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

Sergio Cabello†

Abstract

We show how to compute in $O(n^{11/6}\operatorname{polylog}(n))$ expected time the diameter and the sum of the pairwise distances in an undirected planar graph with n vertices and positive edge weights. These are the first algorithms for these problems using time $O(n^c)$ for some constant $c < 2$.

1 Introduction

Let G be an undirected graph with n vertices and abstract, positive edge lengths $\lambda\colon E(G) \to \mathbb{R}_{>0}$. 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 $\operatorname{diam}(G) = \max\{d_G(x,y) \mid x,y \in V(G)\}$, and the ***sum of the pairwise distances*** of G is $\Sigma(G) = \frac{1}{2}\sum_{(x,y)\in(V(G)^2} d_G(x,y)$. This last concept is essentially equivalent to the average distance in the graph and it is also known as the *Wiener index* in mathematical chemistry. These concepts are also natural for directed graphs.

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 [28] 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 $\Sigma(G) = n(n-1) - |E(G)|$. 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 $O(n^{11/6}\operatorname{polylog}(n))$ expected time. There are efficient algorithms for computing all the distances in a planar graph [10] or a specified subset of the distances; see for example [4, 24]. 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, 6]. For planar graphs, Wulff-Nilsen [31] gives an algorithm to compute the diameter and the sum of pairwise distances in unweighted, undirected planar graphs in $O(n^2 \log\log n/\log n)$ time, which is slightly subquadratic. Wulff-Nilsen [32] extends the result to weighted directed planar graphs with a time bound of $O(n^2(\log\log n)^4/\log n)$.

Researchers have also looked into near-optimal approximations. In particular, Weimann and Yuster [30] provide a $(1+\varepsilon)$-approximation to the diameter of undirected planar graphs in $O((n/\varepsilon^4)\operatorname{polylog} n + 2^{O(1/\varepsilon)}n)$ time. As it was mentioned in [11], a near-linear time randomized $(1+\varepsilon)$-approximation to the sum of pairwise distances in undirected planar graphs can be obtained using random sampling and an oracle for $(1+\varepsilon)$-approximate distances [14, 29]. See [13] 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.

*Supported by the Slovenian Research Agency, program P1-0297.

†Department of Mathematics, IMFM, and Department of Mathematics, FMF, University of Ljubljana, Slovenia. Email address: sergio.cabello@fmf.uni-lj.si.

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 [5]. Since we are aiming for 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 γ in the plane, let $\mathrm{int}(\gamma)$ be the bounded domain of $\mathbb{R}^2 \setminus \gamma$, and let $\mathrm{ext}(\gamma)$ be the unbounded one. For each simple closed curve γ, in particular for a cycle in the dual graph G^*, let $V_{\mathrm{int}}(\gamma, G) = \mathrm{int}(\gamma) \cap V(G)$ and $V_{\mathrm{ext}}(\gamma, G) = \mathrm{ext}(\gamma) \cap V(G)$.

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, such as γ and π. Sets of cycles and paths in the dual graph are with capital Greek letters, like Γor Π.

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

Divisions. The concept of r-division for planar graphs was introduced by Frederickson [9], and then refined and used by several authors; see for example [4, 12, 17, 25] for a sample. For us it is most convenient to use the construction of Klein, Mozes and Sommer [16]. We first state the definitions carefully, almost verbatim from [16].

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,

[1] They use the term "region", which in our opinion is more suitable. However, we are using such a term for so many things in this paper that in our context we prefer to use some other term.

$k = O(n/r)$; each edge of G is in at least one piece; each piece has $O(r)$ vertices; each piece has $O(\sqrt{r})$ boundary vertices; each piece has $O(1)$ holes.

THEOREM 2.1. (KLEIN, MOZES AND SOMMER [16]) *There is a linear-time algorithm that, for any biconnected triangulated plane graph G, outputs an r-division of G 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 [5, 15].

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

As used by several works [3, 25, 8, 23, 24], a shortest path tree in $P_i \cup \mathrm{DDG}(\mathcal{P}, G)$ can be computed in $O((r + nr^{-1/2}) \operatorname{polylog} r)$ 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 2.1. *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 $O((r+nr^{-1/2}) \operatorname{polylog} r)$ time the shortest path tree from x in $P \cup \mathrm{DDG}(\mathcal{P}, G)$.*

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

For the rest of this paper, **we will assume that each piece has a unique hole**. This is a strong assumption that does not hold. The general case is more technical and it is treated in the full version.

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 γ be a cycle in G^*. We are interested in a way to compute $\sigma(V_{\mathrm{int}}(\gamma, G))$ and $\mu(V_{\mathrm{int}}(\gamma, G))$ *locally*, after some preprocessing of G. Here, locally means that we would like to just look at the edges γ and a small neighborhood. For the sum of the weights, $\sigma(\cdot)$, this can be achieved as explained by Park and Phillips [26] and Patel [27]. However, the proof of correctness of those methods heavily uses that the sum has an inverse operation. We are not aware of any such result for computing the maximum weight, $\mu(V_{\mathrm{int}}(\gamma, G))$ or $\mu(V_{\mathrm{ext}}(\gamma, G))$. However, in our very specific case, we will only deal with very special cycles, and we can do something similar.

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

(3.1)
$$\Xi(G, x_0) \;=\; \{\gamma \mid \gamma \text{ is a } x_0\text{-star-shaped cycle of } G^*\}.$$

LEMMA 3.1. *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 γ that encloses x_0, described as a concatenation of k subpaths from Π, return $\sigma(V_{\mathrm{int}}(\gamma, G))$.*
- *Given a cycle γ of $\Xi(G, x_0)$, described as a concatenation of k subpaths from Π, return $\mu(V_{\mathrm{int}}(\gamma, G))$.*

The key idea to obtain Lemma 3.1 is to make a binary search tree for each path π_i of Π such that each subpath can be expressed as the concatenation of $O(\log n)$ canonical subpaths. Then, for each canonical subpath we can precompute and store the local information that is relevant to compute $\sigma(\cdot)$ and $\mu(\cdot)$. Using shortcuts, the result can be extended to non-triangulated graphs. We omit the details.

4 Abstract Voronoi diagrams

Abstract Voronoi diagrams were introduced by Klein [18] 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 [20], as it seems the most recent and general. For the construction, we use the randomized incremental construction of Klein, Mehlhorn and Meiser [21], also discussed in [20] 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 $\mathrm{AJ}(p, q)$ and an open domain $\mathrm{AD}(p, q)$ whose boundary is $\mathrm{AJ}(p, q)$. We refer to the pair $(\mathrm{AJ}(p, q), \mathrm{AD}(p, q))$ as an ***abstract bisector***. Define for each $p \in S$ the ***abstract Voronoi region***

$\mathrm{AVR}(p,S) = \bigcap_{q\in S\setminus\{p\}} \mathrm{AD}(p,q)$. Then the ***abstract Voronoi diagram*** of S, denoted by $\mathrm{AVD}(S)$, is defined as $\mathrm{AVD}(S) = \mathbb{R}^2 \setminus \bigcup_{p\in S} \mathrm{AVR}(p,S)$.

The intuition is that the set $\mathrm{AD}(p,q)$ is the set of points that are closer to p than to q and that $\mathrm{AJ}(p,q)$ plays the role of bisector. Then, $\mathrm{AVR}(p,S)$ stands for the points that are dominated by p, when compared against all $q \in S \setminus \{p\}$. Note that $\mathrm{AVR}(p,S)$ is an open set because it is the intersection of open sets. The abstract Voronoi diagram, $\mathrm{AVD}(S)$ 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 $\{(\mathrm{AJ}(p,q), \mathrm{AD}(p,q)) \mid p,q \in S, p \neq q\}$ is ***admissible*** if it satisfies the following properties:

(A1) For all distinct $p,q \in S$, $\mathrm{AJ}(p,q) = \mathrm{AJ}(q,p)$.

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

(A3) There exists a special point in the plane, which we call p_∞, such that, for all distinct $p,q \in S$, the curve $\mathrm{AJ}(p,q)$ passes through p_∞.[2]

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

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

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 [20] but, as they show in their Theorem 15, they are equivalent. In this regard, our definition is closer to the one given in [19]. 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 $\mathrm{AVD}(S)$ is a plane graph [20, Theorem 10]. This brings a natural concept of ***abstract Voronoi edge*** and ***abstract Voronoi vertex*** as those being vertices (of degree ≥ 3) and edges in the plane graph $\mathrm{AVD}(S)$.

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 diagram, since it is 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 $\mathrm{AVD}(S')$, where $|S'| \leq 4$. Whether $\mathrm{AVD}(S')$ 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 [20], but they discuss the general case.)

THEOREM 4.1. (KLEIN, MEHLHORN, MEISER [21]) *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_∞, 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 [22] or Colin de Verdière [7] 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 $d_G(x,v_s)$.

[2] Usually the axiom tells that the stereographic projection to the sphere of the curve $\mathrm{AJ}(p,q)$ 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_∞.

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

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

See Figure 1 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 $s \in \mathrm{GVR}_G(s,S)$ and each vertex x of $V(G)$ belongs to precisely one graphic Voronoi cell $\mathrm{GVR}_G(s,S)$ over all $s \in S$.

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

$$GVD_G(S) \ = \ \{\mathrm{GVR}_G(s,S) \mid s \in S\}.$$

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

$$\mathrm{GD}_G(s,t) \ = \ \mathrm{GVR}_G(s,\{s,t\}),$$

or equivalently, as

$$\{x \in V(G) \mid d_G(x,s) + w_s \leq d_G(x,t) + w_t\}.$$

The following properties are standard.

LEMMA 5.1. *Let S be a generic, independent set of sites. Then for each $s \in S$ the following holds:*

- *For each x in $\mathrm{GVR}_G(s,S)$, the shortest path from x to s is contained in $\mathrm{GVR}_G(s,S)$.*
- *$\mathrm{GVR}_G(s,S)$ induces a connected subgraph of G.*
- *$\mathrm{GVR}_G(s,S) = \bigcap_{t \in S\backslash\{s\}} \mathrm{GD}_G(s,t)$.*

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 $E_G(s,t)$ to be

$$\{xy \in E(G) \mid x \in \mathrm{GD}_G(s,t), \ y \in \mathrm{GD}_G(t,s)\}.$$

Thus, we are taking the edges that have each endpoint in different graphic Voronoi regions of $GVD_G(\{s,t\})$. We denote by $E^*_G(s,t)$ their dual edges. Standard properties of duality lead to the following.

LEMMA 5.2. *Let $\{s,t\}$ be a generic and independent set of sites. Then the edges of $E^*_G(s,t)$ define a cycle γ in G^*. Moreover, if $s \in V_{\mathrm{int}}(\gamma,G)$, then $V_{\mathrm{int}}(\gamma,G) = \mathrm{GD}_G(s,t)$ and $V_{\mathrm{ext}}(\gamma,G) = \mathrm{GD}_G(t,s)$.*

When s and t are independent and generic, we define the ***bisector*** of s and t, denoted as $\mathrm{bis}_G(s,t)$ 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 set (interior or exterior) defined by the curve $\mathrm{bis}_G(s,t)$ that contains s. We then have

$$D_G(s,t) \ = \ \left(\bigcup_{v \in \mathrm{GD}_G(s,t)} \overline{v^*} \right)^{\circ}.$$

Note that the pair $(\mathrm{bis}_G(s,t), D_G(s,t))$ 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 a common "infinity point" p_∞. However, we can use it when all the sites are in the outer face of G. We next show this.

LEMMA 5.3. *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 $\{(\mathrm{bis}_G(s,t), D_G(s,t)) \mid s,t \in S, s \neq t\}$ is admissible.*

In teh previous lemma, the vertex v_∞ of G^*, dual to the outer face of G, plays the role of p_∞. From now on, whenever we talk about the abstract Voronoi diagram we refer to the abstract Voronoi diagram defined by the system of bisectors $\{(\mathrm{bis}_G(s,t), D_G(s,t)) \mid s,t \in S, s \neq t\}$. 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. We also note a property on the complexity of Voronoi diagrams of three sites.

LEMMA 5.4. *Let S be a generic, independent set of sites. Then, for each $s \in S$, we have $\mathrm{GVR}_G(s,S) = V(G) \cap \mathrm{AVR}(s,S)$. Moreover, the abstract Voronoi diagram of any 3 sites in the outer face of G has at most one vertex, besides v_∞.*

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 5.5. *Consider any two sites $s = (v_s, w_s)$ and $t = (v_t, w_t)$ in the outer face of G. Consider the*

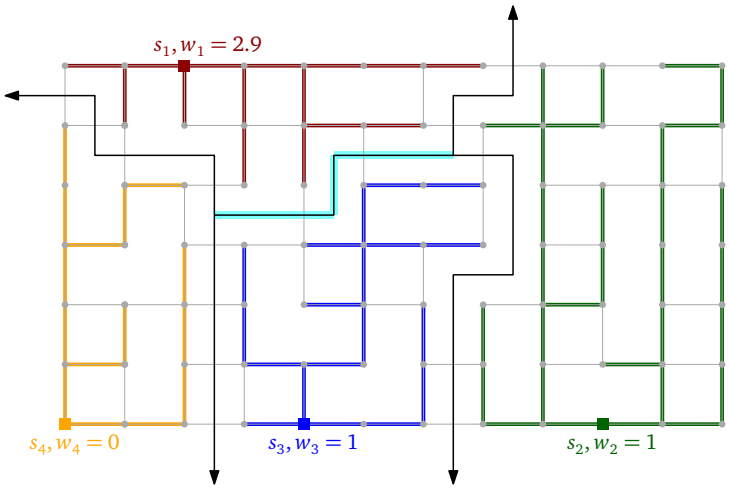

Figure 1: A graphic Voronoi diagram for four sites; the graph is unweighted. An abstract Voronoi edge is marked with thicker pen.

family of bisectors $\mathrm{bis}_G((v_s, w_s), (v_t, w_t))$ *as a function of the weights* w_s *and* w_t. *There are at most* $O(n)$ *different bisectors. We can compute and store all the bisectors in* $O(n^2 \operatorname{polylog} n)$ *time such that, given two values* w_s *and* w_t, *the corresponding representation of* $\mathrm{bis}_G((v_s, w_s), (v_t, w_t))$ *is accessed in* $O(\operatorname{polylog} n)$ *time.*

Proof. [Proof sketch] From the definition it is clear that $\mathrm{bis}_G((v_s, w_s), (v_t, w_t))$ is

$$\begin{cases} \mathrm{bis}_G((v_s, 0), (v_t, w_t - w_s)) & \text{if } w_t \geq w_s, \\ \mathrm{bis}_G((v_s, w_s - w_t), (v_t, 0)) & \text{if } w_t < w_s. \end{cases}$$

Thus, it is enough to consider the bisectors $\mathrm{bis}_G((v_s, 0), (v_t, w))$ and $\mathrm{bis}_G((v_s, w), (v_t, 0))$ parameterized by $w \in \mathbb{R}_{\geq 0}$. Each bisector $\mathrm{bis}((v_s, 0), (v_t, w))$ is a cycle in the dual graph G^* and the cycles are nested. It follows that there are at most $O(n)$ different bisectors.

Computing the distances from v_s and v_t to all vertices of G For each vertex $x \in V(G)$, we can then compute the threshold value η_x such that x is in $\mathrm{GD}_G(s, t)$ when $w < \eta_x$ and in $\mathrm{GD}_G(t, s)$ when $w > \eta_x$. Sorting the values $\{\eta_x \mid x \in V(G)\}$ and storing a representative bisector between any two consecutive values, a binary search provides the answer.

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 $\mathrm{bis}(s, t)$ between dual vertices a and b in some prescribed order, like for example the clockwise order of $\mathrm{bis}(s, t)$.

LEMMA 5.6. *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* $O(b^3 n^2)$. *For any generic, independent set* S *of* 4 *sites placed on the outer face of* G, *the abstract Voronoi diagram* $\mathrm{AVD}(S)$ *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. [Proof sketch] Let X be the set of vertices on the outer face of G. We use Lemma 5.5 to compute and store all the possible bisectors. For each bisector we store information to recognize the circular order of vertices along the bisector. We also store information to decide the circular order of vertices along the outer face of G. Finally, we compute the Voronoi diagrams for each possible subset of combinatorially distinct 3 sites, taking into account the weights. There are $O(b^3 n^2)$ such different Voronoi diagrams, and each can be encoded using $O(1)$ space with pointers to the bisectors.

The next step is to show that, with the stored information we can compute the Voronoi diagram of any 4 sites. For this one distinguishes the case of whether the section of the Voronoi diagram in the bounded faces is connected or disconnected. See Figure 2. In the former, we just join an isolated region with the diagram of 3 sites. In the latter we have to decide between the 2 possible topologies of the abstract Voronoi diagram.

THEOREM 5.1. *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* $O(b^3 n^2)$. *For any generic and independent set* S *of* m *sites placed on the outer face*

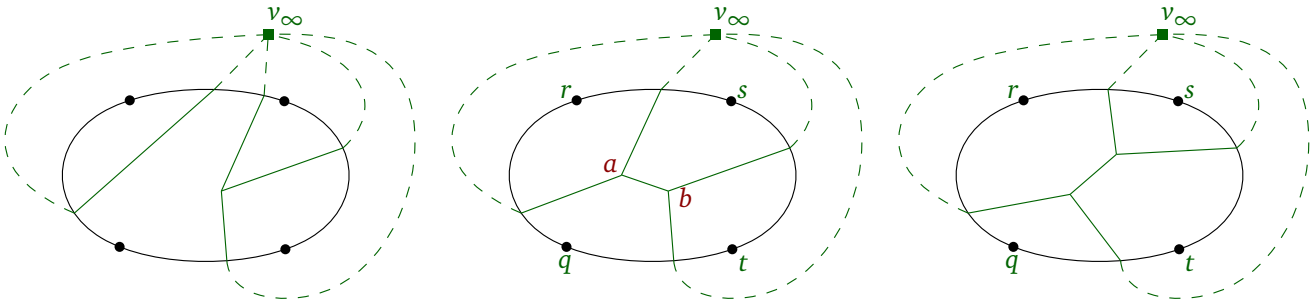

Figure 2: Left: an abstract Voronoi region is bounded by a single bisector. Center and right: possible configurations of the abstract Voronoi diagram of 4 sites, when it is connected.

of G, the abstract Voronoi diagram $\mathrm{AVD}(S)$ *can be computed in $O(m \operatorname{polylog} n)$ 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 5.6. 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 5.4, any three sites have a vertex, besides the one at p_∞ (or v_∞). According to Theorem 4.1, 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 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 5.6, the result follows.

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 $\lambda(x_0x) > 0$ for all $x \in X$. We want to find the distance to the furthest point of G from x_0 in G,

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

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, obtained by a combination of [5, 15] and the version of Dijkstra in [8], helps us detecting this.

LEMMA 6.1. *We can preprocess G in $O(n \operatorname{polylog} n)$ time to answer in $O(b \operatorname{polylog} b)$ 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.*

THEOREM 6.1. *Assume that G is a planar graph with n vertices, $x_0 \in V(G)$, and x_0 has degree b in G. After $O(b^3n^2)$ preprocessing time, we can handle the following queries in $O(b \operatorname{polylog} n)$ expected time: given the weights of the edges incident to x_0 at query time, return* $\operatorname{diam}(x_0, G)$ *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 5.1. We also compute all the bisectors of H. For each vertex $x \in X$ and all the bisectors of the type $\operatorname{bis}(x, \cdot)$ we preprocess H as explained in Lemma 3.1. 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 6.1 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 5.1 tells that we can compute the abstract Voronoi diagram in $O(|X'| \operatorname{polylog} n) = O(b \operatorname{polylog} n)$ expected time. Now, for each site $x \in X'$, we walk along the boundary of the abstract Voronoi region $\mathrm{AVR}(x, X')$ and use the data structure of Lemma 3.1 to collect how many vertices are in $\mathrm{AVR}(x, X')$, how much is the sum of their distances to x', and which one has the largest distance from x'.

For each $\mathrm{AVR}(x, X')$ we spend $O(\mathrm{polylog}\, n)$ times the complexity of the description of $\mathrm{AVR}(x, X')$. Over all X', this takes $O(|X'|\,\mathrm{polylog}\, n) = O(b\,\mathrm{polylog}\, n)$ time. From this information we can extract $\mathrm{diam}(x_0, G)$ and $\Sigma(x_0, G)$ trivially.

6.2 Working over all pairs

THEOREM 6.2. *Let G be a planar graph with n vertices and abstract, positive edge lengths. In expected time $O(n^{11/6}\,\mathrm{polylog}\, n)$ 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.*

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 $\mathcal{P} = \{P_1, \ldots, P_k\}$ of G with few holes, for a parameter r to be specified below. This takes $O(n)$ time (Theorem 2.1). As mentioned earlier, we assume that each piece has a unique hole. We build the dense distance graph $\mathrm{DDG}(\mathcal{R}, G)$ and preprocess it in $O(n \log r)$ time, as mentioned in Theorem 2.1. Then we compute the distances between all vertices of G and all boundary vertices of $\mathcal{P}$. This takes $O(n) \cdot O(nr^{-1/2} \log^2 r) = O(n^2 r^{-1/2}\,\mathrm{polylog}(n))$ time.

For each piece $P_i \in \mathcal{P}$ 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 6.1 for the vertex v_0. This takes $O((\sqrt{r})^3 r^2\,\mathrm{polylog}\, r) = O(r^{7/2}\,\mathrm{polylog}\, r)$ time per piece, for a total of $O((n/r) \cdot r^{7/2})\,\mathrm{polylog}\, r) = O(nr^{5/2}\,\mathrm{polylog}\, r)$ time over all pieces. Now we iterate over all vertices x of G to compute $\mathrm{diam}(x, P_i^+)$ 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, $d_G(x, y) = d_{P_i^+}(v_0, y)$ for all vertices $y \in P_i$. It follows that $\mathrm{diam}(x, P_i) = \mathrm{diam}(v_0, P_i^+)$ and $\Sigma(x, P_i) = \Sigma(v_0, P_i^+)$. Note that the data structure of Theorem 6.1 allows us to compute these values in $O(\sqrt{r}\,\mathrm{polylog}\, r)$ expected time because each piece P_i has $O(\sqrt{r})$ boundary vertices.

Iterating over all vertices x of G and all pieces $P_i \in \mathcal{P}$ we compute the desired values $\mathrm{diam}(x, P_i)$ and $\Sigma(x, P_i)$ for all pairs $(x, P_i) \in V(G) \times \mathcal{P}$ such that x lies in the exterior of P_i. All the pairwise distances within a piece can be computed explicitly in $O(r^{3/2}\,\mathrm{polylog}\, r)$ time (Lemma 2.1).

Making a similar computation where we exchange the role of interior and exterior, it is trivial to obtain $\mathrm{diam}(x, G)$ 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 $(x, P_{i(y)})$.

The total expected time we spend is

$$O(n^2 r^{-1/2}\,\mathrm{polylog}(n)) + O(nr^{5/2}) + O(nr^{1/2}\,\mathrm{polylog}\, r) + n \cdot O(r^{1/2}\,\mathrm{polylog}\, r).$$

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

COROLLARY 6.1. *Let G be a planar graph with n vertices and abstract, positive edge lengths. In expected time $O(n^{11/6}\,\mathrm{polylog}\, n)$ we can compute the diameter and the sum of the pairwise distances in G.*

7 Discussion

In this version we decided to assume that each piece has a unique hole, that is, there is a facial walk of the piece that covers all its boundary vertices. The general case has some additional technicalities, but it just hides the main ideas.

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 the 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 [18, 20]. 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. We will also investigate the extension of our approach to directed graphs and to graphs embedded on surfaces.

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

[1] A. Abboud, V. Vassilevska Williams, and J. Wang. Approximation and fixed parameter subquadratic algorithms for radius and diameter in sparse graphs. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '16, pages 377–391. SIAM, 2016.

[2] C. Bohler, R. Klein, and C. Liu. Forest-like abstract voronoi diagrams in linear time. In *Proceedings of the 26th Canadian Conference on Computational Geometry, CCCG 2014, Halifax, Nova Scotia, Canada, 2014*, 2014.

[3] G. Borradaile, P. Sankowski, and C. Wulff-Nilsen. Min *st*-cut oracle for planar graphs with near-linear preprocessing time. *ACM Transactions on Algorithms*, 11(3):16:1–16:29, 2015.

[4] S. Cabello. Many distances in planar graphs. *Algorithmica*, 62(1-2):361–381, 2012.

[5] S. Cabello, E. W. Chambers, and J. Erickson. Multiple-source shortest paths in embedded graphs. *SIAM J. Comput.*, 42(4):1542–1571, 2013.

[6] S. Cabello and C. Knauer. Algorithms for graphs of bounded treewidth via orthogonal range searching. *Comput. Geom.*, 42(9):815–824, 2009.

[7] É. Colin de Verdière. Shortest cut graph of a surface with prescribed vertex set. In *Algorithms - ESA 2010, 18th Annual European Symposium, Liverpool, UK, September 6-8, 2010. Proceedings, Part II*, volume 6347 of *Lecture Notes in Computer Science*, pages 100–111. Springer, 2010.

[8] J. Fakcharoenphol and S. Rao. Planar graphs, negative weight edges, shortest paths, and near linear time. *J. Comput. Syst. Sci.*, 72(5):868–889, 2006.

[9] G. N. Frederickson. Fast algorithms for shortest paths in planar graphs, with applications. *SIAM J. Comput.*, 16:1004–1022, 1987.

[10] G. N. Frederickson. Planar graph decomposition and all pairs shortest paths. *J. ACM*, 38(1):162–204, 1991.

[11] O. Goldreich and D. Ron. Approximating average parameters of graphs. *Random Struct. Algorithms*, 32(4):473–493, 2008.

[12] M. T. Goodrich. Planar separators and parallel polygon triangulation. *J. Comput. Syst. Sci.*, 51(3):374–389, 1995.

[13] P. Indyk. Sublinear time algorithms for metric space problems. In *Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing*, pages 428–434. ACM, 1999.

[14] K. Kawarabayashi, P. N. Klein, and C. Sommer. Linear-space approximate distance oracles for planar, bounded-genus and minor-free graphs. In *38th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 135–146, 2011.

[15] P. N. Klein. Multiple-source shortest paths in planar graphs. In *SODA '05: Proc. 16th Symp. Discrete algorithms*, pages 146–155, 2005.

[16] P. N. Klein, S. Mozes, and C. Sommer. Structured recursive separator decompositions for planar graphs in linear time. In *Symposium on Theory of Computing Conference, STOC'13*, pages 505–514, 2013. See `http://arxiv.org/abs/1208.2223` for the full version.

[17] P. N. Klein and S. Subramanian. A fully dynamic approximation scheme for shortest paths in planar graphs. *Algorithmica*, 22(3):235–249, 1998.

[18] R. Klein. *Concrete and Abstract Voronoi Diagrams*, volume 400 of *Lecture Notes in Computer Science*. Springer, 1989.

[19] R. Klein. Abstract voronoi diagrams. In M.-Y. Kao, editor, *Encyclopedia of Algorithms*, pages 1–5. Springer Berlin Heidelberg, 2014.

[20] R. Klein, E. Langetepe, and Z. Nilforoushan. Abstract voronoi diagrams revisited. *Comput. Geom.*, 42(9):885–902, 2009.

[21] R. Klein, K. Mehlhorn, and S. Meiser. Randomized incremental construction of abstract voronoi diagrams. *Comput. Geom.*, 3:157–184, 1993.

[22] D. Marx and M. Pilipczuk. Optimal parameterized algorithms for planar facility location problems using voronoi diagrams. *CoRR*, abs/1504.05476, 2015.

[23] S. Mozes, Y. Nussbaum, and O. Weimann. Faster shortest paths in dense distance graphs, with applications. *CoRR*, abs/1404.0977, 2014.

[24] S. Mozes and C. Sommer. Exact distance oracles for planar graphs. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012*, pages 209–222. SIAM, 2012.

[25] G. F. I. Y. Nussbaum, P. Sankowski, and C. Wulff-Nilsen. Improved algorithms for min cut and max flow in undirected planar graphs. In *Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC*, pages 313–322, 2011.

[26] J. K. Park and C. A. Phillips. Finding minimum-quotient cuts in planar graphs. In *Proceedings of the Twenty-fifth Annual ACM Symposium on Theory of Computing*, STOC '93, pages 766–775, 1993.

[27] V. Patel. Determining edge expansion and other connectivity measures of graphs of bounded genus. *SIAM J. Comput.*, 42(3):1113–1131, 2013.

[28] L. Roditty and V. Vassilevska Williams. Fast approximation algorithms for the diameter and radius of sparse graphs. In *Symposium on Theory of Computing Conference, STOC'13*, pages 515–524. ACM, 2013.

[29] M. Thorup. Compact oracles for reachability and approximate distances in planar digraphs. *J. ACM*, 51(6):993–1024, 2004.

[30] O. Weimann and R. Yuster. Approximating the diameter of planar graphs in near linear time. *ACM Transactions on Algorithms*, 12(1):12, 2016. Preliminary

version in SODA 2012.
[31] C. Wulff-Nilsen. Wiener index, diameter, and stretch factor of a weighted planar graph in subquadratic time. Technical Report 08-16, Department of Computer Science, University of Copenhagen, 2008. Preliminary version in EurCG 2009.
[32] C. Wulff-Nilsen. Wiener index, diameter, and stretch factor of a weighted planar digraph in subquadratic time, 2010. Paper M in the PhD thesis of C. Wulff-Nilsen, availalbe at http://www.diku.dk/forskning/phd-studiet/phd/ThesisChristian.pdf.