Truly Subquadratic Exact Distance Oracles with Constant Query Time for Planar Graphs

Viktor Fredslund-Hansen * Shay Mozes † Christian Wulff-Nilsen ‡

Abstract

Given an undirected, unweighted planar graph G with n vertices, we present a truly sub-quadratic size distance oracle for reporting exact shortest-path distances between any pair of vertices of G in constant time. For any $\varepsilon > 0$, our distance oracle takes up $O(n^{5/3+\varepsilon})$ space and is capable of answering shortest-path distance queries exactly for any pair of vertices of G in worst-case time $O(\log(1/\varepsilon))$. Previously no truly sub-quadratic size distance oracles with constant query time for answering exact all-pairs shortest paths distance queries existed.

*Department of Computer Science, University of Copenhagen, viha@di.ku.dk

†Efi Arazi school of Computer Science, IDC Herzliya, Israel. smozes@idc.ac.il https://cs.idc.ac.il/~smozes/. Partially supported by Israel Science Foundation grant no. 592/17.

‡Department of Computer Science, University of Copenhagen, koolooz@di.ku.dk, http://www.diku.dk/~koolooz/. This research is supported by the Starting Grant 7027-00050B from the Independent Research Fund Denmark under the Sapere Aude research career programme.

1 Introduction

Efficiently answering shortest path distance queries between pairs of vertices in a graph is a fundamental algorithmic problem with numerous important applications. Given an n-vertex graph $G = (V, E)$ a *distance oracle* is a compact data-structure capable of efficiently answering shortest path distance queries between pairs of vertices $u, v \in V$. Ideally one would like the data structure to be of linear size and the query time to be constant. However, it is well known that there are graphs for which no distance oracle with $o(n^2)$ bits of space and $O(1)$ query time exists. In fact, even resorting to approximation, Pătraşcu and Roditty [20] showed that there are sparse graphs on $O(n \operatorname{polylog} n)$ edges for which constant query-time distance oracles with stretch less than 2 must be of size $\Omega(n^2 \operatorname{polylog} n)$, assuming the set intersection conjecture. These impossibility results make it natural to consider the probelms in restricted classes of graphs.

In this paper we consider exact distance oracles for planar graphs. Distance oracles for planar graphs are well motivated by important real-world applications, notably in routing, navigation of road and sea maps as well as in the context of computational geometry. To the best of our knowledge there are no non-trivial lower bounds for (static) distance oracles for planar graphs, and thus achieving the "holy grail" of a linear-size distance oracle with constant query time may be possible. Indeed, there has been numerous works over at least three decades developing exact distance oracles for planar graph [10, 2, 3, 8, 11, 18, 19]. However, only recently Cohen-Addad et al. [9] gave the first oracle with truly subquadratic space and polylogarithmic query time. Their result was inspired by Cabello's [4] breakthrough result, who gave the first truly sub-quadratic time algorithm for computing the diameter of planar graphs by a novel use of Voronoi diagrams. The approach of [9] was subsequently improved by [12, 7], who gave an elegant point-location mechanism for Voronoi diagrams in planar graphs, and combined it with a clever recursive scheme to obtain exact distance oracles for directed weighted planar graphs with $O(n^{1+\varepsilon} \operatorname{polylog} n)$ space and $O(\log^{1/\varepsilon} n)$ query time for any small constant ε. We note that even though the oracle of [7] gets quite close to optimal, it remains wide open to support exact queries in *constant time* using truly subquadratic space, even in the most basic case of unweighted undirected planar graphs [25, 5, 24].

Allowing approximate answers does help in planar graphs. Many results reporting $(1+\varepsilon)$-approximate distances with various tradeoffs exist, all with (nearly) linear size and polylogarithmic, or even $O(1/\varepsilon)$ query-time [23, 15, 14, 26]. Gu and Xu[13] presented a size $O(n \operatorname{polylog} n)$ distance oracle capable of reporting $(1+\varepsilon)$-approximate distances in time $O(1)$. While their query time is a constant independent of ε, the preprocessing time and space are nearly linear, but with an exponential dependency on $(1/\varepsilon)$. This exponential dependency was recently improved to polynomial [6].

Thus, despite the large body of work on distance oracles for planar graphs, it has remained an open question to determine whether an exact distance oracle with of size $O(n^{2-\varepsilon})$ with *constant* query-time can be constructed for some constant $\varepsilon > 0$.

Our results and techniques. We answer this question in the affirmative. Our result is presented in the following theorem:

Theorem 1. *Let $G = (V, E)$ be an undirected unweighted n-vertex planar graph. For any $\varepsilon > 0$ there exists a data-structure requiring $O(n^{5/3+\varepsilon})$ space that, for any $s, t \in V$, reports the shortest path distance between s and t in G in time $O(\log(1/\varepsilon))$.*

We remark that the simple distance oracle we present in Section 4 can be distributed into a distance labeling scheme with size $O(n^{3/4})$ per label, such that the distance between any two vertices s, t can be computed in $O(1)$ time given just the labels of s and t.

The main concept we use to obtain our result is that of a *pattern* capturing distances between a vertex and a cycle. This concept was used by [25] , and weas called "distance tuple" by and [17]. Consider a vector storing the distances from a vertex u to the vertices of a cycle β in their cyclic order. The pattern of u w.r.t. β is simply the discrete derivative of this vector. That is, the vector obtained by taking the difference between every pair of consecutive values. Li and Parter [17] observed that when the input graph is planar, the number of different patterns w.r.t. a face with r vertices is $O(r^3)$ regardless of the size of the graph. We next outline how this observation can be used to break the quadratic space barrier.

Roughly speaking, any planar graph can be decomposed into $O(n/r)$ subgraphs, called regions, of size r each, where the boundary of each region (i.e., the vertices that have neighbors outside the region) is a single cycle h.[1] Applying Li and Parter's observation in this setting, the number of different patterns for the hole of each region R is $O(r^3)$. Hence, we can represent the distances from any vertex $s \notin R$ to h by just storing the distance from s to an arbitrarily chosen canonical vertex v_h of h, and a pointer to the pattern of s with respect to h. This requires just $O(n)$ space plus $O(r^3 \cdot r)$ for storing all the patterns for h. Summing over all $O(n/r)$ regions, the space required is $O(n^2/r + nr^3)$. We define the notion of distance from a pattern to a vertex (see Definition 2). While this definition is simple, it is somewhat unnatural because the distance from a pattern to a vertex does not necessarily correspond to the length of any specific path in the graph! However, the distance between s and any vertex $t \in R$ is just the sum of the distance between s and the canonical vertex v_h and the distance from the pattern of s with respect to h to t.

We therefore store the distances from each of the $O(r^3)$ possible patterns of R to each vertex of R. This requires $O(r^3 \cdot r)$ space per region, so $O(nr^3)$ space overall. This way we can report the distance between s and t in constant time by (i) retrieving the pattern p of s w.r.t. h, and (ii) adding the distance from s to the canonical vertex v_h of h and the distance from the pattern p to t. These ideas alone already imply an oracle with space $\tilde{O}(n^{7/4})$ and constant query time. Combining these ideas with recursion yields the improved space bound of Theorem 1.

As we argued in the introduction, breaking the quadratic space barrier for constant query time is important and significant result in its own right. We highlight the following difference between the approach taken in our recursive oracle and the approaches used in all existing distance oracles we are aware of. To the best of our knowledge, all existing distance oracles, both exact and approximate, and both for general graphs and planar graphs, recover the distance from s to t by identifying a vertex or vertices on some (possibly approximate) shortest path between s and t, for which distances have been stored in the preprocessing stage. These vertices are usually referred to as landmarks, portals, hubs, beacons, seeds, or transit nodes [22]. Our oracle, on the other hand, reports the exact shortest path without identifying vertices on the shortest path from s to t. Instead, it "zooms in" on t by recovering distances to the canonical vertices of a sequence of subgraphs of decreasing size that terminates at a constant size graph containing t. We emphasize that *none of these canonical vertices necessarily lies on a shortest path* from s to t. This property may be viewed as a disadvantage if we also want to report the shortest path, but when reporting

[1] In fact, a constant humber of cycles. To readers familiar with the concept, this is just an r-division with a few holes, but *without* the important feature that each region has just $O(\sqrt{r})$ boundary vertices. This is because one cannot triangulate unweighted graphs without changing the distances.

multiple edges on long paths, constant query time is no longer relevant. On the other hand, it may be that just reporting the distance is easier than also reporting an internal vertex on a shortest path. Hence, it may be that developing oracles based on this new approach may lead to further advances on the way to linear size distance oracles for planar graphs with constant query time, and in other related problems.

2 Preliminaries

Let G be a graph. We denote by $V(G)$ and $E(G)$ the vertex and edge-set of G, and denote by $n = |V(G)|$ the number of vertices of G. For a subset S of edges or vertices we denote by $G[S]$ the subgraph of G induced on S. We denote by $u \rightsquigarrow_H v$ a *shortest path* from u to v in the subgraph H, by $\mathbf{d}_H(u, v)$ the length of $u \rightsquigarrow_H v$, and define $u \rightsquigarrow v \equiv u \rightsquigarrow_G v$.

The following definitions will be useful when talking about decompositions of G. A *region* R of G is an edge-induced subgraph of G, and its *boundary* ∂R is the vertices of R that are adjacent to some vertex of $V(G) \setminus V(R)$ in G. Vertices of $V(R) \setminus \partial R$ are called *interior* vertices of R. Observe that for a region R and for $u \in R$ and $v \in V \setminus V(R)$, any path from u to v in G must intersect ∂R.

It will be useful to assume some global strict order on a vertex set V s.t. for any $U \subseteq V$ there is a minimum vertex $\min U \in U$ w.r.t this order. We refer to this as the *canonical vertex* of U.

Faces and holes: We assume the reader is familiar with the the basic definitions of planarity and of planar embeddings. The edges of a plane graph induce maximal open portion of the plane that do not intersect any edges. A *face* of the graph is the closure of one such portion of the plane. We refer to the edges bounding a face as the *boundary* of that face. Given a face f, $V(f)$ is the set of vertices on the boundary of f. We denote by $w(f)$ the *facial walk* of f which is the sequence of vertices encountered when walking along f starting at $\min V(f)$ and going in the clockwise direction. Note that f may be non-simple, so some vertices may appear multiple times in $w(f)$.

A *hole* h in a region R of a graph G is a face of R which is not a face in G. We say that a vertex $u \in V(G) \setminus V(R)$ is *inside* hole h if u lies in the region of the plane corresponding to the face h of R. We denote by $V^\circ(h) = \{u \in V(G) \mid u \text{ is inside } h\}$ all the vertices that are inside h.

Decompositions of unweighted planar graphs. An r-division is a widely used decomposition of planar graphs into regions with small boundary. We use the r-divisions with a few holes as studied in [16], which works for triangulated biconnected graphs:

Lemma 1. *(r-division with few holes for triangulated graphs [16]) Let G be a biconnected, triangulated n-vertex planar embedded graph, and let $0 < r \leq n$. G can be decomposed into $\Theta(n/r)$ connected regions, each of which with $O(r)$ vertices and $O(\sqrt{r})$ boundary vertices. Each region has a constant number of holes. Every boundary vertex lies on some hole, and each hole has $O(\sqrt{r})$ vertices.*

The fact that the boundaries of regions are small (only $O(\sqrt{r})$ boundary vertices for a region with r vertices) is the basis for many efficient algorithms and data structures for planar graphs. Unweighted planar graphs posses additional structure (in comparison to weighted planar graphs), which may also be useful algorithmically. See for example the unit-Monge property in [1], or the limited number of patterns [25, 17], which we use in this work. However, exploiting such additional structure in conjunction with a decomposition into regions with small boundaries has been elusive

because of the seemingly technical requirement in Lemma 1 that the graph be triangulated and biconnected.

Any graph can be triangulated and biconnected by adding to each face f an artificial vertex and infinitely weighted artificial edges from the artificial vertex to each vertex of $V(f)$. This transformation preserves planarity and shortest paths, and ensures that the graph consists only of simple faces of size 3. However, the graph is no longer unweighted. We refer to an *artificial* vertex (edge) of G as a vertex (edge) which was added in the triangulation step, and a *natural* vertex (edge) of G as a vertex (edge) which is not artificial. In order to exploit the structure of the unweighted input graph we will remove the artificial edges and vertices after computing the decomposition using Lemma 1. On the one hand the graph is again unweighted. On the other hand, while the number of boundary vertices in each region remains $O(\sqrt{r})$, the holes may now contain new non-boundary vertices, and the total size of the holes in each region may be $\Theta(r)$. We note, however, that the deletion of artificial edges and vertices does not disconnect regions [16]. We therefore restate the decomposition lemma for unweighted graphs that are not necessarily triangulated or biconnected.

Lemma 2. *(r-division with few holes for non-triangulated graphs) Let G be a n-vertex planar embedded graph G, and let $0 < r \leq n$. G can be decomposed into $\Theta(n/r)$ connected regions, each of which with $O(r)$ vertices and $O(\sqrt{r})$ boundary vertices. Each region has a constant number of holes, and each boundary vertex lies on some hole.*

Recursive r-divisions. Our second construction relies on a *recursive r-division* which is a recursive decomposition of G into r-divisions for varying values of r. Specifically, for a decreasing sequence $\mathbf{r} = r_1, r_2, \ldots$, where $n \geq r_1 > r_2 > \ldots \geq 1$, we want r_i-divisions for all $i = 1, 2, \ldots$, such that each region in the r_i division is the union of regions in the r_{i+1}-division on the next level. We associate with the recursive r-division a decomposition tree, $\mathcal{T}_{\mathbf{r}}$, which is a rooted tree whose nodes correspond to the regions of the recursive decomposition of G. We will refer to nodes and their corresponding regions interchangeably. The root node corresponds to all of G. A node x of $\mathcal{T}_{\mathbf{r}}$ at depth i corresponds to a region of the r_i-division, and its children are the regions of the r_{i+1}-division whose union is the region corresponding to x. We denote by $\mathcal{T}_{\mathbf{r}}^i$ all the nodes at level i. It was shown in [16] that recursive r-divisions can be computed efficiently:

Lemma 3. *(Recursive r-division) Given a biconnected, triangulated n-vertex planar graph G and an exponentially decreasing sequence $\mathbf{r} = n \geq r_1, r_2, \ldots \geq 1$, a decomposition tree, $\mathcal{T}_{\mathbf{r}}$ can be computed in linear time s.t $\mathcal{T}_{\mathbf{r}}^i$ corresponds to an r_i-division of G with few holes for each i.*

3 Patterns

Both [25] and [17] introduce a notion of a "distance tuple" which can be thought of as a vector of shortest-path distances from a vertex to consecutive vertices of some hole. We introduce the following similar notion of a *pattern* (See Figure 1 for an illustration):

Definition 1. *(Pattern) Let G be a graph. Let H be a subgraph of G. Let u be a vertex in H, and let $\beta = b_0, b_1, \ldots, b_k$ be a path in H. The pattern of u (w.r.t. β in H) is a vector $p_{\beta,H}(u) \in \{-1, 0, 1\}^k$ satisfying $p_{\beta,H}(u)[i] = \mathbf{d}_H(u, b_i) - \mathbf{d}_H(u, b_{i-1})$ for $1 \leq i \leq k$. For a region R in G, a hole h of R, and a vertex $u \in V^\circ(h)$, we write $p_{h,G}(u)$ instead of $p_{w(h),G}(u)$.*

Figure 1: Illustration of the pattern of the vertex s w.r.t. f^* in an undirected graph. In this case f^* is the external face of the embedding. The numeric labels indicate the shortest path distances from s to each b_i where $b_i \in V(f^*)$ for $1 \leq i \leq 6$.

Definition 2. *(pattern to vertex distance) Let R be a region in a graph G. Let h be a hole of R. Let $b_0, b_1, \ldots, b_k$ be the vertices of $w(h)$ in their cyclic order. Let p be some pattern w.r.t. h (i.e., $p = p_h(u)$ for some $u \in V^\circ(h)$). For a vertex $v \in R$ we define $\mathbf{d}_G(p, v)$ the distance between p and v to be $\min_{i=0}^{k} \left\{ \mathbf{d}_G(b_i, v) + \sum_{j=0}^{i} p[j] \right\}$.*

Lemma 4. *Let R be a region of a graph G. Let h be a hole of R. For every $u \in V^\circ(h)$ and every $v \in R$, $\mathbf{d}_G(u, v) = \mathbf{d}_G(u, b_0) + \mathbf{d}_G(p_h(u), v))$.*

Proof. By definition of pattern and by a telescoping sum, for every $0 \leq i \leq k$, $\mathbf{d}_G(u, b_i) = \mathbf{d}_G(u, b_0) + \sum_{j=0}^{i} p[j]$. Let b_ℓ be any vertex of $w(h)$ on a shortest u-to-v path (b_ℓ exists since $u \in V^\circ(h)$ and $v \in R$). By choice of b_ℓ, $\mathbf{d}_G(u, v) = \mathbf{d}_G(u, b_\ell) + \mathbf{d}_G(b_\ell, v) = \min_{0 \leq i \leq k} \{\mathbf{d}_G(u, b_i) + \mathbf{d}_G(b_i, v)\} = \min_{0 \leq i \leq k} \left\{ \mathbf{d}_G(u, b_0) + \sum_{j=0}^{i} p[j] + \mathbf{d}_G(b_i, v) \right\} = \mathbf{d}_G(u, b_0) + \mathbf{d}_G(p, v)$. □

Bounding the number of patterns As mentioned, In a recent paper, Li and Parter [17] achieve improved bounds for diameter computation for planar graphs by showing that in unweighted undirected planar graphs the number of patterns is quite small. More specifically, they show that the VC-dimension of a set corresponding to all patterns is at most 3. By the Sauer-Shelah lemma [21], this implies that the number of distinct patterns w.r.t. a face f is in $O(|S|^3)$. Their result is stated in the following lemma:

Lemma 5. *(Pattern compression) [17] Let $G' = (V, E)$ be an n-vertex unweighted undirected planar graph, let f be a face in G', and let S be a set of consecutive vertices on f. Then the number of distinct patterns w.r.t. S, $|\cup_{u\in V}\{p_{S,G'}(u)\}|$, is bounded by $O(|S|^3)$.*

We observe that the bound of Lemma 5 also holds for patterns w.r.t. the entire set of vertices on a hole h of a region R even when distances are defined in the entire graph G.

Corollary 1. *Let R be a region in an n-vertex unweighted undirected planar graph G, and let h be a hole of R. Then the number of distinct patterns w.r.t. h, $|\cup_{u\in V^\circ[h]}\{p_{h,G}(u)\}|$, is bounded by $O(|h|^3)$.*

Proof. Since h is a hole of R, h is a face of $G-(R-h)$. By Lemma 5, $|\cup_{v\in V^\circ[h]}\{p_{h,G-(R-h)}(v)\}| = O(|h|^3)$. The corollary follows since for every two vertices $u, u' \in V^\circ(h)$, $p_{h,G-(R-h)}(u) = p_{h,G-(R-h)}(u')$ implies $p_{h,G}(u) = p_{h,G}(u')$. □

For the remainder of the paper we only deal with distances in G and with patterns in G, so we will omit the subscript G, and write $\mathbf{d}(\cdot,\cdot)$ and $p_h(\cdot)$ instead of $\mathbf{d}(\cdot,\cdot)$ and $p_{h,G}(\cdot)$.

4 $O(n^{7/4})$ space distance oracle

Before presenting our main result, we describe a simpler construction which yields a distance oracle with a larger space requirement of $O(n^{7/4})$ and $O(1)$ query time:

Preprocessing. The preprocessing consists of computing an r-division $\mathcal{R}$ of G with a parameter r to be determined later. For every vertex v of G and every region R of $\mathcal{R}$, we store the hole h of R s.t. v is in $V^\circ(h)$. This requires $O(n \cdot n/r) = O(n^2/r)$ space.

For every region $R \in \mathcal{R}$, for every hole h of R, we maintain the $O(r^3)$ patterns of the vertices in $V^\circ(h)$ w.r.t. h as follows. Let k denote the size of the boundary walk $w(h)$ of h. let v_h be the canonical (i.e., first) vertex of $w(h)$. We maintain the patterns seen so far in a ternary tree $\mathcal{A}$ whose edges are labeled by $\{-1, 0, 1\}$. The depth of $\mathcal{A}$ is $k-1$, and the labels along each root-to-leaf path correspond to a unique pattern, which we associate with that leaf. For every vertex $v \in V^\circ(h)$, we compute the pattern $p_h(v)$ and we make sure that $p_h(v)$ is represented in the tree $\mathcal{A}$ by adding the corresponding labeled edges that are not yet present in $\mathcal{A}$. After all the vertices in $V^\circ(h)$ were handled, the tree $\mathcal{A}$ has $O(r^3)$ leaves. For each leaf of $\mathcal{A}$ with an associated pattern p, we compute and store (i) the distance from p to each vertex of R. This requires $O(r^4)$ time and space for all leaves of $\mathcal{A}$, so a total of $O(n/r \cdot r^4) = O(nr^3)$ space for storing all this information over all regions.

For each vertex $v \in V^\circ(h)$ we store (ii) a pointer to (the leaf of $\mathcal{A}$ that is associated with) the pattern $p_{h,G}(v)$, as well as (iii) the distance $\mathbf{d}(v, v_h)$ between v and the canonical vertex of h. The total space required to store all these pointers and distances is $O(n \cdot n/r) = O(n^2/r)$.

To complete the preprocessing we also store (iv) for each region $R \in \mathcal{R}$, the distance $\mathbf{d}(u, v)$ for all pairs of vertices $u, v \in R$. This takes $O(n/r \cdot r^2)$ additional space, which is dominated by the above terms.

The total space required by the oracle is thus $O(n^2/r) + O(nr^3)$. This is minimized for $r = n^{1/4}$, resulting in an $O(n^{7/4})$-space data structure.

We note that once this information has been computed we no longer need to store the entire tree $\mathcal{A}$. Rather, it suffices to only store just the list of leaves of $\mathcal{A}$ and the distances stored with

each of them. In particular, we no longer need to remember what is the actual pattern associated with each leaf, we only need to know the distances from this pattern to the vertices of the region R. In the current scheme this has no asymptotic effect on the size of the data structure, since each pattern is of size $O(r)$, and we anyway store the $O(r)$ distances from each pattern to all vertices of R. However, in the recursive scheme in the next section this observation will become useful.

Query. To answer a query for the distance between vertices s and t we proceed as follows. If s and t are in the same regions, we simply return the distance $\mathbf{d}(s,t)$ stored in item (iv). Otherwise, let R be the region containing t, and let h be the hole of R such that $s \in V^\circ(h)$. Let v_h be the canonical vertex of h. We return $\mathbf{d}(s,v_h) + \mathbf{d}(p_{h,G}(s),t)$. The correctness is immediate from Lemma 4. We note that $\mathbf{d}(s,v_h)$ is stored in item (iii), a pointer to $p_{h,G}(s)$ is stored in item (ii), and $\mathbf{d}(p_{h,G}(s),t)$ is stored in item (i). The query is illustrated in Figure 4.

Figure 2: Illustration of the query in Section 4. By Lemma 4 the query returns $\mathbf{d}(s,b_0)+\mathbf{d}_R(p,t) = \mathbf{d}(s,b_\ell)+\mathbf{d}_R(b_\ell,t) = \mathbf{d}(s,t)$ where b_ℓ is some boundary vertex of R on $s \rightsquigarrow t$.

As aforementioned this oracle can be distributed into a distance labeling of size $O(n^{3/4})$ per label such that the distance between any two vertices s,t can be computed in $O(1)$ time given just the labels of s and t.

5 $O(n^{5/3+\varepsilon})$ space distance oracle

A bottleneck in the above approach comes from having to store, for each pattern p of a hole h of a region R, the distances from p to all vertices of R. Instead, we use a recursive r-division, in which

we store for p, only the distances to the canonical vertex of a hole h' of each child region R' of R instead of all the vertices in the region. For this information to be useful we also store the pattern induced by p on the hole h', which is defined as follows.

Definition 3. *(Pattern induced by a pattern) Let R be a region in a graph G. Let h be a hole of R and p_h be a pattern of h (w.r.t. a vertex or another pattern). Let R' be a child region of R. Let $v_{h'} = b_0, b_1, \ldots, b_k$ be the vertices of the boundary walk of a hole of h' of R'. The pattern induced by p_h on h' is the vector $p_{h'}$ satisfying $p_{h'}[i] = \mathbf{d}(p, b_i) - \mathbf{d}(p, b_{i-1})$ for $1 \leq i \leq k$.*

Lemma 6. *Consider the settings of Definition 3. If $p_h = p_h(u)$ for some $u \in V^\circ(h)$, then $p_{h'} = p_{h'}(u)$.*

Proof. By Lemma 4, for every $0 \leq i \leq k$, $\mathbf{d}(u, b_i) - \mathbf{d}(u, v_h) = \mathbf{d}(p, b_i)$. Hence for all $1 \leq i \leq k$, $p_{h'}[i] = \mathbf{d}(p, b_i) - \mathbf{d}(p, b_{i-1}) = \mathbf{d}(u, b_i) - \mathbf{d}(u, v_h) - (\mathbf{d}(u, b_{i-1}) - \mathbf{d}(u, v_h)) = \mathbf{d}(u, b_i) - \mathbf{d}(u, b_{i-1})$, which is, by definition, $p_{h'}(u)[i]$. □

Preprocessing. We first compute a $\mathbf{r} = (r_0, r_1, \ldots, r_k, r_{k+1})$-division of G for $\mathbf{r}$ to be determined later, and denote by $\mathcal{T}_\mathbf{r}$ the associated decomposition tree. For convenience, we let $r_0 = n$, $r_{k+1} = 1$ and define $C(R) = \{R' \mid R' \text{ is a child of } R \text{ in } \mathcal{T}_\mathbf{r}\}$. In the following we let P_h denote the set $\{p_h(u) : u \in G\}$. We store the following:

1. For each $u \in V(G)$ we store a list of regions $R_0 \supset R_1 \supset \cdots \supset R_k$ containing u, where $R_i \in \mathcal{T}^i_\mathbf{r}$. (Recall that $\mathcal{T}^i_\mathbf{r}$ is the set of all nodes of $T_\mathbf{r}$ at level i).

2. For each $u \in G$, for each $0 \leq i \leq k-1$, for each region $R \in \mathcal{T}^i_\mathbf{r}$ containing u, for each child region $R' \subset R$ at level-$(i+1)$, let h be the hole of R' such that $u \in V^\circ(h)$. We associate with the pair (u, R') (i) a pointer to $p_h(u)$, (ii) the canonical vertex v_h, and (ii) the distance $\mathbf{d}(u, v_h)$).

3. For each $1 \leq i \leq k$, for each $R \in \mathcal{T}^i_\mathbf{r}$, for each hole h in R, for each $p \in P_h$ and for each $R' \in C(R)$, let h' be the hole of R' such that $v_h \in V^\circ(h')$. We associate with the pair (p, R') (i) a pointer to the pattern $p_{h'}(p)$ induced by p on h', (ii) the canonical vertex $v_{h'}$, and (iii) the distance $\mathbf{d}(p, v_{h'})$).

Space analysis. Storing 1 requires space $O(kn)$. To bound the space for item 2, we note that the number of regions at level i to which a vertes u belongs is bounded by the degree of u. Since the average vertex degree in a planar graph is at most 6, the average number of regions at level i to which u belongs is at most 6. Each such region has r_i/r_{i+1} subregions at level-$(i+1)$, so storing 2 requires space $O(n \sum_{i=0}^{k-1} r_i/r_{i+1}) = O(n^2/r) + O(n \sum_{i=1}^{k-1} r_i/r_{i+1})$. Storing 3 requires space $O(\sum_{i=1}^{k} (n/r_i) \cdot r_i^3 \cdot r_i/r_{i+1}) = O(n \sum_{i=1}^{k} r_i^3/r_{i+1})$. The total space is thus, $O(nk + n^2/r + n \sum_{i=1}^{k} r_i^3/r_{i+1})$.

Query. Algorithm 1 show pseudocode describing the query procedure. To process a query $\mathbf{d}(s, t)$ the query procedure first determines the largest value i for which s and t belong to the same region in $\mathcal{T}^i_\mathbf{r}$. Note that such a region must always exists as the root of $\mathcal{T}_\mathbf{r}$ is all of G. This level can be found in $O(k)$ time by traversing $\mathcal{T}_\mathbf{r}$, starting from a leaf region containing s and a leaf region containing t.

Let R_t be the level-$(i+1)$ region stored for t in item 1. Note that, $t \in R_t$, and, by choice of i, $s \notin R_t$. Hence, s is in some hole h of R_t. We retrieve the pattern $p_h(s)$ and the distance $\mathbf{d}(s, v_h)$ associated with (s, R_t) in item 2. We then proceed iteratively "zooming" into increasingly smaller regions containing t.

We show that the algorithm maintains the invariant that, at the beginning of each iteration, we have a level-i region R_t containing t, the variable d stores $\mathbf{d}(s, v_h)$, where h is the hole of R_t such that $s \in V^\circ(h)$, and the variable p stores (a pointer) to the pattern $p_h(t)$. Thus, when we reach the singleton region containing t, the variable d stores $\mathbf{d}(s, t)$.

We have already established that the invariant is maintained just before the loop is entered for the first time. In each iteration of the loop we retrieve R'_t, a level-$(i+1)$ subregion or R_t containing t (available in item 1), and retrieve $d' \leftarrow \mathbf{d}(p, v_{h'})$ and $p' \leftarrow p_{h'}(p)$ (associated with the pair (p, R'_t) in item 3). By Lemma 4, $d + d' = \mathbf{d}(s, v_h) + \mathbf{d}(p_h(u), v_{h'}) = \mathbf{d}(s, v_{h'})$. By Lemma 6, $p' = p_{h'}(t)$. Hence, after the assignements in Line 9, the invariant is restored.

The time complexity of the query is clearly $O(k)$.

Choosing parameters: Recall that the space requirement is $O(nk + n^2/r + n\sum_{i=1}^{k} r_i^3/r_{i+1})$. Picking each r_i s.t. $r_i/r_{i+1} = r_1^\varepsilon$ results in $r_k = \Theta(1)$ when $k = \Theta(1/\varepsilon)$, and in a query time of $O(1/\varepsilon)$. Choosing $r_1 = n^{1/3+\varepsilon}$, the total space used becomes

$$O\left(n\sum_{i=1}^{k} r_i^3/r_{i+1}\right) = O(nr_1^2 r_1^\varepsilon) = O(n^{1+2/3+2\varepsilon+\varepsilon/3+\varepsilon^2}) = O(n^{5/3+\varepsilon'})$$

for a suitable choice of ε'.

One can decrease the sizes of regions more aggressively to get the query time of $k = O(\log(1/\varepsilon))$ of Theorem 1. To this end we choose $\mathbf{r}$ such that $r_i^3/r_{i+1} = n^{2/3+\varepsilon}$, and $r_1 = n^{1/3}$. Then the space requirement is $O(n^{5/3} + nkn^{2/3+\varepsilon}) = O(kn^{5/3+\varepsilon})$. It is not hard to verify that one gets $r_i = O(n^{1/3-\varepsilon\frac{3^{i-2}-1}{2}})$, so $r_k = O(1)$ with $k = O(\log(1/\varepsilon))$.

As a last remark we note that the smallest interesting choice of ε in Theorem 1 is $\Theta(1/\log n)$, giving $O(n^{5/3})$ space and $O(\log\log n)$ query-time, which is a faster query-time than was previously known for this amount of space [9, 7].

Algorithm 1 Query procedure for the $O(n^{5/3+\varepsilon})$ construction.

```
1:  procedure QUERY(s, t)
2:      i ← the largest i s.t. the region R_i stored in item 1 for t contains both s and t
3:      R_t ← level (i + 1) region stored in item 1 for t
4:      (p, d) ← the tuple associated with (s, R_t)
5:      i ← i + 1
6:      while i ≤ k do
7:          R'_t ← level (i + 1) subregion of R_t stored in item 1 for t
8:          (p', d') ← the tuple associated with (p, R'_t)
9:          d ← d + d' ; p ← p' ; R_t ← R'_t ; i ← i + 1
10:     return d
```

References

[1] A. Abboud, P. Gawrychowski, S. Mozes, and O. Weimann. Near-Optimal Compression for the Planar Graph Metric. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 530–549. Society for Industrial and Applied Mathematics, Philadelphia, PA, jan 2018.

[2] S. Arikati, D. Z. Chen, L. P. Chew, G. Das, M. Smid, and C. D. Zaroliagis. Planar spanners and approximate shortest path queries among obstacles in the plane. In J. Diaz and M. Serna, editors, *Algorithms — ESA '96*, pages 514–528, Berlin, Heidelberg, 1996. Springer Berlin Heidelberg.

[3] S. Cabello. Many Distances in Planar Graphs. *Algorithmica*, 62(1-2):361–381, feb 2012.

[4] S. Cabello. Subquadratic Algorithms for the Diameter and the Sum of Pairwise Distances in Planar Graphs. *ACM Transactions on Algorithms*, 15(2):1–38, dec 2018.

[5] T. M. Chan. All-pairs shortest paths for unweighted undirected graphs in $o(mn)$ time. *ACM Trans. Algorithms*, 8(4):34:1–34:17, 2012.

[6] T. M. Chan and D. Skrepetos. Faster Approximate Diameter and Distance Oracles in Planar Graphs. *Algorithmica*, 81(8):3075–3098, aug 2019.

[7] P. Charalampopoulos, P. Gawrychowski, S. Mozes, and O. Weimann. Almost optimal distance oracles for planar graphs. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2019, pages 138–151, New York, NY, USA, 2019. Association for Computing Machinery.

[8] D. Z. Chen and J. Xu. Shortest path queries in planar graphs. In *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing*, STOC 2000, pages 469–478, New York, NY, USA, 2000. Association for Computing Machinery.

[9] V. Cohen-Addad, S. Dahlgaard, and C. Wulff-Nilsen. Fast and Compact Exact Distance Oracle for Planar Graphs. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 962–973. IEEE, oct 2017.

[10] H. Djidjev. On-line algorithms for shortest path problems on planar digraphs. In *Proceedings of the 22nd International Workshop on Graph-Theoretic Concepts in Computer Science*, WG 1996, pages 151–165, Berlin, Heidelberg, 1996. Springer-Verlag.

[11] J. Fakcharoenphol and S. Rao. Planar graphs, negative weight edges, shortest paths, and near linear time. *Journal of Computer and System Sciences*, 72(5):868–889, aug 2006.

[12] P. Gawrychowski, S. Mozes, O. Weimann, and C. Wulff-Nilsen. Better tradeoffs for exact distance oracles in planar graphs. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA 2018, pages 515–529, USA, 2018. Society for Industrial and Applied Mathematics.

[13] Q. Gu and G. Xu. Constant query time $(1+\epsilon)$ -approximate distance oracle for planar graphs. In *26th ISAAC*, pages 625–636, 2015.

[14] K.-i. Kawarabayashi, P. N. Klein, and C. Sommer. Linear-Space Approximate Distance Oracles for Planar, Bounded-Genus, and Minor-Free Graphs. apr 2011.

[15] P. Klein. Preprocessing an undirected planar network to enable fast approximate distance queries. In *Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '02, pages 820–827, USA, 2002. Society for Industrial and Applied Mathematics.

[16] P. N. Klein, S. Mozes, and C. Sommer. Structured recursive separator decompositions for planar graphs in linear time. In *Proceedings of the 45th annual ACM symposium on Symposium on theory of computing - STOC '13*, page 505, New York, New York, USA, 2013. ACM Press.

[17] J. Li and M. Parter. Planar diameter via metric compression. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2019, pages 152–163, New York, NY, USA, 2019. Association for Computing Machinery.

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

[19] Y. Nussbaum. Improved distance queries in planar graphs. In *Proceedings of the 12th International Conference on Algorithms and Data Structures*, WADS 2011, pages 642–653, Berlin, Heidelberg, 2011. Springer-Verlag.

[20] M. Pătraşcu and L. Roditty. Distance Oracles beyond the Thorup–Zwick Bound. *SIAM Journal on Computing*, 43(1):300–311, jan 2014.

[21] N. Sauer. On the density of families of sets. *Journal of Combinatorial Theory, Series A*, 13(1):145–147, jul 1972.

[22] C. Sommer. Shortest-path queries in static networks. *ACM Computing Surveys*, 46(4):1–31, apr 2014.

[23] M. Thorup. Compact oracles for reachability and approximate distances in planar digraphs. *Journal of the ACM (JACM)*, 51(6):993–1024, nov 2004.

[24] C. Wulff-Nilsen. *Algorithms for Planar Graphs and Graphs in Metric Spaces.* PhD thesis, 2010.

[25] C. Wulff-Nilsen. Constant time distance queries in planar unweighted graphs with subquadratic preprocessing time. *Computational Geometry*, 46(7):831–838, oct 2013.

[26] C. Wulff-Nilsen. Approximate distance oracles for planar graphs with improved query time-space tradeoff. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '16, pages 351–362, USA, 2016. Society for Industrial and Applied Mathematics.