Voronoi Diagrams on Planar Graphs, and Computing the Diameter in Deterministic $\tilde{O}(n^{5/3})$ Time

Paweł Gawrychowski* Haim Kaplan† Shay Mozes‡ Micha Sharir†
Oren Weimann*

Abstract

We present an explicit and efficient construction of additively weighted Voronoi diagrams on planar graphs. Let G be a planar graph with n vertices and b sites that lie on a constant number of faces. We show how to preprocess G in $\tilde{O}(nb^2)$ time[1] so that one can compute any additively weighted Voronoi diagram for these sites in $\tilde{O}(b)$ time.

We use this construction to compute the diameter of a directed planar graph with real arc lengths in $\tilde{O}(n^{5/3})$ time. This improves the recent breakthrough result of Cabello (SODA'17), both by improving the running time (from $\tilde{O}(n^{11/6})$), and by providing a deterministic algorithm. It is in fact the first truly subquadratic *deterministic* algorithm for this problem. Our use of Voronoi diagrams to compute the diameter follows that of Cabello, but he used abstract Voronoi diagrams, which makes his diameter algorithm more involved, more expensive, and randomized.

As in Cabello's work, our algorithm can also compute the Wiener index of a planar graph (i.e., the sum of all pairwise distances) within the same bound. Our construction of Voronoi diagrams for planar graphs is of independent interest. It has already been used to obtain fast exact distance oracles for planar graphs [Cohen-Addad et al., FOCS'17].

1 Introduction

Computing the diameter of a (directed, weighted) graph (the largest distance between a pair of vertices) is a fundamental problem in graph algorithms, with numerous research papers studying its complexity.

For general graphs, the current fastest way to compute the diameter is by computing all pairs of shortest paths (APSP) between its vertices. Computing APSP can be done in cubic $O(n^3)$ time, using the classical Floyd-Warshall algorithm [30, 56]. Following a long line of improvements by polylogarithmic factors [17, 18, 25, 32, 34–36, 54, 55, 62], the current fastest APSP algorithm is the $O(n^3/2^{\Omega(\log n)^{1/2}})$-time algorithm by Williams [57]. However, no truly subcubic, i.e. $O(n^{3-\varepsilon})$-time algorithm, for any fixed $\varepsilon > 0$, is known for either APSP or the problem of computing the diameter, referred to as DIAMETER. We also do not know if DIAMETER is as hard as APSP (i.e., if a truly subcubic algorithm for DIAMETER implies a truly subcubic algorithm for APSP). This is in contrast to many other related problems that were recently shown to be as hard as APSP [1–4, 6, 9, 10, 13, 51, 58, 59] (including the problem of computing the radius of a graph [2]).

For sparse graphs (with $\tilde{O}(n)$ edges), the problem is better understood. We can solve APSP (and thus DIAMETER) in $\tilde{O}(n^2)$ time by running a single-source shortest path algorithm from every vertex. There is clearly no truly subquadratic $O(n^{2-\varepsilon})$ algorithm for APSP (as the output size is $\Omega(n^2)$). However, Chan [16] showed that, for undirected unweighted sparse graphs, APSP can be represented and computed in $O(n^2/\log n)$ time. Interestingly, assuming the strong exponential time hypothesis (SETH) [41], there is also no truly subquadratic algorithm for DIAMETER. In fact, even distinguishing between diameter 2 and 3 requires quadratic time assuming SETH [50]. This holds even for undirected, unweighted graphs with treewidth $O(\log n)$. For graphs of bounded treewidth, the diameter can be computed in near-linear time [5] (see also [27, 40] for algorithms with time bounds that depend on the value of the diameter itself). Near-linear time algorithms were developed for many other restricted graph families, see e.g. [7, 11, 19–22, 26, 29, 39, 49].

For planar graphs, we know that the diameter can in fact be computed faster than APSP. This was illustrated by Wulff-Nilsen who gave an algorithm for computing the diameter of unweighted, undirected planar graphs in $O(n^2 \log\log n/\log n)$ time [60] and of weighted directed planar graphs in $O(n^2(\log\log n)^4/\log n)$ time [61]. The question of whether a truly subquadratic ($O(n^{2-\varepsilon})$-

*University of Haifa, Department of Computer Science, gawry@mimuw.edu.pl, oren@cs.haifa.ac.il. Partially supported by the Israel Science Foundation (grant No. 794/13 and 592/17).

†Tel Aviv University, Blavatnik School of Computer Science, {haimk,michas}@post.tau.ac.il. Partially supported by Len Blavatnik and the Blavatnik Research Fund in Computer Science at Tel Aviv University, by the Israeli Centers of Research Excellence (I-CORE) program (center No. 4/11), by the Israel Science Foundation (grant No. 1841-14), by GIF (grant no. 1161 and 1367), and by the Hermann Minkowski-MINERVA Center for Geometry at Tel Aviv University.

‡Interdisciplinary Center Herzliya, Efi Arazi School of Computer Science, smozes@idc.ac.il. Partially supported by the Israel Science Foundation (grant No. 794/13 and 592/17).

[1]The $\tilde{O}$ notation hides polylogarithmic factors.

time) algorithm exists (even for undirected, unweighted planar graphs) was a major open problem in planar graph algorithms. In SODA'17, a breakthrough result of Cabello [14] showed that the diameter of weighted directed planar graphs can be computed in truly subquadratic $\tilde{O}(n^{11/6})$ expected time. Whether the exponent 11/6 could be substantially reduced became a main open problem.

Voronoi diagrams on planar graphs. The breakthrough in Cabello's work is his novel use of Voronoi diagrams in planar graphs.[2] An *r-division* [31] of a planar graph G is a decomposition of G into $\Theta(n/r)$ pieces, each of them with $O(r)$ vertices and $O(\sqrt{r})$ boundary vertices (vertices shared with other pieces). There is an $O(n)$-time (deterministic) algorithm that computes an r–division with the additional property that the boundary vertices in each piece lie on a constant number of faces of the piece (called *holes*) [28, 44]. Unsurprisingly, it turns out that the difficult case for computing the diameter is when the farthest pair of vertices lie in different pieces.

Consider some source vertex v_0 outside of some piece P. For every boundary vertex u of P, let $\omega(u)$ denote the v_0-to-u distance in G. The *additively weighted Voronoi diagram* of P with respect to $\omega(\cdot)$ is a partition of the vertices of P into pairwise disjoint sets (Voronoi cells), each associated with a unique boundary vertex (site) u. The vertices in the cell $\mathrm{Vor}(u)$ of u are all the vertices v of P such that u minimizes the quantity $\omega(u)$ plus the length of the u-to-v shortest path inside P (i.e., u is the last boundary vertex of P on the v_0-to-v shortest path in G). The *boundary* of a cell $\mathrm{Vor}(u)$ consists of all edges of P that have exactly one endpoint in $\mathrm{Vor}(u)$. For example, in a Voronoi diagram of just two sites u and v, the boundary of the cell $\mathrm{Vor}(u)$ is a uv-cut and is therefore a cycle in the dual graph. We call this cycle the *uv-bisector*. The *complexity* of a Voronoi diagram is defined as the number of faces of P that contain vertices belonging to more than two Voronoi cells.

For every v_0, computing the farthest vertex from v_0 in P thus boils down to computing, for each site u, the farthest vertex from u in $\mathrm{Vor}(u)$, and returning the maximum of these quantities. The main challenges with this approach are: (1) to efficiently compute the Voronoi diagram of a piece P (that is, to identify the partition of the vertices of P into Voronoi cells), with respect to boundary weights equal to the v_0-to-boundary distances, and (2) to find the maximum site-to-vertex distance in each cell. To appreciate these issues, note that we need to perform tasks (1) and (2) much faster than $O(|P|) = O(r)$ time! Otherwise, since there are n possible sources v_0, and $\Theta(n/r)$ pieces, the overall running time would be $\Omega(n^2)$. Thus, even task (2), which is trivial to implement by simply inspecting all vertices of P, requires a much more sophisticated approach.

Voronoi diagrams of geometric objects of many kinds, and with respect to different metrics, have been extensively studied (see, e.g., [8]). One of the basic approaches for computing Voronoi diagrams is divide and conquer. Using this approach, we split the objects (sites) into two groups, compute the Voronoi diagram of each group recursively, and then somehow merge the diagrams. Using this approach, Shamos amd Hoey [53] gave the first $O(n \log n)$ deterministic algorithm for computing the Voronoi diagram of n points in the plane with respect to the Euclidean metric. Since then other approaches, such as randomized incremental construction, have also been developed.

Cabello's approach. For every source v_0 and every piece P, Cabello uses the "heavy hammer" of *abstract Voronoi diagrams* in order to compute the Voronoi diagram of P with respect to the v_0-to-boundary distances as additive weights. Klein [45] introduced the abstract Voronoi diagram framework, trying to abstract the properties of objects and metric that make the computation of Voronoi diagrams efficient. Cabello uses (as a black box) the randomized construction of abstract Voronoi diagrams by Klein, Mehlhorn and Meiser [47] (see also [46]). This construction, when applied to our special setup, is highly efficient, requiring only $\tilde{O}(\sqrt{r})$ time. However, in order to use it, certain requirements must be met.

First, it requires knowing in advance the Voronoi diagrams induced by every subset of four sites (boundary vertices). Cabello addresses this by first showing that the planar case only requires the Voronoi diagrams for any triple of sites to be known in advance. The Voronoi diagram of each triple is composed of segments of bisectors. The situation is further complicated because each pair of sites can have many bisectors, depending on (the difference between) the weights of the sites. Cabello shows that for each pair of sites, there are only $O(r)$ different bisectors over all possible weight differences for this pair. His algorithm computes each of these bisectors separately in $O(r)$ time, so computing all the bisectors for a single pair of sites takes $O(r^2)$ time. This yields a procedure that precomputes all 3-site weighted Voronoi diagrams from the bisectors of pairs of sites, in $\tilde{O}(r^{7/2})$ time.

The second requirement is that all the sites must

[2] Using Voronoi diagrams for algorithms in planar graphs was done before (see e.g. [24,48]), but never for a polynomially solvable problem (where we care about polynomial factors).

lie on the boundary of a single hole. However, from the properties of r–divisions, the boundary vertices of P may lie on several (albeit a constant number of) holes. Overcoming this is a significant technical difficulty in Cabello's paper. He achieves this by allowing the bisectors to venture through the subgraphs enclosed by the holes (i.e., through other pieces). Informally, Cabello's algorithm "fills up" the holes in order to apply the mechanism of abstract Voronoi diagrams.

Our result. We present a deterministic $\tilde{O}(n^{5/3})$-time algorithm for computing the diameter of a directed planar graph with no negative-length cycles. This improves Cabello's bound by a polynomial factor, and is the first deterministic truly subquadratic algorithm for this problem.

Our algorithm follows the general high-level approach of Cabello, but differs in the way it is implemented. Our main technical contribution is an efficient and deterministic construction of additively weighted Voronoi diagrams on planar graphs (under the assumption that the sites lie on the boundaries of a constant number of faces, which holds in the context considered here). In contrast to Cabello, we only need to precompute the bisectors between *pairs* of sites, and then use a technique, developed in this paper, for intersecting three bisectors in $\tilde{O}(1)$ time. Moreover, we compute the bisectors faster, so that all possible bisectors between a given pair of sites can be computed in $\tilde{O}(r)$ time, compared to $O(r^2)$ time in [14]. Another difference is that we use a deterministic divide-and-conquer approach that exploits the planar structure, rather than the randomized incremental construction of abstract Voronoi diagrams used by Cabello. Formally, we prove the following result.

THEOREM 1.1. *Let P be a directed planar graph with real arc lengths, r vertices, and no negative length cycles. Let S be a set of b sites that lie on the boundaries of a constant number of faces (holes) of P. One can preprocess P in $\tilde{O}(r \cdot b^2)$ time, so that, given any subset $S' \subseteq S$ of the sites, and a weight $\omega(u)$ for each $u \in S'$, one can construct a representation of the additively weighted Voronoi diagram of S' with respect to the weights $\omega(\cdot)$, in $\tilde{O}(|S'|)$ time. With this representation of the Voronoi diagram we can, for any site $u \in S'$, (i) report the boundary of the Voronoi cell of u in $\tilde{O}(1)$ time per edge, and (ii) query the maximum distance from u to a vertex in the Voronoi cell of u in $\tilde{O}(|\partial\mathrm{Vor}(u)|)$ time, where $|\partial\mathrm{Vor}(u)|$ denotes the complexity of the Voronoi cell of u.*

The above $\tilde{O}(|S'|)$ construction time is significant because $|S'| \leq b = O(\sqrt{r}) \ll |P| = O(r)$. This fast construction comes with the price of the cost of the preprocessing stage. In other words, a one-time construction of the diagram is less efficient than a brute force $\tilde{O}(br)$ algorithm (that simply computes a shortest path tree from each site), because we need to account for the preprocessing cost too. Nevertheless, the preprocessing is *independent* of the weights of the sites. Hence, the overall approach, which consists of a one-time preprocessing stage (per piece), followed by many calls to the diagram construction procedure, makes the overall algorithm efficient. This general approach is borrowed from Cabello, except that we achieve a simpler, more efficient and deterministic algorithm for DIAMETER.

As in Cabello's work, our algorithm can also compute the Wiener index (sum of all pairwise distances) of a planar graph within the same performance bounds. Our detailed study of Voronoi diagrams for planar graphs, and the resulting algorithm of Theorem 1.1 are of independent interest, and we believe it will find additional uses. For example, in a very recent work, Cohen-Addad, Dahlgaard, and Wulff-Nilsen [23] show that the Voronoi diagram approach can be used to construct exact *distance oracles* for planar graphs with subquadratic space and polylogarithmic query time. They show an oracle of size $\tilde{O}(n^{5/3})$, query time $O(\log n)$, and preprocessing time $O(n^2)$. They also improve the state-of-the-art tradeoff between space and query time (see also [33]). Using Theorem 1.1 yields an optimal preprocessing time for the oracles in [23], matching their space requirements up to polylogarithmic factors.

1.1 The diameter algorithm.

Let G be a directed planar graph with non-negative arc lengths.[3] The algorithm computes an r–division of G with few holes per piece. For a vertex v that is not a boundary vertex, let P_v be the unique piece of the r–division that contains v and let ∂P_v denote the boundary vertices of P_v. The diameter of G is the length of a shortest u-to-v path for some pair of vertices u, v. There are three cases: (i) at least one of u, v is a boundary vertex, (ii) none of u, v is a boundary vertex and $P_u = P_v$, and (iii) none of u, v is a boundary vertex and $P_u \neq P_v$.

To take care of case (i), we reverse all arcs in G and compute single-source shortest paths trees rooted at each boundary vertex. This takes $O(n \cdot n/\sqrt{r})$ time using the linear time algorithm of Henzinger et al. [38],[4] and computes the distance from every vertex in G to every boundary vertex.

To take care of case (ii), the algorithm next com-

[3] Negative arc length can be handled so long as there are no negative-length cycles. This is done using the standard technique of feasible price functions, which make all the weights non-negative. See, e.g., [14, 28].

[4] In fact, for our purposes it suffices to use Dijkstra's algorithm that takes $O(n \log n)$ time.

putes the distances in G from v to all vertices of P_v. This is done by a single-source shortest-path computation in P_v, with the distance labels of vertices of ∂P_v initialized to their distance from v in the entire graph G (these distances were already computed in case (i)). Note that in doing so we take care of the possibility that the shortest path from u to v traverses other pieces of G, even though u and v lie in the same piece. This takes $O(r)$ time per vertex, for a total of $O(nr)$ time.

It remains to take care of case (iii), that is, to compute the maximum distance between vertices that are not in the same piece. To this end we first invoke Theorem 1.1, on every piece P in the r-division, preprocessing P in $\tilde{O}(r(\sqrt{r})^2) = \tilde{O}(r^2)$ time. Then, for every non-boundary source vertex $v_0 \in G$, and every piece P in the r-division (except for the piece of v_0) we compute, in $\tilde{O}(\sqrt{r})$ time, the additively weighted Voronoi diagram of P, where the weight $\omega(v)$, for each $v \in \partial P$, is the v_0-to-v distance in G (these distances have been computed in case (i)). We then use the efficient maximum query in item (ii) of Theorem 1.1 to compute the farthest vertex from v_0 in each cell of the Voronoi diagram. Each query takes time proportional to the complexity of the Voronoi cell, which together sums up to the complexity of the Voronoi diagram, which is $\tilde{O}(\sqrt{r})$. Thus, we find the farthest vertex from v_0 in P in $\tilde{O}(\sqrt{r})$ time. The total preprocessing time over all pieces is $\tilde{O}(\frac{n}{r} \cdot r^2) = \tilde{O}(nr)$, and the total query time over all sources v_0 and all pieces P is $\tilde{O}(n \cdot \frac{n}{r} \cdot \sqrt{r}) = \tilde{O}(n^2/\sqrt{r})$. Summing the preprocessing and query bounds, we get an overall cost of $\tilde{O}(nr + n^2/\sqrt{r})$.

Concerning the running time of the entire algorithm, we note that computing the r–division takes $O(n)$ time [44]. The total running time is thus dominated by the $\tilde{O}(nr + n^2/\sqrt{r})$ of case (iii). Setting $r = n^{2/3}$ yields the claimed $\tilde{O}(n^{5/3})$ bound.

1.2 A high-level description of the Voronoi diagram algorithm. We now outline the proof of Theorem 1.1. The description is given in the context of the diameter algorithm, so the input planar graph is called a piece P, and the faces to which the sites are incident are called holes. The set of sites is denoted by S. To avoid clutter, and without loss of generality, we assume that the subset S' of sites of the desired Voronoi diagram is the entire set S. Hence, $|S'| = |S| = b = O(\sqrt{r})$. Given a weight assignment $\omega(\cdot)$ to the sites in S, the algorithm computes (an implicit representation of) the *additively weighted Voronoi diagram* of S in P, denoted as $VD(S, \omega)$, or just $VD(S)$ for short. Recall that our goal is to construct many instances of $VD(S)$ in $\tilde{O}(b) = \tilde{O}(\sqrt{r})$ time each. What enables us to achieve this is the fact that these instances differ only in the weight assignment $\omega(\cdot)$. We exploit this by carrying out a preprocessing stage, which is weight-independent. We use the information collected in the preprocessing stage to make the construction of the weighted diagrams efficient.

The structure of $VD(S)$. Each Voronoi cell $\mathrm{Vor}(v)$ is in fact a tree rooted at v, which is a subtree of the shortest-path tree from v (within P). We also consider the dual graph P^* of P, and use the following dual representation of $VD(S)$ within P^*, which we denote as $VD^*(S)$. For each pair of distinct sites u, v, we define the *bisector* $\beta^*(u, v)$ of u and v to be the collection of all edges $(pq)^*$ of P^* that are dual to edges pq of P for which p is nearer to u than to v and q is nearer to v than to u. Each bisector is a simple cycle in P^*.[5] In general, in the presence of other sites, only part of $\beta^*(u, v)$ appears in $VD^*(S)$. The set of maximal connected segments of $\beta^*(u, v)$ is a collection of one or several *Voronoi edges* that separate between the cells $\mathrm{Vor}(u)$ and $\mathrm{Vor}(v)$.[6] Each Voronoi edge terminates at a pair of *Voronoi vertices*, where such a vertex f^* is dual to a face f of P whose vertices belong to three or more distinct Voronoi cells. To simplify matters, we triangulate each face of P (except for the holes), so all Voronoi vertices (except those dual to the holes, if any) are of degree 3. If we contract the edges of each maximal connected segment comprising a Voronoi edge into a single abstract edge connecting its endpoints, we get a planar map of size $O(b)$ (by Euler's formula). This planar map is the dual Voronoi diagram $VD^*(S)$. A useful property is that, in the presence of just three sites, the diagram VD^* has at most two vertices. In fact, even if there are more than three sites, but assuming that the sites lie on the boundaries of only three holes, the diagram has at most two "trichromatic" vertices, that is, Voronoi vertices whose dual faces have vertices in the Voronoi cells of sites from all three holes. See Sections 2 and 5.

Computing all bisectors. The heart of the preprocessing step is the computation of bisectors of every pair of sites u, v, for every possible pair of weights $\omega(u), \omega(v)$ that can be assigned to them. Note that the bisector $\beta^*(u, v)$ only depends on the *difference* $\delta = \omega(v) - \omega(u)$ between the weights of these sites, and that, due to the discrete nature of the setup, the bisector changes only at a discrete set of differences, and one can show

[5] In this high level description we treat bisectors as undirected objects. In the precise definition, given in Section 2, they are directed.

[6] Even in the case of standard additively weighted Euclidean Voronoi diagrams, two cells can have a disconnected common boundary.

that there are only $O(r)$ different bisectors for each pair (u, v). Computing a representation of the $O(r)$ possible bisectors for a fixed pair of sites (u, v) is done in $\tilde{O}(r)$ time by varying δ from $+\infty$ to $-\infty$. As we do this, the bisector "sweeps" over the piece, moving farther from u and closer to v. This is reminiscent of the multiple source shortest paths (MSSP) algorithm of [15, 43], except that in our case u and v do not necessarily lie on the same face. We represent all the possible bisectors for (u, v) using a persistent binary search tree [52] that requires $\tilde{O}(r)$ space. Summing over all $O((\sqrt{r})^2) = O(r)$ pairs of sites, the total preprocessing time is $\tilde{O}(r^2)$. See Section 3.

Computing $VD(S)$. We compute the diagram in four steps: (i) We first compute the diagram for the sites on the boundary of a single hole (the "monochromatic" case). We do this using a divide-and-conquer technique, whose main ingredient is merging two sub-diagrams into a larger diagram. We compute one diagram for each of the t holes. (ii) We then compute the diagram of the sites that lie on the boundaries of a fixed pair of holes (the "bichromatic case"), for all $\binom{t}{2} = O(1)$ pairs of holes. Each such diagram is obtained by merging the two corresponding monochromatic diagrams. (iii) We then compute all "trichromatic" diagrams, each of which involves the sites that lie on the boundaries of three specific holes. Here too we merge the three (already computed) corresponding bichromatic diagrams. (iv) Since each Voronoi edge (resp., vertex) is defined by two (resp., three) sites (even if a Voronoi vertex is defined by more than three sites, it is uniquely determined by any three of them), we now have a superset of the vertices of $VD(S)$. Each Voronoi edge of $VD(S)$, say separating the cells of sites u, v, is contained in the appropriate Voronoi edges of all trichromatic diagrams involving the holes of u and v. A simple merging step along each bisector constructs the true Voronoi edges and vertices. See section 6.

Intersecting bisectors, finding trichromatic vertices, and merging diagrams. The main technical part of the algorithm is an efficient implementation of steps (i)–(iii) above. A crucial procedure that the algorithm repeatedly uses is an efficient construction of the (at most two) Voronoi vertices of three-sites diagrams, where the cost of this step is only $\tilde{O}(1)$. In fact, we generalize this procedure so that it can compute the (at most two) trichromatic Voronoi vertices of a diagram of the form $VD(\{x, y, Z\})$, where x and y are two sites and Z is a sequence of sites on a fixed hole. This extended version also takes only $\tilde{O}(1)$ time. See Section 4.

Having such a procedure at hand, each merging of two subdiagrams $VD(S_1)$, $VD(S_2)$, into the joint diagram $VD(S_1 \cup S_2)$, is performed by tracing the S_1S_2-*bisector*, which is the collection of all Voronoi edges of the merged diagram that separate between a cell of a site in S_1 and a cell of a site in S_2. In all the scenarios where such a merge is performed, the S_1S_2-bisector is also a cycle, which we can trace segment by segment. In each step of the trace, we are on a bisector of the form $\beta^*(u_1, u_2)$, for $u_1 \in S_1$ and $u_2 \in S_2$. We then take the cell of u_1 in $VD(S_1)$, and compute the trichromatic vertices for u_1, u_2, and $S_1 \setminus \{u_1\}$. We do the same for the cell of u_2 in $VD(S_2)$, and the two steps together determine the first exit point of $\beta^*(u_1, u_2)$ from one of these two cells, and thereby obtain the terminal vertex of the portion of $\beta^*(u_1, u_2)$ that we trace, in the combined diagram. Assume that we have left the cell of u_1 and have entered the cell of u_1' in $VD(S_1)$. We then apply the same procedure to this new bisector $\beta^*(u_1', u_2)$, and keep doing so until all of the S_1S_2-bisector is traced. See Section 6.

The details concerning the identification of trichromatic vertices are rather involved, and we only give a few hints in this overview. Let x, y, z be our three sites. We keep the bisector $\beta^*(y, z)$ fixed, meaning that we keep the weights $\omega(y)$, $\omega(z)$ fixed, and vary the weight $\omega(x)$ from $+\infty$ to $-\infty$. As already reviewed, this causes the cell $\mathrm{Vor}(x)$, which is initially empty, to gradually expand, "sweeping" through P. This expansion occurs at discrete critical values of $\omega(x)$. We show that, as $\mathrm{Vor}(x)$ expands, it annexes a contiguous portion of $\beta^*(y, z)$, which keeps growing as $\omega(x)$ decreases. The terminal vertices of the annexed portion of $\beta^*(y, z)$ are the desired trichromatic vertices. By a rather involved variant of binary search through $\beta^*(y, z)$ (with the given $\omega(x)$ as a key), we find those endpoints, in $\tilde{O}(1)$ time. See Section 4.

To support this binary search we need to store, at the preprocessing step, for each vertex p of P and for each pair of sites x, y, the "time" (i.e, difference $\omega(x) - \omega(g)$) at which $\beta^*(x, g)$ sweeps through p. Note that this can be done within the $O(r^2)$ time and space that it takes to precompute the bisectors.

Putting all the above ingredients together (where the details that we skimmed through in this overview are spelled out later on), we get an overall algorithm that constructs the Voronoi diagram, within a single piece and under a given weight assignment, in $\tilde{O}(b) = \tilde{O}(\sqrt{r})$ time.

Finding the farthest vertex in each Voronoi cell. To be useful for the diameter algorithm, our representation of Voronoi diagrams must be augmented to report the farthest vertex in each Voronoi cell from the site of the cell in nearly linear time in the number of Voronoi vertices of the cell. Such a mechanism has

been developed by Cabello [14, Section 3], but only for pieces with a single hole, where a Voronoi cell is bounded by a single cycle. We extend this procedure to pieces with a constant number of holes by exploiting the structure of Voronoi diagrams, and the interaction between a shortest path tree, its cotree, and the Voronoi diagram. See Section 7.

1.3 Discussion of the relation to Cabello's work. We have already mentioned the similarities and differences of our work and Cabello's. To summarize and further clarify the relation of the two papers, we distinguish three aspects in which our construction of Voronoi diagrams differs from his. The first main difference is the faster computation of bisectors, the representation of bisectors, and the new capability to compute trichromatic vertices on the fly, which has no analogue in [14]. One could try to plug in just these components into Cabello's algorithm (i.e., still using the randomized incremental construction of abstact Voronoi diagrams) in order to obtain an $\tilde{O}(n^{5/3})$-time randomized algorithm for DIAMETER. Doing so, however, is not trivial since there are difficulties in modifying Cabello's technique for "filling-up" the holes to work with our persistent representation of the bisectors. This seems doable, but seems to require quite a bit of technical work.

The second main difference is that we develop a deterministic divide-and-conquer construction of Voronoi diagrams, while Cabello uses the randomized incremental construction for abstract Voronoi diagrams [47]. This makes the algorithm more explicit and deterministic.

The third main difference is that our construction of Voronoi diagrams works when the sites lie on multiple holes,[7] whereas Cabello's use of abstract Voronoi diagrams requires the sites to lie on a single hole. Assuming that the sites lie on a single hole would significantly simplify multiple components of our construction of Voronoi diagrams, but has its drawbacks. Most concretely, it leads to a more complicated and less elegant algorithm for diameter due to the need to "fill-up" holes. Indeed, in [14], Cabello defers the entire description of this technical issue to the full version of his paper. More generally, allowing multiple holes in the construction of Voronoi diagrams leads to a stronger interface that can be more suitable, easier to use, and perhaps even crucial for other applications of Voronoi diagrams on planar graphs beyond DIAMETER. As an anonymous reviewer pointed out, computing the diameter of a graph embedded on a surface of genus g seems to reduce to the planar case with $O(g)$ holes.

[7] We assume throughout the paper that the number of holes is constant but, in fact, the dependency of the construction time in our algorithm on the number of holes is polynomial, so we could tolerate a non-constant number of holes.

2 Preliminaries

Planar embedded graphs. We assume basic familiarity with planar embedded graphs and planar duality. We treat the graph $G = (V, E)$ as a directed object, where V is the set of vertices, and E is the set of arcs. We assume that for every arc $e = uv$ there is an antiparallel arc $rev(e) = vu$ that is embedded on the same curve in the plane as e. The lengths of an arc and its reverse need not be equal. We call u the *tail* of e and v the *head* of e. We use the term *edge* when the direction plays no role (i.e., when we wish to refer to the undirected object, not distinguishing between the two antiparallel arcs). The dual of a planar graph G is a planar embedded graph $G^* = (V^*, E^*)$, where the nodes in V^* represent faces in G, and the dual arcs in E^* stand in 1-1 correspondence with the arcs in E, in the sense that the arc e^* dual to an arc e connects the face to the left of e to the face to the right of e. We use some well known properties of planar graphs, see e.g., [42]. If T is a spanning tree of G then the edges not in T form a spanning tree T^* of the dual G^*. The tree T^* is called the *cotree* of T. If $(X, \bar{X})$ is a partition of V, and the subgraphs induced by X and by $\bar{X}$ are both connected, then the set of duals of the arcs whose tail is in X and whose head is in $\bar{X}$ forms a simple cycle in G^*.

Assumptions. By a *piece* $P = (V, E)$ we mean an embedded directed planar graph with $t = O(1)$ distinguished faces $h_1, \ldots, h_t$, to which we refer as *holes*. (In our context, the pieces are the subgraphs of the input graph G produced by an r-division.) Each arc $uv \in E$ has a *length* $\ell(uv)$ associated with it. We assume, for the diameter algorithm, that arc lengths are non-negative. This assumption can be enforced using the standard technique of price functions and reduced lengths (see, e.g., [14, 28]). We assume that all faces of G, except possibly for the holes, are triangulated. This assumption can be enforced by triangulating each non-triangular face that is not a hole by infinite length diagonal edges. Except for the dual nodes representing the holes, all other nodes of P^* therefore have degree 3.

We assume that shortest paths are unique. Our assumption is identical to the one made in [15]. More specifically, let P be a piece with a set S of boundary sites and additive weights $\omega(\cdot)$. Consider the process in which we vary the weight $\omega(u)$ of exactly one site $u \in S$. We assume that vertex-to-vertex distances in P are distinct, and that shortest paths are unique, except at a discrete set of critical values of $\omega(u)$ where there

is a unique *tense* arc (see Section 3 for the definition). This assumption can be achieved deterministically with $\tilde{O}(1)$ time overhead using a deterministic lexicographic perturbation [15, 37].

Additively weighted Voronoi diagram in a piece. We are given a piece $P = (V, E)$ with $|V| = O(r)$ (and so $|E| = O(r)$). We are also given a set $S \subseteq V$ of $b = O(\sqrt{r})$ vertices, each of which is a vertex of one of the $O(1)$ holes $h_1, \ldots, h_t$; S is a subset of the boundary vertices of the piece P, and its elements are the *sites* of the Voronoi diagram that we are going to define. Each site $v \in S$ has a real *weight* $\omega(v)$ associated with it. We think of ω as a weight function on S. For every pair of vertices u and v, we denote by $\pi(u, v)$ the shortest path from u to v in P. We denote the length of $\pi(u, v)$ by $d(u, v)$. The (additively weighted) *distance* between a site $u \in S$ and a vertex $v \in V$, is $\omega(u) + d(u, v)$.

The *additively weighted Voronoi diagram* of (S, ω) within P, denoted by $VD(S, \omega)$ (we will often drop ω from this notation, although the diagram does depend on ω), is a partition of V into pairwise disjoint sets, one set, denoted by $\mathrm{Vor}_{S,\omega}(u)$, for each site $u \in S$. We omit the subscript when it is clear from the context. The set $\mathrm{Vor}(u)$ contains all vertices closer (by additively weighted distance) to u than to any other site $u' \neq u \in S$. We call $\mathrm{Vor}(u)$ the (primal) *Voronoi cell of* u. Note that a Voronoi cell might be empty. In what follows, we denote the shortest-path tree rooted at a site $u \in S$ as T_u. See Figure 1 for an illustration of some of the definitions in this section. The Voronoi diagram has the following basic properties.

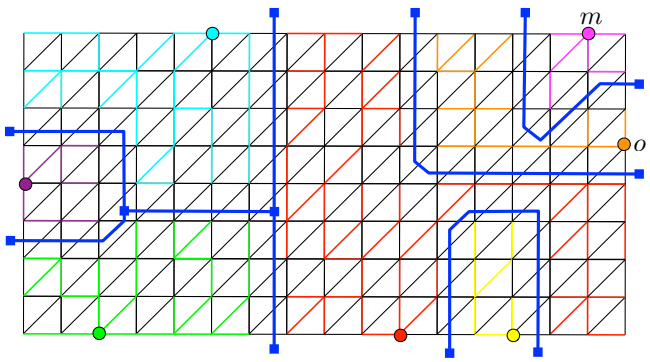

Figure 1: A graph (triangulated grid) with 7 sites on the infinite face (colored circles). Edge lengths and additive weights are not indicated. For each site u, the subtree of T_u spanning $\mathrm{Vor}(u)$ is indicated by edges with the same color as u. The dual Voronoi diagram $VD^*(S)$ is shown in blue. To avoid clutter, edges of $VD^*(S)$ incident to the infinite face are not shown to have a common endpoint. In this example $VD^*(S)$ has 3 Voronoi vertices which are indicated by solid blue squares (all the blue squares in the infinite face are in fact the same vertex). In this example, The boundary between the cells of the magenta site m and the orange site o happens to be the entire bisector $\beta^*(m, o)$, which is a simple dual cycle through the infinite face.

LEMMA 2.1. *For each $u \in S$, the vertices in $\mathrm{Vor}(u)$ form a connected subtree (rooted at u) of T_u.*

Proof. This is an immediate consequence of the property that shortest paths from a single source cannot cross one another (under our non-degeneracy assumption); in fact, they cannot even meet one another except at a common prefix. □

Let $B \subseteq E$ be the set of edges $vw \in E$ such that the sites u_1, u_2, for which $v \in \mathrm{Vor}(u_1)$ and $w \in \mathrm{Vor}(u_2)$, are distinct. Let B^* denote the set of the edges that are dual to the edges of B, and let $VD^*(S, \omega)$ (or $VD^*(S)$ for short) denote the subgraph of P^* with B^* as a set of edges. The following consequence of Lemma 2.1 gives some structural properties of $VD^*(S)$.

LEMMA 2.2. *The graph $VD^*(S)$ consists of at most $|S|$ faces, so that each of its faces corresponds to a site $u \in S$ and is the union of all faces of P^* that are dual to the vertices of* $\mathrm{Vor}(u)$.

Proof. For each $u \in S$, the union of all faces of P^* that are dual to the vertices of $\mathrm{Vor}(u)$ is connected, since $\mathrm{Vor}(u)$ is a tree. Moreover, by construction, the dual edge that separates two adjacent such faces cannot be part of any bisector. Each face of P^* belongs to exactly one face of $VD^*(S)$, because the trees $\mathrm{Vor}(u)$ are pairwise disjoint, and form a partition of V. Hence the faces of $VD^*(S)$ stand in 1-1 correspondence with the trees $\mathrm{Vor}(u)$, for $u \in S$, in the sense asserted in the lemma, and the claim follows. □

We refer to the face of $VD^*(S)$ corresponding to a site $u \in S$ as the *dual Voronoi cell* of u, and denote it as $\mathrm{Vor}^*(u)$. ($\mathrm{Vor}^*(u)$ is empty when $\mathrm{Vor}(u)$ is empty.) By Lemma 2.2, $\mathrm{Vor}^*(u)$ is the union of the faces dual to the vertices of $\mathrm{Vor}(u)$. Once we fix a concrete way in which we draw the dual edges in B^* in the plane, we can regard each $\mathrm{Vor}^*(u)$ as a concrete embedded planar region. Since the sets $\mathrm{Vor}(u)$ form a partition of V, it follows that the sets $\mathrm{Vor}^*(u)$ induce a partition of the sets of dual faces of P^*.

We define a vertex $f^* \in VD^*(S)$ to be a *Voronoi vertex* if its degree in $VD^*(S)$ is at least 3. This means that there exist at least three distinct sites whose primal Voronoi cells contain vertices incident to the primal face f dual to f^*. The next corollary follows directly from Euler's formula for planar graphs.

COROLLARY 2.1. *The graph $VD^*(S)$ consists of $O(b)$ vertices of degree ≥ 3 (which are the Voronoi vertices); all other vertices are of degree 2. The only vertices of degree strictly larger than 3 are those corresponding to the non-triangular holes among $h_1, \ldots, h_t$.*

LEMMA 2.3. *For any three distinct sites u, v, w in S there are at most two faces f of P such that each of the cells $\mathrm{Vor}(u)$, $\mathrm{Vor}(v)$, $\mathrm{Vor}(w)$ contains a vertex of f.*

Proof. Assume to the contrary that there are three such faces f_1, f_2, f_3. Let p_i, q_i, r_i, for $i = 1, 2, 3$, denote the vertices of f_i satisfying $p_i \in \mathrm{Vor}(u)$, $q_i \in \mathrm{Vor}(v)$, and $r_i \in \mathrm{Vor}(w)$. Let f_i^* be an arbitrary point inside f_i, for $i = 1, 2, 3$. We construct the following embedding of $K_{3,3}$ in the plane. One set of vertices is $\{f_1^*, f_2^*, f_3^*\}$ and the other is $\{u, v, w\}$. To connect f_i^* with u, say, we connect u to p_i via the shortest path $\pi(u, p_i)$, concatenated with the segment $p_i f_i^*$. The connections with v, w are drawn analogously. It is easily verified that the edges in this drawing do not cross one another (because shortest paths do not cross one another), so we get an impossible planar embedding of $K_{3,3}$, a contradiction that implies the claim. See Figure 2. □

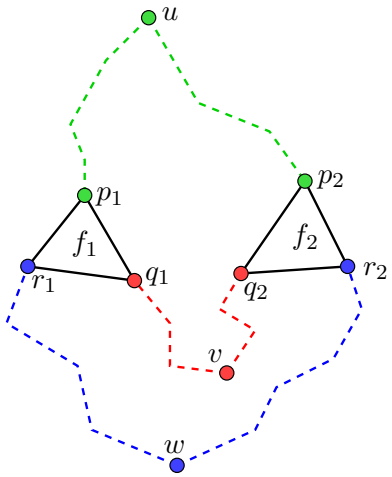

Figure 2: Illustration of the proof of Lemma 2.3.

Let u and v be two distinct sites in S. Let D denote the doubleton $\{u, v\}$. The *bisector* between u and v (with respect to their assigned weights), denoted as $\beta^*(u, v)$, is defined to be the set of arcs of P^* whose corresponding primal arcs have their tail in $\mathrm{Vor}_{D,\omega}(u)$ and their head in $\mathrm{Vor}_{D,\omega}(v)$. In other words, $\beta^*(u, v)$ is the set of duals of arcs of P whose tail is closer (w.r.t. ω) to u than to v, and whose head is closer to v than to u. Note that, unless we explicitly say otherwise, the bisector $\beta^*(u, v)$ is a directed object, and $\beta^*(v, u)$ consists of the reverses of the arcs of $\beta^*(u, v)$. Bisectors satisfy the following crucial property.

LEMMA 2.4. *$\beta^*(u, v)$ is a simple cycle of arcs of P^*. If u and v are incident to the same hole h and $\beta^*(u, v)$ is nonempty then $\beta^*(u, v)$ is incident to h^*.*

Proof. In the diagram of only the two sites u, v, $\mathrm{Vor}(u)$ and $\mathrm{Vor}(v)$ form a partition of the vertices of P into two connected sets. Therefore, the set B of arcs with tail in $\mathrm{Vor}(u)$ and head in $\mathrm{Vor}(v)$ is a simple cut in P. By the duality of simple cuts and simple cycles , $B^* = \beta^*(u, v)$ is a simple cycle. If u and v are incident to the same hole and $\beta^*(u, v)$ is nonempty, the simple cut defined by the partition $(\mathrm{Vor}(u), \mathrm{Vor}(v))$ must contain an arc e on the boundary of h. Therefore, e^* is an arc of $\beta^*(u, v)$ that is incident to h^*. □

By our conventions about the relation between primal and dual arcs, the bisector $\beta^*(u, v)$ is a directed clockwise cycle around u, and $\beta^*(v, u)$ is a directed clockwise cycle around v. Viewed as an undirected object, $\beta^*(u, v)$ corresponds to the dual Voronoi diagram $VD^*(\{u, v\}, \omega)$. See Figure 1.

3 Computing bisectors during preprocessing

To facilitate an efficient implementation of the algorithm, we carry out a preprocessing stage, in which we compute the bisectors $\beta^*(u, v)$ for every pair of sites $u, v \in S$ and for every pair of weights that can be assigned to u and v. To clarify this statement, note that $\beta^*(u, v)$ only depends on the *difference* $\delta = \omega(v) - \omega(u)$ between the weights of u and v, and that, due to the discrete nature of the setup, $\beta^*(u, v)$ changes only at a discrete set of differences. The preprocessing stage computes, for each pair $u, v \in S$, all possible bisectors, by varying δ from $+\infty$ to $-\infty$. As we do this, $\beta^*(u, v)$ "sweeps" over P^*, moving farther from u and closer to v, in a sense that will be made more precise shortly. We find all the critical values of δ at which $\beta^*(u, v)$ changes, and store all versions of $\beta^*(u, v)$ in a (partially) persistent binary search tree [52]. Each version of the bisector is represented as a binary search tree on the (cyclic) list of its dual vertices and edges (which we cut open at some arbitrary point, to make the list linear). Hence, we can find the k-th edge on any bisector $\beta^*(u, v)$ in $O(\log |\beta^*(u, v)|) = O(\log r)$ time, for any $1 \leq k \leq |\beta^*(u, v)|$ (where $|\beta^*(u, v)|$ denotes the number of edges on the bisector $\beta^*(u, v)$).

Consider a critical value of $\delta = \omega(v) - \omega(u)$ at which $\beta^*(u, v)$ changes. We assume (see Section 2) that there is a unique arc $yz \in T_v$ such that for $\delta' > \delta$, $y \in \mathrm{Vor}(v)$

and $z \in \text{Vor}(u)$, and $\delta = d(u,z) - d(v,z)$. We say that yz is *tense* at δ. For $\delta' > \delta$, z was a node in the shortest-path (sub)tree $\text{Vor}(u)$, and for $\delta' < \delta$ it becomes a node of $\text{Vor}(v)$, as a child of y. If z was not a leaf of $\text{Vor}(u)$ (at the time of the switch), then the entire subtree rooted at z moves with z from $\text{Vor}(u)$ to $\text{Vor}(v)$ (this is an easy consequence of the property that, under our unique shortest paths assumption, shortest paths from a single source do not meet or cross one another). These dynamics imply the following crucial property of bisectors (the proof is deferred to Section 5).

LEMMA 3.1. *Consider some critical value δ of $\omega(v) - \omega(u)$ where $\beta^*(u,v)$ changes. The dual edges that newly join $\beta^*(u,v)$ at δ form a contiguous portion of the new bisector, and the dual edges that leave $\beta^*(u,v)$ at δ form a contiguous portion of the old bisector.*

We compute and store, for each vertex p, and for each pair of sites u, v, the unique value of δ at which p moves from $\text{Vor}(u)$ to $\text{Vor}(v)$. Denote this value as $\delta^{uv}(p)$. Namely, $\delta^{uv}(p) = d(u,p) - d(v,p)$. Thus, for $\delta > \delta^{uv}(p)$, $p \in \text{Vor}(u)$, and for $\delta < \delta^{uv}(p)$, $p \in \text{Vor}(v)$. Consider an arc pq of P. If $\delta^{uv}(p) = \delta^{uv}(q)$ (i.e. pq is not the tense arc) then both endpoints of the arc move together from $\text{Vor}(u)$ to $\text{Vor}(v)$, so the dual arc never becomes an arc of $\beta^*(u,v)$. If $\delta^{uv}(p) < \delta^{uv}(q)$ then q moves first (as δ decreases) and right after time $\delta^{uv}(q)$ the arc dual to pq becomes an arc of $\beta^*(u,v)$. It stops being an arc of $\beta^*(u,v)$ at $\delta^{uv}(p)$. Similarly, if $\delta^{uv}(q) < \delta^{uv}(p)$ then p moves first and right after time $\delta^{uv}(p)$ the arc dual to qp becomes an arc of $\beta^*(u,v)$. It stops being an arc of $\beta^*(u,v)$ at $\delta^{uv}(q)$.

We compute the bisectors $\beta^*(u,v)$ by adding and deleting arcs at the appropriate values of δ. By Lemma 3.1, the arcs that leave and enter $\beta^*(u,v)$ at any particular value of δ form a single subpath of $\beta^*(u,v)$. We can infer the appropriate position of each arc by ensuring that the order of tails of the arcs of $\beta^*(u,v)$ respects the preorder traversal of T_u.[8] Note that there are $O(b^2)$ pairs of sites in S, so for each vertex $p \in V$ we compute and store $O(b^2)$ values of δ, for a total of $O(rb^2)$ storage. To compute the bisectors for each pair of sites, we perform $O(r)$ updates to the corresponding persistent search tree. We have thus established the following towards the preprocessing part of Theorem 1.1.

THEOREM 3.1. *Consider the settings of Theorem 1.1. One can compute in $\tilde{O}(rb^2)$ time and space the persistent binary search tree representations of all possible bisectors for all pairs of sites in S.*

4 Computing Voronoi vertices

Consider the settings of Theorem 1.1. We will henceforth only deal with a single subgraph P, so to simplify notation we denote the size of P by n (rather than r). Recall that, by Lemma 2.3, a Voronoi diagram with three sites has at most two Voronoi vertices. In this section we prove the following theorem.

THEOREM 4.1. *Let P be a directed planar graph with real arc lengths, n vertices, and no negative length cycles. Let S be a set of sites that lie on the boundaries of a constant number of faces (holes) of P. One can preprocess P in $\tilde{O}(n|S|^2)$ time so that, given any three sites $r, g, b \in S$ with additive weights $\omega(\cdot)$, one can find the (at most two) Voronoi vertices of $VD^*(\{r,g,b\})$ in $\tilde{O}(1)$ time.*

We will actually prove a more general theorem that extends the single site b to a sequence of sites B on the same hole. More formally, given any set $B \subseteq S$ of sites, we say that the sites B are *nearly consecutive* on hole h if they all lie on a subpath of the boundary of h, and this subpath contains a constant number (say, 2, for concreteness) of nodes in $S \setminus B$. Thus, B can be represented as the set of sites on at most three subpaths of the boundary of h. Let r, g be two sites, and let $B \subseteq S \setminus \{r,g\}$ be a set of nearly consecutive sites on some hole h. Consider adding an artificial site v_B embedded inside h and connected to all sites in B with artificial arcs whose lengths are the corresponding additive weights of the sites in B. Denote by P' be the resulting graph. We define the bisector $\beta^*(g,B)$ in P to be the bisector $\beta^*(g,v_B)$ in P' (ignoring the artificial arcs). Similarly, we define the diagram $VD(\{r,g,B\})$ in P to be the diagram $VD(\{r,g,v_B\})$ in P'. The cell $\text{Vor}(B)$ contains all vertices closer (by additively weighted distance) to some $b_i \in B$ than to any other site $u' \in \{r,g\}$. By Lemma 2.3, the dual diargram $VD^*(\{r,g,B\})$ consists of at most two vertices, each corresponding to a face in P that contains a vertex in $\text{Vor}(r)$, a vertex in $\text{Vor}(g)$, and a vertex in $\text{Vor}(b)$ for some $b \in B$.

THEOREM 4.2. *Consider the settings of Theorem 4.1. One can preprocess P in $\tilde{O}(n|S|^2)$ time so that the following procedure takes $\tilde{O}(1)$ time. The inputs to the procedure are two sites $r, g \in S$, a set $B \subset S \setminus \{r,g\}$ of nearly consecutive sites on a hole[9] h, with respective*

[8] By preorder traversal we mean breadth first search where descendants of a node are visited according to their cyclic order in the embedding.

[9] Since the sites in B are nearly consecutive, B can be implicitly represented by the endpoints of at most 3 subpaths of the boundary of the hole h.

additive weights $\omega(\cdot)$, as well as a representation of the bisector $\beta^*(g, B)$ that allows one to retrieve the k'th vertex of $\beta^*(g, B)$ in $\tilde{O}(1)$ time. The output of the procedure are the (at most two) vertices of $VD^*(\{r, g, B\})$.

Due to space constraints, we only prove Theorem 4.1 in this extended abstract. The modifications to the proof required for establishing Theorem 4.2 will appear in the full version. In the case of Theorem 4.1, there are three sites, and the Voronoi diagram has at most three cells. We call a face f of P *trichromatic* if it has an incident vertex in each of the three cells of the diagram. *Monochromatic* and *bichromatic* faces are defined similarly. (This definition also includes faces that are holes.) We say that a vertex v of P is *red*, *green*, or *blue* if v is in the Voronoi cell of r, g, or b, respectively. By definition, the Voronoi vertices that we seek are precisely those dual to the trichromatic faces of P. Let $\beta^*(g, b)$ denote the bisector of g and b (with respect to the additive weights $\omega(g), \omega(b)$). By Lemma 2.4, $\beta^*(g, b)$ is a simple cycle in P^*. For $c \in \{g, b\}$, and for each vertex $v \in V(P)$, define $\tilde{\delta}^{rc}(v) := \omega(c) + d(c, v) - d(r, v)$. Equivalently, $\tilde{\delta}^{rc} = \omega(c) - \delta^{rc}$. Define $\Delta^r(v) := \min\{\tilde{\delta}^{rg}(v), \tilde{\delta}^{rb}(v)\}$. That is, $\Delta^r(v)$ is the maximum weight that can be assigned to r so that v is red (and v will be red also for any smaller assigned weight). Indeed, if $\omega(r) > \tilde{\delta}^{rg}(v)$, say, then $\omega(r) + d(r, v) > \omega(g) + d(g, v)$, so v is not red. For each edge uv of P, define $\Delta^r(uv) := \max\{\Delta^r(u), \Delta^r(v)\}$. That is, $\Delta^r(uv)$ is the maximum weight that can be assigned to r so that at least one endpoint of uv is red. For an edge e^*, dual to a primal edge e, we put $\Delta^r(e^*) = \Delta^r(e)$.

For any real x, we denote by VD_x the Voronoi diagram of r, g, b, with respective additive weights $x, \omega(g), \omega(b)$. We define

$$Q^*_{\geq x} := \{e^* \in \beta^*(g, b) \mid \Delta^r(e^*) \geq x\} ,$$
$$Q^*_{=x} := \{e^* \in \beta^*(g, b) \mid \Delta^r(e^*) = x\} .$$

4.1 Warmup: the case of sites on a single hole.

We begin with describing the case in which r, g and b are all incident to the same hole h. The multiple hole case follows the same principles, but is much more complicated. Note that in the single hole case, by Lemma 2.4, $\beta^*(g, b)$ is a simple cycle incident to h^*. We treat $\beta^*(g, b)$ as a path starting and ending at h^*. This defines a natural order on the arcs and vertices of $\beta^*(g, b)$.

LEMMA 4.1. *If r, g and b are incident to the same hole h then for any $\infty > x > -\infty$, the edges of $Q^*_{\geq x}$ form a prefix or a suffix $\beta^*(g, b)$. Futhermore, the duals of the endpoints of $Q^*_{\geq x}$ are the trichromatic faces in VD_x.*

Proof. Consider h as the infinite face of P. Since $\beta^*(g, b)$ is a simple cycle incident to h^*, it partitions the cycle h into two paths, one containing g and the other containing b. We refer to these subpaths of h as the green and blue subpaths, respectively. Assume that r belongs to the green subpath. Further assume that r lies before g on the green subpath, where the order of the vertices on the green subpath is defined such that the first vertex is an endpoint of the first arc of $\beta^*(g, b)$, and the last vertex is an endpoint of the last arc of $\beta^*(g, b)$. See Figure 3. Under these assumptions we will prove that the edges of $Q^*_{\geq x}$ form a prefix of $\beta^*(g, b)$. The other cases are analogous.

Consider any value of x. Let e^* be the last arc of $\beta^*(g, b)$ such that $\Delta^r(e^*) \geq x$. In other words, e^* is the last arc of $\beta^*(g, b)$ such that e has an endpoint in the cell of r in VD_x. If e^* does not exist then $Q^*_{\geq x}$ is empty and the lemma trivially holds. It suffices to prove that for every edge e^*_1 that appears on $\beta^*(g, b)$ before e^*, e has an endpoint in the cell of r in VD_x.

Let v be an endpoint of e in the cell of r in VD_x. Let γ be the r-to-v shortest path in P. By planarity of P, for any edge e^*_1 that appears on $\beta^*(g, b)$ before e^*, the path γ intersects either the shortest path from g to the endpoint of e_1 that is closer to g than to b, or the shortest path from b to the endpoint of e_1 that is closer to b than to g. In either case, this implies that e_1 has an endpoint that is closer to r than to both g and b, i.e., an endpoint in the cell of r in VD_x.

To prove the second statement of the lemma observe that one endpoint of $Q^*_{\geq x}$ is always the hole h, which is a trichromatic face assuming VD_x indeed has three nonempty cells. Consider the other endpoint f of $Q^*_{\geq x}$. Let e^* be the arc of $Q^*_{\geq x}$ incident to f, and let e^*_1 be the arc of $\beta^*(g, b)$ that does not belong to $Q^*_{\geq x}$ and is also incident to f. By definition of $Q^*_{\geq x}$, e has an endpoint in the cell of r in VD_x. Since $e^*_1 \notin Q^*_{\geq x}$, both endpoints of e_1 are not in the cell of r in VD_x. Therefore, since $e^*_1 \in \beta^*(g, b)$, e_1 has one endpoint in the cell of g in VD_x, and one endpoint in the cell of b in VD_x. Therefore, f is trichromatic. □

We define $\mathbf{\Delta}^r_\beta$ to be the cyclic sequence $\Big(\Delta^r(e^*) \mid e^* \in \beta^*(g, b)\Big)$. Note that $\Delta^r(e^*)$ is not known at preprocessing time since it depends on three sites and on their relative weights. An immediate consequence of Lemma 4.1 is:

COROLLARY 4.1. *The sequence $\mathbf{\Delta}^r_\beta$ is weakly monotone.*

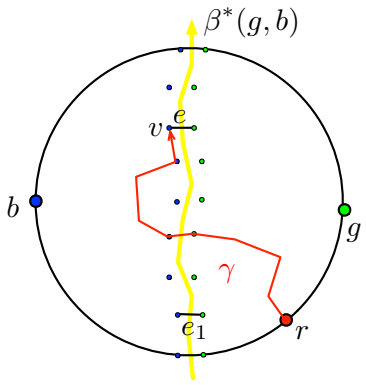

Figure 3: Illustration of the proof of Lemma 4.1. The bisector $\beta^*(g,b)$ is shown in yellow. The r-to-v shortest path γ is shown in red.

Proof. By Lemma 4.1, $Q^*_{\geq x}$ is a prefix or a suffix of $\beta^*(g,b)$, for each $\infty > x > -\infty$, and by definition we have $Q^*_{\geq x'} \subseteq Q^*_{\geq x}$ for every pair $x' \geq x$. This clearly implies the corollary. □

By the second part of Lemma 4.1, to find the trichromatic vertices of VD_x one needs to find the endpoints of $Q^*_{\geq x}$. By Corollary 4.1, one can find it by binary searching for x in $\mathbf{\Delta}^r_\beta$. Note that a single element of $\mathbf{\Delta}^r_\beta$ can be computed on the fly in constant time given the shortest path trees rooted at r, g and b. Also note that our representation of bisectors supports retrieving the k'th arc of the bisector in $\tilde{O}(1)$ time. Therefore, the binary search can be implemented in $\tilde{O}(1)$ time as well. This proves Theorem 4.1 for the case of a single hole.

4.2 The case of sites on multiple holes. In the remainder of this section we treat the general case when sites are not necessarily on the same hole. In this case $Q^*_{\geq x}$ is a subpath of $\beta^*(g,b)$, but not necessarily a prefix or a suffix. As a consequence, $\mathbf{\Delta}^r_\beta$ is no longer weakly monotonic, but *weakly bitonic*. This makes the binary search procedure much more involved. We first establish the bitonicity of $\mathbf{\Delta}^r_\beta$ and describe the bitonic search. We then elaborate on the various steps of the search. The following lemma proves that $Q^*_{\geq x}$ is a subpath of $\beta^*(g,b)$.

LEMMA 4.2. *For any $\infty > x > -\infty$, the edges of $Q^*_{\geq x}$ form a subpath of $\beta^*(g,b)$. Furthermore, if $Q^*_{\geq x}$ is non-empty and does not contain all the edges of $\beta^*(g,b)$, then the trichromatic faces in VD_x are the duals of the endpoints of $Q^*_{\geq x}$; that is, these are the faces whose duals have exactly one incident edge in $Q^*_{\geq x}$.*

Proof. Let Q^* be a maximal subset of edges in $Q^*_{\geq x}$ that form a (contiguous) subpath of $\beta^*(g,b)$. Assume that there exists at least one edge of $\beta^*(g,b)$ that is not in Q^*; otherwise $Q^* = \beta^*(g,b)$, which we assume not to be the case.

Enumerate the dual vertices incident to the edges of Q^* as $f^*_1, f^*_2, \ldots, f^*_j$ in their (cyclic) order along $\beta^*(g,b)$. The vertex f^*_1 has an incident edge $f^*_1 f^*_2$ in $Q^*_{\geq x}$, and another incident edge, call it $f^*_0 f^*_1$, that is in $\beta^*(g,b)$ but not in $Q^*_{\geq x}$. In the primal, the face f_1, dual to f^*_1, has an incident edge uv that is dual to $f^*_1 f^*_2$, such that, by construction, at least one of u, v is red, and another incident edge $u'v'$, dual to $f^*_0 f^*_1$, such that none of u', v' is red (note that when f_1 is a triangle, one of u', v' coincides with one of u, v). Moreover, since $f^*_0 f^*_1$ is an edge of the bisector $\beta^*(g,b)$, exactly one of u', v' is blue and the other one is green. Therefore, the face f_1 is trichromatic. A similar argument shows that f_j is also trichromatic. Note that the argument does not rely on the faces being triangles, so it also applies in the presence of holes. See Figure 4.

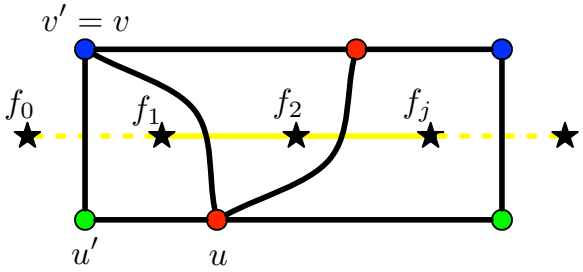

Figure 4: Illustration of the proof of Lemma 4.2. A set of consecutive edges in $\beta^*(g,b)$ is shown in yellow. Edges of Q^* are solid, and edges not in Q^* are dashed. Dual vertices are indicated by stars. Primal vertices (circles) are colored according to the Voronoi region they belong to. Primal edges are shown in black. In this example u, u' are distinct vertices, but $v' = v$.

This shows that every maximal subset of edges in $Q^*_{\geq x}$ that forms a subpath of $\beta^*(g,b)$ gives rise to two distinct trichromatic faces. Since, by Lemma 2.3, there are at most two trichromatic faces we must have that $Q^* = Q^*_{\geq x}$ and the lemma follows. □

Using standard notation, we say that a linear sequence is strictly (weakly) *bitonic* if it consists of a strictly (weakly) decreasing sequence followed by a strictly (weakly) increasing sequence. A cyclic sequence is strictly (weakly) bitonic if there exists a cyclic shift that makes it strictly (weakly) bitonic; this shift starts and ends at the (a) maximum element of the sequence. Recall that we defined $\mathbf{\Delta}^r_\beta$ to be the cyclic sequence $\left(\Delta^r(e^*) \mid e^* \in \beta^*(g,b)\right)$. Note that $\Delta^r(e^*)$ is not known at preprocessing time since it depends on three sites and

on their relative weights. An immediate consequence of Lemma 4.2 is:

COROLLARY 4.2. *The cyclic sequence* $\mathbf{\Delta}^r_\beta$ *is weakly bitonic.*

Proof. By Lemma 4.2, $Q^*_{\geq x}$ is a subpath of $\beta^*(g,b)$, for each $\infty > x > -\infty$, and by definition we have $Q^*_{\geq x'} \subseteq Q^*_{\geq x}$ for every pair $x' \geq x$. This clearly implies the corollary. □

We will show that we can find the maximum of $\mathbf{\Delta}^r_\beta$ in $\tilde{O}(1)$ time. This will allow us to turn the weakly bitonic cyclic sequence $\mathbf{\Delta}^r_\beta$ into a weakly bitonic linear sequence. We will then use binary search on $\mathbf{\Delta}^r_\beta$ to find the endpoints of $Q^*_{\geq \omega(r)}$, which, by the second part of Lemma 4.2, are the trichromatic faces of the Voronoi diagram of r, g, b with respective additive weights $\omega(r), \omega(g), \omega(b)$. The search might of course fail to find these vertices when they do not exist, either because $\omega(r)$ is too small, in which case $\mathrm{Vor}^*(r)$ completely "swallows" $\beta^*(g,b)$, or because $\omega(r)$ is too large, in which case $\beta^*(g,b)$ either appears in full in $VD^*(\{r,g,b\},\omega)$ or not at all, and there are either no Voronoi vertices, or there is a single Voronoi vertex which is dual to one of the holes.

Before we show how to find the maximum of $\mathbf{\Delta}^r_\beta$, we briefly discuss a general strategy for conducting binary search on linear bitonic sequences. This search is not trivial, especially when the sequence is weakly bitonic. We first consider the case of strict bitonicity, and then show how to extend it to the weakly bitonic case.

Searching in a strictly bitonic linear sequence. Given a strictly bitonic linear sequence σ and a value y, one can find the two "gaps" in σ that contain y, where each such gap is a pair of consecutive elements of σ such that y lies between their values. (For simplicity of presentation, and with no real loss of generality, we only consider the case where y is not equal to any element of the sequence.) This is done by the following variant of binary search.

The search consists of two phases. In the first phase the interval that the binary search maintains still contains both gaps (if they exist), and in the second phase we have already managed to separate between them, and we conduct two separate standard binary searches to identify each of them.

Consider a step where the search examines a specific entry $x = \sigma(i)$ of σ.

(i) If $x > y$, we compute the discrete derivative of σ at i. If the derivative is positive (resp., negative), we update the upper (resp., lower) bound of the search to i. (This rule holds for both phases.)

(ii) If $x < y$ and we are in the first phase, we have managed to separate the two gaps, and we move on to the second phase with two searches, one with upper bound i and one with lower bound i. If we are already in the second phase, we set the upper (resp., lower) bound to i if we are in the lower (resp., higher) binary search.

Searching in a weakly bitonic linear sequence. This procedure does not work for a weakly bitonic sequence (consider, e.g., a sequence all of whose elements, except for one, are equal). This is because the discrete derivative in (i) might be locally 0 in the weakly bitonic case. This difficulty can be overcome if, given an element i in σ such that $\sigma(i) = x$, we can efficiently find the endpoints of the maximal interval I of σ that contains i and all its elements are of value equal to x (note that in general σ might contain up to two intervals of elements of value equal to x, only one of which contains i). We can then compute the discrete derivatives at the endpoints of I, and use them to guide the search, similar to the manner described for the strict case.

Unfortunately, now focusing on the specific context under consideration, given an edge $\hat{e}^* \in \beta^*(g,b)$ such that $\Delta^r(\hat{e}^*) = x$, we do not know how to find the maximal interval I of $\beta^*(g,b)$ such $\hat{e}^* \in I$ and for every edge $e^* \in I$, $\Delta^r(e^*) = x$. Instead, we provide a procedure that returns the smallest interval I^+ of (the cyclic) $\beta^*(g,b)$ that contains all edges $e^* \in \beta^*(g,b)$ for which $\Delta^r(e^*) = x$, and no edge e^* for which $\Delta^r(e^*) < x$. Note that I^+ might contain edges $e^* \in \beta^*(g,b)$ for which $\Delta^r(e^*) > x$. When there is just one interval of elements of value equal to x then I^+ is in fact that interval. When there are two intervals of elements of value equal to x then I^+ also contains all edges $e^* \in \beta^*(g,b)$ for which $\Delta^r(e^*) > x$, and, in particular, an edge $e^*_{max} \in \beta^*(g,b)$ maximizing $\Delta^r(\cdot)$. In this case, if $\hat{e}^*$ appears before (resp., after) e^*_{max} in I^+ then $\hat{e}^*$ is in the increasing (resp., decreasing) part of $\mathbf{\Delta}^r_\beta$, and we should set the upper (resp., lower) bound of the search in (i) to the beginning (resp., end) of the interval I^+.

In Section 4.3 we describe the procedure that finds an edge $e^*_{\max} \in \beta^*(g,b)$ with a largest value of Δ^r. In Section 4.4 we introduce a new definition and describe some additional preprocessing that is required for the mechanism for finding I^+ with the properties described above. This mechanism is described in Section 4.5.

4.3 Finding the maximum in $\mathbf{\Delta}^r_\beta$. We now describe a procedure for finding $x_{max} := \max(\mathbf{\Delta}^r_\beta)$, or more precisely, finding some edge $e^*_{\max} \in \beta^*(g,b)$ such that $\Delta^r(e^*_{\max}) = x_{max}$. Let h_r be the hole to which the site r is incident. We check whether r belongs to $\mathrm{Vor}(g)$

or to $\mathrm{Vor}(b)$ in $VD(\{g,b\})$ by comparing the distances from g to r and from b to r. Assume that r belongs to $\mathrm{Vor}(g)$ (the case where r belongs to $\mathrm{Vor}(b)$ is symmetric). Let T_g^* be the cotree of the shortest-path tree T_g. Define the label $\ell_g^{h_r}(f^*)$, for each dual vertex f^*, to be equal to the number of edges on the f^*-to-h_r^* path in T_g^*. Note that, because the number of holes is constant, these labels can be computed during preprocessing, when we compute the tree T_g^*, without changing the asymptotic preprocessing time. Furthermore, we can augment the persistent search tree representation of the bisectors with these labels, so that, given $\beta^*(g,b)$ and r, we can retrieve in $\tilde{O}(1)$ time the vertex of $\beta^*(g,b)$ minimizing $\ell_g^{h_r}$.

LEMMA 4.3. *The value of $\Delta^r(\cdot)$ is x_{max} for at least one of the two arcs of $\beta^*(g,b)$ incident to the dual vertex $f^* \in \beta^*(g,b)$ minimizing $\ell_g^{h_r}(\cdot)$.*

We omit the proof. See the full version for details.

4.4 Arc labels. We now define labels for the arcs of P. This is an extension of preorder traversal labels of a spanning tree of P to labels to all arcs of P. These labels will be instrumental in identifying the interval I^+ defined in Section 4.2. Let $c \in S$ be a site, and let T_c be the shortest path tree rooted at c. We define *labels* for the arcs of P with respect to the site c, as follows. For the sake of definition only, we modify P as follows. For each arc uv of P, we create two artificial vertices u_{uv} and v_{uv}, and two artificial arcs uv_{uv} and vu_{uv}. The arc uv_{uv} (resp., vu_{uv}) is embedded so that it immediately precedes uv (resp., the reverse of uv) in the counterclockwise cyclic order of arcs around u (resp., v). See Figure 5. (We apply this construction only once for each pair of anti-parallel arcs.) Let T_c' be the tree obtained by adding to T_c all the artificial arcs, all of which are leaf arcs in T_c'. For each arc $uv \in P$, define its *label* $\mathrm{pre}_c(uv)$ to be the preorder index of the artificial vertex v_{uv} in T_c', in a *CCW-first* search of T_c'.[10] Similarly, define the label $\mathrm{pre}_c(vu)$ of the reverse arc vu to be the preorder index of the vertex u_{uv} in T_c'. We define $\mathrm{pre}_c(e^*)$ to be $\mathrm{pre}_c(e)$. The goal of the artificial arcs and vertices is to enable us to extend the definition of the preorder labels to all arcs of P and their reverses. Note that this order is consistent with the usual preorder on just the arcs of T_c.

LEMMA 4.4. *Let $g,b \in S$ be sites and let $c \in \{g,b\}$ be a site. Let u be a node in the cell $\mathrm{Vor}(c)$ in $VD(\{g,b\},\{\omega(g),\omega(b)\})$. Let p be the parent of u in T_c. Then the following hold:*

1. *The arcs in $R_u^* := \{e^* \in \beta^*(g,b) \mid \mathrm{pre}_c(pu) < \mathrm{pre}_c(e^*) < \mathrm{pre}_c(up)\}$ form a subpath of $\beta^*(g,b)$.*
2. *The labels $\mathrm{pre}_c(e^*)$ are strictly monotone along R_u^*.*

Proof. Consider the cells $\mathrm{Vor}(g)$ and $\mathrm{Vor}(b)$ of g and b in $VD(\{g,b\},\{\omega(g),\omega(b)\})$. They classify the faces of P into three types: (i) those that are not incident to any vertex of $\mathrm{Vor}(g)$, (ii) those that are not incident to any vertex of $\mathrm{Vor}(b)$, and (iii) those that are incident to a vertex of each set. Faces of type (iii) are the faces dual to the vertices of $\beta^*(g,b)$. We denote by f_g the union of the faces of type (i) and (iii), and by f_b the union of the faces of type (ii) and (iii). It follows that for every arc $e^* \in \beta^*(g,b)$, such that $e = vw$ (so v belongs to $\mathrm{Vor}(g)$), both v and w lie in the face f_g, where v lies on its boundary and w in its interior. See Figure 6 for an illustration. Viewed as sets of edges in P, ∂f_g and ∂f_b are cycles, which we denote by β_g and β_b, respectively. Note that the arcs with tail on β_g and head on β_b are exactly the dual arcs of $\beta^*(g,b)$. Note also that, since the restriction of T_g to $\mathrm{Vor}(g)$ is a connected subtree of T_g (Lemma 2.1), a branch of T_g that enters f_g (through β_g) does not leave it.

Assume, without loss of generality, that $c = g$. Consider the pair of arcs $e_1^*, e_3^* \in \beta^*(g,b)$ such that $\mathrm{pre}_g(pu) < \mathrm{pre}_g(e_1) < \mathrm{pre}_g(e_3) < \mathrm{pre}_g(up)$ (that is, e_1^*, $e_3^* \in R_u^*$), so that $\mathrm{pre}_g(e_1)$ is smallest and $\mathrm{pre}_g(e_3)$ is largest under the above constraints. Denote $e_i = v_i w_i$, for $i = 1,3$ (so v_i is the endpoint in $\mathrm{Vor}(g)$). By definition of preorder, both v_1 and v_3 are descendants of u in T_g. Consider the (undirected) cycle γ formed by the u-to-v_1 path in T_g, the u-to-v_3 path in T_g, and one of the two v_1-to-v_3 subpaths of β_g, chosen so that γ does not enclose the blue site b. Let e_2^* be any edge in R_u^*, with a primal edge $e_2 = v_2 w_2$, where $v_2 \in \mathrm{Vor}(g)$. By choice of e_1^* and e_3^*, and by definition of R_u^*, $\mathrm{pre}_g(pu) < \mathrm{pre}_g(e_1) \le \mathrm{pre}_g(e_2) \le \mathrm{pre}_g(e_3) < \mathrm{pre}_g(up)$, so v_2 is also a descendant of u in T_g, which is visited in between the visits of v_1 and v_3. By definition of the CCW-first search preorder, at the node where the paths in T_g to v_1 and to v_2 (resp., to v_2 and to v_3) bifurcate, the arc towards v_2 lies clockwise to the arc towards v_1 (resp., counterclockwise to the arc towards v_3). Since no branch of T_g exits f_g, and no pair of paths in T_g cross each other, it follows that v_2 lies between v_1 and v_3 on $\beta_g \cap \gamma$. This establishes (1). Establishing (2) is done with a similar, slightly modified argument, applied to any pair of edges e_1, e_3 satisfying $\mathrm{pre}_g(pu) < \mathrm{pre}_g(e_1) < \mathrm{pre}_g(e_3) < \mathrm{pre}_g(up)$. □

[10] A CCW-first search is a DFS that always visits the unvisited child of a node u that lies most counter-clockwise with respect to the other unvisited children, where the counter-clockwise order starts and ends at the incoming edge of u.

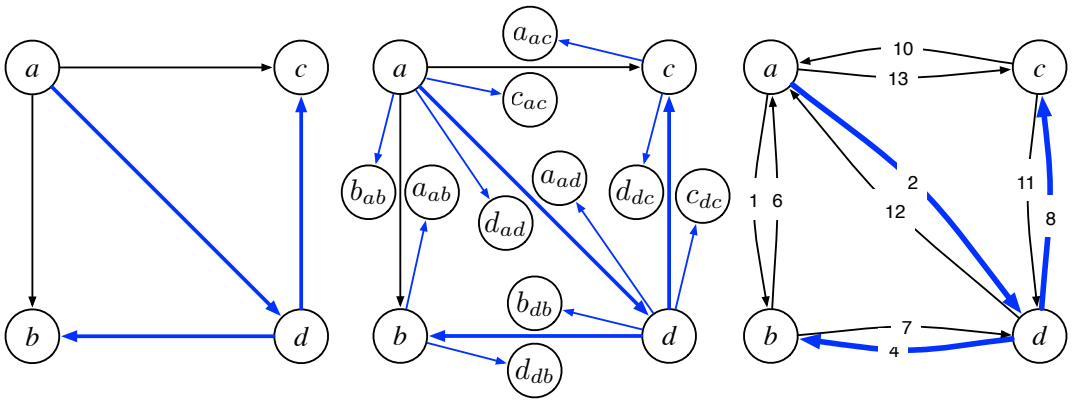

Figure 5: Illustration of the definition of labels $\text{pre}_c(\cdot)$ for all arcs. Left: A directed graph with a spanning tree T_a (in blue), rooted at vertex a. Middle: The same graph with artificial arcs and vertices. The tree T'_a is shown in blue. Right: The labels $\text{pre}_a(\cdot)$ for all arcs.

4.5 The mechanism. We now return to presenting our mechanism for finding I described in the beginning of Section 4. To this end, we need to exploit some structure of the shortest path trees rooted at the three sites, and the evolution of the Voronoi diagram $VD_x := VD(\{r, g, b\})$, with the weights $\omega(g), \omega(b)$ kept fixed and $\omega(r) = x$, as x decreases from $+\infty$ to $-\infty$. For $c \in \{r, g, b\}$, let $\text{Vor}_x(c)$ denote the current version of $\text{Vor}(c)$ (for the weight $\omega(r) = x$); recall that it is a subtree of T_c that spans the vertices in $\text{Vor}(c)$ in VD_x. These subtrees form a spanning forest F^x of P. We call any edge not in F^x with one endpoint in $\text{Vor}_x(c_1)$ and the other in $\text{Vor}_x(c_2)$, for a pair of distinct sites $c_1 \neq c_2$, c_1c_2-*bichromatic*. To handle the case where $\text{Vor}_x(r)$ is empty (e.g., at $x = +\infty$), we think of adding a super source s connected to each site c with an edge sc of weight $\omega(c)$ (in general, these edges cannot be embedded in the plane together with P), and define the edge sr to be bichromatic.

We now define some bichromatic arcs to be tense at certain critical values of x in a way similar to the definition of tense arcs in Section 3. The difference is that here an arc uv is tense at x if v becomes closer to r at x than to *both* g and b (rather than to just one other site as in Section 3). Specifically, we say that an arc uv is *tense* at x if uv is rc-bichromatic just before x (i.e., for values slightly larger than x), and uv is an arc of the full tree T_r, and $x + d_{T_r}(r, v) = \omega(c) + d_{T_c}(c, v) = d(s, v)$. The values of x at which some arc becomes tense are a subset of the critical values at which the rb-bisector or the rg-bisector change. Recall that we assume that at each critical value x only one arc is tense. At the critical value x, the red tree $\text{Vor}_x(r)$ takes over the node v of the tense arc uv. Let $\text{Vor}^+_x(c)$ be the primal Voronoi cell of c just before the critical value x. For any descendant w of v in $\text{Vor}^+_x(c)$, we have

$$\begin{aligned} d(s, w) &= \omega(c) + d_{T_c}(c, v) + d_{T_c}(v, w) \\ &= x + d_{T_r}(r, v) + d_{T_c}(v, w), \end{aligned}$$

so the red tree also takes over the entire subtree of v in $\text{Vor}^+_x(c)$. This establishes the following lemma.

LEMMA 4.5. *At any critical value x of $\omega(r)$ the vertices that change color at x are precisely the descendants of v in $\text{Vor}^+_x(c)$ where uv is the unique tense arc at x.*

Consider the sequence $\mathbf{\Delta}^r_\beta$, and let $x' > x$ be two consecutive values in it. Recall that both $Q^*_{\geq x'}$ and $Q^*_{\geq x}$ are contiguous (cyclic) subsequences of $\beta^*(g, b)$ (Lemma 4.2). Assume $Q^*_{\geq x'} = \beta^*(g, b)[j \dots k]$, and $Q^*_{\geq x} = \beta^*(g, b)[i \dots l]$ (where the indices are taken modulo the length of $\beta^*(g, b)$). Since, by definition, $Q^*_{\geq x'} \subset Q^*_{\geq x}$, the set $Q^*_{=x} := Q^*_{\geq x} \setminus Q^*_{\geq x'}$ corresponding to elements of $\mathbf{\Delta}^r_\beta$ with value exactly x, forms two intervals $[i \dots j), (k \dots l]$ of $\beta^*(g, b)$, at least one of which is non-empty. Let uv be the unique tense arc at critical value x. Let $c \in \{g, b\}$ be such that uv is rc-bichromatic just before x. By the preceding arguments, including Lemma 4.5, the nodes w with $\Delta^r(w) = x$ are descendants of v in the subtree of T_c spanning $\text{Vor}^+_x(c)$. Furthermore, the subtree of T_c spanning $\text{Vor}_\infty(c)$ (this is the cell of c in $VD(\{g, b\})$ when r is at distance ∞ from all vertices) contains the subtree of T_c spanning $\text{Vor}^+_x(c)$ and additional nodes w for which $\Delta^r(w) = \tilde{\delta}^{rc}(w) \geq x$ that switched to the red tree at critical values greater than x (recall that x is decreasing).

Recall the definition of the path R^*_v in the statement of Lemma 4.4 (with respect to the bisector $\beta^*(g, b)$). It follows, by the discussion above, that any edge e^* of

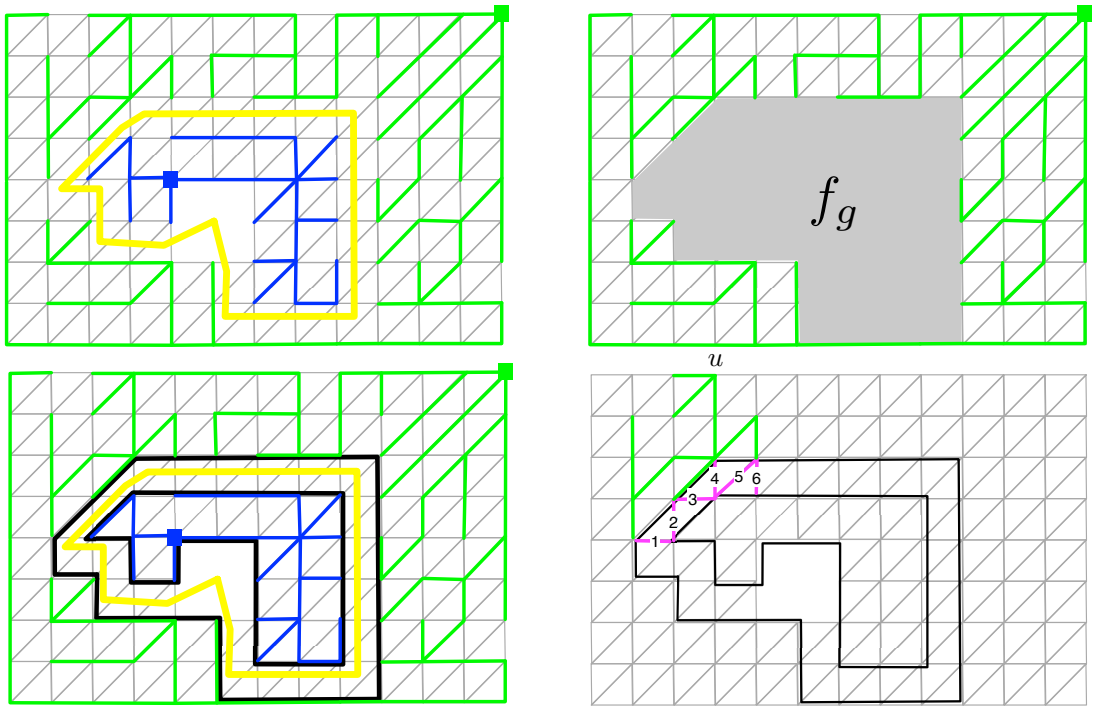

Figure 6: Illustration of the proof of Lemma 4.4. Top left: A graph P with two sites g, b. The bisector $\beta^*(g,b)$ is shown in yellow. The subtrees of T_g and T_b spanning the nodes of $\mathrm{Vor}(g)$ and $\mathrm{Vor}(b)$ are shown in green and blue, respectively. Top right: The Voronoi region $\mathrm{Vor}(g)$ is shown. The face f_g is grey. Bottom left: The cycles β_g and β_b corresponding to the boundaries of f_g in $\mathrm{Vor}(g)$ and of f_b in $\mathrm{Vor}(b)$, respectively, are indicated in black. Bottom right: A node $u \in \mathrm{Vor}(g)$ is indicated along with the subtree of T_g rooted at u. The primal edges of R^*_u are shown in magenta. Their relative order with respect to the labels $\mathrm{pre}_g(\cdot)$ is indicated.

R^*_v has $\Delta^r(e^*) \geq x$, and all edges e^* with $\Delta^r(e^*) = x$ belong to R^*_v. Let e^*_1, and e^*_2 be the first and last edges of R^*_v, respectively. By Lemma 4.4, e^*_1 and e^*_2 are also the edges with minimum and maximum values of $\mathrm{pre}_c(\cdot)$ in R^*_v, respectively. It follows that at least one of $\Delta^r(e^*_1), \Delta^r(e^*_2)$ is x (and the other is $\geq x$), and that, moreover, either e^*_1 or e^*_2 is an extreme edge in the maximal interval of edges $Q^*_{\geq x}$ of $\beta^*(g,b)$.

Exploiting these properties, we design the following procedure GetInterval($\hat{e}^*$). The input is an edge $\hat{e}^* \in \beta^*(g,b)$ with $\Delta^r(\hat{e}^*) = x$. The output is an interval I of extreme edges e^* of $Q^*_{\geq x}$ such that (1) either the first or the last edge of I is of value x, (2) I contains all edges of value x in $\beta^*(g,b)$ (and $\hat{e}^*$ in particular).

GetInterval($\hat{e}^*$)

1. Let $\hat{e} = vw$ be the primal edge of $\hat{e}^*$. Find the endpoint of $\hat{e}$ whose $\Delta^r(\cdot)$ value is x (Lemma 4.5 implies that there is only one such endpoint). Suppose, without loss of generality, that $\Delta^r(v) = x$.
2. Find the Voronoi cell to which v belongs in $VD(\{g,b\}, \{\omega(g), \omega(b)\})$. Let c be the corresponding site. So $\Delta^r(v) = \tilde{\delta}^{rc}(v) = x$.
3. Find the ancestor u of v in T_c that is nearest to the root, such that $\tilde{\delta}^{rc}(u) = x$. Note that the node u is an endpoint of the unique tense edge at critical value x. Finding u can be done by binary search on the root-to-v path in T_c since, by Lemma 4.5, all the ancestors of u on this path have strictly smaller $\tilde{\delta}^{rc}$-values.
4. Let p be the parent of u in T_c. Return the interval that starts at the successor of $\mathrm{pre}_c(pu)$ and ends at the predecessor of $\mathrm{pre}_c(up)$ in $\beta^*(g,b)$ (with respect to the $\mathrm{pre}_c(\cdot)$ numbering of the edges of $\beta^*(g,b)$). Since, by Lemma 4.4, the cyclic order on $\beta^*(g,b)$ is consistent with $\mathrm{pre}_c(\cdot)$, we can find these successor and predecessor by using $\mathrm{pre}_c(\cdot)$ as an additional key in the search tree representing $\beta^*(g,b)$.

To efficiently implement GetInterval($\hat{e}^*$), we retrieve and compare the distances from r, g, and b to v

and w. This gives $\Delta^r(v)$, $\Delta^r(w)$, and thereby $\Delta^r(\hat{e})$. It also reveals the Voronoi cell in $VD(\{g,b\},\{\omega(g),\omega(b)\})$ containing v. All this is done easily in $O(1)$ time. To carry out the binary search to find the ancestor u of v in T_c that is nearest to the root, such that $\tilde{\delta}^{rc}(u) = x$, we use a level ancestor data structure on T_c which we prepare during preprocessing. Each query to this data structure takes $O(1)$ time.

Using the weak bitonicity of $\mathbf{\Delta}^r_\beta$ (Corollary 4.2), the procedeure GetInterval, and the procedure for finding $e^*_{\max}$ described in Section 4.3, we can find the trichromatic faces that are dual to the Voronoi vertices of $VD^*(\{r,g,b\})$, under the respective weights $\omega(r),\omega(g),\omega(b)$ by the variation of binary search described before.

Several additional enhancements of the preprocessing stage are needed to support an efficient implementation of this procedure. First, we need to store T_c for each individual site c. Second, for each bisector $\beta^*(c_1,c_2)$ we need to store in its persistent search tree representation two secondary keys $\mathrm{pre}_{c_1}(\cdot)$ and $\mathrm{pre}_{c_2}(\cdot)$. As we discussed above, these keys are consistent with the cyclic order of $\beta^*(c_1,c_2)$. Clearly, all these enhancements do not increase the preprocessing time asymptotically. We have thus established Theorem 4.1.

The proof of Theorem 4.2 builds upon that of Theorem 4.1. Some additional preprocessing is required to handle a set of nearly consecutive sites B instead of a single site b, as well as some changes due to the weaker structure of $\beta^*(g,B)$. The details will appear in the full version.

5 Additional structure of Voronoi diagrams

In this section we study the structural properties that will be used in the fast construction of a Voronoi diagram from the precomputed bisectors (Section 6). We first provide the deferred proof of Lemma 3.1, and then continue with additional properties and terminology.

Lemma 3.1. *Consider some critical value δ of $\omega(v)-\omega(u)$ where $\beta^*(u,v)$ changes. The dual edges that newly join $\beta^*(u,v)$ at δ form a contiguous portion of the new bisector, and the dual edges that leave $\beta^*(u,v)$ at δ form a contiguous portion of the old bisector.*

Proof. Let $\mathrm{Vor}^-(u)$ (resp., $\mathrm{Vor}^+(v)$) be the primal Voronoi cell of u right before (resp., after) δ. Let yz be the tense edge that triggers the switch, so z is the root of the subtree that moves from $\mathrm{Vor}^-(u)$ to $\mathrm{Vor}^+(v)$ at δ. Let $\beta^{*-}(u,v)$ and $\beta^{*+}(u,v)$ denote, respectively, the uv-bisector immediately before and after δ. By Lemma 4.4 the preorder numbers (in T_u, say) of the edges along $\beta^{*-}(u,v)$ are monotonically increasing and therefore the arcs of $\beta^{*-}(u,v)$ whose tails are in the subtree of z must form a continuous portion of $\beta^{*-}(u,v)$. An analogous argument applies to the arcs of $\beta^{*+}(u,v)$ whose heads are in the subtree of z. □

Lemma 5.1. *Let f^* and g^* be two Voronoi vertices of $VD^*(S)$, which are consecutive on the common boundary between the cells $\mathrm{Vor}^*(u)$ and $\mathrm{Vor}^*(v)$. Then the path between f^* and g^* along this boundary in $VD^*(S)$ is a (connected) segment of $\beta^*(u,v)$.*

We omit the (trivial) proof. We next generalize the notion of bisectors to sets of sites. Let h_g, h_b be (not necessarily distinct) holes of P. Let $G \subset S$ be a set of "green" sites incident to h_g and let $B \subset S$ be a set of "blue" sites incident to h_b; when $h_g = h_b$ we require that G and B be separated along ∂h_g. Define $\beta^*(G,B)$ to be the set of edges of P^* whose corresponding primal arcs have their tail in $\mathrm{Vor}(g_i)$ and head in $\mathrm{Vor}(b_j)$, for some $g_i \in G$, and $b_j \in B$.

Lemma 5.2. *If $h_g \neq h_b$, or if $h_g = h_b$, and the sets G,B are separated along the boundary of h_g, then $\beta^*(G,B)$ is a non-self-crossing cycle of arcs of P^*. If $|G| > 1$ (resp., $|B| > 1$) then h^*_g (resp., h^*_b) may have degree greater than 2 in $\beta^*(G,B)$. All other dual vertices have degree 0 or 2 in $\beta^*(G,B)$. Furthermore, if $h_g = h_b$ and if $\beta^*(G,B)$ is nonempty, then $\beta^*(G,B)$ contains h^*_g (possibly multiple times).*

Proof. Embed a "super-green" vertex g inside h_g, and a "super-blue" vertex b inside h_b, and assign weight 0 to both g and b. This can be done without violating planarity also when $h_g = h_b$, since in this case G,B are separated along ∂h_g. Connect h_g (resp., h_b) to the green (resp., blue) sites with arcs gg_i (resp., bb_i) of weight $\omega(g_i)$ (resp., $\omega(b_i)$). By Lemma 2.4, the bisector $\beta^*(g,b)$ is a simple cycle in this auxiliary graph. However, this cycle can go through the artificial faces created inside the holes h_g, h_b. Deleting the artificial arcs in the primal is equivalent to contracting them in the dual, which contracts all the artificial faces into the dual faces h^*_g and h^*_b. This contraction turns $\beta^*(g,b)$ into $\beta^*(G,B)$, as is easily checked, and might give rise to non-simplicities of $\beta^*(G,B)$ at h^*_g and h^*_b. See Figure 7.

The proof of the final property is identical to the one in Lemma 2.4. Namely, If $h_g = h_b$ and $\beta^*(G,B)$ is nonempty, then the simple cut defined by the partition $(\mathrm{Vor}(g),\mathrm{Vor}(b))$ must contain at least one arc e on the boundary of h_g. Therefore, e^* is an arc of $\beta^*(u,v)$ that is incident to h^*_g, and multiplicities can arise when there are several such arcs e. □

6 Computing the Voronoi diagram

In this section we describe an algorithm that, given access to the pre-computed representation of the bisec-

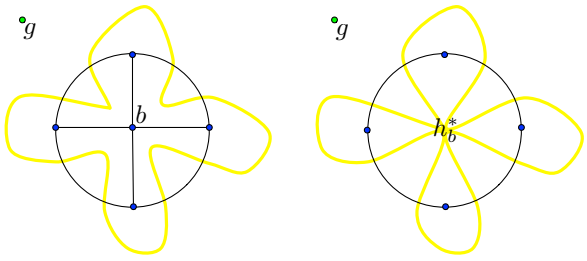

Figure 7: The structure of the bisector $\beta^*(G, B)$ when $|G| = 1$. Left: a set B of four blue sites on the hole h_b (black cycle). An artificial blue site b is embedded in h_b and connected to all blue sites. The bisector $\beta^*(g, b)$ is a simple cycle. Right: When removing the artificial arcs the bisector $\beta^*(G, B)$ visits h_b^* multiple times.

tors (Theorem 3.1), and the mechanism for computing trichromatic vertices provided by Theorem 4.2, computes $VD^*(S)$ in $\tilde{O}(|S|)$ time. Thus we establish all parts of Theorem 1.1, except for the mechanism for maximum queries in a Voronoi cell (point (ii) in Theorem 1.1), which is shown in Section 7. The presentation proceeds through several stages that handle the cases where the sites lie on the boundary of a single hole, of two holes, of three holes, and finally the general case. Due to space constraint we only include the description of the single hole case. The details will appear in the full version.

Representing the diagram. We represent $VD^*(S)$ by a reduced graph in which we contract all the vertices of degree 2. That is, we replace each path $p^* = p_1^*, p_2^*, \ldots, p_l^* = q^*$ in $VD^*(S)$, where p^* and q^* are Voronoi vertices, and $p_2^*, p_3^*, \ldots, p_{l-1}^*$ are vertices of degree 2, by the single edge (p^*, q^*). We represent this reduced graph using the standard DCEL data structure for representing planar maps (see [12] for details). Note that, $VD^*(S)$ and its reduced representation may be disconnected, and may contain parallel edges and self loops. The standard DCEL data structure can represent disconnected and non-simple planar maps. By Lemma 5.1, the path $(p^* = p_1^*, p_2^*, \ldots, p_l^* = q^*)$ is a contiguous portion of some bisector $\beta^*(u_1, u_2)$. We store with the contracted edge (p^*, q^*) pointers that (a) identify the sites u_1, u_2 whose dual cells are adjacent to this edge, and (b) point to the first and the last edges of this portion, namely to the edges (p^*, p_2^*) and (p_{l-1}^*, q^*).

The divide-and-conquer mechanism. Assume that all sites lie on a single face (hole) h_1 of the graph P, and let h_1^* denote its dual vertex. Note that here we only assume that the sites of the diagram are on a single hole, not that there is just a single hole (non-triangulated face of P). We describe a divide-and-conquer algorithm for constructing $VD^*(S)$ in $\tilde{O}(|S|)$ time. Note that in this case the diagram does not contain an isolated loop; that is, its edges form a connected graph (see below for a more precise statement and justification).

We partition the set S of sites into two contiguous subsets of (roughly) the same size $k \approx |S|/2$. Each subset is consecutive in the cyclic order around ∂h_1; to simplify the notation, we assume that they have exactly the same size. The sites in one subset, Y, denoted as $y_1, \ldots, y_k$ in their clockwise order around h_1, are referred to as the *yellow* sites, and those of the other subset O, denoted as $o_1, \ldots, o_k$ in their clockwise order around h_1, are the *orange* sites. If $k > 3$ we recursively compute the Voronoi diagram of Y, denoted as $VD^*(Y)$, and the Voronoi diagram of O, denoted as $VD^*(O)$. If $k = 3$ then we compute $VD^*(Y)$ (resp., $VD^*(O)$) using the algorithm of Section 4 and if $k = 2$, $VD^*(Y)$ (resp., $VD^*(O)$) is the bisector of the two sites in Y (resp., O). We now describe how to merge these two diagrams and obtain $VD^*(S)$, in $\tilde{O}(|S|)$ time.

Constructing $\boldsymbol{\beta^*(Y, O)}$. To merge $VD^*(Y)$ and $VD^*(O)$, we have to identify all segments of the bisectors $\beta^*(y, o)$, for $y \in Y$ and $o \in O$, that belong to $VD^*(S)$. Recall that we denoted the subgraph of $VD^*(S)$ induced by the edges of these segments by $\beta^*(Y, O)$. The structure of $\beta^*(Y, O)$ is specified by Lemma 5.2 (this is the case where $h_g = h_b$ in the statement of the lemma).

Our algorithm consists of two main stages. In the first stage we identify the edges of $\beta^*(Y, O)$ that are incident to h_1^*; that is, edges of $\beta^*(Y, O)$ that are dual to edges on ∂h_1. We denote this subset of $\beta^*(Y, O)$ by $\beta_I^*(Y, O)$. In the second stage we compute the paths that comprise $\beta^*(Y, O)$, each of which connects a pair of edges of $\beta_I^*(Y, O)$, as obtained in Stage 1. Each such path is a concatenation of contiguous portions of "bichromatic" bisectors of the form $\beta^*(y_i, o_j)$.

Stage 1. We partition ∂h_1 into segments by cutting off its edges whose duals are in $VD^*(Y) \cup VD^*(O)$. The vertices in each such segment belong to a single Voronoi cell $\mathrm{Vor}(y_i)$ of $VD(Y)$, and to a single Voronoi cell $\mathrm{Vor}(o_j)$ of $VD(O)$, and we then denote this segment by A_{ij}. Consider the bisector $\beta^*(y_i, o_j)$. It is a simple cycle incident to h_1^*, so it has at most two edges that cross ∂h_1. These edges are stored with $\beta^*(y_i, o_j)$ when it is constructed by the sweeping procedure at preprocessing. For each segment A_{ij}, we check whether one of the edges, e^*, of $\beta^*(y_i, o_j)$ that is incident to h_1^* is dual to an edge in A_{ij}. This can be done by assigning to each edge a number according to the cyclic walk along h_1, and checking if the number assigned to e is between

the numbers assigned to the endpoints of A_{ij}. If we find such an edge e^*, it belongs to $\beta_I^*(Y,O)$, and we add it to this set.

Another way in which an edge e^* can belong to $\beta_I^*(Y,O)$ is when it is a delimiter between two consecutive segments A_{ij} and $A_{i'j'}$ (where we have $i = i'$ or $j = j'$, but not both). Let $e = xx'$ be the primal edge of ∂h dual to e^*, with $x \in A_{ij}$ and $x' \in A_{i'j'}$. In this case, the site in $Y \cup O$ nearest to x must be either y_i or o_j, and the site nearest to x' must be either $y_{i'}$ or $o_{j'}$. Then e^* is an edge of $\beta_I^*(Y,O)$ if and only if the site nearest to x and the site nearest to x' are of different colors (a test that we can perform in constant time).

Since the preceding arguments exhaust all possibilities, we obtain the following lemma that asserts the correctness of this stage.

LEMMA 6.1. *The procedure described above identifies all edges of $\beta_I^*(Y,O)$.*

Stage 2. By Lemma 5.2, $\beta^*(Y,O)$ is a union of edge-disjoint simple cycles, all passing through h_1^*; we refer to these sub-cycles simply as cycles. The set $\beta_I^*(Y,O)$ produced in Stage 1 consists of the edges incident to h_1^* on each of these cycles. Moreover, as already argued, each of these cycles contains exactly two such edges. Saying it slightly differently (with a bit of notation abuse) ignoring h_1^*, each of these cycles is a simple path in P^* that starts and ends at a pair of respective edges of $\beta_I^*(Y,O)$, and otherwise does not meet ∂h_1. Stage 2 constructs these paths one at a time, picking an edge e_1^* of $\beta_I^*(Y,O)$, and tracing the path γ that starts at e_1^* until it reaches ∂h_1 again, at another "matched" edge $e_2^* \in \beta_I^*(Y,O)$; the stage repeats this tracing step until all the edges of $\beta_I^*(Y,O)$ are exhausted.

We assume that each of $VD^*(Y)$ and $VD^*(O)$ has at least two non-empty cells. The procedure for the other cases is a degenerate version of the one we describe. We compute the path $\gamma \in \beta^*(Y,O)$ containing an initial edge $e_1^* \in \beta_I^*(Y,O)$ as follows. Let $\beta^*(y_i,o_j)$ be the bisector that contains e_1^* in $\beta^*(Y,O)$. It follows that e_1^* is either contained in, or lies on the boundary of $\mathrm{Vor}^*(y_i)$ in $VD^*(Y)$, and the same holds for $\mathrm{Vor}^*(o_j)$ in $VD^*(O)$. We use Theorem 4.2 with $o_j, y_i, Y \setminus \{y_i\}$ to find the (one or two) trichromatic vertices of $VD^*(o_j, y_i, Y \setminus \{y_i\})$. Note that the theorem applies since $Y \setminus \{y_j\}$ is a nearly consecutive set of sites. Similarly, we use Theorem 4.2 with $y_i, o_j, O \setminus \{o_j\}$ to find the (one or two) trichromatic vertices of $VD^*(y_i, o_j, O \setminus \{o_j\})$. Note that h_1^* is a trichromatic vertex in both diagrams, so it is always one of the vertices returned by the algorithm of Theorem 4.2.

Let I_1 be the segment of the bisector $\beta^*(y_i,o_j)$ between the two trichromatic vertices of $VD^*(o_j, y_i, Y \setminus \{y_i\})$ that contains e_1^* (or all of $\beta^*(y_i,o_j)$ in case there is just one trichromatic vertex). Let I_2 be the segment of the bisector $\beta^*(y_i,o_j)$ between the two trichromatic vertices of $VD^*(y_i, o_j, O \setminus \{o_j\})$ that contains e_1^* (or all of $\beta^*(y_i,o_j)$ in case there is just one trichromatic vertex).

Let I be the shorter of the segments I_1 and I_2. The segment I of $\beta^*(y_i,o_j)$ is a bichromatic Voronoi edge (of the final diagram) in $\beta^*(Y,O)$. If I is the entire $\beta^*(y_i,o_j)$ then $\gamma = I$ and we are done. Otherwise, each endpoint of I is a Voronoi vertex of $VD^*(S)$ adjacent to the cells $\mathrm{Vor}(y_i)$, $\mathrm{Vor}(o_j)$, and to a third cell which is either $\mathrm{Vor}(y_{i'})$ for some yellow site $y_{i'} \neq y_i$, or $\mathrm{Vor}(o_{j'})$ for some orange site $o_{j'} \neq o_j$. Let g^* be the endpoint of I different from h_1^*. Assume for concreteness that g^* is adjacent in $VD^*(S)$ to $\mathrm{Vor}(y_i)$, $\mathrm{Vor}(o_j)$, and to some other $\mathrm{Vor}(y_{i'})$. Let e_2^* be the edge of $\beta^*(y_{i'},o_j)$ incident to g^* that is on the boundary of $\mathrm{Vor}(y_{i'})$ in $VD^*(S)$ (the third edge adjacent to g^* in $VD^*(S)$ is an edge of $\beta^*(y_i,y_{i'})$ from $VD^*(Y)$, and is not part of $\beta^*(Y,O)$). We continue tracing γ by repeating the above procedure with g^* taking the role of h_1^*, and e_2^* taking the role of e_1^*. We continue tracing γ in this manner, identifying the Voronoi edges on γ one by one, until we get back to h_1^* through an edge $a^* \in \beta_I^*(Y,O)$. We then discard e_1^* and a^* from $\beta_I^*(Y,O)$, and keep performing this tracing procedure from another edge of $\beta_I^*(Y,O)$, until all edges of $\beta_I^*(Y,O)$ are exhausted, obtaining in this way all the cycles of $\beta^*(Y,O)$.

Wrap-up. Once we have computed the cycles of $\beta^*(Y,O)$, we can assemble the entire Voronoi diagram $VD^*(S)$ by pasting parts of $VD^*(Y)$ and $VD^*(O)$ to $\beta^*(Y,O)$ as follows. Consider the bisector $\beta^*(Y,O)$ as a cycle in the plane. It intersects the embedding of $VD^*(Y)$ at the endpoints of segments of $\beta^*(Y,O)$ (each such intersection is either a Voronoi vertex of $VD^*(Y)$ or a degree 2 vertex on some Voronoi edge of $VD^*(Y)$). We use the DCEL representation to cut $VD^*(Y)$ along $\beta^*(Y,O)$, keeping just parts of $VD^*(Y)$ that belong to the Y side of $\beta^*(Y,O)$. We repeat the process for $VD^*(O)$, and then glue together $\beta^*(Y,O)$ and the parts we kept from $VD^*(Y)$ and $VD^*(O)$. This is done by identifying the endpoints of segments of $\beta^*(Y,O)$ that appear in multiple parts.

Running time. Each call to Theorem 4.2 takes $\tilde{O}(1)$ time, and the number of calls is proportional to the overall complexity of $VD(Y)$, $VD(O)$, and $VD(S)$, which is $O(|S|)$. It follows that the overall cost of the divide-and-conquer mechanism is $\tilde{O}(|S|)$.

7 Preprocessing for max queries

In this section we establish the last part (point (ii)) of Theorem 1.1. Recall that P contains r vertices and b

sites. We wish to preprocess P (independently of any weight assignment to the sites) in $\tilde{O}(rb^2)$ time, so that, given the cell $\mathrm{Vor}^*(v)$ of a site v, we can return the vertex $w \in \mathrm{Vor}(v)$ maximizing $d(v, w)$ in time that is linear (up to polylogarithmic factors) in the number of Voronoi vertices of $\mathrm{Vor}^*(v)$. Cabello described a similar mechanism, but since in his Voronoi diagrams the sites are on a single hole, VD^* is connected, so the boundary of each Voronoi region is a single cycle. As our treatment is more general, we need to handle the case of sites on multiple holes, where the boundary of Voronoi cells might consist of multiple cycles. Our approach for the single hole case is essentially that of Cabello, although our presentation is different. We then extend the technique to handle sites on multiple holes. Due to space constraints the details will appear in the full version.

References

[1] A. Abboud and S. Dahlgaard. Popular conjectures as a barrier for dynamic planar graph algorithms. In *FOCS*, pages 476–486, 2016.

[2] A. Abboud, F. Grandoni, and V. V. Williams. Subcubic equivalences between graph centrality problems, APSP and diameter. In *SODA*, pages 1681–1697, 2015.

[3] A. Abboud and K. Lewi. Exact weight subgraphs and the k-SUM conjecture. In *ICALP*, pages 1–12, 2013.

[4] A. Abboud and V. V. Williams. Popular conjectures imply strong lower bounds for dynamic problems. In *FOCS*, pages 434–443, 2014.

[5] A. Abboud, V. V. Williams, and J. R. Wang. Approximation and fixed parameter subquadratic algorithms for radius and diameter in sparse graphs. In *SODA*, pages 377–391, 2016.

[6] A. Abboud, V. V. Williams, and H. Yu. Matching triangles and basing hardness on an extremely popular conjecture. In *STOC*, pages 41–50, 2015.

[7] A. Aggarwal, M. M. Klawe, S. Moran, P. Shor, and R. Wilber. Geometric applications of a matrix-searching algorithm. *Algorithmica*, 2(1):195–208, 1987.

[8] F. Aurenhammer. Voronoi diagrams - A survey of a fundamental geometric data structure. *ACM Comput. Surv.*, 23(3):345–405, 1991.

[9] A. Backurs, N. Dikkala, and C. Tzamos. Tight hardness results for maximum weight rectangles. In *ICALP*, pages 81:1–81:13, 2016.

[10] A. Backurs and C. Tzamos. Improving Viterbi is hard: Better runtimes imply faster clique algorithms. *Arxiv 1607.04229*, 2016.

[11] B. Ben-Moshe, B. K. Bhattacharya, Q. Shi, and A. Tamir. Efficient algorithms for center problems in cactus networks. *Theor. Comput. Sci.*, 378(3):237–252, 2007.

[12] M. d. Berg, O. Cheong, M. v. Kreveld, and M. Overmars. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, 3rd edition edition, 2008.

[13] K. Bringmann, P. Gawrychowski, S. Mozes, and O. Weimann. Tree edit distance cannot be computed in strongly subcubic time (unless apsp can). In *SODA*, 2018.

[14] S. Cabello. Subquadratic algorithms for the diameter and the sum of pairwise distances in planar graphs. In *SODA*, pages 2143–2152, 2017. Full version in *Arxiv 1702.07815*.

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

[16] T. M. Chan. All-pairs shortest paths for unweighted undirected graphs in *o(mn)* time. In *SODA*, pages 514–523, 2006.

[17] T. M. Chan. All-pairs shortest paths with real weights in $O(n^3/\log n)$ time. *Algorithmica*, 50(2):236–243, 2008.

[18] T. M. Chan. More algorithms for all-pairs shortest paths in weighted graphs. *SIAM J. Comput.*, 39(5):2075–2089, 2010.

[19] V. Chepoi and F. F. Dragan. A linear-time algorithm for finding a central vertex of a chordal graph. In *ESA*, pages 159–170, 1994.

[20] V. Chepoi, F. F. Dragan, B. Estellon, M. Habib, and Y. Vaxès. Diameters, centers, and approximating trees of delta-hyperbolicgeodesic spaces and graphs. In *SoCG*, pages 59–68, 2008.

[21] V. Chepoi, F. F. Dragan, and Y. Vaxès. Center and diameter problems in plane triangulations and quadrangulations. In *SODA*, pages 346–355, 2002.

[22] K. Clarkson and P. Shor. Applications of random sampling in computational geometry, ii. *Discrete & Computational Geometry*, 4(5):387–421, 1989.

[23] V. Cohen-Addad, S. Dahlgaard, and C. Wulff-Nilsen. Fast and compact exact distance oracle for planar graphs. In *FOCS*, pages 962–973, 2017.

[24] É. C. de Verdière. Shortest cut graph of a surface with prescribed vertex set. In *ESA*, pages 100–111.

[25] W. Dobosiewicz. A more efficient algorithm for the min-plus multiplication. *International Journal of Computer Mathematics*, 32(1-2):49–60, 1990.

[26] F. F. Dragan and F. Nicolai. Lexbfs-orderings of distance-hereditary graphs with application to the diametral pair problem. *Discrete Applied Mathematics*, 98(3):191–207, 2000.

[27] D. Eppstein. Subgraph isomorphism in planar graphs and related problems. In *SODA*, pages 632–640, 1995.

[28] 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.

[29] A. M. Farley and A. Proskurowski. Computation of the center and diameter of outerplanar graphs. *Discrete Applied Mathematics*, 2(3):185–191, 1980.

[30] R. Floyd. Algorithm 97: Shortest path. *Commun. ACM*, 5(6):345, 1962.

[31] G. N. Frederickson. Fast algorithms for shortest paths

in planar graphs, with applications. *SIAM J. Comput.*, 16(6):1004–1022, 1987.
[32] M. Fredman. New bounds on the complexity of the shortest path problem. *SIAM J. Comput.*, 5(1):83–89, 1976.
[33] P. Gawrychowski, S. Mozes, O. Weimann, and C. Wulff-Nilsen. Better tradeoffs for exact distance oracles in planar graphs. In *SODA*, 2018.
[34] Y. Han. Improved algorithm for all pairs shortest paths. *Information Processing Letters*, 91(5):245–250, 2004.
[35] Y. Han. An $O(n^3(\log\log n/\log n)^{5/4})$ time algorithm for all pairs shortest paths. *Algorithmica*, 51(4):428–434, 2008.
[36] Y. Han and T. Takaoka. An $O(n^3\log\log n/\log^2 n)$time algorithm for all pairs shortest paths. In *SWAT*, pages 131–141, 2012.
[37] D. Hartvigsen and R. Mardon. The all-pairs min cut problem and the minimum cycle basis problem on planar graphs. *Journal of Discrete Mathematics*, 7(3):403–418, 1994.
[38] M. R. Henzinger, P. N. Klein, S. Rao, and S. Subramanian. Faster shortest-path algorithms for planar graphs. *Journal of Computer and System Sciences*, 55(1):3–23, 1997.
[39] J. Hershberger and S. Suri. Matrix searching with the shortest-path metric. *SIAM J. Comput.*, 26(6):1612–1634, 1997.
[40] T. Husfeldt. Computing graph distances parameterized by treewidth and diameter. In *IPEC*, pages 16:1–16:11, 2016.
[41] R. Impagliazzo, R. Paturi, and F. Zane. Which problems have strongly exponential complexity? *Journal of Computer and System Sciences*, 63(4):512–530, 2001.
[42] P. Klein and S. Mozes. Optimization algorithms for planar graphs. http://planarity.org. Book draft.
[43] P. N. Klein. Multiple-source shortest paths in planar graphs. In *SODA*, pages 146–155, 2005.
[44] P. N. Klein, S. Mozes, and C. Sommer. Structured recursive separator decompositions for planar graphs in linear time. In *STOC*, pages 505–514, 2013.
[45] R. Klein. *Concrete and Abstract Voronoi Diagrams*, volume 400 of *Lecture Notes in Computer Science*. Springer, 1989.
[46] R. Klein, E. Langetepe, and Z. Nilforoushan. Abstract voronoi diagrams revisited. *Comput. Geom.*, 42(9):885–902, 2009.
[47] R. Klein, K. Mehlhorn, and S. Meiser. Randomized incremental construction of abstract voronoi diagrams. *Comput. Geom.*, 3:157–184, 1993.
[48] D. Marx and M. Pilipczuk. Optimal parameterized algorithms for planar facility location problems using voronoi diagrams. In *ESA*, pages 865–877, 2015.
[49] S. Olariu. A simple linear-time algorithm for computing the center of an interval graph. *International Journal of Computer Mathematics*, 34:121–128, 1990.
[50] L. Roditty and V. V. Williams. Fast approximation algorithms for the diameter and radius of sparse graphs. In *STOC*, pages 515–524, 2913.
[51] L. Roditty and U. Zwick. On dynamic shortest paths problems. *Algorithmica*, 61(2):389–401, 2011.
[52] N. Sarnak and R. E. Tarjan. Planar point location using persistent search trees. *Commun. ACM*, 29(7):669–679, 1986.
[53] M. I. Shamos and D. Hoey. Closest-point problems. In *SFCS*, pages 151–162, 1975.
[54] T. Takaoka. A new upper bound on the complexity of the all pairs shortest path problem. *Information Processing Letters*, 43(4):195–199, 1992.
[55] T. Takaoka. An $O(n^3\log\log n/\log n)$ time algorithm for the all-pairs shortest path problem. *Information Processing Letters*, 96(5):155–161, 2005.
[56] S. Warshall. A theorem on boolean matrices. *J. ACM*, 9(1):11–12, 1962.
[57] R. Williams. Faster all-pairs shortest paths via circuit complexity. In *STOC*, pages 664–673, 2014.
[58] V. V. Williams and R. Williams. Subcubic equivalences between path, matrix and triangle problems. In *FOCS*, pages 645–654, 2010.
[59] V. V. Williams and R. Williams. Finding, minimizing, and counting weighted subgraphs. *SIAM J. Comput.*, 42(3):831–854, 2013.
[60] C. Wulff-Nilsen. Wiener index and diameter of a planar graph in subquadratic time. Technical report, 08-16, Department of Computer Science, University of Copenhagen, 2008. Available at http://www.diku.dk/OLD/publikationer/tekniske.rapporter/rapporter/08-16.pdf. Preliminary version in EurCG 2009.
[61] C. Wulff-Nilsen. Wiener index, diameter, and stretch factor of a weighted planar graph in subquadratic time, 2010. Paper M in the PhD thesis of C. Wulff-Nilsen, available at http://www.diku.dk/forskning/phd-studiet/phd/ThesisChristian.pdf.
[62] U. Zwick. A slightly improved sub-cubic algorithm for the all pairs shortest paths problem with real edge lengths. In *ISAAC*, pages 921–932, 2004.