S) \) because of the previous item. \( \square \)

For each two sites \( s \) and \( t \), we define the graphic dominance region of \( s \) over \( t \) as

\[
GD_G(s, t) = GVR_G(s, \{s, t\}) = \{ x \in V(G) \mid d_G(x, s) + w_s \leq d_G(x, t) + w_t \}.
\]

**Lemma 10.** For each \( s \in S \) we have \( GVR_G(s, S) = \bigcap_{t \in S \setminus \{s\}} GD_G(s, t) \).

**Proof.** We note that

\[
GVR_G(s, S) = \{ x \in V(G) \mid \forall t \in S \setminus \{s\} : d_G(x, s) + w_s \leq d_G(x, t) + w_t \}
= \bigcap_{t \in S \setminus \{s\}} \{ x \in V(G) \mid d_G(x, s) + w_s \leq d_G(x, t) + w_t \}
= \bigcap_{t \in S \setminus \{s\}} GD_G(s, t).
\]

\( \square \)
5.2 Plane graphs

Now we will make use of graph duality to provide an alternative description of additively weighted Voronoi diagrams in plane graphs. The aim is to define Voronoi diagrams geometrically using bisectors, where a bisector is just going to be a cycle in the dual graph.

Consider two sites \( s \) and \( t \) in \( G \) and define

\[
E_G(s, t) = \{ xy \in E(G) \mid x \in GD_G(s, t), \ y \in GD_G(t, s) \}.
\]

Thus, we are taking the edges that have each endpoint in different Voronoi cells of \( GVD_G(s, t) \). We denote by \( E^*_G(s, t) \) their dual edges.

**Lemma 11.** Let \( \{ s, t \} \) be a generic and independent set of sites. Then the edges of \( E^*_G(s, t) \) define a cycle \( \gamma \) in \( G^* \). Moreover, if \( s \in V_{\text{int}}(\gamma, G) \), then \( V_{\text{int}}(\gamma, G) = GD_G(s, t) \) and \( V_{\text{ext}}(\gamma, G) = GD_G(t, s) \).

**Proof.** Let \( A^* \) be an arbitrary set of dual edges. It is well known that \( A^* \) is the edge set of a cycle if and only if \( G - A \) has precisely two connected components. Moreover, two faces \( u^* \) and \( v^* \) of \( G^* \) are in the same side of the cycle defined by \( A^* \) if and only if \( u \) and \( v \) are in the same connected component of \( G - A \). See for example the proof in [9, Proposition 4.6.1] or [3, Theorem 10.16].

When \( \{ s, t \} \) is generic and independent, we have \( V(G) = GD_G(s, t) \cup GD_G(t, s) \) with \( GD_G(s, t), \ GD_G(t, s) \neq \emptyset \). This means that \( E_G(s, t) \) is the edge cut between \( GD_G(s, t) \) and its complement, \( GD_G(t, s) \). Moreover, by Lemma 9, the subgraphs of \( G \) induced by \( GD_G(s, t) \) and by \( GD_G(t, s) \) are connected. Therefore \( G - E_G(s, t) \) has precisely two connected components, and thus \( E^*_G(s, t) \) is the edge set of a cycle \( \gamma \) in \( G^* \).

Assume that \( s \in V_{\text{int}}(\gamma, G) \). Since \( GD_G(s, t) \) is the vertex set of the connected component of \( G - E_G(s, t) \) that contains \( s \), the faces of \( \{ u^* \mid u \in GD_G(s, t) \} \) are in \( \text{int}(\gamma) \) and the faces \( \{ v^* \mid v \in GD_G(t, s) \} \) are in \( \text{ext}(\gamma) \). Since a vertex \( u \) of \( G \) is the unique vertex of \( G \) contained in the dual face \( u^* \) of \( G^* \), the result follows.

When \( s \) and \( t \) are independent and generic, we define the **bisector** of \( s \) and \( t \), denoted as \( \text{bis}_G(s, t) \) as the curve in the plane defined by the cycle of \( E^*_G(s, t) \), as guaranteed in the previous lemma. We also define \( D_G(s, t) \) as the face defined by \( \text{bis}_G(s, t) \) that contains \( s \). We then have

\[
D_G(s, t) = \left( \bigcup_{v \in GD_G(s, t)} v^* \right)^\circ.
\]

Note that the pair \( (\text{bis}_G(s, t), D_G(s, t)) \) is the type of pair used to define abstract Voronoi diagrams. However, we cannot use the machinery of abstract Voronoi diagrams for arbitrary sites because of axiom (A3). In our case bisectors may not pass through the “infinity point” \( p_\infty \). However, we can use it when all the sites are in the outer face of \( G \). We next show this.

**Lemma 12.** Let \( G \) be a plane graph and let \( S \) be a generic, independent set of sites located in the outer face of \( G \). Then the system of abstract bisectors \( \{(\text{bis}_G(s, t), D_G(s, t)) \mid s, t \in S, s \neq t \} \) is admissible.

**Proof.** It is clear that the system of abstract bisectors \( \{(\text{bis}_G(s, t), D_G(s, t)) \mid s, t \in S, s \neq t \} \) satisfies axioms (A1) and (A2) of the definition.

We next show the validity of axiom (A3). Let \( v_\infty \) be the vertex of \( G^* \) dual to the outer face of \( G \). Consider any two sites \( s \) and \( t \) of \( S \). Since \( GVR_G(s, S) \) and \( GVR_G(t, S) \) are nonempty, also \( GD_G(s, t) \) and \( GD_G(t, s) \) are nonempty. Since \( s \) and \( t \) are located in the outer face of \( G \) the bisector \( \text{bis}_G(s, t) \) passes through \( v_\infty \). Indeed, the dual faces \( s^* \) and \( t^* \) have to be in different sides of the dual cycle \( \text{bis}_G(s, t) \) and, since \( s \) and \( t \) are on the outer face of \( G \), that can happen only if \( \text{bis}_G(s, t) \) passes through \( v_\infty \). Thus,
if we take the geometric position of \( v_\infty \) as \( p_\infty \), all the bisecting curves pass through \( p_\infty \) and axiom (A3) holds.

For axiom (A4), consider any three sites \( r, s, t \) of \( S \) and let \( S' = \{r, s, t\} \). Following the definition of \( \text{AVR}(\cdot, \cdot) \) we have

\[
\text{AVR}(s, S') = D(s, r) \cap D(s, t).
\]

Because of Lemma 11, \( D_G(s, r) \) is the interior of the union of the closure of the faces dual to \( G_D(s, r) \).

Thus we have

\[
\text{AVR}(s, S') = \bigcap_{v \in G_D(s, r)} \overline{v^*} \bigcap_{v \in G_D(s, t)} \overline{v^*} \bigcap_{v \in \overline{G_V}(s, S')} v^* = \bigcup_{v \in \overline{V(G)}} v^* = \mathbb{R}^2.
\]

where in the last inequality we have used Lemma 10. Since the vertices of \( G_V(s, S') \) form a connected subgraph of \( G \) (Lemma 9), the domains \( \overline{v^*} \), when \( v \) iterates over \( G_V(s, S') \), are glued through the primal edges, and \( \text{AVR}(s, S') \) is path connected. This proves axiom (A4).

Axiom (A5) is shown similarly. Following the notation and the observations from the previous paragraph, we use that

\[
\overline{\text{AVR}(s, S')} = \bigcup_{v \in \overline{G_V}(s, S')} v^*
\]

and that \( V(G) = G_V(r, S') \cup G_V(s, S') \cup G_V(t, S') \), we conclude that

\[
\overline{\text{AVR}(r, S')} \cup \overline{\text{AVR}(s, S')} \cup \overline{\text{AVR}(t, S')} = \bigcup_{v \in \overline{V(G)}} v^* = \mathbb{R}^2.
\]

From now on, whenever we talk about the abstract Voronoi diagram we refer to the abstract Voronoi diagram defined by the system of bisectors \( \{ (\text{bis}_G(s, t), D_G(s, t)) \mid s, t \in S, s \neq t \} \). We have defined Voronoi regions of plane graphs in two different ways: using distances in the primal graph \( G \), called graphic Voronoi regions, and using bisectors defined as curves in the plane, called abstract Voronoi regions. We next make sure that the definitions match, when restricted to vertices of \( G \).

**Lemma 13.** Let \( S \) be a generic, independent set of sites. Then, for each \( s \in S \), we have \( G_V(s, S) = V(G) \cap \text{AVR}(s, S) \).

**Proof.** Recall the definition

\[
\text{AVR}(s, S) = \bigcap_{t \in S \setminus \{s\}} D_G(s, t).
\]

Using that

\[
D_G(s, t) = \left( \bigcup_{v \in G_D(s, t)} \overline{v^*} \right)^*.
\]
we obtain that
\[
AVR(s, S) = \bigcap_{r \in S \setminus \{s\}} \left( \bigcup_{v \in GD_r(s, t)} \overline{v}^s \right) = \bigcup_{v \in \{r \in S \}: GD_r(s, t)} \overline{v}^s
\]
where in the last equality we used Lemma 10. Since the only vertex of \( V(G) \) contained in the dual face \( \overline{v}^s \) is precisely \( v \), and it lies in the interior of \( v^s \), we get that \( V(G) \cap AVR(s, S) = GVR(s, S) \).

We know that the abstract Voronoi diagram AVD(S) is a plane graph, and by construction it is contained in the dual graph \( G^* \). An abstract Voronoi vertex corresponds to a vertex in the dual graph \( G^* \). An abstract Voronoi edge corresponds to a path in the dual graph \( G^* \). More precisely, any abstract Voronoi edge corresponds to a portion of a bisector \( bis_G(s, t) \) whose endpoints are vertices of \( G^* \).

**Lemma 14.** The abstract Voronoi diagram of any 3 sites in the outer face of \( G \) has at most one vertex, besides \( v_\infty \).

**Proof.** Assume that \( S \) is the set of 3 sites. Since each site \( s \in S \) is in the outer face, the abstract Voronoi diagram AVD(\( s, S \)) contains the dual face \( s^* \), which is incident to \( v_\infty \). It follows that all faces have a common vertex in \( v_\infty \). Since a plane graph with 3 faces can have at most 2 vertices, the result follows.

### 5.3 Algorithmic aspects

We next provide tools to manipulate portions of the bisectors. For the rest of this section, we assume that \( G \) is a plane graph with \( n \) vertices.

**Lemma 15.** For any two generic, independent sites \( \{s, t\} \) in the outer face of \( P \) we can compute \( bis_G(s, t) \) in \( O(r) \) time.

**Proof.** We add a vertex \( v_0 \) and edges \( v_0t, v_0s \). We set the length of \( v_0s \) to the weight \( w_s \) of \( s \), and similarly the length of \( v_0t \) to \( w_t \). Now we compute a single source shortest path tree from \( v_0 \) in linear time [15]. The vertices that are reached from \( v_0 \) through \( s \) belong to \( GVR(s, \{s, t\}) \), while the others belong to \( GVR(t, \{s, t\}) \). Now we can mark the edges of \( A_G(s, t) \) and construct the cycle \( bis_G(s, t) \) in the dual graph.

**Lemma 16.** Consider any two sites \( s = (v_s, w_s) \) and \( t = (v_t, w_t) \) in the outer face of \( G \). Consider the family of bisectors \( bis_G((v_s, w_s), (v_t, w_t)) \) as a function of the weights \( w_s \) and \( w_t \). There are at most \( O(n^2) \) different bisectors. We can compute and store all the bisectors in \( O(n^2) \) time such that, given two values \( w_s \) and \( w_t \), the corresponding representation of \( bis_G((v_s, w_s), (v_t, w_t)) \) is accessed in \( O(log n) \) time.

**Proof.** From the definition it is clear that
\[
bis_G((v_s, w_s), (v_t, w_t)) = \begin{cases} 
  bis_G((v_s, 0), (v_t, w_t - w_s)) & \text{if } w_t \geq w_s, \\
  bis_G((v_s, w_s - w_t), (v_t, 0)) & \text{if } w_t < w_s.
\end{cases}
\]

Thus, it is enough to consider the bisectors \( bis_G((v_s, 0), (v_t, w)) \) and \( bis_G((v_s, w), (v_t, 0)) \) parameterized by \( w \in \mathbb{R}_{\geq 0} \). Each bisector \( bis((v_s, 0), (v_t, w)) \) is a cycle in the dual graph \( G^* \) and the cycles are nested: as \( w \) increases, the graphic dominance region \( GD_G(s, t) \) monotonically grows and \( D_G(s, t) \) also monotonically...
increases. Since any two different cycles that are nested must differ by at least one face of $G^*$, there are at most $O(n)$ different bisectors.

We compute the distances from $v_s$ to all vertices of $G$ and the distances from $v_t$ to all vertices of $G$ using twice a linear-time single source shortest path [15]. For each vertex $x \in V(G)$, define the value \( \eta_x = d_G(v_s, x) - d_G(v_t, x) \). The vertex $x$ is in $GD_G(s, t)$ when $w < \eta_x$, in $GD_G(t, s)$ when $w > \eta_x$, and we have a degenerate case when $w = \eta_x$. Thus, we can compute the values $\{\eta_x \mid x \in V(G)\}$, sort them and store them sorted in a table. For each $w$ between two consecutive values of $\{\eta_x \mid x \in V(G)\}$ we compute the bisector using Lemma 15 and store it with its predecessor of $\{\eta_x \mid x \in V(G)\}$. Given a query with shifts $w_s \leq w_t$, we use binary search in $O(\log n)$ time for the value $w_t - w_s$ and locate the relevant bisector. The case $w_s > w_t$ is symmetric.

As mentioned before, an abstract Voronoi vertex is just a vertex of $G^*$ and an abstract Voronoi edge is encoded in the dual graph by a tuple $(s, t, a, b)$, meaning that the edge is the portion of bis$(s, t)$ between dual vertices $a$ and $b$ in some prescribed order, like for example the clockwise order of bis$(s, t)$.

Lemma 17. Let $G$ be a connected plane graph with $n$ vertices and $b$ vertices in the outer face of $G$. There is a data structure with the following properties. The preprocessing time is $O(b^3n^2)$. For any generic, independent set $S$ of 4 sites placed on the outer face of $G$, the abstract Voronoi diagram $AVD(S)$ can be computed in $O(\log n)$ time. The output is given combinatorially as a collection of abstract Voronoi vertices and edges encoded in the dual graph $G^*$.

Proof. Let $X$ be the set of vertices on the outer face of $G$. We use Lemma 16 to compute and store all the possible bisectors. Since there are $b^2$ different possible sites, for each pair of sites there are $O(n)$ different bisectors, and for each bisector we spend $O(n)$ space and preprocessing time, we have spent a total of $O(b^2n^2)$ time.

For each bisector $\beta$, we preprocess it to quickly figure out the circular order of its vertices: given two vertices $a$ and $b$ on $\beta$, is the clockwise order along $\beta$ given by $a, b, v_{\infty}$ or by $b, a, v_{\infty}$? For each bisector $\beta$ we can make a table $T_\beta[\cdot]$ indexed by the vertices such that $T_\beta[a]$ is the position of $a$ along $\beta$, when we walk $\beta$ clockwise starting from $v_{\infty}$. We set $T_\beta[a]$ to undefined when $a$ does not appear in $\beta$. Thus, given 2 vertices of $G^*$, we can decide their relative order along $\beta$ in $O(1)$ time. The time and space for this, over all bisectors, is also $O(b^2n^2)$.

Next, for each possible 3 sites located in $X$, we compute and store its Voronoi diagram. The description of the Voronoi diagram is as a plane graph $H$. Each vertex of $H$ is a vertex of $G^*$. Each edge of $H$ is described as a triple $(s, t, a, b)$ that means that the edge is the portion of bis$(s, t)$ between the dual vertex $a$ and the dual vertex $b$. There are $O(b^3n^2)$ different Voronoi diagrams. We have $O(|X|^2) = O(b^3)$ choices for the vertices hosting the sites and $n^2$ combinatorially different weights because only the difference between the weights is relevant. For example, if $w_s$ is the smallest of the weights, we can replace $w_s$ by $0$, $w_t$ by $w_t - w_s$ and $w_r$ by $w_r - w_s$. We store the Voronoi diagram indexed by the tuple $(r, s, t, w_r - w_s, w_r - w_t)$. For the first 3 elements we use a 3-dimensional table and for the last 2 elements we use a binary search, as also done in Lemma 16. Thus, for a given triple $(t, w_t, )$, we can find the representation of the Voronoi diagram of those three sites in $O(\log n)$ time. The representation of one such Voronoi diagram takes $O(1)$ time, since we only need to provide pointers to parts of the bisectors.

Finally, we make a table $T_X[\cdot]$ such that, for $u \in X$, $T_X[u]$ is the rank of $u$ when walking along the boundary of $G$ and, for $u \notin X$, we have $T_X[u]$ undefined. Thus, given 3 vertices of $X$ we can deduce their circular ordering along the boundary of the outer face of $G$ in $O(1)$ time. This finishes the preprocessing.

Assume that we are given a set $S$ of 4 sites and we want to compute its abstract Voronoi diagram. We recover the abstract Voronoi diagrams for each subset $\binom{S}{3}$ in $O(\log n)$ time, using the stored data.
If there are two sites \( s, t \in S \) such that their bisector \( \text{bis}(s, t) \) is in full in the Voronoi diagram of subset \( S' \) with \( |S'| = 3 \) and \( \{s, t\} \subset S' \subset S \), then in the abstract Voronoi diagram of \( S \) there is a region bounded only by \( \text{bis}(s, t) \). We can then compose that bisector and the abstract Voronoi diagram of the other three sites to obtain the final Voronoi diagram. See the left of Figure 7. (It may be that we have more than one such “isolated” abstract Voronoi region.)

In the opposite case, in the abstract Voronoi diagram there is no abstract Voronoi region that is bounded by a unique bisector. The abstract Voronoi diagram restricted to the interior faces of \( G \) is connected. The shape of such a Voronoi diagram can be only one of two, depending on which opposite sites share a common edge. See the center and right side of Figure 7. Let \( q, r, s, t \) be the sites in clockwise order along the boundary of \( G \). We can infer this order in \( O(1) \) time through the table \( T_X[\cdot] \). Assume, by renaming the sites if needed, that \( \text{bis}(s, q) \) encloses \( s \). We compute in \( O(\log n) \) time the vertex \( a \) of \( \text{AVD}(\{q, r, s\}) \) and the vertex \( b \) of \( \text{AVD}(\{q, s, t\}) \). If the cyclic order of \( v_\infty, b, a \) along \( \text{bis}(q, s) \) is clockwise, then \( a \) and \( b \) are vertices of the abstract Voronoi diagram of \( \{q, r, s, t\} \). Otherwise, the vertices of the abstract Voronoi diagram of \( \{q, r, s, t\} \) are the vertices of \( \text{AVD}(\{r, s, t\}) \) and \( \text{AVD}(\{t, q, r\}) \). From this information we can easily construct the abstract Voronoi diagram of \( \{q, r, s, t\} \).

**Theorem 18.** Let \( G \) be a connected plane graph with \( n \) vertices and \( b \) vertices in the outer face of \( G \). There is a data structure with the following properties. The preprocessing time is \( O(b^3n^2) \). For any generic and independent set \( S \) of \( m \) sites placed on the outer face of \( G \), the abstract Voronoi diagram \( \text{AVD}(S) \) can be computed in \( O(m \text{ polylog } n) \) expected time. The output is given combinatorially as a collection of abstract Voronoi vertices and edges encoded in the dual graph \( G^* \).

**Proof.** We apply the preprocessing of Lemma 17. We spend \( O(b^3n^2) \) time and, given any four sites on the outer face of \( G \), we can compute its abstract Voronoi diagram in \( O(\log n) \) time.

Assume that we are given a set \( S \) of \( m \) sites placed in the outer face of \( G \). Because of Lemma 14, any three sites have a vertex, besides the one at \( p_\infty \) (or \( v_\infty \)). According to Theorem 8, we can compute the abstract Voronoi diagram using \( O(m \log m) = O(m \log n) \) expected time and expected elementary operations, where an elementary operation is the the computation of an abstract Voronoi diagram of 4 sites. Since each elementary operation takes \( O(\log n) \) time because of the data structure of Lemma 17, the result follows.

**Dealing with holes** We will need a result like Theorem 18 for plane graphs where some interior holes are created.

Let \( G \) be a plane graph with \( b \) vertices on the outerface. Assume that we have a family \( \mathcal{C} \) of \( O(1) \) interior-disjoing cycles in \( G \), with a total of \( c \) vertices. Let \( H = G \setminus \mathcal{C} \) be the graph obtained from \( G \) by removing the vertices and the (open) edges in the interior of each cycle \( C \in \mathcal{C} \). We use \( n \) for the number
of vertices in the resulting graph $G \setminus \mathcal{C}$. Assume that, for each cycle $C \in \mathcal{C}$, we know the distances in $G$ between each two vertices of $C$. We would like to preprocess $H$ such that, when we get some sites on the outer face of $G$, we can quickly compute the restriction of the abstract Voronoi diagram to $H$. We want to avoid spending time that depends on the size of the removed parts. The running time should depend on $n$, $b$ and $c$.

Since the bisectors may go through the interior of some cycles of $\mathcal{C}$, some care is needed. For example in Figure 5 the dual cycle $\gamma$ could be a bisector in $G$, but the projection onto $H^*$ is not a simple curve. As it was used at the of Section 3, for a dual cycle $\gamma$ in $G^*$, we use $H^*(\gamma)$ for the projection of $\gamma$ on $H^*$, the dual graph of $H$.

We can add to $H$ the edges between any two vertices on the same cycle of $\mathcal{C}$ with length equal to their distance in $G$. With this we get a graph with $n$ vertices and $O(n + c^2)$ edges. Since we can compute single source shortest paths in this graph in $O((n + c^2)\log n)$ time, we can compute the projection $H^*(\text{bis}(s, t))$ of any bisector of $G$ in $O((n + c^2)\log n)$ time.

The same argument that is used in Lemma 16 shows that we can have at most $O(n)$ different projected bisectors for any two locations $v_\ell$ and $v_\nu$ of the sites. Indeed, when the bisector sweeps over the vertices of $H$ inside a cycle $\mathcal{C}$, the projection of the bisector onto $H^*$ does not change.

We have the following technical problem to use abstract Voronoi diagrams: the projection of a bisector onto $H^*$ is not a simple curve. Combinatorially, we keep encoding the projection of a bisector as a closed walk in the dual graph $H^*$. However, the geometric curve associated to a description goes out of the dual graph, as follows. For each two consecutive edges $aa'$ and $a'a''$ of each projected cycle, we always make a small shortcut in a small neighborhood of $a'$ that avoids $a'$. In such a way we obtain true geometric simple curves associated to each projection of each cycle, and we can use the technology of abstract Voronoi diagrams. The rest of the presentation goes essentially unchanged.

**Theorem 19.** Let $G$ be a connected plane graph with $b$ vertices in the outer face of $G$. Let $\mathcal{C}$ be a family of interior-disjoint cycles in $G$ with a total of $c$ vertices and assume that the pairwise distances between vertices of $\mathcal{C}$ are available. Let $n$ be the number of vertices in $H = G \setminus \mathcal{C}$. There is a data structure with the following properties. The preprocessing time is $O((b^2 nc^2 + b^3 n^2)\log n)$. For any generic and independent set $S$ of $m$ sites placed on the outer face of $G$, the restriction of the abstract Voronoi diagram $\text{AVD}(S)$ to $H$ can be computed in $O(m\log n)$ expected time. The output is given combinatorially as a collection of abstract Voronoi vertices and edges encoded in the dual graph $H^*$.

### 6 Putting the pieces together

#### 6.1 Data structure per piece

Let $G$ be a plane graph with $n$ vertices, let $x_0$ be a distinguished vertex of $G$, let $X$ be the neighbors of $x_0$ in $G$, and let $b = |X|$. We assume that $G$ has positive edge weights. We want to preprocess $G$ for the following type of queries. At query time, the weight of the edges incident to $x_0$ will be reset to new positive values. Effectively, we can assume that the weights of the $b$ edges incident to $v_0$ are undefined at preprocessing time, but at query time we get the values $\lambda(x_0, x) > 0$ for all $x \in X$. We want to find the distance to the furthest point of $G$ from $x_0$ in $G$,

$$\text{diam}(x_0, G) = \max\{d_G(x_0, x) \mid x \in V(G)\},$$

and the sum of the distances from $v_0$ to all vertices of $G$

$$\Sigma(x_0, G) = \sum_{x \in V(G)} d_G(x_0, x).$$
Note that the vertex $x_0$ is specified at preprocessing time. It can happen that for some $x \in X$ the edge $x_0x$ is never used in any shortest path tree from $x_0$. The following Lemma helps us detecting this.

**Lemma 20.** We can preprocess $G$ in $O(n \text{ polylog } n)$ time to answer in $O(b \text{ polylog } b)$ the following type of queries: given the weights $\lambda(x_0x) > 0$ for all $x \in X$, detect the vertices of $X$ that in the shortest path tree from $x_0$ have as ancestor some other vertex of $X$.

**Proof.** In the graph $H = G - x_0$ all the vertices of $X$ are cofacial. This means that we can compute all the pairwise distances between the $b$ vertices of $X$ in $O(b^2 \text{ polylog } b)$ time [6, 18]. With this data we can run the version of Dijkstra in [10] in $O(b \text{ polylog } b)$ time. □

**Theorem 21.** Assume that $G$ is a planar graph with $n$ vertices, $x_0 \in V(G)$, and $x_0$ has degree $b$ in $G$. After $O(b^3 r^2)$ preprocessing time, we can handle the following queries in $O(b \text{ polylog } n)$ expected time: given the weights of the edges incident to $x_0$ at query time, return $\text{diam}(x_0, G)$ and $\Sigma(x_0, G)$.

**Proof.** Take a plane embedding of $G$ such that $x_0$ is in the outer face. Let $H = G - x_0$. Let $X$ be neighbours of $x_0$ in $G$. We preprocess $H$ as described in Theorem 18. We also compute all the bisectors of $H$. For each vertex $x \in X$ and all the bisectors of the type $\text{bis}(x, y)$ we preprocess $H$ as explained in Proposition 7. We do this for two types of weights. The first weight is $w(y) = d_H(x, y)$ for all $y \in V(H)$. The second weight is just $w(y) = 1$ for all vertices $y$. This finishes the preprocessing.

Consider now a query specified by the edge weights $\lambda(v_0x)$, for all $x \in X$. First, we use Lemma 20 to get rid of vertices of $X$ with some ancestor in $X$. Let $X'$ be the remaining vertices of $X$. Define the weight $w_x = \lambda(v_0x)$ for all $x \in X'$ and compute the weighted Voronoi diagram with respect to the sites $((x, w_x))_{x \in X'}$. Since $X'$ is on the outer face of $H$, Theorem 18 tells that we can compute the abstract Voronoi diagram in $O(|X'| \text{ polylog } n) = O(b \text{ polylog } n)$ expected time. Now, for each site $x \in X'$, we walk along the boundary of the abstract Voronoi region $AVR(x, X')$ and use the data structure of Proposition 7 to collect how many vertices are in $AVR(x, X')$, how much the sum of their distance to $x'$, and which one has the largest distance from $x'$. For each $AVR(x, X')$ we spend $O(\text{polylog } n)$ times the complexity of the description of $AVR(x, X')$. Over all $X'$, this takes $O(|X'| \text{ polylog } n) = O(b \text{ polylog } n)$ time. From this information we can extract $\text{diam}(x_0, G)$ and $\Sigma(x_0, G)$ trivially. □

When in $G$ we remove some interior-disjoint cycles, we can use the same approach, but using Theorem 19. We omit the details.

**Theorem 22.** Assume that $G$ is a planar graph, let $\mathcal{C}$ be a family of interior-disjoint cycles in $G$ with a total of $c$ vertices and assume that the pairwise distances between vertices of $\mathcal{C}$ are available. Let $n$ be the number of vertices in $H = G \setminus \mathcal{C}$, let $x_0$ be a vertex of $H$ and let $b$ be the degree of $x_0$ in $H$. There is a data structure with the following properties. After $O((b^2 nc^2 + b^3 n^2) \text{ polylog } n)$ preprocessing time, we can handle the following queries in $O(b \text{ polylog } n)$ expected time: given the weights of the edges incident to $x_0$ at query time, return $\text{diam}(x_0, H)$ and $\Sigma(x_0, H)$, where the distances are distances in $G$.

### 6.2 Working over all pairs

**Theorem 23.** Let $G$ be a planar graph with $n$ vertices and abstract, positive edge lengths. In expected time $O(n^{11/6} \text{ polylog } n)$ we can compute for all vertices $x$ in $G$ the following:

- the sum of the distances from $x$ to all vertices of $G$ and
- the distance from $x$ to the furthest vertex of $x$ in $G$.  

19
Proof. We assume that $G$ is connected, as otherwise the problem is trivial. We triangulate $G$ adding edges of sufficiently large lengths that do not affect any distance. We also embed $G$. With a slight abuse of notation, we keep using $G$ for the resulting embedded, triangulated graph.

We compute an $r$-division $\mathcal{D} = \{P_1, \ldots, P_k\}$ of $G$ with few holes, for a parameter $r$ to be specified below. This takes $O(n)$ time (Theorem 1). We build the dense distance graph $\text{DDG}(\mathcal{D}, G)$ and preprocess it in $O(n \log r)$ time, as mentioned in Theorem 2. Then we compute the distances between all vertices of $G$ and all boundary vertices of $\mathcal{D}$. This takes $O(n) \cdot O(nr^{-1/2} \log^2 r) = O(n^2 r^{-1/2} \text{polylog}(n))$ time.

For each piece $P_i \in \mathcal{D}$ we do the following. Let $X_i$ be its boundary vertices in the outer face of $P_i$. We build a plane graph $P_i^+$ by adding a vertex $v_0$ to $P_i$ that is connected to each vertex of $X_i$. We preprocess $P_i^+$ as indicated in Theorem 22 for the vertex $v_0$. This takes $O((\sqrt{r})^{3/2} \text{polylog } r) = O(r^{7/2} \text{polylog } r)$ time per piece, for a total of $O((n/r) \cdot r^{7/2}) \text{polylog } r) = O(nr^{5/2} \text{polylog } r)$ time over all pieces. Now we iterate over all vertices $x$ of $G$ to compute $\text{diam}(x, P_i^+)$ and $\Sigma(x, P_i^+)$, where the distances are taken in $G$, as follows. For a vertex $x$ of $G$, we assign length $d_{G}(x, u)$ to the edge $v_0u$, for all $u \in X_i$. In such a way, $d_G(x, y) = d_{P_i^+}(x, y)$ for all vertices $y \in P_i$. It follows that $\text{diam}(x, P_i) = \text{diam}(v_0, P_i^+)$ and $\Sigma(x, P_i) = \Sigma(x, P_i^+)$. Note that the data structure of Theorem 22 allows us to compute these values in $O(\sqrt{r} \text{polylog } r)$ expected time because each piece $P_i$ has $O(\sqrt{r})$ boundary vertices.

Iterating over all vertices $x$ of $G$ and all pieces $P_i \in \mathcal{D}$ we compute the desired values $\text{diam}(x, P_i)$ and $\Sigma(x, P_i)$ for all pairs $(x, P_i) \in V(G) \times \mathcal{D}$ such that $x$ lies in the exterior of $P_i$. All the pairwise distances within a piece can be computed explicitly in $O(r^{3/2} \text{polylog } r)$ time (Lemma 2).

Making a similar computation where we exchange the role of interior and exterior, it is trivial to obtain $\text{diam}(x, G)$ for each vertex $x$. In the case of the sum, we have to avoid double counting some distances. For this, we just have to make sure that each vertex $y$ of $G$ is assigned to a unique piece $P_i(y)$, and then we only count the distance to $y$ when looking at the pair $(y, P_i(y))$.

The total expected time we spend is

\[
O(n^2 r^{-1/2} \text{polylog}(n)) + O(nr^{5/2}) + O(nr^{1/2} \text{polylog } r) + n \cdot O(r^{1/2} \text{polylog } r).
\]

Taking $r = n^{1/3}$ the running time is $O(n^{11/6} \text{polylog } n)$ in expectation. \qed

Corollary 24. Let $G$ be a planar graph with $n$ vertices and abstract, positive edge lengths. In expected time $O(n^{11/6} \text{polylog } n)$ we can compute the diameter and the sum of the pairwise distances in $G$.

7 Discussion

We have decided to explain the construction through the use of abstract Voronoi diagrams, instead of providing an algorithm tailored to our case. It is not clear to author which option would be better. In any case, for people familiar with randomized incremental constructions, it should be clear that the details can be worked out, once the representation through the dual graph is clear. Using a direct algorithm perhaps we could get rid of the assumption that the sites have to be in the outer face and perhaps we could actually build a deterministic algorithm.

There are deterministic algorithms to compute abstract Voronoi diagrams [21, 23]. However, they require additional elementary operations and properties. Also, when the abstract Voronoi diagram has a forest-like shape, it can be computed in linear time [2]. At this point, it is unclear to us whether these results are applicable in our case, and we will investigate this in the future.

At this point, we think that the main problem is whether the exponent $11/6$ can be further reduced.
Acknowledgments

This work was initiated at the Dagstuhl seminar Algorithms for Optimization Problems in Planar Graphs, 2016. I am very grateful to Kyle Fox, Shay Mozes, Oren Weimann, and Christian Wulff-Nilsen for several discussions on the problems treated here.

References


