Noname manuscript No.
(will be inserted by the editor)

Faster Approximate Diameter and Distance Oracles in Planar Graphs

Timothy M. Chan · Dimitrios Skrepetos

the date of receipt and acceptance should be inserted later

Abstract We present an $O\left(n\log^2 n + (1/\varepsilon)^5 n\log n\right)$-time algorithm that computes a $(1+\varepsilon)$-approximation of the *diameter* of a non-negatively-weighted, undirected planar graph of n vertices. This is an improvement over the algorithm of Weimann and Yuster [ICALP 2012] of $O\left((1/\varepsilon)^4 n\log^4 n + 2^{O(1/\varepsilon)}n\right)$ time in two regards. First we eliminate the exponential dependency on $1/\varepsilon$ by adapting and specializing Cabello's recent Voronoi-diagram-based technique [SODA 2017] for approximation purposes. Second we shave off two logarithmic factors by choosing a better sequence of error parameters in the recursion.

Moreover, using similar techniques we obtain a variant of Gu and Xu's $(1+\varepsilon)$-approximate distance oracle [ISAAC 2015] with polynomial dependency on $1/\varepsilon$ in the preprocessing time and space and $O(\log(1/\varepsilon))$ query time.

1 Introduction

In this paper we study the approximate versions of two fundamental shortest-path problems in non-negatively-weighted, undirected planar graphs. The first problem is that of approximating the *diameter*, defined as the longest shortest-path distance of any two vertices. The second problem is that of constructing

Timothy M. Chan
Department of Computer Science, University of Illinois at Urbana-Champaign `tmc@illinois.edu`

Dimitrios Skrepetos
Cheriton School of Computer Science, University of Waterloo `dskrepet@uwaterloo.ca`

The conference version of this paper appeared in ESA 2017 [7]. There are two differences between the conference and the journal versions of the paper. First the running time of our diameter algorithm is now deterministic. Second the correct query time of our $(1+\varepsilon)$-approximate distance oracles is $O(\log(1/\varepsilon))$ instead of $O(1)$ as we earlier claimed.

Work of the first author has been supported in part by NSF Grant CCF-1814026.

approximate *distance oracles*, i.e., data structures that support the following kind of queries: given any two vertices, return an approximation of their shortest-path distance. Let n be the number of vertices of the discussed graphs.

Diameter. Ever since Frederickson [12] in 1983 solved the problem exactly in $O\left(n^2\right)$ time (with his all-pairs shortest paths (APSP) algorithm), a natural question arose as to whether the diameter can be computed in subquadratic time. Eppstein [11] in 1995 gave a partial answer in unweighted planar graphs with a linear-time algorithm for the special case of the diameter being a constant. Chan [6] in 2006 and Wulff-Nilsen [30] in 2008 developed two slightly-subquadratic-time solutions (for arbitrary diameter), both of them requiring $O\left(n^2 \cdot \frac{\log\log n}{\log n}\right)$ time in unweighted planar graphs. The algorithm of Wulff-Nilsen also works for the weighted case but in $O\left(n^2 \cdot \frac{(\log\log n)^4}{\log n}\right)$ time. However a truly-subquadratic-time algorithm, i.e., one that runs in $O\left(n^{2-\delta}\right)$ time for some constant $\delta > 0$, still eluded researchers for many years, thus leading them to consider approximation algorithms.

A c-approximation of the diameter Δ is a value $\widetilde{\Delta}$ with $\Delta \leq \widetilde{\Delta} \leq c\Delta$. The linear-time single-source shortest paths (SSSP) algorithm of Henzinger et al. [17] can trivially compute a 2-approximation. In 2007 Berman and Kasiviswanathan [2] gave a $(3/2)$-approximation algorithm of $\widetilde{O}\left(n^{3/2}\right)$ time, where $\widetilde{O}\left(f(n)\right)$ denotes $O\left(f(n)\log^{O(1)} n\right)$. In 2012, Weimann and Yuster [29] in a breakthrough gave a $(1+\varepsilon)$-approximation algorithm with a near linear time of $O\left((1/\varepsilon)^4 n \log^4 n + 2^{O(1/\varepsilon)} n\right)$. However, Weimann and Yuster's result did not settle the problem completely, as the running time has *exponential* dependency on $1/\varepsilon$, and moreover, there are multiple (four) $\log n$ factors.

Unexpectedly the next result came in the context of exact algorithms. In 2017, Cabello [4] (full paper in [5]) made a breakthrough with an $\widetilde{O}\left(n^{11/6}\right)$-expected-time algorithm. His techniques are as interesting as the result itself, for they involved a seemingly alien concept to planar graphs, *Voronoi diagrams*, originating from computational geometry. Gawrychowski et al. [14], again using Voronoi diagrams, derandomized Cabello's algorithm and improved its running time to $\widetilde{O}\left(n^{5/3}\right)$. No lower bound is available presently, but Cabello [5] conjectured that the diameter cannot be computed exactly in time faster than $O\left(n^{1+\delta}\right)$ for some constant $\delta > 0$.

We show that Cabello's idea of employing Voronoi diagrams in planar graphs can be combined nicely with Weimann and Yuster's recursive scheme to get better approximation algorithms, eliminating the $2^{O(1/\varepsilon)}$ factor from the running time of the latter. Compared with Cabello's approach, which had to deal with the general case of site weights being real numbers, our version of Voronoi diagrams is much simplified because in the approximate setting we can map the site weights to small integers. We also eliminate two of the four logarithmic factors by using a better sequence of error parameters in the recursion and by employing the multiple shortest paths data structure of Klein [21].

Theorem 1 (Approximate diameter) *We can find a $(1+\varepsilon)$-approximation of the diameter of a non-negatively-weighted, undirected planar graph of n vertices in $O\left(n\log^2 n + (1/\varepsilon)^5 n\log n\right)$ time.*

Distance oracles. A c-approximate distance oracle for a graph is a data structure that supports the following kind of query: given any two vertices at shortest-path distance d, return a value $\widetilde{d}$ with $d \le \widetilde{d} \le (1+\varepsilon)d$. The seminal paper of Thorup [28] presented $(1+\varepsilon)$-approximate distance oracles in non-negatively-weighted, undirected planar graphs with $O\left((1/\varepsilon)^2 n\log^3 n\right)$ preprocessing time, $O\left((1/\varepsilon)n\log n\right)$ space, and $O\left(1/\varepsilon\right)$ query time. These oracles were later simplified by Klein [20]. Kawarabayashi et al. [18] constructed an oracle of linear space but $O\left((1/\varepsilon)^2\log^2 n\right)$ query time, and then Kawarabayashi et al. [19] improved the dependency on $1/\varepsilon$ of the space-query-time product from $1/\varepsilon^2$ to $1/\varepsilon$. Wulff-Nilsen [31] described another trade-off, with $O\left(n(\log\log n)^2\right)$ space and $O\left((\log\log n)^3\right)$ query time, ignoring dependency on ε. In the word RAM model, Gu and Xu [16] combined the techniques of the above results with those of Weimann and Yuster [29] for the approximate diameter problem to obtain the first distance oracle with constant query time (independent of both n and ε). However, the preprocessing time and space of Gu and Xu's data structure have exponential dependency on $1/\varepsilon$.

We employ techniques similar to those of our diameter algorithm to develop in the word RAM model the first $(1+\varepsilon)$-approximate distance oracle with $o(1/\varepsilon)$ query time and $O\left((1/\varepsilon)^{O(1)}\, n\, \log^{O(1)} n\right)$ preprocessing time and space. Specifically, the preprocessing time and space of our oracle are $O\left(n\log^3 n + (1/\varepsilon)^5 n\log^2 n\right)$ and $O\left((1/\varepsilon)n\log^2 n + (1/\varepsilon)^{4+\delta} n\log n\right)$ respectively for any constant $\delta > 0$, and the query time is $O\left(\log\left(1/\varepsilon\right)\right)$. Although we slightly increase the query time of the oracle of Gu and Xu (from $O\left(1\right)$ to $O\left(\log\left(1/\varepsilon\right)\right)$), we significantly improve its preprocessing time and space by eliminating the exponential dependency on $1/\varepsilon$.

Theorem 2 (Approximate Distance Oracles) *We can construct a $(1+\varepsilon)$-approximate distance oracle for a non-negatively-weighted, undirected planar graph of n vertices with $O\left(n\log^3 n + (1/\varepsilon)^5 n\log^2 n\right)$ preprocessing time, $O\left((1/\varepsilon)n\log^2 n + (1/\varepsilon)^{4+\delta} n\log n\right)$ space, and $O\left(\log\left(1/\varepsilon\right)\right)$ query time in the word RAM model, for any constant $\delta > 0$.*

2 Preliminaries

2.1 Definitions and notations

Let $G = (V, E)$ be a non-negatively-weighted, undirected planar graph, i.e., a graph that can be drawn in the plane such that edges intersect only at their endpoints. We also refer to V and E as $V(G)$ and $E(G)$ respectively. Let G^* be the dual of G. We assume that G and any graph under discussion in this chapter comes with a fixed embedding (we can find such an embedding in

linear time [9,26]) and is triangulated (we can triangulate naively, in linear time).

For any $u, v \in V$, we denote the u-to-v shortest path in G by $\pi_G[u, v]$ and its length by $dist_G[u, v]$. We also refer to $dist_G[u, v]$ as shortest-path distance or simply distance. By $pred_G[s, t]$ we denote t's predecessor on $\pi_G[s, t]$. The shortest-path tree of each $u \in V$ is a spanning tree of G rooted at u such that the u-to-v shortest-path distance for each $v \in V$ in that tree corresponds to $dist_G[s, t]$.

We are interested in the following shortest-path problems in G.

- The problem of computing a $(1 + \varepsilon)$-approximation of the *diameter* $\Delta = \max_{u,v \in V} dist_G[u, v]$ of G, i.e. a value $\widetilde{\Delta}$ with $\Delta \leq \widetilde{\Delta} \leq (1 + \varepsilon)\Delta$. Similar problems are computing $(1 + \varepsilon)$-approximations of related parameters, such as the *radius* (i.e., $\min_{u \in V} \max_{v \in V} dist_G[u, v]$), the *eccentricity* of each vertex $u \in V$ (i.e., $\max_{v \in V} dist_G[u, v]$), et cetera.
- The problem of constructing $(1+\varepsilon)$-*approximate distance oracles*, i.e., data structures that support the following kind of queries: given any $s, t \in V$, compute a value $\widetilde{d}$ with $dist_G[s, t] \leq \widetilde{d} \leq (1 + \varepsilon) dist_G[s, t]$ and the predecessor of t in an s-to-t path of length $\widetilde{d}$.

2.2 Model of computation

Throughout the paper, except for Section 5, we use the standard (real) RAM model of computation. Specifically, we have random access to an array of words, each storing a real number, a $\Theta(\log n)$-bit integer (where n is the input size), or a pointer to another word. Moreover, we can perform any standard arithmetic operation, such as addition, subtraction, multiplication, division, and comparison, that involves a constant number of words in constant time.

In Section 5 we work in the word RAM model of computation, where the input values are assumed to be w-bit integers ($w \geq \log n$). We assume that standard arithmetic and bit-wise logical operations on w-bit integers take constant time.

3 A farthest-neighbor data structure

Here, we want to construct a data structure for the following *farthest-neighbor* problem in planar graphs, which is crucial in obtaining our diameter algorithm in Section 4. Let $[W] = \{0, 1, \ldots, W - 1\}$ for an integer $W > 0$.

Problem 1 (Farthest neighbor) *Let $H = (V, E)$ be a triangulated planar graph of n vertices with a fixed embedding. Also, let X be a set of b vertices on the boundary of its outer face, and let U be a subset of V. Finally, let H^+ be the graph that is obtained by adding to H a vertex z_0 and an edge of unspecified weight from z_0 to each $x \in X$.*

Construct a data structure that supports the following kind of queries: given a weight $\ell(\cdot)$ for each edge (z_0, x), where $x \in X$, find the distance to the farthest neighbor of z_0 in H^+ among the vertices of U, i.e., compute $\max_{u \in U} dist_{H^+}(z_0, u)$.

Cabello [5, Theorem 21] employed Voronoi diagrams in planar graphs to develop a farthest-neighbor data structure with $\widetilde{O}\left(n^2b^3 + b^4\right)$ preprocessing time and $\widetilde{O}\left(b \log b\right)$ expected query time, under a non-degeneracy assumption. His result works for the more general version of Problem 1 where the weight $\ell(\cdot)$ of each edge (z_0, x) $(x \in X)$ is a real number. We show that when these weights are instead small integers, we can employ Voronoi diagrams in a simpler way to construct a farthest-neighbor data structure in time only near linear in n. Moreover, our query time does not have any $\log n$ factors.

3.1 Defining Voronoi diagrams in planar graphs

The concepts of standard geometric Voronoi diagrams can be extended to the planar graph $H = (V, E)$ from the setting of Problem 1. Each *site* s is a pair (v_s, w_s), where v_s is its placement (i.e., a vertex of H), and w_s is its weight. Given a set of sites S, the *graphic Voronoi region* of $s \in S$ in H is defined as

$$\mathrm{VR}(s, S) \;=\; \{u \in V \mid dist_H[v_s, u] + w_s \leq dist_H[v_t, u] + w_t,\, \forall\, t \in S - \{s\}\},$$

i.e., as the set of all vertices closer to s than to any other site under the weighted metric. The (additively weighted) *graphic Voronoi diagram* $\mathrm{VD}(S)$ of S in H is simply the collection of $\mathrm{VR}(s, S)$ over all $s \in S$. See Figure 1(a)-(b). Since we discuss Voronoi diagrams only in H, we drop the subscripts and refer to $\mathrm{bis}(\cdot,\cdot)$, $\mathrm{VR}(\cdot,\cdot)$, and $\mathrm{VD}(\cdot)$ from now on.

Henceforth, we assume that S is a set of b sites, each placed at a vertex on the boundary of the infinite face of H. We assume that S is *generic*, i.e., for each $s, t \in S$ and $u \in V$, we have $dist_H[u, v_s] + w_s \neq dist_H[u, v_t] + w_t$. We also assume that S is *independent*, i.e., each region of the graphic Voronoi diagram of S is non-empty. Given $s, t \in S$, We define their *bisector* $\mathrm{bis}(s, t)$ to be the set of the duals of the edges in

$$\{(u, v) \in E \mid dist_H[u, v_s] + w_s < dist_H[u, v_t] + w_t \text{ and } dist_H[v, v_t] + w_t < dist_H[v, v_s] + w_s\}.$$

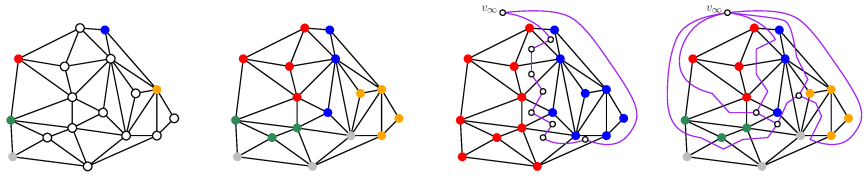
In other words, $\mathrm{bis}(s, t)$ is composed of the duals of the edges whose endpoints are not both closer to the same site. From [5, Lemma 10], each such bisector is a simple cycle in H^*. See Figure 1(c).

Klein [22] introduced the framework of *abstract Voronoi diagrams* to unify the treatment of various Voronoi diagrams in the plane. Cabello defined a system of *abstract bisectors* as

$$\left\{ \left(\mathrm{bis}(s, t), \left(\bigcup_{v \in \mathrm{VR}(s, \{s,t\})} \overline{v^*} \right)^{\circ} \right) \mid s, t \in S, s \neq t \right\},$$

where v^* is the dual face of $v \in V$, $\overline{A}$ is the closure of $A \subset \mathbb{R}^2$, and A° is the interior of A. Cabello showed [5, Lemma 12] that his system of abstract bisectors fulfills Klein's *admissibility* axioms, as long as S is generic and independent. Thus, the graphic Voronoi diagram of S in H can be implicitly represented as a planar graph with a fixed combinatorial embedding of $O(b)$ *Voronoi arcs* and *Voronoi nodes*. Notice that an explicit representation would require $O(n)$ space. Henceforth, every reference to a Voronoi diagram is to the implicit representation of the corresponding graphic Voronoi diagram in H. Each Voronoi arc in a such diagram is a contiguous portion of a bisector (thus, a simple path in H^*), and each Voronoi node is a vertex of H^* incident to three Voronoi arcs. See Figure 1(d).

We can compute the Voronoi diagram of S with the algorithm of Klein et al. [23, Theorem 1], which is an extension of a standard randomized incremental construction algorithm for Euclidean Voronoi diagrams [8,25]. The worst-case running time of that algorithm is quadratic. Also, each Voronoi arc (respectively, node) is represented with a pointer to a Voronoi arc (respectively, node) in the Voronoi diagram of four sites of S [5, Section 4]. We state the result of Klein et al. tailored for our setting.



(a) (b) (c) (d)

Fig. 1: (a) Five sites (the colored vertices) of a planar graph. (b) Their graphic Voronoi diagram. (c) A bisector of the blue and red sites. (d) The Voronoi diagram of the five sites.

Theorem 3 (Via abstract Voronoi diagrams) *Let H be a triangulated planar graph with a fixed embedding, and let S be a generic and independent set of b sites placed on vertices on the boundary of its outer face.*

We can construct the Voronoi diagram of S by a randomized algorithm using an expected number of $O(b \log b)$ elementary operations and $O(b \log b)$ additional time, or by a deterministic algorithm with using $O(b^2)$ elementary operations and $O(b^2)$ additional time. Here, an elementary operation is the computation of the Voronoi diagram of any four sites of S.

3.2 Constructing Voronoi diagrams in planar graphs

We now show how to perform the elementary operations of the Voronoi diagrams algorithm of Theorem 3, i.e., compute the Voronoi diagram of any

four sites of a given generic and independent set $S' \subseteq S$. For the rest of this section, we assume that each $s \in S'$ is assigned a weight drawn from $[W]$. We also assume that without loss of generality all subsets of at most four sites of S are generic and independent (we can easily find those that are not and ignore them).

As Cabello did, we first compute all possible bisectors for each pair of sites of S. Our approach builds upon that of [5, Lemma 17] and requires only $O(nW)$ time, whereas Cabello's needed $O(n^2)$ time for site weights being real numbers. Also, we can return a pointer to a bisector in $O(1)$ time instead of $O(\log n)$ time.

Lemma 1 (Bisectors) *Let H be a triangulated planar graph with a fixed embedding, and let S be a set of b sites placed on vertices on the boundary of its outer face with unspecified weights.*

Given two sites $s, t \in S$, there are $O(W)$ distinct bisectors in the family of $\mathrm{bis}((v_s, w_s), (v_t, w_t))$ over all possible weights w_s and w_t drawn from $[W]$. Moreover, we can compute all of them in $O(nW)$ total time, such that given weights $w_s, w_t \in [W]$ for any generic and independent set of two sites $s, t \in S$, we can return a pointer to their bisector in $O(1)$ time.

Proof Assuming without loss of generality that $w_s \geq w_t$, we can write $\mathrm{bis}((v_s, w_s), (v_t, w_t))$ as $\mathrm{bis}((v_s, w), (v_t, 0))$, where $w = w_s - w_t \in [W]$. Thus, there are $O(W)$ distinct bisectors in that family. We represent each such bisector with a linked list that contains its edges (which form a cycle) in clockwise order.

In the preprocessing phase, we find the shortest-path trees from v_s and v_t in linear time [17] and compute the value $\eta_u = dist_H[v_t, u] - dist_H[v_s, u]$ for each $u \in V$. For each of the $O(W)$ values of w, we find with a linear scan every edge $uv \in E$, such that $w < \eta_u$ and $w > \eta_v$, and insert its dual to the linked list of $\mathrm{bis}((v_s, w), (v_t, 0))$. We might need to rearrange that list in linear time. Finally, we store all these lists in a table, such that given weights $w_s, w_t \in [W]$, where without loss of generality $w_s \geq w_t$, for any two sites $s, t \in S$, we can return a pointer to the linked list of $\mathrm{bis}((v_s, w_s - w_t), (v_t, 0))$ in $O(1)$ time. □

Next, we show how to compute all Voronoi diagrams of any three sites of S in $O(nW^2)$ time. We improve upon the approach of Cabello [5, Lemma 18], requiring $O(n^2)$ time for site weights being real numbers, and we also simplify it, as we employ neither line arrangements nor amortization. We can return a pointer to a Voronoi diagram of any three sites of S in $O(1)$ time instead of $O(\log n)$.

Lemma 2 (Voronoi diagrams of 3 sites) *Let H be a triangulated planar graph with a fixed embedding, and let S be a set of b sites placed on vertices on the boundary of its outer face with unspecified weights.*

Given three sites $s, t, q \in S$, there are $O(W^2)$ distinct Voronoi diagrams in the family $\mathrm{VD}(\{(v_s, w_s), (v_t, w_t), (v_q, w_q)\})$ over all possible weights w_s, w_t, and w_q drawn from $[W]$. Moreover, we can compute all of them in $O(nW^2)$ total

time, such that given weights $w_s, w_t, w_q \in [W]$ for any generic and independent set of three sites $s, t, q \in S$, we can return a pointer to their Voronoi diagram in $O(1)$ time.

Proof Similarly to the proof of Lemma 1, assuming without loss of generality that $w_s, w_t \geq w_q$, we can write $\mathrm{VD}(\{(v_s, w_s), (v_t, w_t), (v_q, w_q)\})$ as $\mathrm{VD}(\{(v_s, w'_s), (v_t, w'_t), (v_q, 0)\})$, where $w'_s = w_s - w_q, w'_t = w_t - w_q \in [W]$. Hence, that family of Voronoi diagrams has size $O(W^2)$. According to [5, Lemma 13], a Voronoi diagram of three sites has at most one Voronoi node (besides v_∞), which we represent with a pointer. Also, each Voronoi arc is a contiguous portion $e^*_1, e^*_2, \ldots, e^*_k$ for some k, of the bisector of (s, t), (s, q), or (t, q) [5, Section 5.2]. Thus, we represent that arc with pointers to e^*_1, to e^*_k, and to the relevant bisector.

In the preprocessing phase, we invoke Lemma 1 to compute and store all bisectors of (s, t), (s, q), and (t, q) in $O(nW)$ time. Then we find the shortest-path trees from v_s, v_t, and v_q in linear time [17], and we compute for each vertex $u \in V$ the values $\eta^{st}_u = dist_H[v_s, u] - dist_H[v_t, u]$, $\eta^{qt}_u = dist_H(v_q, u) - dist_H(v_t, u)$, and $\eta^{sq}_u = dist_H(v_s, u) - dist_H(v_q, u)$.

For each of the $O(W^2)$ values of w'_s and w'_t, we use the η values to find with a linear clockwise scan that starts at v_∞ the first and the last edge $(uv)^*$ (i.e., the dual of $uv \in E$) of $\mathrm{bis}(s, t)$ such that $u \in \mathrm{VR}(s, \{s, t, q\})$ and $v \in \mathrm{VR}(t, \{s, t, q\})$. If these edges exist, we properly create the pointers for a Voronoi arc and then repeat with $\mathrm{bis}(s, q)$ and $\mathrm{bis}(t, q)$. If $\mathrm{VD}(\{s, t, q\})$ has three Voronoi arcs, we can find and store a pointer to its Voronoi node in $O(1)$ time by determining the vertex of H^* where these arcs meet. Last, we store a pointer to each computed Voronoi diagram in a two-dimensional table, such that given weights $w_s, w_t, w_q \in [W]$, where without loss of generality $w_s, w_t \geq w_q$, we can return pointers to the node and to the arcs of $\mathrm{VD}(\{(v_s, w_s - s_q), (v_t, w_t - w_q), (v_q, 0)\})$ in $O(1)$ time. □

We now give a data structure that given any four sites of S, constructs their Voronoi diagram, thus supporting the elementary operation of the Voronoi diagrams algorithm of Theorem 3. The proof is similar to that in [5, Lemma 19], but using Lemmas 1 and 2 for bisectors and Voronoi diagrams of three sites respectively, we achieve $O(nb^3W^2)$ preprocessing time instead of $\widetilde{O}(n^2b^3)$. Also, the query time here is $O(1)$ instead of $O(\log n)$. The proof we give below is a paraphrase of that of Cabello, and we include it for the sake of completeness.

Lemma 3 (Voronoi diagrams of 4 sites) *Let H be a triangulated planar graph with a fixed embedding, and let S be a set of b sites placed on vertices on the boundary of its outer face with unspecified weights.*

We can construct in $O(nb^3W^2)$ time a data structure that supports the following kind of queries: given a generic and independent set of four sites of S, whose weights are drawn from $[W]$, we can return a pointer to their Voronoi diagram in $O(1)$ time.

Proof In the preprocessing phase, we apply the methods of Lemmas 1 and 2 to compute all distinct bisectors (respectively Voronoi diagrams) of any two (respectively three) sites of S in $O\left(nW^2\right)$ time. Let s, t, q, r be four given sites of S with weights drawn from $[W]$. We assume without loss of generality that the clockwise order of the four sites on the boundary of the outer face of S is s, t, q, r. For each pair s, t we check in constant time whether $\text{bis}(s, t)$ fully participates in $\text{VD}(\{s, t, q\})$ and $\text{VD}(\{s, t, r\})$.

If that is so, then $\text{bis}(s, t)$ encloses exactly the vertices of H that are closest to s than to any of the other three sites, i.e., the vertices of $\text{VR}(s, \{s, t, q, r\})$. Thus, $\text{VD}(\{s, t, q, r\})$ is composed of $\text{bis}(s, t)$ and $\text{VD}(\{t, q, r\})$, and we can easily generate its Voronoi nodes and arcs. Notice that in this case, the part of the Voronoi diagram that is restricted in the interior faces of H is not connected. See Figure 2(a).

Else, no bisector fully bounds a Voronoi region. In other words, the part of the Voronoi diagram that is restricted in the interior faces of H is connected. That implies that $\text{VD}(\{s, t, q, r\})$ has two Voronoi nodes, besides v_∞ (remember that H^* is triangulated). See Figure 2(b). There are two cases for these nodes. First, they are the meeting points of each $\text{bis}(s, \cdot)$ and of each $\text{bis}(r, \cdot)$, respectively (Figure 2(c)). Second, they are the meeting points of each $\text{bis}(q, \cdot)$ and of each $\text{bis}(t, \cdot)$, respectively (Figure 2(d)).

We can determine which case we are in by finding whether $\text{bis}(s, r)$ participates in $\text{VD}(\{s, t, q, r\})$. Notice that the intersection of $\text{VR}(s, \{s, r, q\})$ and $\text{VR}(s, \{s, r, t\})$ gives $\text{VR}(s, \{s, r, t, q\})$ because these two Voronoi diagrams contain each bisector $\text{bis}(s, \cdot)$. Thus, we can find the Voronoi nodes v and v' of $\text{VD}(\{s, r, q\})$ and $\text{VD}(\{s, r, t\})$ respectively, and compare the order of v_∞, v', and v along $\text{bis}(s, r)$. That can be done in constant time after linear preprocessing time per bisector. If it is clockwise, we are in the first case (see Figure 2(e)); else, we are in the second (see Figure 2(f)). After determining the case we are in, we can generate the Voronoi nodes and arcs of $\text{VD}(\{s, t, q, r\})$ by properly using the information of the relevant Voronoi diagrams of triples of sites. □

Combining the above lemma with the Voronoi diagram algorithm of Theorem 3, we obtain a data structure that given any generic and independent set $S' \subseteq S$, whose weights are drawn from $[W]$, computes their Voronoi diagram. Its preprocessing time is $O\left(nb^3W^2\right)$, while the structure of Cabello required $\widetilde{O}\left(n^2b^3\right)$ construction time for real site weights. Also, the query time here is $O\left(b \log b\right)$ expected, while that of Cabello had multiple $\log n$ factors.

Theorem 4 (Voronoi diagram data structure) *Let H be a triangulated planar graph with a fixed embedding, and let S be a set of b sites placed on vertices on the boundary of its outer face with unspecified weights.*

We can construct in $O\left(nb^3W^2\right)$ time a data structure that supports the following kind of queries: given a generic and independent set $S' \subseteq S$, whose weights are drawn from $[W]$, we can compute the Voronoi diagram of S' in $O\left(b \log b\right)$ expected or $O\left(b^2\right)$ worst-case time.

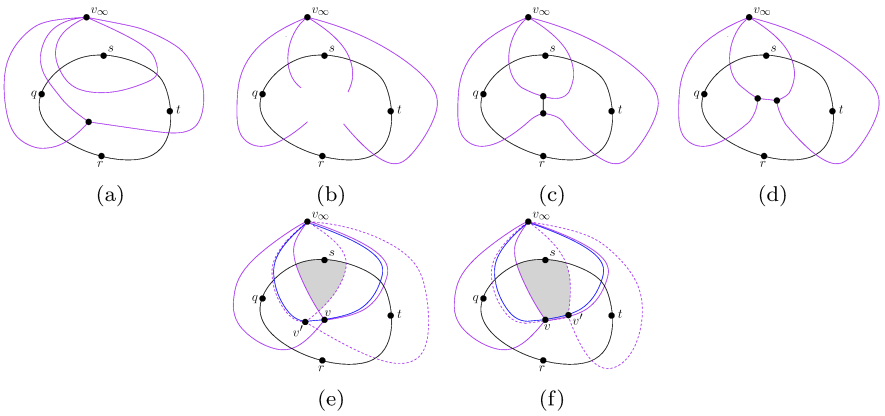


Fig. 2: (a) A Voronoi diagram where a bisector bounds a Voronoi region. (b) Parts of the bisectors of consecutive sites when no bisector bounds a Voronoi region. (c)-(d) The Voronoi diagram for the two cases of (b). (e)-(f) The dual faces of the vertices in $\mathrm{VR}(s, \{s, t, q, r\})$ (gray), $\mathrm{VD}(\{s, r, q\})$ (bold), $\mathrm{VD}(\{s, r, t\})$ (dotted), and $\mathrm{bis}(s, r)$ (blue), for the two cases of (b).

3.3 Constructing the farthest-neighbor data structure

Before describing our farthest-neighbor data structure for Problem 1, we state the following lemma, taken almost verbatim from [5, Corollary 6]. Given a cycle γ in H^*, let $V_{\mathrm{int}}(\gamma, H)$ be the set of vertices of H enclosed by it. For a vertex $v_0 \in V$, a v_0-star-shaped cycle γ in H^* is a cycle that encloses the shortest path from v_0 to every $v \in V_{\mathrm{int}}(\gamma, H)$.

Lemma 4 (Preprocessing for farthest-neighbor queries) *Let v_0 be a vertex of H. Assume that each vertex u of H has a* cost *$c(u) > 0$, and let $\Pi = \{\pi_1, \ldots, \pi_\ell\}$ be a family of simple paths in H^* with a total of h edges, counted with multiplicity.*

After $O(n + h)$ preprocessing time, we can answer the following kind of queries in $O(k)$ time: given a v_0-star-shaped cycle γ in H^*, described as a concatenation of k subpaths from Π, return $\max_{u \in V_{\mathrm{int}}(\gamma, H)} c(u)$.

Proof Sketch For each pair e_i and e_j of consecutive edges of a path in Π, we define $\chi(e_i^*, e_j^*)$ to be the maximum cost of the vertices in the region "sandwiched" between the shortest paths from v_0 to an endpoint of the dual edges of e_i and e_j. Then, a query can be reduced to $O(k)$ *range maximum queries* in the sequence of these $\chi(\cdot, \cdot)$ numbers of each such path. See [5] for more details. □

Theorem 5 (Farthest neighbor) *We can construct a farthest-neighbor data structure for Problem 1 with $O(nb^3W^2)$ preprocessing time and $O(b^2)$ query time.*

Proof Let S be a set of b sites as in Section 3.1, i.e., each $s \in S$ is placed at a vertex of X. The weights of the sites of S are unspecified. We construct

the Voronoi diagram data structure of Theorem 4 and then apply Lemma 1 to compute all bisectors of each pair of sites. For each $s \in S$ we assign a *cost* $c_s(u) = dist_H(v_s, u)$ to each $u \in U$ and $c_s(u) = 0$ to each $u \in V - U$. Then, we construct the data structure of Lemma 4, where Π is the set of all $\mathrm{bis}(s, \cdot)$, $\ell = bW$, $h = nbW$, and $k = b$. Note that for any $s, t \in S$, $\mathrm{bis}(s, t)$ is an s-star-shaped cycle in H^* because $\mathrm{VR}(s, \{s, t\})$ is a connected subtree of the shortest-path tree of s with the same root. To avoid degeneracies, we arbitrarily order the edges of H that are incident to vertices of X and *perturb* the weight of the i-th such edge by adding to it a number $\rho \cdot i$, where $\rho > 0$ is infinitesimal. We also compute the shortest path tree from every $x \in X$. The preprocessing time is $O\left(nb^3W^2\right)$.

In a query, we create naively in $O\left(b^2\right)$ time a set $S' \subseteq S$ by deleting from S each site s with $\ell(z_0, v_s) > \ell(z_0, v_t) + dist_H[v_s, v_t]$ for some $t \in S$. That, along with the perturbation in the preprocessing, guarantees that S' is generic and independent respectively. For each $s' \in S'$ we set $w_{s'}$ equal to $\ell(z_0, v_{s'})$ and then query the data structure of Theorem 4 to compute the Voronoi diagram of S' in $O\left(b^2\right)$ worst-case time (we do not need the faster $O(b \log b)$ randomized bound here). For each $s' \in S'$ the boundary of $\mathrm{VR}(s', S')$ is the concatenation of at most b subpaths of the bisectors $\mathrm{bis}(s', \cdot)$ because each Voronoi arc is a contiguous part of a bisector, as mentioned earlier. Thus, we can find $\max_{u \in \mathrm{VR}(s', S')} dist_H(s', u)$ by employing Lemma 4. Finally, we return the maximum of these distances, for a total of $O\left(b^2\right)$ query time. □

4 Approximate diameter

Given a non-negatively-weighted, undirected planar graph $\mathcal{G}$ of N vertices and of diameter Δ, we show how to compute a $(1 + O(\varepsilon))$-approximation of Δ, which is equivalent to computing an $O(\varepsilon\Delta)$-additive approximation. Let $\widetilde{\Delta}$ be a 2-approximation of Δ, which we can compute in linear time [17].

We adapt the recursive scheme of Weimann and Yuster [29]. The input to our algorithm is a non-negatively-weighted, undirected planar graph G whose vertices are either marked or unmarked, and the output is an $O(\varepsilon\Delta)$-additive approximation of the longest shortest-path distance of any two marked vertices of G. At the beginning, $G = \mathcal{G}$, all vertices of G are marked, and $n = N$. Let G_1 and G_2 be two of G's subgraphs such that each marked vertex of G lies in at least one of them. We denote by $d(G_1, G_2, G)$ the longest shortest-path distance in G between a marked vertex in G_1 and another in G_2. Notice that $d(G, G, G) = \max\{d(G_1, G_2, G), d(G_1, G_1, G), d(G_2, G_2, G)\}$. We make the following assumption for the distances between marked vertices of G, which states the distance in G between any marked vertices is an $O(\varepsilon\Delta)$-additive approximation of their distance in $\mathcal{G}$.

Assumption 1 (Distances between marked vertices) *For every two marked vertices s, t of G, we have $dist_{\mathcal{G}}[s, t] \le dist_G[s, t] \le dist_{\mathcal{G}}[s, t] + \varepsilon\Delta$.*

Recall that a *separator* of a planar graph is a subset of its vertices whose removal decomposes it into at least two disjoint, induced subgraphs. If the size of each is at most a constant fraction α of that of the original graph, the separator is said to be α-*balanced*. Moreover, if the vertices of the separator are shortest paths with common root, it is called *shortest-path* separator.

Our algorithm for computing $d(G, G, G)$ performs the following steps.

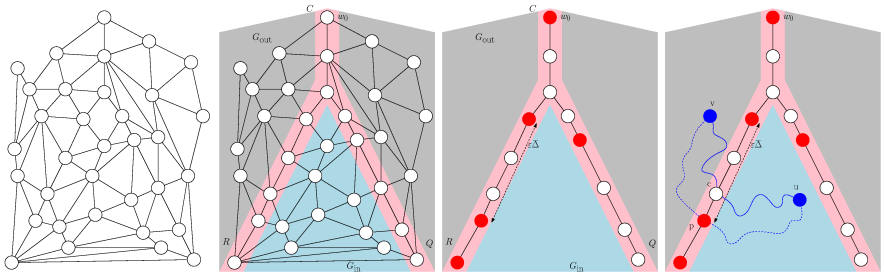
1. Find an α-balanced shortest-path separator C of G, for some constant $\alpha < 1$, such that the removal of its vertices decomposes G into two disjoint subgraphs A and B. Let $G_{\text{in}} = A \cup C$ and $G_{\text{out}} = B \cup C$. See Section 4.1.
2. Compute an $O(\varepsilon\Delta)$-additive approximation of $d(G_{\text{in}}, G_{\text{out}}, G)$. See Section 4.2.
3. Unmark each vertex of C in both G_{in} and G_{out} (these vertices appear in both graphs, so we have already considered all such pairs of marked vertices). *Augment* G_{in} with extra (unmarked) vertices and edges into a non-negatively-weighted, undirected planar graph G^+_{in} which satisfies Assumption 1 and its size is roughly the same as the number of marked vertices of G_{in}. Construct a similarly defined graph G^+_{out} for G_{out}. Recursively solve the problem in G^+_{in} and G^+_{out} to compute $d\left(G^+_{\text{in}}, G^+_{\text{in}}, G^+_{\text{in}}\right)$ and $d\left(G^+_{\text{out}}, G^+_{\text{out}}, G^+_{\text{out}}\right)$. See Section 4.3.
4. Return $\max\left\{d(G_{\text{in}}, G_{\text{out}}, G), d\left(G^+_{\text{in}}, G^+_{\text{in}}, G^+_{\text{in}}\right), d\left(G^+_{\text{out}}, G^+_{\text{out}}, G^+_{\text{out}}\right)\right\}$.

4.1 Decomposing G

To compute the shortest-path separator C in Step 1 we first find the shortest-path tree T from any marked vertex of G in linear time [17]. As Lipton and Tarjan showed [24, Lemma 2], there are two root paths R and Q in T, which can be computed in linear time, such that the removal of their vertices decomposes G into two disjoint planar subgraphs A and B of $2n/3$ vertices each. However, the size of $C = R \cup Q$ can be as big as n. See Figure 3(a)-(b). Let $G_{\text{in}} = A \cup C$ and $G_{\text{out}} = B \cup C$.

4.2 Approximating $d(G_{\text{in}}, G_{\text{out}}, G)$

Let M_{in} (respectively M_{out}) be the set of marked vertices of G_{in} (respectively G_{out}). We want to $O(\varepsilon\Delta)$-additively approximate the distance from each $v \in M_{\text{out}}$ to its farthest neighbor in M_{in} (i.e., $\max_{u \in M_{\text{in}}} \min_{c \in C}(dist_G[v, c] + dist_G[c, u])$), and return the maximum. To do that, we would like to construct the farthest-neighbor data structure of Theorem 5 with $H = G_{\text{in}}$, $X = C$, and $U = M_{\text{in}}$. Then, for each $v \in M_{\text{out}}$ we want to query that data structure with $z_0 = v$, $X' = X$, and $\ell(z_0, x') = dist_G[z_0, x']$ for each $x' \in X'$. However, there are two issues with that approach. First, C could have $O(n)$ vertices, thus leading to superlinear total time. Second, the edge weights $\ell(z_0, x')$, where $x' \in X'$, for each query are not necessarily small integers (because the distances $dist_G[z_0, x']$ are in general non-negative numbers).



(a) (b) (c) (d)

Fig. 3: (a) A planar graph. (b) A shortest-path separator (its vertices on the pink background) and the resulting decomposition. (c) The portals (red vertices). (d) Detour through portals.

For the first issue, we show that by increasing the error by $O(\varepsilon\Delta)$, it suffices to choose only a small subset of the vertices of C. Specifically, we build a set P of $O(1/\varepsilon)$ vertices, called *portals*, and route any v-to-u shortest path through them, where $v \in M_{\text{out}}$ and $u \in M_{\text{in}}$. To construct P, we start from the root of C and perform a linear walk in the $8\widetilde{\Delta}$-prefix of both R and Q. Notice that the diameter of G can be much bigger than Δ because when we recursively solve the problem in G^+_{in} and G^+_{out} only the distances between marked vertices are approximately preserved. With these walks, we select $O(1/\varepsilon)$ portals on each path, such that any consecutive pair of them is at distance at most $\varepsilon\Delta$ thereon and discard any duplicates. See Figure 3(c). Let

$$d'(G_{\text{in}}, G_{\text{out}}, G) = \max_{v \in M_{\text{out}}, u \in M_{\text{in}}} \min_{p \in P} (dist_G[v,p] + dist_G[p,u]).$$

Lemma 5 (Approximation with portals) *We have that* $d(G_{\text{in}}, G_{\text{out}}, G) \le d'(G_{\text{in}}, G_{\text{out}}, G) \le d(G_{\text{in}}, G_{\text{out}}, G) + O(\varepsilon\Delta)$.

Proof Since $d'(G_{\text{in}}, G_{\text{out}}, G)$ corresponds to a path in G, we have $d(G_{\text{in}}, G_{\text{out}}, G) \le d'(G_{\text{in}}, G_{\text{out}}, G)$. Let $v \in M_{\text{out}}$ and $u \in M_{\text{in}}$ be such that $dist_G[u,v] = d(G_{\text{in}}, G_{\text{out}}, G)$. Recall that from Assumption 1 we have $dist_{\mathcal{G}}[s,t] \le dist_G[s,t] \le dist_{\mathcal{G}}[s,t] + \varepsilon\Delta$ for any two marked vertices s, t of G. Since $dist_{\mathcal{G}}[s,t] \le \Delta$, we have $dist_G[s,t] \le (1+\varepsilon)\Delta$.

Let $c \in C$ be a vertex that the v-to-u shortest path in G crosses and assume without loss of generality that $c \in R$. We now show that c lies in $8\widetilde{\Delta}$-prefix of R. Let w_0 be the root of C. All of v, u, and w_0 are marked, so the w_0-to-v and the v-to-u shortest paths are of length at most $(1+\varepsilon)\Delta$. Since c lies on the v-to-u shortest path, the length of the v-to-c shortest path is also upper-bounded by $(1+\varepsilon)\Delta$. Hence, the concatenation of the w_0-to-v and the v-to-c shortest paths has length $2(1+\varepsilon)\Delta$, which is indeed smaller than $8\widetilde{\Delta}$ for $\varepsilon < 1$ (since $\Delta \le 2\widetilde{\Delta}$).

Let $p \in P$ be the portal on R that is closest to c. From the way we built P, we have $dist_G[p,c] \le \varepsilon\Delta$. The triangle inequality implies that $dist_G[v,p] \le$

$dist_G[v,c] + dist_G[c,p] \leq dist_G[v,c] + \varepsilon\Delta$ and $dist_G[p,u] \leq dist_G[c,u] + \varepsilon\Delta$. Using these inequalities, we have

$$\begin{aligned} d'(G_{\text{in}}, G_{\text{out}}, G) &= dist_G[v,p] + dist_G[p,u] \\ &\leq dist_G[v,c] + dist_G[c,u] + 2\varepsilon\Delta \\ &\leq dist_G[v,u] + 2\varepsilon\Delta \;\leq\; d(G_{\text{in}}, G_{\text{out}}, G) + 2\varepsilon\Delta. \end{aligned}$$

See Figure 3(d). □

For the second issue, we show how to ensure that whenever we query the data structure of Theorem 5 each $\ell(z_0, x)$ $(x \in X)$ is a small integer. We assume that without loss of generality $1/\varepsilon$ is an integer. First, we compute the shortest-path tree in G from every $p \in P$ in $O((1/\varepsilon)n)$ time [17] and create a value $\widehat{d}[v,p]$ for each $v \in M_{\text{out}}$ and $p \in P$ by first rounding $dist_G[v,p]$ to the closest multiple of $\varepsilon\widetilde{\Delta}$ and then dividing it with that number. If $dist_G[v,p] \geq 4\widetilde{\Delta}$, then $dist_G[v,p]$ will be irrelevant in approximating $d'(G_{\text{in}}, G_{\text{out}}, G)$. Thus, in this case we set $\widehat{d}[v,p] = 4/\varepsilon$. Notice that now $\widehat{d}[v,p] \in [4/\varepsilon]$. We divide (but not round) every edge weight of G_{in} by $\varepsilon\widetilde{\Delta}$ and denote the resulting graph by $\widehat{G}_{\text{in}}$. Let

$$\widehat{d} = \varepsilon\widetilde{\Delta} \cdot \max_{v \in M_{\text{out}}, u \in M_{\text{in}}} \min_{p \in P} (\widehat{d}[v,p] + dist_{\widehat{G}_{\text{in}}}[p,u]).$$

Thus, we have $d(G_{\text{in}}, G_{\text{out}}, G) \leq \widehat{d} \leq d(G_{\text{in}}, G_{\text{out}}, G) + O(\varepsilon\Delta)$.

We can finally construct the farthest-neighbor data structure of Theorem 5 with $H = \widehat{G}_{\text{in}}$, $X = P$, $U = M_{\text{in}}$, $b = O(1/\varepsilon)$, and $W = 4/\varepsilon$. For each $v \in M_{\text{out}}$, we query the data structure with $z_0 = v$, $X = P$, and $\ell(z_0, x) = \widehat{d}_G[v,x]$, where $x \in X$, and multiply the answer by $\varepsilon\widetilde{\Delta}$. The total time to $O(\varepsilon\Delta)$-additively approximate $d(G_{\text{in}}, G_{\text{out}}, G)$ is $O\left(nb^3W^2 + nb^2\right) = O\left((1/\varepsilon)^5 n\right)$.

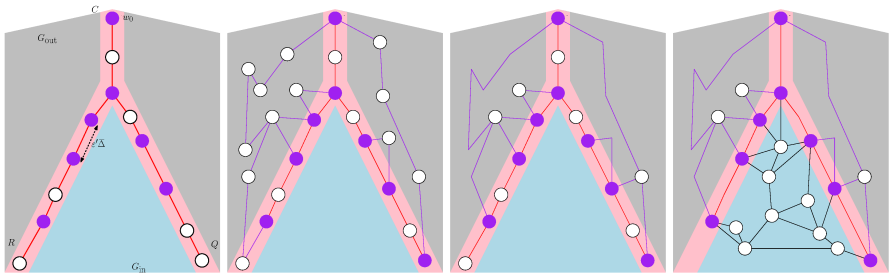
Contrary to our approach, Weimann and Yuster [29] employed a brute-force search, after observing that there are only $2^{O(1/\varepsilon)}$ combinatorially different vertices of M_{in} and M_{out} (in terms of their vectors of distances to the portals).

4.3 Recursively solving the problem in G^+_{in} and G^+_{out}

The vertices on C appear in both G_{in} and G_{out}, so we have already considered all such pairs of marked vertices and can unmark them in both graphs. Then, we need to augment G_{in} into a graph G^+_{in} to ensure that Assumption 1 is satisfied, i.e., $dist_{\mathcal{G}}[s,t] \leq dist_{G^+_{\text{in}}}[s,t] \leq dist_{\mathcal{G}}[s,t] + \varepsilon\Delta$ for any two marked vertices s, t of G^+_{in}. Also, we need to ensure that the size of G^+_{in} is roughly the same as the number of marked vertices of G_{in}. We want to augment G_{out} to a similarly defined graph G^+_{out} and use recursion in G^+_{in} and G^+_{out}.

We start at the common root of R and Q and select with a linear walk on their $8\widetilde{\Delta}$-prefices $1/\varepsilon'$ of their vertices, called *dense portals*, where $\varepsilon' \ll \varepsilon$ is a parameter to be set later, such that any consecutive pair of them is at distance $\varepsilon'\Delta$ thereon. The union B_{in} of the $(1/\varepsilon')^2$ shortest paths in G_{out} of

every pair of dense portals has at most $(1/\varepsilon')^4$ vertices of degree more than two. That is guaranteed by generating these shortest paths with the data structure of Klein [21], which ensures that any two of them share at most a common subpath (see Theorem 6). Thus, we can shrink the rest to obtain a planar graph B'_{in}. Then, we unmark the vertices of B'_{in} and create a new graph $G'_{\text{in}} = G_{\text{in}} \cup B'_{\text{in}}$ of $|V(G)| + (1/\varepsilon')^4$ vertices. Its size can be reduced to $\beta + (1/\varepsilon')^4$, where $\beta \leq 2|V(G)|/3$ is the set of vertices of $V(G_{\text{in}}) - V(C)$. To do so, we delete each non-dense-portal c and redirect each edge cu, where $u \in V(G_{\text{in}}) - V(C)$, from c to its closest dense portal p and change its weight to $dist_G[p, u]$. We also create an edge between every consecutive pair s, t of dense portals on R and Q with weight $dist_G[s, t]$. Notice that all these edges can be created without affecting the planarity. Let G^+_{in} be the ensuing graph. See Figure 4.



(a) (b) (c) (d)

Fig. 4: (a) The dense portals (purple vertices) for the planar graph of Figure 3(a). (b)-(c) The graphs B_{in} and B'_{in}, respectively (on the grey background). (d) The graph G^+_{in}.

Lemma 6 (Recursion with dense portals) *For any two marked vertices u, v of G^+_{in}, we have $dist_G[s, t] \leq dist_{G^+_{\text{in}}}[s, t] \leq dist_G[s, t] + O(\varepsilon' \Delta)$.*

Proof A path in G^+_{in} corresponds to a path in G, and u and v are marked in both G and G^+_{in}. Thus, $dist_G[u, v] \leq dist_{G^+_{\text{in}}}[u, v]$. If $\pi_G[u, v]$ uses only edges incident to vertices in $V(G_{\text{in}}) - V(C)$, then it also exists in G^+_{in}, and we trivially have $dist_G[u, v] = dist_{G^+_{\text{in}}}[u, v]$.

Otherwise, we assume without loss of generality that $\pi_G[u, v]$ is composed of the u-to-c shortest path in G_{in}, the c-to-c' shortest path in G (which either lies entirely in G_{in} or not), and the c'-to-v shortest path in G_{in}, where $c \in R$ and $p \in Q$. As in the proof of Lemma 5, we can show that c and c' lie in the $8\widetilde{\Delta}$-prefix of R and Q respectively, i.e., $dist_G[w_0, c], dist_G[w_0, c'] \leq 8\widetilde{\Delta}$, where w_0 is the common root of P and Q. Let p and p' be the closest dense portals to c and c' respectively, so $dist_G[c, p], dist_G[c', p'] \leq \varepsilon' \Delta$. From the way we

deleted the non-dense-portals and redirected their incident edges, there is a u-to-p path in G_{in}^+ of length at most $dist_G[u,c]+dist_G[c,p] \leq dist_G[u,c]+\varepsilon'\Delta$ and a p'-to-v path of length at most $dist_G[p',c'] + dist_G[c',v] \leq dist_G[c',v] + \varepsilon'\Delta$.

If the c-to-c' shortest path in G does not lie entirely in G_{in}, then from the way we constructed B_{in}^+, we know

$$dist_{G_{\text{in}}^+}[p,p'] \leq dist_{G_{\text{out}}}[c,c'] + dist_{G_{\text{out}}}[p,c] + dist_{G_{\text{out}}}[p',c'] \leq dist_G[c,c'] + 2\varepsilon'\Delta.$$

Otherwise, let the c-to-c' shortest path in G be composed of the c-to-x shortest path in R, the x-to-x' shortest path in $V\,(G_{\text{in}}) - V\,(C)$, and the x'-to-c' shortest path in Q, where $x \in R$ and $x' \in Q$. We can similarly show that in this case we also have $dist_{G_{\text{in}}^+}[p,p'] \leq dist_G[c,c'] + 2\varepsilon'\Delta$.

Therefore, the concatenation of the u-to-p, p-to-p', and p'-to-v shortest paths in G_{in}^+ yields a path of length

$$\begin{aligned} &dist_{G_{\text{in}}^+}[u,p] + dist_{G_{\text{in}}^+}[p,p'] + dist_{G_{\text{in}}^+}[p',v] \\ &\leq dist_{G_{\text{in}}}[u,c] + dist_G[c,c'] + dist_{G_{\text{in}}}[c',v] + O\,(\varepsilon'\Delta) \\ &\leq dist_G[u,v] + O\,(\varepsilon'\Delta)\,. \qquad \square \end{aligned}$$

It remains to show how to construct B'_{in} and how to set ε'. For the first, contrary to Weimann and Yuster, who constructed B'_{in} explicitly in $O\,((1/\varepsilon')n)$ time, we employ a method based on a slightly modified version of the multiple shortest paths data structure of Klein [21].

Theorem 6 (Augmented graph) *We can build B'_{in} in $O\left(n \log n + (1/\varepsilon')^4 \log^2 n\right)$ time.*

Proof As Klein showed, we can construct an implicit representation of the shortest-path tree $T(u)$ of each vertex u on the boundary of the outer face of G_{in} in $O\,(n \log n)$ total time. The order of the children $w_1, w_2, \ldots, w_\ell$ of a vertex v in $T(u)$ is specified as follows: w_i is to the left of w_j if and only if vw_i, vw_j, and vp are in counterclockwise order around v, where p is v's parent in $T(u)$. Klein used a *persistent* [10] version of *dynamic trees* [27] to represent the $T(u)$'s, so we can find the u-to-v shortest-path distance, for any $v \in V(G_{\text{in}})$, in $O\,(\log n)$ time. We want to augment Klein's data structure to also support the following two queries on each $T(u)$: (i) find the lowest common ancestor of any two vertices; and (ii) find the level ancestor of any vertex and any level. We can accommodate both queries in $O\,(\log n)$ time by merely replacing the dynamic trees with the (persistent) *top-tree* structures of Alstrup et al. [1].

To build B'_{in}, we construct the modified version of Klein's data structure for G_{out}, after redrawing it in linear time (if needed), such that the vertices of C lie on the outer face. We will describe how to use this data structure to find all vertices of G_{out} of degree more than two in B_{in}; as argued before, there are $O\left((1/\varepsilon')^4\right)$ such vertices. There are three cases for each pair of shortest paths between dense portals: they (i) do not intersect, (ii) intersect only at one vertex, or (iii) share a common subpath. For any four dense portals a, b, c, and d, assuming without loss of generality that the latter case holds for

$\pi_G[a,b]$ and $\pi_G[c,d]$ and that c is between a and b on the boundary of the outer face of G_{in}, we want to find the first and the last vertices, p_1 and p_2, on that subpath.

We find p_1 with a binary search on $\pi_G[a,b]$ and T_c (finding p_2 is similar). Let p' and p'' be initially set to a and b respectively, and let p be the vertex midway between p' and p'' on $\pi_G[a,b]$, which can be found by a level ancestor query on T_a. We want to determine whether p is (i) between p_1 and p_2 (i.e., on $\pi_G[c,d]$), (ii) between a and p_1, or (iii) between p_2 and b. To do so, we find the lowest common ancestor lca of p and d on T_c. If $lca = p$, we are in case (i) because p is on $\pi_G[c,d]$. See Figure 5(a) and (c). Else, we perform a level ancestor query for p and d to find the children $\hat{p}$ and $\hat{d}$ of lca that lie on $\pi_G[p, lca]$ and $\pi_G[d, lca]$, respectively, and compare their order around lca. If $\hat{p}$ is to the right of $\hat{d}$, we are in case (ii), else we are in case (iii). See Figure 5(b) and (d). For case (i) or (iii) we recurse with $p' = p$, for case (ii) with $p'' = p$, and we stop when $p' = p''$. Each of the $O\left((1/\varepsilon')^4\right)$ binary searches requires $O(\log n)$ queries to the data structure and takes $O(\log^2 n)$ time. Once we have processed every pair of dense portals, we shrink the vertices of $V(G_{\text{out}})$ of degree at most two with a linear scan, thus obtaining B'_{in}. □

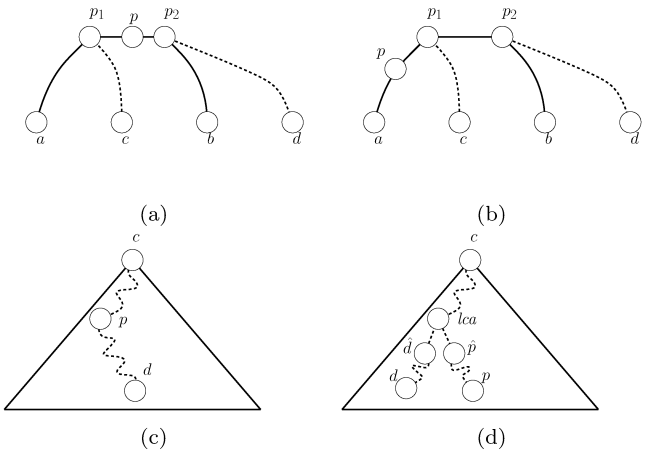


Fig. 5: (a), (c) p lies between p_1 and p_2. (b), (d) p lies between a and p_1.

Finally, Weimann and Yuster used a fixed constant for ε' throughout their algorithm. There are $O(\log N)$ recursion levels, and the error accumulates, hence they set $\varepsilon' = \varepsilon/\log N$. However, with that choice of ε', the four $O(1/\varepsilon')$ factors of the size of B'_{in} lead to four $O(\log N)$ factors in the running time of their solution. Instead, we make ε' adaptive (i.e., dependent on the current input size n), namely equal to ε/n^{δ} for a sufficiently small constant $\delta > 0$, thus shaving off two $\log N$ factors, while retaining the $1+O(\varepsilon)$ approximation factor, as we show next.

4.4 Analyzing our algorithm

Approximation factor. As described in Section 4.2, we compute an $O(\varepsilon\Delta)$-additive approximation of $d\left(G_{\text{in}}^{(\mu)}, G_{\text{out}}^{(\mu)}, G^{(\mu)}\right)$ for each node μ of the recursion tree, where $G^{(\mu)}$ is the graph associated with μ, and $G_{\text{in}}^{(\mu)}$ and $G_{\text{out}}^{(\mu)}$ are the two graphs created by decomposing $G^{(\mu)}$ as in Section 4.1. As explained in Section 4.3, from the way we apply recursion, in each child ν of μ we have that $d\left(G_{\text{in}}^{(\nu)}, G_{\text{out}}^{(\nu)}, G^{(\nu)}\right) \leq d\left(G_{\text{in}}^{(\mu)}, G_{\text{out}}^{(\mu)}, G^{(\mu)}\right) + O(\varepsilon'\Delta)$ and $\left|V\left(G^{(\nu)}\right)\right| \leq (2/3)\left|V\left(G^{(\mu)}\right)\right| + O\left((1/\varepsilon')^4\right)$, where $\varepsilon' = \varepsilon \Big/ \left|V\left(G^{(\mu)}\right)\right|^{\delta}$ for some sufficiently small constant $\delta > 0$. At the root κ of the recursion tree, we have $G^{(\kappa)} = \mathcal{G}$, while in each leaf λ, $\left|V\left(G^{(\lambda)}\right)\right| = \Theta(N_0)$ for some parameter $N_0 \geq (1/\varepsilon)^{\Omega(1)}$ to be determined later.

Thus, the additive error of our algorithm is $O(\varepsilon\Delta) + O\left(\sum_i \varepsilon_i\Delta\right)$, where $\varepsilon_i = \varepsilon/n_i^{\delta}$ for some sequence $n_1, n_2, \ldots, n_k$ that satisfies $n_{i-1}/3 \leq n_i \leq 2n_{i-1}/3 + O\left((1/\varepsilon_i)^4\right)$ with $n_1 = N$ and $n_k = \Theta(N_0)$. Since n_i decreases at least exponentially, ε_i grows likewise. Hence, $\sum_i \varepsilon_i$ is similar to a geometric series and can be upper-bounded by the last term, which is $O(\varepsilon_k) = O\left(\varepsilon/N_0^{\delta}\right) = O\left(\varepsilon^{1+\Omega(1)}\right)$. We conclude that the additive error is $O(\varepsilon\Delta)$, implying that the approximation factor of our algorithm is $1+O(\varepsilon)$. That can be refined to $1+\varepsilon$ after adjusting ε by a constant factor.

Assumption 1 is true. We now show that for any two marked vertices u and v of a graph G encountered during the recursion, we have $dist_{\mathcal{G}}[u,v] \leq dist_G[u,v] \leq dist_{\mathcal{G}}[u,v] + \varepsilon\Delta$.

Fix a non-leaf node ν of $\mathcal{T}$ and its parent μ, such that u and v are marked in both and $G^{(\nu)} = G$. As explained in Section 4.3, from the way we use recursion, we have $dist_{G^{(\mu)}}[u,v] \leq dist_{G^{(\nu)}}[u,v] \leq dist_{G^{(\mu)}}[u,v] + O(\varepsilon'\Delta)$.

Thus, $dist_{\mathcal{G}}[u,v] \leq dist_G[u,v] \leq dist_{\mathcal{G}}[u,v] + O\left(\sum_i \varepsilon_i\Delta\right)$, where $\varepsilon_i = \varepsilon/n_i^{\delta}$, for some sequence $n_1, n_2, \ldots, n_k$ that satisfies $n_{i-1}/3 \leq n_i \leq 2n_{i-1}/3 + O\left((1/\varepsilon_i)^4\right)$ with $n_1 = N$ and $n_k = \Theta(N_0)$. As above, $\sum_i \varepsilon_i$ can be upper-bounded by $\varepsilon^{1+\Omega(1)}$. Hence, for any two marked vertices u and v of $G^{(\mu)}$, we have $dist_{\mathcal{G}}[u,v] \leq dist_G[u,v] \leq dist_{\mathcal{G}}[u,v] + O\left(\varepsilon^{1+\Omega(1)}\Delta\right)$, thus proving (a stronger version of) the assumption.

Running time. In a graph of size n, the running time $T(n)$ of our algorithm satisfies the following recurrence

$$T(n) \leq \max_{1/3 \leq \alpha \leq 2/3} \left(T\left(\alpha n + O\left((1/\varepsilon)^4 n^{4\delta}\right)\right) + T\left((1-\alpha)n + O\left((1/\varepsilon)^4 n^{4\delta}\right)\right) + O\left(n\log n + (1/\varepsilon)^5 n\right)\right),$$

since the algorithm in Section 4.2 takes $O\left((1/\varepsilon)^5 n\right)$ time and Theorem 6 takes $O\left(n\log n + (1/\varepsilon')^4 \log^2 n\right) = O\left(n\log n + (1/\varepsilon)^4 n^{4\delta}\log^2 n\right)$ time—the last term is dominated by $O\left((1/\varepsilon)^5 n\right)$ when $\delta < 1/4$.

We choose N_0 so that $(1/\varepsilon)^4 N_0^{4\delta} = \Theta(N_0)$, i.e., $N_0 = \Theta((1/\varepsilon)^{4/(1-4\delta)})$. In the base case there are $O(N/N_0)$ graphs of $O(N_0)$ vertices each, so it can be addressed in $O\left(N/N_0 \cdot N_0^2\right) = O(N_0 N)$ time [12], which is dominated by $O\left((1/\varepsilon)^5 N\right)$ when $\delta < 1/20$. Solving the recurrence gives $T(N) = O\left(N \log^2 N + (1/\varepsilon)^5 N \log N\right)$.

Theorem 7 (Approximate diameter) *We can find a $(1+\varepsilon)$-approximation of the diameter of a non-negatively-weighted, undirected planar graph of n vertices in $O\left(n \log^2 n + (1/\varepsilon)^5 n \log n\right)$ time.*

Remarks:

- Gawrychowski et al. [14] recently improved Cabello's algorithm [5] for computing the diameter in planar graphs exactly. Their algorithm is deterministic instead of randomized and requires $\widetilde{O}\left(n^{5/3}\right)$ time instead of $\widetilde{O}\left(n^{11/6}\right)$. It is likely that their techniques can be used to shave off some $1/\varepsilon$ factors.
- An interesting consequence of our result is that we can compute the *exact* diameter of an *unweighted* planar graph in $O\left(n \log^2 n + \Delta^{O(1)} n \log n\right)$ time, where Δ is the diameter, simply by setting ε near $1/\Delta$. If one wants running time near linear in n, the best previous result we are aware of was by Eppstein [11] and had exponential dependence in Δ (namely, the time bound is $O\left(n 2^{O(\Delta \log \Delta)}\right)$). Note that our result beats Cabello's or Gawrychowski et al.'s algorithm when the diameter is smaller than n^δ for some constant δ.

By keeping track throughout the algorithm of the distance of each vertex to its farthest neighbor, we can compute a $(1+\varepsilon)$-approximation of its eccentricity. Hence, we can also compute a $(1+\varepsilon)$-approximation of the radius (i.e., the minimum eccentricity) of the graph.

Corollary 1 (Approximate eccentricities, farthest neighbors, and radius) *Given a non-negatively weighted, undirected planar graph of n vertices, we can compute a $(1+\varepsilon)$-approximation of the radius and of the eccentricity of each vertex (and an approximate farthest neighbor) in $O\left(n \log^2 n + (1/\varepsilon)^5 n \log n\right)$ time.*

5 Approximate distance oracles

To construct an approximate distance oracle in the word RAM model, we build upon the general framework of the oracles of Kawarabayashi et al. [19] and of Gu and Xu [16]. Specifically, given a non-negatively-weighted, undirected planar graph $\mathcal{G}$ of N vertices and of diameter Δ, we focus in Section 5.1 on constructing a distance oracle with additive stretch $O(\varepsilon\Delta)$, i.e., a data structure that supports the following kind of queries: given for any two vertices u and v of $\mathcal{G}$, return a value $\widetilde{d}$ with $dist_{\mathcal{G}}(u,v) \le \widetilde{d} \le dist_{\mathcal{G}}(u,v) + O(\varepsilon\Delta)$. Then, we can obtain a $(1+\varepsilon)$-approximate distance oracle with an approach based on *sparse neighborhood covers*, as explained in Section 5.2. Let $\widetilde{\Delta}$ be a 2-approximation of Δ.

5.1 Distance oracles with additive stretch

The decomposition tree. We recursively decompose $\mathcal{G}$ as in Sections 4.1 and 4.3, but now we also store the ensuing graphs in a *recursive decomposition tree* $\mathcal{T}$ with the following properties.

- $\mathcal{T}$ has degree two and height $O(\log N)$.
- Each non-leaf node μ of $\mathcal{T}$ is associated with a graph $G^{(\mu)}$. The graph of each child of μ has at most $2|V\left(G^{(\mu)}\right)|/3 + O\left((1/\varepsilon')^4\right)$ vertices, where $\varepsilon' = \varepsilon/|V\left(G^{(\mu)}\right)|^{\delta}$ for a sufficiently small constant $\delta > 0$. At the root κ of $\mathcal{T}$, $G^{(\kappa)} = \mathcal{G}$, and at each leaf, λ $|V\left(G^{(\lambda)}\right)| = \Theta(N_0)$ for some parameter $N_0 \geq (1/\varepsilon)^{\Omega(1)}$.
- Each non-leaf node μ of $\mathcal{T}$ is also associated with a shortest-path separator $C^{(\mu)}$. We call a vertex of $G^{(\mu)}$ *marked* if and only if it does not lie in $C^{(\nu)}$ of any ancestor ν of μ. Each marked vertex of $G^{(\mu)}$ that is not on $C^{(\mu)}$ is contained in the graph of one child of μ.
- Fix a non-leaf node ν of $\mathcal{T}$ and let μ be its parent. For each pair of marked vertices $u, v \in V(G^{(\nu)})$, we have $dist_{G^{(\mu)}}[u,v] \leq dist_{G^{(\nu)}}[u,v] \leq dist_{G^{(\mu)}}[u,v] + O(\varepsilon'\Delta)$.

Our data structure. In each non-leaf node μ of $\mathcal{T}$, we find in linear time, as in Section 4.2, a set $P^{(\mu)}$ of $O(1/\varepsilon)$ portals on $C^{(\mu)}$, such that for any two marked vertices $u, v \in V\left(G^{(\mu)}\right)$ on different sides of $C^{(\mu)}$,

$$\begin{aligned} dist_{G^{(\mu)}}[u,v] &\leq \min_{p \in P^{(\mu)}} \left(dist_{G^{(\mu)}}[u,p] + dist_{G^{(\mu)}}[p,v]\right) \\ &\leq dist_{G^{(\mu)}}[u,v] + O(\varepsilon\Delta)\,. \end{aligned}$$

Then, we run an SSSP algorithm from each $p \in P^{(\mu)}$. Let ν_1 and ν_2 be the children of μ. For every marked vertex v of ν_1, we create a value $\widehat{d}_{G^{(\mu)}}[v,p] \in [4/\varepsilon]$ as in Section 4. We also divide every edge weight of $G^{(\mu)}$ by $\varepsilon\widetilde{\Delta}$. Thus,

$$\begin{aligned} dist_{G^{(\mu)}}[u,v] &\leq \varepsilon\widetilde{\Delta} \cdot \min_{p \in P^{(\mu)}} \left(\widehat{d}_{G^{(\mu)}}[u,p] + dist_{G^{(\mu)}}[p,v]\right) \\ &\leq dist_{G^{(\mu)}}[u,v] + O(\varepsilon\Delta)\,. \end{aligned}$$

We create a set S of $O(1/\varepsilon)$ sites with unspecified weights in $G^{(\nu_2)}$, place each at a vertex of $P^{(\mu)}$. Also, we perturb the weights of the edges incident to the sites as in Section 4.2. Then, we construct for $G^{(\nu_2)}$ and S a data structure for Voronoi diagrams, similarly to that of Theorem 4, in $O\left((1/\varepsilon)^5 n\right)$ time. For each marked vertex v of $G^{(\nu_1)}$, we construct an empty set P' and insert to it every $p \in P^{(\mu)}$, such that the last edge on the v-to-p shortest path in $G^{(\mu)}$ does not lie in $G^{(\nu_2)}$, which we can determine by inspecting p's shortest-path tree. Next, we query the data structure of Theorem 4 to construct the Voronoi diagram in $G^{(\nu_2)}$ of a set $S' \subseteq S$ of sites placed at the vertices of P', where the weight of each $s' \in S$ is equal to $\widehat{d}_G[v, v_{s'}]$. Notice that from the perturbation and from the choice of P', we have that S' is generic and independent. Finally,

we build for that Voronoi diagram the *vertex location* data structure of [15, Section 6]. That data structure preprocesses once in $O(bn)$ time and space a planar graph of n vertices and a set of b sites with unspecified weights that lie on the boundary of its outer face to support the following operation: preprocess any given Voronoi diagram of a subset of these sites in $O(b)$ time and space, such that the site that contains a given vertex can be found in $O(\log b)$ time. In our application, $b = 1/\varepsilon$, so we need $O((1/\varepsilon)n)$ total time and space for preprocessing all Voronoi diagrams related to the marked vertices of $G^{(\nu_1)}$.

Given two vertices u and v, we find in $O(1)$ time the lowest node μ of $\mathcal{T}$ where both are marked (that can be done with linear preprocessing time). If none exists, u and v must reside in a leaf, and we return their shortest-path distance by looking up the distance matrix therein (which we can precompute). Else, we assume without loss of generality that $u \in V\left(G^{(\nu_2)}\right)$ and $v \in V\left(G^{(\nu_1)}\right)$, where ν_1 and ν_2 are the children of μ, and that we have computed the Voronoi diagram in $G^{(\nu_2)}$ for sites and weights prescribed by v. Then, we query, in $O(\log(1/\varepsilon))$ time, the vertex location data structure for that Voronoi diagram to find the site p therein whose Voronoi region contains u. Finally, we return $\widetilde{d} = \varepsilon\widetilde{\Delta} \cdot (dist_{G^{(\mu)}}[v,p] + dist_{G^{(\nu_2)}}[p,u])$.

Additive stretch. We now show that for any two vertices u, v of $\mathcal{G}$, our oracle returns a value $\widetilde{d}$ with $dist_{\mathcal{G}}[u,v] \le \widetilde{d} \le dist_{\mathcal{G}}[u,v] + O(\varepsilon\Delta)$.

Fix a non-leaf node ν of $\mathcal{T}$ and its parent μ, such that u and v are marked in both. From the way we use recursion, we have $dist_{G^{(\mu)}}[u,v] \le dist_{G^{(\nu)}}[u,v] \le dist_{G^{(\mu)}}[u,v] + O(\varepsilon'\Delta)$, where $\varepsilon' = \varepsilon/|V(G^{(\mu)})|^{\delta}$. At the root κ of the recursion tree, $G^{(\kappa)} = \mathcal{G}$, while at each leaf λ, $\left|V\left(G^{(\lambda)}\right)\right| = \Theta(N_0)$. Also, from the way we approximate $dist_{G^{(\mu)}}[u,v]$, we have $dist_{G^{(\mu)}}[u,v] \le \widetilde{d} \le dist_{G^{(\mu)}}[u,v] + O(\varepsilon\Delta)$.

Thus, the value $\widetilde{d}$ that our oracle returns is such that $dist_{\mathcal{G}}[u,v] \le \widetilde{d} \le dist_{\mathcal{G}}[u,v] + O(\varepsilon\Delta) + O(\sum_i \varepsilon_i\Delta)$, where $\varepsilon_i = \varepsilon/n_i^{\delta}$, for some sequence $n_1, n_2, \ldots, n_k$ that satisfies $n_{i-1}/3 \le n_i \le 2n_{i-1}/3 + O\left((1/\varepsilon_i)^4\right)$ with $n_1 = N$ and $n_k = \Theta(N_0)$. As in Section 4.4, $\sum_i \varepsilon_i$ can be upper-bounded by $O\left(\varepsilon^{1+\Omega(1)}\right)$, hence yielding the lemma.

Preprocessing time and space. The preprocessing time $T(n)$ and space $S(n)$ satisfy the following recurrences:

$$T(n) \le \max_{1/3 \le \alpha \le 2/3} \left(T\left(\alpha n + O\left((1/\varepsilon)^4 n^{4\delta}\right)\right) + T\left((1-\alpha)n + O\left((1/\varepsilon)^4 n^{4\delta}\right)\right) + O\left(n\log n + (1/\varepsilon)^5 n\right)\right),$$

$$S(n) \le \max_{1/3 \le \alpha \le 2/3} \left(S\left(\alpha n + O\left((1/\varepsilon)^4 n^{4\delta}\right)\right) + S\left((1-\alpha)n + O\left((1/\varepsilon)^4 n^{4\delta}\right)\right) + O\left((1/\varepsilon)n\right)\right).$$

As before, we choose $N_0 = \Theta((1/\varepsilon)^{4/(1-4\delta)})$. In the base case there are $O(N/N_0)$ graphs of $O(N_0)$ vertices each, so we need

$O\left(N/N_0 \cdot N_0^2\right) = O\left(N_0 N\right)$ time and space. Solving the recurrences gives $T(N) = O\left(N \log^2 N + (1/\varepsilon)^5 N \log N\right)$ and $S(N) = O\left((1/\varepsilon) N \log N + (1/\varepsilon)^{4+O(\delta)} N\right)$. We can readjust δ by a constant factor.

Theorem 8 (Oracle of additive stretch) *Given a non-negatively weighted, undirected planar graph of n vertices and of diameter Δ, we can construct for it an oracle of $O\left(\varepsilon\Delta\right)$ additive stretch with $O\left(n \log^2 n + (1/\varepsilon)^5 n \log n\right)$ preprocessing time, $O\left((1/\varepsilon) n \log n + (1/\varepsilon)^{4+\delta} n\right)$ space, and $O\left(\log\left(1/\varepsilon\right)\right)$ query time for any constant $\delta > 0$.*

5.2 Approximate distance oracles

Now we show how to use our oracle of additive stretch as a building block to construct a $(1+\varepsilon)$-approximate distance oracle with an existing technique based on *sparse neighborhood covers*, as done by Kawarabayashi et al. [19] and Gu and Xu [16]. The following lemma is due to Busch et al. [3] and Kawarabayashi et al. [19].

Lemma 7 (Sparse neighborhood covers) *Given a planar graph $G = (V, E)$ of n vertices and an integer r, we can construct a collection of subsets V_i of V in $O\left(n \log n\right)$ time, such that (i) the diameter of the subgraph of $\mathcal{G}$ induced by each V_i is at most $24r - 18$, (ii) every vertex resides in $O\left(1\right)$ subsets V_i, and (iii) for every vertex v, the set of all vertices at distance at most r from v is contained in at least one of the V_i's.*

We assume without loss of generality that each edge of $\mathcal{G}$ has weight at least one. That can be ensured by dividing every edge weight with the minimum. Then, for every scale $r \in \left\{2^0, 2^1, \ldots, 2^{\log \Delta}\right\}$, we consider the graph $\mathcal{G}^{(r)}$ that ensues after deleting the edges of $\mathcal{G}$ of weight at least $24r$ and contracting those of weight at most r/N^2. Thus, each edge appears in the graphs of $O\left(\log N\right)$ scales, which we can identify in that much time. Let R be the set that contains each scale r such that $\mathcal{G}^{(r)}$ has at least one edge. For each $r \in R$, we construct $\mathcal{G}^{(r)}$ and the sparse neighborhood cover of Lemma 7 for it, hence obtaining a collection of subsets $V_i^{(r)}$. Then, for the induced graph of $\mathcal{G}^{(r)}$ of each such subset, we build our distance oracle of $O\left(\varepsilon r\right)$ additive stretch of Theorem 8. Finally, we build the 2-approximate distance oracle of $O\left(N \log N\right)$ space and $O\left(1\right)$ query time of Thorup [28] in $O\left(N \log^3 N\right)$ time. We assume that $\varepsilon \geq 2/N$ because otherwise we can just run a linear-time SSSP algorithm [17].

Given two vertices of u and v, we obtain a 2-approximation d of the u-to-v distance by querying the oracle of Thorup [28]. Then, we compute the most significant bit [13] (this is where we need the assumption of the word RAM model) of d to identify a scale r such that $r/2 \leq d \leq r$ in constant time. Finally, we visit the oracle of each of the $O\left(1\right)$ subsets $V_j^{(r)}$ that contain u, compute an approximation $\widehat{d}_j$ of additive stretch of the u-to-v distance therein, and set $\widetilde{d}$ to be the minimum of the values $\widehat{d}_j + \varepsilon r/2$.

Approximation factor. We now show that for any two vertices u, v of $\mathcal{G}$, our oracle returns a value $\widetilde{d}$ with $dist_{\mathcal{G}}[u,v] \leq \widetilde{d} \leq (1+O(\varepsilon))dist_{\mathcal{G}}[u,v]$.

We first claim that $r \in R$. To see this, recall that in $\mathcal{G}^{(r)}$ all edges of weight at least $24r$ have been deleted and those of weight at most r/n^2 have been contracted. Notice that the former edges do not participate in the u-to-v shortest path in $\mathcal{G}$ because the length of that path is at most r. Since the diameter of $\mathcal{G}^{(r)}$ is at most $24r-18$, these edges are not used in any shortest path therein. Let L be the largest summation of distances in any contracted path of $\mathcal{G}^{(r)}$. Since $\varepsilon > 2/N$, we have $L \leq r/N \leq \varepsilon r/2$. The length of the u-to-v shortest path in $\mathcal{G}$ is at least $r/2$, so not all of its edges are contracted in $\mathcal{G}^{(r)}$, i.e., $r \in R$. This also shows that $dist_{\mathcal{G}^{(r)}}[u,v] \leq dist_{\mathcal{G}}[u,v] \leq dist_{\mathcal{G}^{(r)}}[u,v] + \varepsilon r/2$.

From the first property of sparse neighborhood covers, we know that there is at least one subset $V_i^{(r)}$ that contains both u and v, such that the induced subgraph of $\mathcal{G}^{(r)}$ has diameter at most $24r-18$. From the third, we have $dist_{\mathcal{G}_i^{(r)}}[u,v] = dist_{\mathcal{G}^{(r)}}[u,v]$. Moreover, for the u-to-v distance $\widehat{d}_i$ returned by our oracle of additive stretch for $\mathcal{G}_i^{(r)}$, we have $dist_{\mathcal{G}_i^{(r)}}[u,v] \leq \widehat{d}_i \leq dist_{\mathcal{G}_i^{(r)}}[u,v] + O(\varepsilon r)$. Combining everything,

$$\begin{aligned} dist_{\mathcal{G}}[u,v] &\leq \widehat{d}_i + \varepsilon r/2 \\ &\leq dist_{\mathcal{G}}[u,v] + O(\varepsilon dist_{\mathcal{G}}[u,v]) \;\leq\; (1+O(\varepsilon))dist_{\mathcal{G}}[u,v]. \end{aligned}$$

By adjusting ε by a constant factor, the approximation factor of our oracle can become $1+\varepsilon$.

Time and space analysis. The preprocessing time is dominated by the time required to build the oracles of additive stretch for the graphs $\mathcal{G}_i^{(r)}$ associated with the sparse neighborhood cover of $\mathcal{G}^{(r)}$ for each scale $r \in R$ (the same reasoning applies for the space). We can construct each such oracle in $O\left(n\log^2 n + (1/\varepsilon)n\log n\right)$ time, where n is the number of vertices of the corresponding graph. Since each edge of $\mathcal{G}$ appears in $O(\log N)$ graphs $\mathcal{G}_i^{(r)}$, the summation of n's over all oracles of additive stretch is $O(N\log N)$. Therefore, the total preprocessing time and space of our $(1+\varepsilon)$-approximate oracle is $O\left(N\log^3 N + (1/\varepsilon)^5 N\log^2 N\right)$ and $O\left((1/\varepsilon)N\log^2 N + (1/\varepsilon)^{4+\delta}N\log N\right)$ respectively. The query time is $O(\log(1/\varepsilon))$ because we spend $O(1)$ time to find the appropriate scale and query $O(1)$ oracles of additive stretch, each in $O(\log(1/\varepsilon))$ time.

Theorem 9 (Approximate distance oracle) *We can construct a $(1+\varepsilon)$-approximate distance oracle for a non-negatively-weighted, undirected planar graph of n vertices with $O\left(n\log^3 n + (1/\varepsilon)^5 n\log^2 n\right)$ preprocessing time, $O\left((1/\varepsilon)n\log^2 n + (1/\varepsilon)^{4+\delta}n\log n\right)$ space, and $O(\log(1/\varepsilon))$ query time in the word RAM model, for any constant $\delta > 0$.*

6 Conclusion

It would be interesting to investigate whether the logarithmic factors of our near-$O\left(n \log^2 n\right)$-time approximation algorithm for the diameter of a non-negatively-weighted, undirected planar graph of n vertices can be improved.

Another problem that remains open is reducing the space of our $(1+\varepsilon)$-approximate distance oracles to linear while keeping the query time independent of n. Kawarabayashi et al. [18] presented oracles of linear space but nearly $O\left(\log^2 n\right)$ query time.

Acknowledgements We thank the reviewers for their careful reading and comments.

References

1. Stephen Alstrup, Jacob Holm, Kristian de Lichtenberg, and Mikkel Thorup. Maintaining information in fully dynamic trees with top trees. *ACM Transactions on Algorithms*, 1(2):243–264, 2005.
2. Piotr Berman and Shiva Prasad Kasiviswanathan. Faster approximation of distances in graphs. In *Proceedings of the 10th International Symposium on Algorithms and Data Structures (WADS)*, pages 541–552, 2007.
3. Costas Busch, Ryan LaFortune, and Srikanta Tirthapura. Improved sparse covers for graphs excluding a fixed minor. In *Proceedings of the 26th ACM Symposium on Principles of Distributed Computing (PODC)*, pages 61–70, 2007.
4. Sergio Cabello. Subquadratic algorithms for the diameter and the sum of pairwise distances in planar graphs. In *Proceedings of the 28th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2143–2152, 2017.
5. Sergio Cabello. Subquadratic algorithms for the diameter and the sum of pairwise distances in planar graphs. *CoRR*, abs/1702.07815v1, 2017.
6. Timothy M. Chan. All-pairs shortest paths for unweighted undirected graphs in $o(mn)$ time. *ACM Transactions on Algorithms*, 8:1–17, 2012.
7. Timothy M. Chan and Dimitrios Skrepetos. Faster approximate diameter and distance oracles in planar graphs. In *Proceeding of the 25th European Symposium on Algorithms (ESA)*, pages 25:1–25:13, 2017.
8. Kenneth L. Clarkson and Peter W. Shor. Applications of random sampling in computational geometry, II. *Discrete & Computational Geometry*, 4(5):387–421, 1989.
9. Hubert De Fraysseix, János Pach, and Richard Pollack. How to draw a planar graph on a grid. *Combinatorica*, 10(1):41–51, 1990.
10. James R. Driscoll, Neil Sarnak, Daniel D. Sleator, and Robert E. Tarjan. Making data structures persistent. *Journal of Computer and System Sciences*, 38(1):86–124, 1989.
11. David Eppstein. Subgraph isomorphism in planar graphs and related problems. *Journal of Graph Algorithms and Applications*, 3(3):283–309, 1999.
12. Greg N. Frederickson. Fast algorithms for shortest paths in planar graphs, with applications. *SIAM Journal on Computing*, 16:1004–1022, 1987.
13. Michael L. Fredman and Dan E. Willard. Surpassing the information theoretic bound with fusion trees. *Journal of Computer and System Sciences*, 47(3):424–436, 1993.
14. Pawel Gawrychowski, Haim Kaplan, Shay Mozes, Micha Sharir, and Oren Weimann. Voronoi diagrams on planar graphs, and computing the diameter in deterministic $\widetilde{O}\left(n^{5/3}\right)$ time. In *Proceedings of the 29th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 495–514, 2018.
15. Pawel Gawrychowski, Shay Mozes, Oren Weimann, and Christian Wulff-Nilsen. Better tradeoffs for exact distance oracles in planar graphs. In *Proceedings of the 24th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 515–529, 2018.

16. Qian-Ping Gu and Gengchun Xu. Constant query time $(1 + \varepsilon)$-approximate distance oracle for planar graphs. In *Proceedings of the 26th International Symposium on Algorithms and Computation (ISAAC)*, pages 625–636, 2015. *Theoretical Computer Science*, to appear.
17. Monika R. Henzinger, Philip Klein, Satish Rao, and Sairam Subramanian. Faster shortest-path algorithms for planar graphs. *Journal of Computer and System Sciences*, 55(1):3–23, 1997.
18. Ken-ichi Kawarabayashi, Philip N. Klein, and Christian Sommer. Linear-space approximate distance oracles for planar, bounded-genus and minor-free graphs. In *Proceedings of the 38th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 135–146, 2011.
19. Ken-ichi Kawarabayashi, Christian Sommer, and Mikkel Thorup. More compact oracles for approximate distances in undirected planar graphs. In *Proceedings of the 24th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 550–563, 2013.
20. Philip N. Klein. Preprocessing an undirected planar network to enable fast approximate distance queries. In *Proceedings of the 13th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 820–827, 2002.
21. Philip N. Klein. Multiple-source shortest paths in planar graphs. In *Proceedings of the 16th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 146–155, 2005.
22. Rolf Klein. *Concrete and Abstract Voronoi Diagrams*, volume 400 of *Lecture Notes in Computer Science*. Springer, 1989.
23. Rolf Klein, Kurt Mehlhorn, and Stefan Meiser. Randomized incremental construction of abstract Voronoi diagrams. *Computational Geometry*, 3(3):157–184, 1993.
24. Richard J. Lipton and Robert Endre Tarjan. A separator theorem for planar graphs. *SIAM Journal on Applied Mathematics*, 36(2):177–189, 1979.
25. Ketan Mulmuley. *Computational Geometry: An Introduction Through Randomized Algorithms*. Prentice Hall, 1994.
26. Walter Schnyder. Embedding planar graphs on the grid. In *Proceedings of the 1st ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 138–148, 1990.
27. Daniel D. Sleator and Robert Endre Tarjan. A data structure for dynamic trees. *Journal of Computer and System Sciences*, 26(3):362–391, 1983.
28. Mikkel Thorup. Compact oracles for reachability and approximate distances in planar digraphs. *Journal of the ACM*, 51(6):993–1024, 2004.
29. Oren Weimann and Raphael Yuster. Approximating the diameter of planar graphs in near linear time. *ACM Transactions on Algorithms*, 12(1):12, 2016.
30. Christian Wulff-Nilsen. *Algorithms for planar graphs and graphs in metric spaces*. PhD thesis, University of Copenhagen, 2010.
31. Christian Wulff-Nilsen. Approximate distance oracles for planar graphs with improved query time-space tradeoff. In *Proceedings of the 27th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 351–362, 2016.